1. Implement the following substitution and Transposition techniques:

- Caesar Cipher
- Columnar Transposition Cipher

```
1#ceaser cipher technique
def encrypt(text,s):
  res=""
  for i in range(len(text)):
    char=text[i]
    if (char.isupper()):
      res=res+chr((ord(char)+s-65)%26+65)
    else:
      res=res+chr((ord(char)+s-97)%26+97)
  return res
text="HEllo"
print("Input:",text)
print("\nAfter applying ceaser cipher:")
s=4
print("Output:",encrypt(text,s))
```

```
Input: HEllo

After applying ceaser cipher:
Output: LIpps
```

```
[5] #rail road transposition technique(row column transposition)
    s="hello world"
    s = s.replace(" ","")
    num_cols = 4
    num_rows = (len(s)+num_cols-1)//num_cols
    s+=' '*(num_cols*num_rows-len(s))
    columns= ['' for _ in range(num_cols)]
    for i in range(len(s)):
      col_index = i% num_cols
      columns[col_index] += s[i]

    encrypted_message = ''.join(columns)
    print("Input:")
    print("Original Text:",s)
    print("\nOutput:")
    print("Encrypted Message:", encrypted_message)
    print("Decrypted Text:",s)
```

```
Input:
Original Text: helloworld

Output:
Encrypted Message: holewdlo lr
Decrypted Text: helloworld
```

2. Implement Diffie-Hellman Algorithm to calculate key for Sender and Receiver.

```
2# Diffie-Hellman
# Power function to return value of a^b mod P
def power(a, b, p):
    if b == 1:
        return a
    else:
        return pow(a, b) % p

# Main function
def main():
    print("Input:")
    P = 23
    print("The value of P:", P)
    # A primitive root for P, G is taken
    G = 9
    print("The value of G:", G)
    # Alice chooses the private key a
    # a is the chosen private key
    a = 4
    print("The private key a for Alice:", a)
    # Gets the generated key
    x = power(G, a, P)
    # b is the chosen private key
    b = 3
    print("The private key b for Bob:", b)
    # Gets the generated key
    y = power(G, b, P)
    # Generating the secret key after the exchange of keys
    ka = power(y, a, P)  # Secret key for Alice
    kb = power(x, b, P)  # Secret key for Bob
    print("\nOutput:")
    print("Secret key for Alice is:", ka)
    print("Secret key for Bob is:", kb)
if __name__ == "__main__":
    main()
```

```
Input:
The value of P: 23
The value of G: 9
The private key a for Alice: 4
The private key b for Bob: 3

Output:
Secret key for Alice is: 9
Secret key for Bob is: 9
```

3. Implement following Brute Force Attack

- Dictionary Attack

```python
#Dictionary attack
def dictionary_attack():
    possible_passwords = ['123456', 'password', 'admin', 'letmein', 'welcome', 'qwerty']

    while True:
        target_password = input("Input:\nEnter the target password: ")

        for password in possible_passwords:
            if password == target_password:
                print("Final Output:\nPassword cracked successfully!")
                return

        print("Output:\nPassword not cracked. Try again.")

# Start the attack
dictionary_attack()
```

```
Input:
Enter the target password: hello
Output:
Password not cracked. Try again.
Input:
Enter the target password: admin
Final Output:
Password cracked successfully!
```

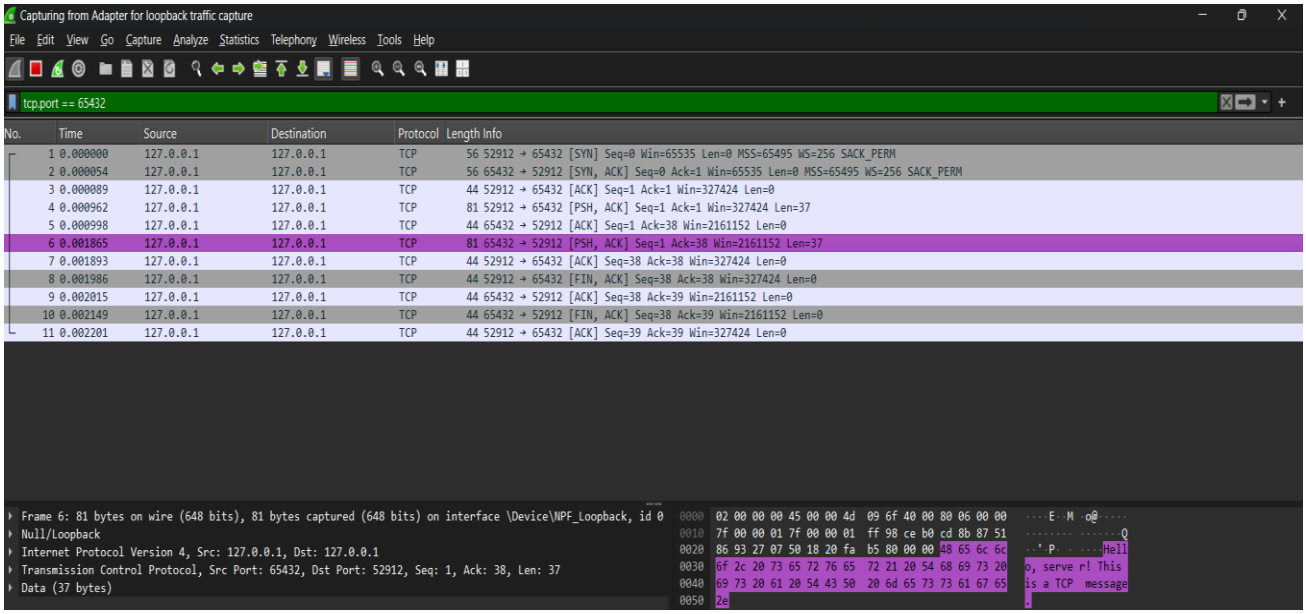
4. Demonstrate message exchange and data transmission between server and client to demonstrate TCP using Wireshark.

tcp_server.py

```
import socket
HOST = '127.0.0.1'  # Localhost
PORT = 65432        # Port to listen on
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
    server_socket.bind((HOST, PORT))
    server_socket.listen()
    print(f"Server started. Listening on {HOST}:{PORT}...")
    conn, addr = server_socket.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print(f"Received: {data.decode()}")
            conn.sendall(data)  # Echo the data back to the client
```

tcp_client.py

```
import socket
HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 65432        # The port used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
    client_socket.connect((HOST, PORT))
    print("Connected to the server.")
    client_socket.sendall(b'Hello, server! This is a TCP message.')
    data = client_socket.recv(1024)
print(f"Received back: {data.decode()}")
```



5. To build a simple python client server application to understand basic socket programming.

```python
#5 client server application for basic socket programming
import socket
import threading

# Server Code
def server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 12345))  # Bind to localhost and port 12345
    server_socket.listen(1)  # Listen for incoming connections
    print("Server is listening on port 12345...")

    conn, addr = server_socket.accept()  # Accept a connection
    print(f"Connection established with {addr}")

    data = conn.recv(1024).decode()  # Receive data
    print(f"Server received: {data}")

    conn.send("Hello from Server!".encode())  # Send a response
    conn.close()  # Close the connection
    server_socket.close()  # Close the server socket

# Client Code
def client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 12345))  # Connect to the server

    client_socket.send("Hello from Client!".encode())  # Send data
    response = client_socket.recv(1024).decode()  # Receive the response
    print(f"Client received: {response}")

    client_socket.close()  # Close the client socket

# Run server and client using threading
server_thread = threading.Thread(target=server)
client_thread = threading.Thread(target=client)

server_thread.start()
client_thread.start()

server_thread.join()
client_thread.join()
```

```
Server is listening on port 12345...
Connection established with ('127.0.0.1', 45960)
Server received: Hello from Client!
Client received: Hello from Server!
```

6. How to create web scraper in python to gather data from websites.

```python
#web scraper in python to gather data from websites
#importing requests (for fetching web pages) and BeautifulSoup (for parsing HTML).
import requests
from bs4 import BeautifulSoup

# Step 1: Define the URL to scrape
url = "https://www.bbc.com/news"

# Step 2: Fetch the web page content
response = requests.get(url)
if response.status_code == 200:
    print("Successfully fetched the webpage!")
else:
    print("Failed to fetch the webpage. Status code:", response.status_code)

# Step 3: Parse the HTML content
soup = BeautifulSoup(response.content, 'html.parser')

# Step 4: Extract headlines based on the correct structure
# On BBC News, headlines are often within <h3> tags with specific classes
headlines = soup.find_all('h3', class_='gs-c-promo-heading__title')

# Step 5: Print the extracted headlines
if headlines:
    print("Top Headlines:")
    for idx, headline in enumerate(headlines[:10], start=1):  # Limit to top 10 headlines
        print(f"{idx}. {headline.get_text(strip=True)}")
else:
    print("No headlines found. The HTML structure might have changed.")
```

```
Successfully fetched the webpage!
No headlines found. The HTML structure might have changed.
```

7. Write a python scripts for basic static malware analysis.

```python
#Python script for basic static malware analysis
# Step 1: Create the EICAR test file
eicar_string = "X5O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*"
with open("eicar.com", "w") as f:
    f.write(eicar_string)
print("EICAR test file created: eicar.com")

# Step 2: Install necessary library for PE file analysis
!pip install pefile

import os
import hashlib
import re
import pefile

# Step 3: File to Analyze
file_name = "eicar.com"  # Analyze the created EICAR test file

# Compute Hashes
def compute_hashes(file_path):
    hashes = {"MD5": None, "SHA1": None, "SHA256": None}
    with open(file_path, "rb") as f:
        data = f.read()
        hashes["MD5"] = hashlib.md5(data).hexdigest()
        hashes["SHA1"] = hashlib.sha1(data).hexdigest()
        hashes["SHA256"] = hashlib.sha256(data).hexdigest()
    return hashes

# Extract Strings
def extract_strings(file_path):
    with open(file_path, "rb") as f:
        data = f.read()
    # Find printable ASCII strings
    return re.findall(rb"[ -~]{4,}", data)
```

```python
# PE File Analysis (for Windows executables)
def analyze_pe(file_path):
    try:
        pe = pefile.PE(file_path)
        return {
            "Entry Point": hex(pe.OPTIONAL_HEADER.AddressOfEntryPoint),
            "Imported DLLs": [entry.dll.decode() for entry in pe.DIRECTORY_ENTRY_IMPORT],
        }
    except Exception as e:
        return {"Error": str(e)}

# Perform Analysis
print(f"\nAnalyzing file: {file_name}")
file_hashes = compute_hashes(file_name)
print("\nFile Hashes:")
for hash_type, hash_value in file_hashes.items():
    print(f"{hash_type}: {hash_value}")

print("\nExtracted Strings (Preview):")
strings = extract_strings(file_name)
print("\n".join([s.decode('utf-8', errors='ignore') for s in strings[:10]]))  # Show first 10 strings

if file_name.endswith(".exe"):
    print("\nPE File Analysis:")
    pe_analysis = analyze_pe(file_name)
    for key, value in pe_analysis.items():
        print(f"{key}: {value}")
else:
    print("\nPE File Analysis: Skipped (Not a Windows executable)")
```

```
EICAR test file created: eicar.com
Requirement already satisfied: pefile in /usr/local/lib/python3.10/dist-packages (2024.8.26)

Analyzing file: eicar.com

File Hashes:
MD5: 44d88612fea8a8f36de82e1278abb02f
SHA1: 3395856ce81f2b7382dee72602f798b642f14140
SHA256: 275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f

Extracted Strings (Preview):
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

PE File Analysis: Skipped (Not a Windows executable)
```