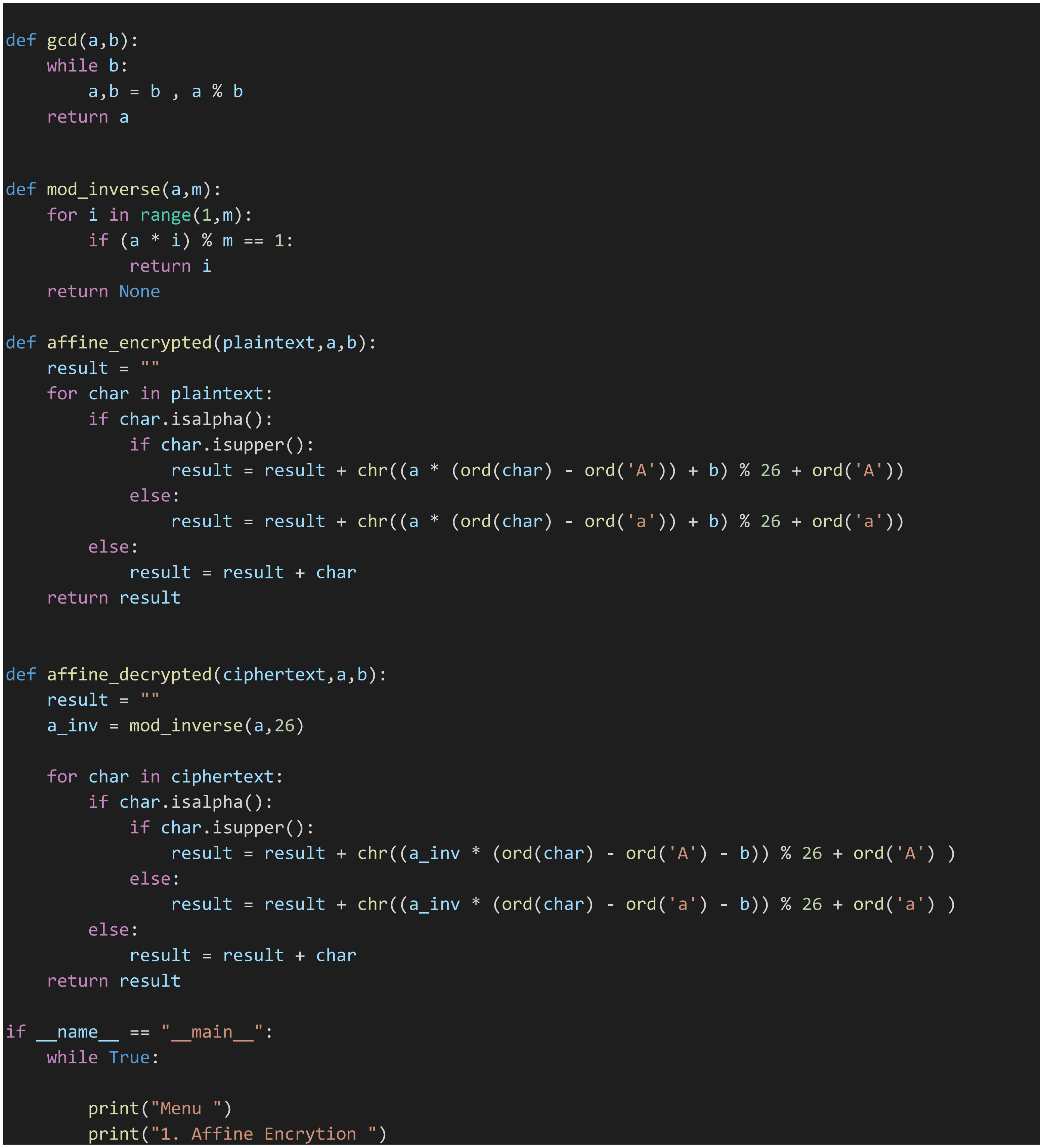
All codes:

Casear(Shift Cipher) – Multiplicative and Additive

```python
def additive_encrypt(message,key):
    result = ""
    for char in message:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            encrypted_char =chr((ord(char) - base + key) % 26 + base)
            result = result + encrypted_char
        else:
            result = result + char
    return result

def additive_decrypt(message,key):
    return additive_encrypt(message, -key)

def multiplicative_encrypt(message,key):
    result = ""
    for char in message:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            encrypted_char =chr(((ord(char) - base) * key) % 26 + base)
            result = result + encrypted_char
        else:
            result = result + char
    return result

def multiplicative_decrypt(message, key):

  inverse_key = None

  for i in range(1,26):
    if (i * key) % 26 == 1:
      inverse_key = i
      break

  if inverse_key is None:
    raise ValueError("Invalid key!")

  return multiplicative_encrypt(message, inverse_key)
```

```python
if __name__ == "__main__":
    while True:
        print("Menu")
        print("1. Additive Cipher Encryption")
        print("2. Additive Cipher Decryption")
        print("3. Multiplicative Cipher Encryption")
        print("4. Multiplicative Cipher Decryption")
        print("0. Exit")

        try:
            choice = int(input("Enter Your Choice: "))

            if choice == 1:
                message = input("Enter the Plaintext: ")
                key = int(input("Enter the number(an integer): "))
                print("The Encrypted text is: ",additive_encrypt(message, key))

            elif choice == 2:
                message = input("Enter the Ciphertext: ")
                key = int(input("Enter the number(an integer): "))
                print("The Decrypted text is: ",additive_decrypt(message, key))

            elif choice == 3:
                message = input("Enter the Plaintext: ")
                key = int(input("Enter the number(an integer): "))
                print("The Encrypted text is: ",multiplicative_encrypt(message, key))

            elif choice == 4:
                message = input("Enter the Ciphertext: ")
                key = int(input("Enter the number(an integer): "))
                print("The Decrypted text is: ",multiplicative_decrypt(message, key))

            elif choice == 0:
                print("exiting..")
                break

            else:
                print("Invalid Choice! Please choose the number from Menu.")
        except ValueError:
            print("Invalid Choice! Enter valid choice as number")
```

Affine Cipher:



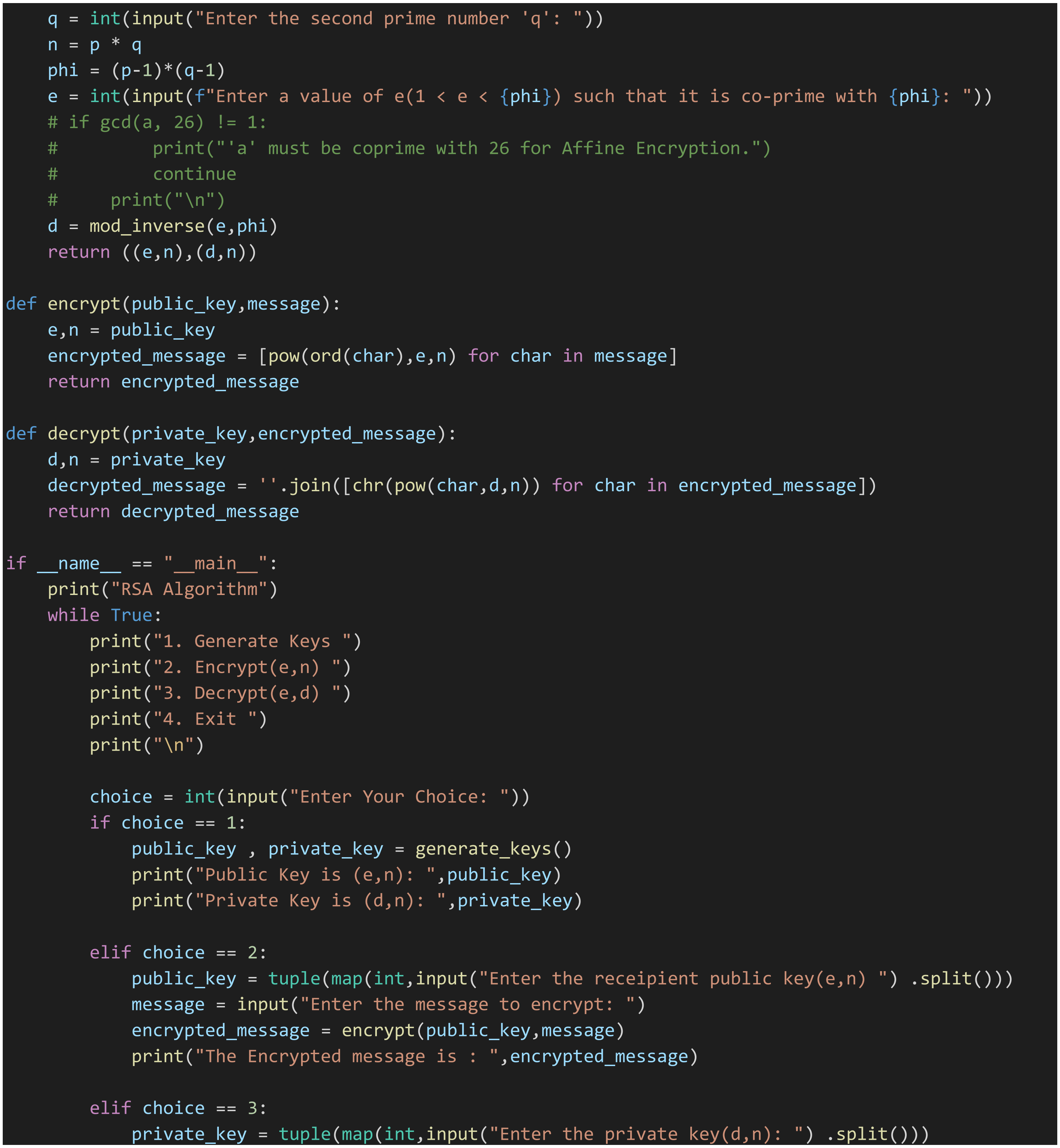
```python
def gcd(a,b):
    while b:
        a,b = b , a % b
    return a


def mod_inverse(a,m):
    for i in range(1,m):
        if (a * i) % m == 1:
            return i
    return None

def affine_encrypted(plaintext,a,b):
    result = ""
    for char in plaintext:
        if char.isalpha():
            if char.isupper():
                result = result + chr((a * (ord(char) - ord('A')) + b) % 26 + ord('A'))
            else:
                result = result + chr((a * (ord(char) - ord('a')) + b) % 26 + ord('a'))
        else:
            result = result + char
    return result


def affine_decrypted(ciphertext,a,b):
    result = ""
    a_inv = mod_inverse(a,26)

    for char in ciphertext:
        if char.isalpha():
            if char.isupper():
                result = result + chr((a_inv * (ord(char) - ord('A') - b)) % 26 + ord('A') )
            else:
                result = result + chr((a_inv * (ord(char) - ord('a') - b)) % 26 + ord('a') )
        else:
            result = result + char
    return result

if __name__ == "__main__":
    while True:

        print("Menu ")
        print("1. Affine Encrytion ")
        print("2. Affine Decrytion ")
        print("0. Exit ")

        try:
            choice = int(input("Enter Your Choice: "))

            if choice == 1:
                plaintext = input("Enter the Plaintext: ")
                a = int(input("Enter the first Number 'a' (co-prime with 26): "))
                b = int(input("Enter the second Number 'b': "))

                print("The Encypted text: " , affine_encrypted(plaintext,a,b))
                if gcd(a, 26) != 1:
                    print("'a' must be coprime with 26 for Affine Encryption.")
                    continue
                print("\n")

            elif choice == 2:
                ciphertext = input("Enter the Ciphertext: ")
                a = int(input("Enter the first Number 'a' (co-prime with 26): "))
                b = int(input("Enter the second Number 'b': "))

                print("The Decrypted text: " , affine_decrypted(ciphertext,a,b))
                print("\n")

            elif choice == 0:
                print("exiting..")
                break

            else:
                print("Invalid Choice!! Please enter valid number from Menu")

        except ValueError:
            print("Invalid Choice!! Please enter valid number not an alphabet!!")
```

Vignere Cipher

```python
def vignere_encrpyt(Plaintext,key):
    result = ""
    key = key.upper()
    key_index = 0

    for char in Plaintext:
        shift = ord(key[key_index]) - ord('A')
        if char.isalpha():
            if char.islower():
                result = result + chr((ord(char) + shift - ord('A')) % 26 + ord('A'))
            else:
                result = result + chr((ord(char) + shift - ord('a')) % 26 + ord('a'))
            key_index = (key_index + 1) % len(key)

        else:
            result = result + char

    return result

def vignere_decrpyt(Ciphertext,key):
    result = ""
    key = key.upper()
    key_index = 0

    for char in Ciphertext:
        shift = ord(key[key_index]) - ord('A')
        if char.isalpha():
            if char.islower():
                result = result + chr((ord(char) - shift - ord('A')) % 26 + ord('A'))
            else:
                result = result + chr((ord(char) - shift - ord('a')) % 26 + ord('a'))
            key_index = (key_index + 1) % len(key)
        else:
            result = result + char

    return result


if __name__ == "__main__":
    while True:
        print("Menu")
        print("1. Vignere Cipher Encryption")
        print("2. Vignere Cipher Decryption")
        print("0. Exit")

        try:
            choice = int(input("Enter Your Choice: "))

            if choice == 1:
                Plaintext = input("Enter the Plaintext: ")
                key = input("Enter the key: ")
                print("The Encrypted text is : " , vignere_encrpyt(Plaintext,key))
            elif choice == 2:
                Ciphertext = input("Enter the Ciphertext: ")
                key = input("Enter the key: ")
                print("The Decrypted text is : " , vignere_decrpyt(Ciphertext,key))
            elif choice == 0:
                print("exiting..")
                break

            else:
                print("Invalid Choice! Please choose the number from Menu.")
        except ValueError:
            print("Invalid Choice! Enter valid choice as number")
```

RSA(Rivest,Shamir,adlemen)

```python
def gcd(a,b):
    while b:
        a,b = b,a % b
    return a

def mod_inverse(e,phi):

    for d in range(2,phi):
        if (d * e) % phi == 1:
            return d
    return None

def generate_keys():
    p = int(input("Enter the first prime number 'p': "))
```



```python
    q = int(input("Enter the second prime number 'q': "))
    n = p * q
    phi = (p-1)*(q-1)
    e = int(input(f"Enter a value of e(1 < e < {phi}) such that it is co-prime with {phi}: "))
    # if gcd(a, 26) != 1:
    #         print("'a' must be coprime with 26 for Affine Encryption.")
    #         continue
    #     print("\n")
    d = mod_inverse(e,phi)
    return ((e,n),(d,n))

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

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

if __name__ == "__main__":
    print("RSA Algorithm")
    while True:
        print("1. Generate Keys ")
        print("2. Encrypt(e,n) ")
        print("3. Decrypt(e,d) ")
        print("4. Exit ")
        print("\n")

        choice = int(input("Enter Your Choice: "))
        if choice == 1:
            public_key , private_key = generate_keys()
            print("Public Key is (e,n): ",public_key)
            print("Private Key is (d,n): ",private_key)

        elif choice == 2:
            public_key = tuple(map(int,input("Enter the receipient public key(e,n) ") .split()))
            message = input("Enter the message to encrypt: ")
            encrypted_message = encrypt(public_key,message)
            print("The Encrypted message is : ",encrypted_message)

        elif choice == 3:
            private_key = tuple(map(int,input("Enter the private key(d,n): ") .split()))
            encrypted_message = list(map(int,input("Enter the encrypted message (space-separated integers): ") . split()))
            decrypted_message = decrypt(private_key,encrypted_message)
            print("The Decrypted message is : ",decrypted_message)
        elif choice == 4:
            print("exiting..")
            break
```

A5/1 Cipher:

```
x = input("Enter the value of Register X: ")
y = input("Enter the value of Register Y: ")
z = input("Enter the value of Register Z: ")

print("Initially")
print("Register X: ",x)
print("Register Y:",y)
print("Register Z: ",z)

keystream = []

for i in range(10):
    c = int(x[8]) + int(y[10]) + int(z[10])
    if c <= 1:
        m = 0
    else:
        m = 1

    if int(x[8]) == m:
        tx = int(x[13]) ^ int(x[16]) ^ int(x[17]) ^ int(x[18])
        x1 = x[0:18]
        x = str(tx) + x1

    if int(y[10]) == m:
        ty = int(y[20]) ^ int(y[21])
        y1 = y[0:21]
        y = str(ty) + y1

    if int(z[10]) == m:
        tz = int(z[7]) ^ int(z[20]) ^ int(z[21]) ^ int(z[22])
        z1 = z[0:22]
        z = str(tz) + z1


    print("x:",x)
    print("y:",y)
    print("z:",z)

    k = int(x[18]) ^ int(y[21]) ^ int(z[22])
    print("For Iteration: ",i+1,",keystream bit: ",k)
    print("\n")
    keystream.append(k)


print("After all iterations: ")
print("Register X: ",x)
print("Register Y: ",y)
print("Register Z: ",z)
print("keystream: ",keystream)


plaintext = input("Enter the plaintext(o or 1): ")

ctr = 0
key = ''

for i in keystream:
    key = key + str(i)
    ctr = ctr + 1
    if ctr == len(plaintext):
        break


ct = ''

for i,j in zip(key,plaintext):
    x = int(i) ^ int(j)
    ct = ct + str(x)

print("The ciphertext is : ",ct)
```

Diffie Hellmen

Server-side:

```python
import socket
import random

try:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        server_socket.bind(('127.0.0.1', 12344))
        server_socket.listen()
        print("Server Listening")

        client_socket, client_address = server_socket.accept()
        print("Connected to:", client_address)

        p_g = client_socket.recv(1024).decode().split()
        p, g = map(int, p_g)

        b = random.randint(1, p - 1)
        RB = pow(g, b, p)
        RA = client_socket.recv(1024).decode()

        str_RB = str(RB)
        client_socket.send(str_RB.encode())

        KAB = pow(int(RA), b, p)
        print("KAB:", KAB)

except Exception as e:
    print(f"Server error: {e}")
```

client side

```python
import socket
import random
import time

try:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
        client_socket.connect(('127.0.0.1', 12344))
        p = 23
        g = random.randint(2, p - 1)
        p_g = f'{p} {g}'
        client_socket.send(p_g.encode())

        a = random.randint(1, p - 1)
        RA = pow(g, a, p)
        str_RA = str(RA)
        time.sleep(3)

        client_socket.send(str_RA.encode())
        RB = client_socket.recv(1024).decode()

        KAB = pow(int(RB), a, p)
        print("KAB:", KAB)

except Exception as e:
    print(f"Client error: {e}")
```

Challenge Response:

Server-side

```
import random
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as bob_socket:
    bob_socket.bind(('127.0.0.1', 12345))
    bob_socket.listen()
    print('Server listening')

    alice, adr = bob_socket.accept()

    PSWD = int(alice.recv(1024).decode())
    RAND = random.randint(100, 999)
    alice.send(str(RAND).encode())

    RB = pow(PSWD, 1, RAND)

    RA = int(alice.recv(1024).decode())

    if str(RA) == str(RB):
        print("Authentication Successful")
```

client-side:

```python
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as alice_socket:
    alice_socket.connect(('127.0.0.1', 12345))

    PSWD = 100
    alice_socket.send(str(PSWD).encode())

    RAND = alice_socket.recv(1024).decode()
    remainder = pow(int(PSWD), 1, int(RAND))
    alice_socket.send(str(remainder).encode())
```