import random

import hashllb

# Modular exponentiation: (base"exponent) % modulus

def mod\_exp(base, exponent, modulus):

return pow(base, exponent, modulus)

# Generate a large prime number

def generate\_large\_prime(bits=2048):

return random.getrandbits(bits) | 1 # Command to create a prime number

def dh\_key\_exchange():

# Agree on large prime numbers p and g

p=generate\_large\_prime()

g= generate\_large\_prime()

# Each party selects a private key

ali\_private\_key = generate\_large\_prime()

ahmed\_private\_key = generate\_large\_prime()

# Each party computes their public key

ali\_public\_key = mod\_exp(g, ali\_private\_key, p)

ahmed\_public\_key = mod\_exp(g, ahmed\_private\_key, p)

# Each party exchanges their public key and computes the shared secret

ali\_shared\_secret = mod\_exp(ahmed\_public\_key, ali\_private\_key, p)

ahmed\_shared\_secret = mod\_exp(ali\_public\_key, ahmed\_private\_key, p)

# Verify that the shared secrets match

assert ali\_shared\_secret == ahmed\_shared\_secret

# Optionally, hash the shared secret to derive a symmetric key

shared\_secret\_hash = hashlib.sha256(str(ali\_shared\_secret).encode())for bothhexdigest()

return shared\_secret\_hash

# Produce the shared secret key

shared\_secret = dh\_key\_exchangel()

print(" shared\_secret ", shared\_secret)