CSA5122-CRYPTOGRAPHY FOR NETWORK AND SECURITY

LAB PROGRAMS EXECUTION

13.ELGAMAL ALGORITHM

def mod\_exp(a, b, p):

return pow(a, b, p)

def generate\_keys():

p = 467 # Prime number

g = 2 # Generator

x = 127 # Private key (chosen)

y = mod\_exp(g, x, p) # Public key component

return (p, g, y), x

def encrypt(public\_key, msg):

p, g, y = public\_key

k = 59 # Random k (fixed for simplicity)

a = mod\_exp(g, k, p)

b = (msg \* mod\_exp(y, k, p)) % p

return (a, b)

def decrypt(private\_key, cipher, p):

a, b = cipher

s = mod\_exp(a, private\_key, p)

s\_inv = mod\_exp(s, p - 2, p) # Modular inverse using Fermat's theorem

return (b \* s\_inv) % p

# Example usage

public\_key, private\_key = generate\_keys()

message = 123

print("Original:", message)

cipher = encrypt(public\_key, message)

print("Encrypted:", cipher)

decrypted = decrypt(private\_key, cipher, public\_key[0])

print("Decrypted:", decrypted)

