**IMPLEMENT ELGAMAL DIGITAL SIGNATURE SCHEME.**

import math

z = 16

class signAlgo:

def \_\_init\_\_(self, a, b, c, d):

self.p = a

self.alpha = int(b) # Convert 'alpha' to an integer

self.beta = int(pow(self.alpha, z) % self.p)

self.m = c

self.k = d

self.r = self.createR()

self.s = self.createS()

print("The Signed Message Triplet generated is: (" + str(self.m) + "," + str(self.r) + "," + str(self.s) + ")")

def gcd(self, a, b):

if a < b:

return gcd(self, b, a)

elif a % b == 0:

return b

else:

return gcd(b, a % b)

def invK(self):

kc = self.k

mc = self.p - 1

y = 0

x = 1

if mc == 1:

return 0

while kc > 1:

quotient = kc // mc

temp = mc

mc = kc % mc

kc = temp

temp = y

y = x - quotient \* y

x = temp

if x < 0:

x = x + self.p - 1

return x

def createR(self):

a = int((self.alpha \*\* self.k) % self.p)

return a

def createS(self):

a = (self.invK() \* (self.m - z \* self.r)) % (self.p - 1)

return a

class verify:

def \_\_init\_\_(self, a, b, c, d, e, f):

self.p = a

self.alpha = b

self.beta = c

self.m = d

self.r = e

self.s = f

def v1(self, b, c, d):

a = int(((b \*\* c) \* (c \*\* d)) % self.p)

return a

def v2(self, b, c):

a = int((b \*\* c) % self.p)

return a

def verified(self):

if self.v1(self.beta, self.r, self.s) == self.v2(self.alpha, self.m):

print("Signature verified using ElGamal.")

print("The value of v1 mod p: " + str(self.v1(self.beta, self.r, self.s)))

print("The value of v2 mod p: " + str(self.v2(self.alpha, self.m)))

else:

print("Signature mismatch")

print("The value of v1 mod p: " + str(self.v1(self.beta, self.r, self.s)))

print("The value of v2 mod p: " + str(self.v2(self.alpha, self.m)))

p = int(input("Enter the value of p : "))

alpha = int(input("Enter the value of alpha : "))

m = int(input("Enter the value of m : "))

k = int(input("Enter the value of k : "))

sign = signAlgo(p, alpha, m, k)

print()

print("Verification of Elgamal Signature")

v = verify(sign.p, sign.alpha, sign.beta, sign.m, sign.r, sign.s)

v.verified()

**OUTPUT**

