Question 2)

Write a Sage function that takes 3 successive outputs from a linear congruential RNG, as well as the modulus *m* of the internal state, that returns a and c OR indicates that it cannot find these values. Generate a linear congruential state, and 3 successive outputs and show your function working.

Question 2 – Solution:

Write a Sage function that takes 3 successive outputs from a linear congruential RNG, as well as the modulus *m* of the internal state, that returns a and c OR indicates that it cannot find these values. Generate a linear congruential state, and 3 successive outputs and show your function working.

The Sage function should behave something like:

def AttackLinearCongruential(X,m):

r"""

This is a function that takes 3 output states of a Linear Congruential RNG and m

and returns the a and c parameters, or indicates that the attack fails.

Takes a list of 3 subsequent outputs from the PRNG and the value of the modulus m.

"""

(X0,X1,X2) = (X[0], X[1], X[2])

# (1) Attempt to recover a by calculating X1 - X0 = a\*(X2 - X1) mod m

# let t = X1 - X0 so a = (X3-X2)\*t^-1 mod m

# errors out if t is uninvertible

t = X1 - X0

(gcd\_t\_m, t\_inv, m\_inv) = xgcd(t,m)

if (gcd\_t\_m != 1):

print "Could not invert X2 and X1 mod m."

return None

a = (X2 - X1)\*t\_inv % m

# (2) calculate c from X1 and X0

c = (X1 - a\*X0) % m;

return (a, c)

An Example of this function running:

sage: state = LinearCongruential\_Initialize(33, 47, 100, 1)

sage: LinearCongruential\_Generate(state)

80

sage: LinearCongruential\_Generate(state)

87

sage: LinearCongruential\_Generate(state)

18

sage: X = [80, 87, 18]

sage: m = 100

sage: AttackLinearCongruential(X, m)

(33, 47)