Question 3)

Previously we saw that factoring can be reduced to breaking Blum Blum Shub’s security. In this problem we will see the other direction, namely that Blum Blum Shub’s security can be broken by factoring. For this problem you may use the fact that finding a square root mod a prime is a solved problem. In fact if *p* ≡ 3 (mod 4) and *x* is a square mod *p* then the square root of *x* is given by *x*(*p*+1)/4 mod *p*. Either use this formula or the built in sage functionality to compute square roots in a prime field to write a function that takes a Blum Blum Shub internal state, and the two prime factors *p*, *q* (both 3 mod 4) of the modulus, and outputs a list of at most 4 possibilities for the previous Blum Blum Shub State. Generate a Blum Blum Shub state (with your own *p*,*q*) and show that your function works. [Hint: use the CRT.]

Question 3 - Solution

Previously we saw that factoring can be reduced to breaking Blum Blum Shub’s security. In this problem we will see the other direction, namely that Blum Blum Shub’s security can be broken by factoring. For this problem you may use the fact that finding a square root mod a prime is a solved problem. In fact if *p* ≡ 3 (mod 4) and *x* is a square mod *p* then the square root of *x* is given by *x*(*p*+1)/4 mod *p*. Either use this formula or the built in sage functionality to compute square roots in a prime field to write a function that takes a Blum Blum Shub internal state, and the two prime factors *p*, *q* (both 3 mod 4) of the modulus, and outputs a list of at most 4 possibilities for the previous Blum Blum Shub State. Generate a Blum Blum Shub state (with your own *p*,*q*) and show that your function works. [Hint: use the CRT.]

The Function to break BBS by factoring:

def BreakBBSWithFactors(X, p, q):

r"""

Given an output of a BlumBlumShub PRNG

and p,q, factorization of N (where p,q are both 3 mod 4):

Returns the previous state of the BlumBlumShub RNG.

"""

Fp = GF(p)

Fq = GF(q)

Xp = Fp(X)

Xq = Fq(X)

Xp\_sqrt = Xp^((p+1)/4) # or Xp\_sqrt = Xp.sqrt()

Xq\_sqrt = Xq^((q+1)/4) # or Xq\_sqrt = Xq.sqrt()

M = [p,q];

sqrt0 = [Xp\_sqrt.lift(), Xq\_sqrt.lift()]

sqrt1 = [-Xp\_sqrt.lift(), Xq\_sqrt.lift()]

sqrt2 = [Xp\_sqrt.lift(), -Xq\_sqrt.lift()]

sqrt3 = [-Xp\_sqrt.lift(), -Xq\_sqrt.lift()]

return [CRT\_list(sqrt0, M), CRT\_list(sqrt1, M), \

CRT\_list(sqrt2, M), CRT\_list(sqrt3, M)]

And an example of this code working is:

sage: state = [565184539 \* 1038996839, 12345678]

sage: BreakBBSWithFactors(state[1], 565184539, 1038996839)

[587224949460326543, 50025230168011697, 537199719304660524, 12345678]

We can see that the previous state 12345678 is indeed in this list.