**1. In Aug 31’s lecture (slide 5), we discussed a recursive multiplication algorithm. If the**

**input is an m-bit number x and an n-bit number y, how long does it take to multiply**

**x and y? Justify your answer.**

Usually, it will be O() or O() for smaller int values (assuming our processor and memory can contain both). However, as n and m get sufficiently large and we can’t hold both in the limited amount of registers on the processor then it will go to O(n\*m).

**2. Compute gcd(770, 546) in the following three different ways. Show your steps.**

**(a) By finding the factorization of each number;**

So, we can combine common prime factors to:

so gcd(770,546)=14

**(b) By using the Euclid algorithm;**

so gcd(770,546)=14

**(c) By using the extended Euclid algorithm (also finds x and y).**

Building off 2b):

so 14 = – 17\*770 + 24\*546 (x=-17, y=24)

**3. What is the result of 7^7293 (mod 342)? Show your steps.**

We will use modular multiplication properties to get the result of this large power:

1. Compute the powers of 7:

Noticing the pattern of alternating values 7 and 49, we can easily extrapolate:

Now given 7293, we convert it to binary so we can more easily compute our required value:

1110001111101

= 13841287201 mod 342 = 1

**4. Results:**

Testing for 1024 :

Key Generation took 0.00112414360046 seconds

Encryption took 0.000118017196655 seconds

Decryption took 3.81469726562e-06 seconds

\*\*\*\*

Testing for 4096 :

Key Generation took 0.0105400085449 seconds

Encryption took 0.000447034835815 seconds

Decryption took 4.05311584473e-06 seconds

\*\*\*\*

Testing for 8192 :

Key Generation took 0.0437829494476 seconds

Encryption took 0.00170278549194 seconds

Decryption took 1.50203704834e-05 seconds

\*\*\*\*

[Finished in 0.1s]