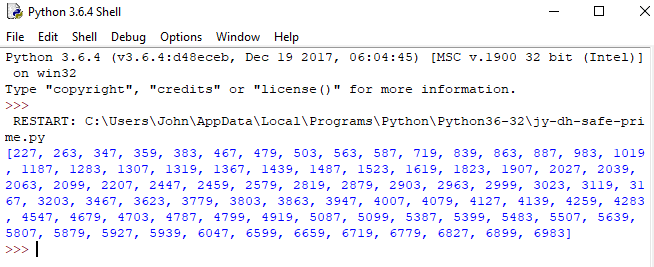
# Cryptography Homework 6

## Diffie-Hellman KeyExchange

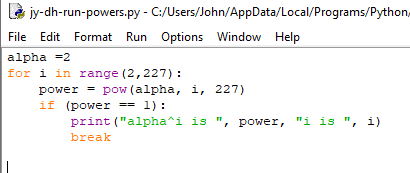
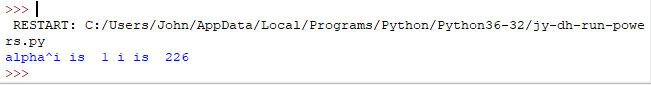
## Required Reading

Cryptology6 slides  
<https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange>

## Select p and α

The safe primes script at the end of this lesson gives the following output.  


The students can choose any number, so I’ll guess they choose the smallest one, 227. (227 -1)/2 is 113, which is also prime.

As an extra check (not required of the students, I computed the size of the subgroup by brute force.  
  


The subgroup with alpha = 2 has 225 elements, good.

## Select your private keys

This is a student choice, so it can be anything. Alice will choose 42, and Bob will choose 97.

## Compute and share your public keys

## 

Alice key is (227, 2, 213)

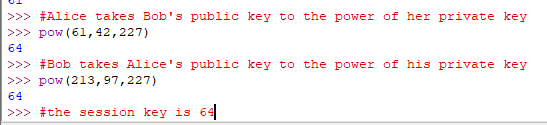
Bob key is (227, 2, 61)

## Compute the session key

Alice should take Bob’s public key, B, to the power of her private key, a.  
session key = Ba mod p = αab mod p

Bob should take Alice’s public key, A, to the power of her private key, b.  
session key = Ab mod p = αab mod p

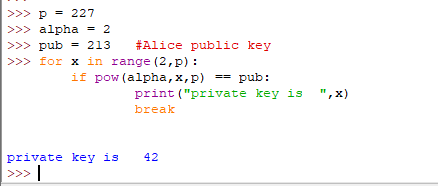
Compare your keys. They should be the same.



## Crack the private keys

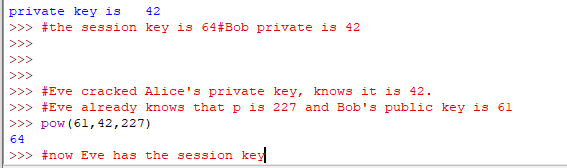
Since your prime number, p, is small, you should be able to solve the discrete logarithm problem by brute force. Here is some Python code that tries every value of x until it solves pub = alphax mod p. It is taken from the example for Alice’s key in the slides. Demonstrate that if Eve solves this problem she can derive the session key.

Eve knows p , alpha, Alice’s public key, and Bob’s public key (226, 2, 213, 61). She chooses to break Alice’s public key.



Alice’s private key is found to be 42.

Now, Eve can take Bob’s public key (61) to the 42 power just like Alice did, and she has the session key.



Note: this worked because we are using very small numbers. If we were using huge numbers, the brute force attack would not have worked.

## Hand In

Fill the blanks in the form below for the DHKE you did with your neighbor. Also hand in a screenshot of your Python script breaking the key.

p \_\_\_\_227\_\_\_\_\_\_\_

α \_\_\_\_2\_\_\_\_\_\_\_

**Alice**

Select a \_\_\_42\_\_\_\_\_\_\_\_

Compute A \_\_\_213\_\_\_\_\_\_\_\_ A = αa mod p

Give A to Bob

**Bob**

Select b \_\_\_97\_\_\_\_\_\_\_\_

Compute B \_\_\_61\_\_\_\_\_\_\_\_ B = αb mod p

Give B to Alice

**Alice computes key** = Ba mod p (she picked a, Bob gave her B ) \_\_\_\_64\_\_\_\_\_\_\_

**Bob computes key** = Ab mod p (he picked b, Alice gave him A) \_\_\_\_\_64\_\_\_\_\_\_

Now that Alice and Bob have the same key, they could transfer data with a symmetric algorithm like AES.

Attach a screenshot of you breaking the private key.

## Appendix: Python script to pick a prime, p, where (p-1)/2 is also prime.

#This requires pycryptodome to be installed, but we've been using it for a while

from Crypto.Util.number import isPrime

# generate a list of prime numbers

primes = []

for num in range(200, 7000):

  if isPrime(num):

    primes.append(num)

print('There are {0} primes between 200 and 7000\n'.format(len(primes)))

#check to see if (p-1)/2 is a prime so p is "safe"

safe = []

for p in primes:

    if isPrime( int((p-1)/2) ):

        safe.append(p)

print('There are {0} safe prime numbers between 200 and 7000\n'.format(len(safe)))

print(safe)