Cryptography Homework 6

Diffie-Hellman Key Exchange

# Required Reading

Cryptology6 slides  
This is a good, non-mathematical description of Diffie-Hellman. <https://www.youtube.com/watch?v=NmM9HA2MQGI>  
Here is the same person describing the math.  
<https://www.youtube.com/watch?v=Yjrfm_oRO0w>

Here is the Wikipedia explanation <https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange>

Do this lab in groups of two (or three if there is an odd number of students.) You can also do this by yourself, you just have to be both Alice and Bob…

This lab leads you through the process of a Diffie-Hellman Key Exchange (DHKE). If you get confused, look at the form you are filling out on page 2. It can give you an overview.

# Select p and α

Working together select a prime number, p, where q = (p-1)/2 is also prime. You can use a list of the first 1000 prime numbers from <https://primes.utm.edu/lists/small/1000.txt>. You should be able to pick a prime greater than 200 that meets the criteria within a few attempts. (**Or**, you can write a short Python script, see the end of this document.)

Because you have chosen a “safe” prime number, α can be almost anything you wish. It cannot be 1, or p-1, because they form their own subgroup with 1 and p-1 as the only members. The other subgroup, which you will use, contains all numbers in {2 … q-1}, where q = (p-1)/2. Usually α is set to a small number, but any number in the group should work.

Record your choices for p and α.

# Select your private keys

Each person should select their own private exponent (if there are three in the group, two select one exponent while one selects another exponent.) The exponent (a or b, depending on who is Alice and who is Bob) should be in the range 2 < a < q-1.

# Compute and share your public keys

Each person should compute their public key, A = αa mod p, or B = αb mod p. Share your key with the other person in the group. Note: the entire key consists of (p, α, and A) or (p, α, and B)

# Compute the session key

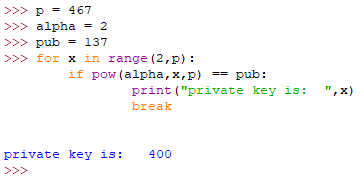
Alice should take Bob’s public key, B, to the power of her private key, a.  
session key = Ba mod p = αab mod p

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.



# Hand In

1) Fill the blanks in the form below for the DHKE you did with your neighbor. Remember, to compute  
αa mod p  
use  
pow(α, a, p).

p \_\_\_\_\_\_\_\_\_\_\_

α \_\_\_\_\_\_\_\_\_\_\_

**Alice**

Select a \_\_\_\_\_\_\_\_\_\_\_ (Private)

Compute A \_\_\_\_\_\_\_\_\_\_\_ A = αa mod p (Public)

Give A to Bob

**Bob**

Select b \_\_\_\_\_\_\_\_\_\_\_ (Private)

Compute B \_\_\_\_\_\_\_\_\_\_\_ B = αb mod p (Public)

Give B to Alice

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

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

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

2) Attach a screenshot of you breaking the private key. (Hint: use a Python for loop that tests all values of a in αa mod p until you find the one that equals A. Finding b that gives you B in αb mod p works too. Example code is just above, in the Cracking Public Keys paragraph.

Now that you have a (or b if you decided to crack B), you can compute the shared key.  
key = pow(B, a, p) (if you cracked A to get a) or,   
key = pow(A, b, p) (if you cracked B to get b)  
What is the shared key?

3) If p = 19, what is a good choice for α? (Hint: Modify the script gen\_p\_17\_with\_break.py from Canvas. Find a value of alpha that creates a large subgroup.)

4) Use the script order\_calculate.py to examine these choices for p and α. Which is best? Which is worst?  
a) p = 449, α = 324

b) p = 337, α = 302

c) p = 479, α = 163

# Appendix: Scripts

## safe-primes.py

#Uses the isPrime function from PyCryptodome

from Crypto.Util.number import isPrime

# check numbers starting at 99 to see if they are prime

# only odd numbers can be prime, so use range(99, 6999, 2)

# stopping at 7000 will get about 900 primes

primes = [x for x in range(99, 7000, 2) if isPrime(x)]

print('There are {0} primes between 99 and 6999'.format(len(primes)))

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

safe = []

unsafe = 0

for p in primes:

if isPrime( int((p-1)/2) ):

safe.append(p)

else:

unsafe += 1

print('There are {0} "unsafe" primes'.format(unsafe))

print('Some safe primes are:')

print(safe)

## order\_calculate.py

#This calculates the order (number of elements)

#that each value of alpha generates

from Crypto.Util.number import isPrime

p = 337

order = []

for alpha in range(1,p):

powers = []

for i in range(1,p):

x = pow(alpha, i, p)

if x == 1:

print(alpha, i)

if i not in order:

order.append(i)

break

order.sort()

print(('Possible order sizes are: {0}').format(order))

if not isPrime(p):

print("Whoops!! p is not prime!")

if isPrime((p-1)//2):

print(('p = {0} is a safe prime').format(p))

else:

print(('p = {0} is NOT a safe prime').format(p))

## gen\_p\_17\_with\_break.py

# this calculates the subgroup that each

# possible value of alpha generates

# It breaks when the subgroup begins to

# repeat to make the subgroup size more

# obvious

p = 17

for alpha in range(1,p):

powers = []

for i in range(1,p):

x = pow(alpha, i, p)

if x == 1:

powers.append(x)

break

else:

powers.append(x)

print(alpha, powers)