Kaitlyn Peterson

Diffie-Hellman

First, I (Eve) interpreted the numbers that I intercepted from Bob and Alice in their Diffie Hellman Key Exchange. I know that Alice and Bob chose random values *a* and *b* respectively and used them to calculate *A* and *B* using the below formulas:

$A = g^a \bmod p$ (equation 1) $B = g^b \bmod p$ (equation 2)

Substituting in the values that I intercepted (g=11, p=59, A=57, and B=44) sent between Alice and Bob, I got:

$57 = 11^a \bmod 59$ (equation 1) $44 = 11^b \bmod 59$ (equation 2)

Further, I know that their secret shared key (*K*) will be defined by the following formula:

$K = B^a \bmod p = A^b \bmod p$
$K = 44^a \bmod 59 = 57^b \bmod 59$ (equation 3)

First, I will discern the value of a using equation 1. Using the following piece of python code, I tested all possible integer values of *a*, beginning with 1, until I found a value of *a* that satisfies this equation.

```
a = 1
remainder = 1
while (remainder != 57):
  power = 11**a
  remainder = power % 59
  if (remainder == 57):
    print(a)
  a = a + 1
```

I found the value a = 36. Next, I repeated this process on equation 2 to solve for b.

```
b = 1
remainder = 1
while (remainder != 44):
  power = 11**b
  remainder = power % 59
  if (remainder == 44):
```

```
    print(b)
  b = b + 1
```

Here I found that b = 15. Finally, I tested that equation 3 holds true; that both Alice and Bob will calculate a shared *K* using the values of *a* and *b*.

$K = B^a \bmod p = A^b \bmod p$

$K = 44^{36} \bmod 59 = 57^{15} \bmod 59$

$K = 36 = 36$

Thus, the secret key is 36. Here, with such small integers chosen, I (Eve) only had to check 36 possible values of *a* and 15 of *b*. However, if Bob and Alice had chosen larger integers, I (Eve) would have trouble with completing my calculations in a reasonable amount of time. The brute-force method I chose to implement to discover *a* and *b* would soon become incredibly inefficient, as I have to check every possible value of *a* and *b.* If the integers became large enough, this algorithm would take an unrealistic amount of time and effort to actually run;.

I was also lucky that the first *a* and *b* values I found were equivalent. It is possible that with different original integers, I would have had to continue searching for subsequent possible values of *a* and *b* to find two that were equal.

(Note: I could have chosen to test every possible *K,* rather than *a* and *b,* but this would have been similarly inefficient.)

RSA

As I know Bob's public key is (e_{Bob}, n_{Bob}) = (13, 5561), I can compute the possible values of *p* and *q* that satisfy *pq* = *n*. I wrote the following piece of python code to check if a number is prime and to print all possible prime factorizations that comprised two integers. This gives me the set of all possible values of *p* and *q*.

```
def isPrime(k):
  for i in range(2, int(k/2)+1):
    if (k % i) == 0:
      return False
  return True

for p in range(5562):
  if((p!=0) and isPrime(p)):
    q = 5561/p
    if ((q%1==0) and isPrime(q)):
      print("p:", p, "q:", q)
```

After running this piece of code, I knew that the only possible values of *p* and *q* are:

p: 67 q: 83
p: 83 q: 67

The next step in the RSA algorithm is to select an *e* value; however I had already intercepted e_{Bob} = 13. This value works with all restraints on *e* and the other values I had found so far:

e<(p-1)(q-1)	gcd(e, (p-1)(q-1)) = 1
13<5412	gcd(13, 5412) = 1

Next, to decrypt the message, I will need to calculate *d*, which is Bob's private key. Bob would use his private key to read all messages sent to him, since they are encrypted using his public key. As RSA uses the following equation to choose a *d* value, I wrote the following piece of code to find the first suitable *d* value that satisfies this equation.

ed mod (p-1)(q-1) → d = (mod5412)/13

```
i=1
while (True):
  d = (1 + (5412*i)) / 13
  if (d%1==0):
    print ("d:", d)
    break
  i = i + 1
```

From this, I found that d = 1249 satisfies the equation above. I used this key to decrypt each integer in the message sent from Alice to Bob. I used the following equation on each integer:
$DecryptedM = M^{1249} \bmod 5561$

To facilitate this process, I wrote the following piece of code:

```
decryptedMessage = []
for m in encryptedMessage:
  d = (m**1249)%5561
  decryptedMessage.append(d)
print(decryptedMessage)
```

After decrypting the entire message I had the following integers:
[72, 101, 121, 32, 66, 111, 98, 46, 32, 73, 116, 39, 115, 32, 101, 118, 101, 110, 32, 119, 111, 114, 115, 101, 32, 116, 104, 97, 110, 32, 119, 101, 32, 116, 104, 111, 117, 103, 104, 116, 33, 32, 89, 111, 117, 114, 32, 112, 97, 108, 44, 32, 65, 108, 105, 99, 101, 46, 32, 104, 116, 116, 112, 115, 58, 47, 47, 119, 119, 119, 46, 115, 99, 104, 110, 101, 105, 101, 114, 46, 99, 111, 109, 47, 98, 108, 111, 103, 47, 97, 114, 99, 104, 105, 118, 101, 115, 47, 50, 48, 50, 50, 47, 48, 52, 47, 97, 105, 114, 116, 97, 103, 115, 45, 97, 114, 101, 45, 117, 115, 101, 100, 45, 102, 111, 114, 45, 115, 116, 97, 108, 107, 105, 110, 103, 45, 102, 97, 114, 45, 109, 111, 114, 101, 45, 116, 104, 97, 110, 45, 112, 114, 101, 118, 105, 111, 117, 115, 108, 121, 45, 114, 101, 112, 111, 114, 116, 101, 100, 46, 104, 116, 109, 108]

This is a message encoded using ASCII characters. Thus, I decoded this message using an ASCII chart on each 2 or 3 digit integer. I found the following message, which includes a link to a concerning article about using Airtags as a method of stalking:
Hey Bob. It's even worse than we thought! Your pal, Alice.
https://www.schneier.com/blog/archives/2022/04/airtags-are-used-for-stalking-far-more-than-previously-reported.html

While I was able to discern Alice and Bob's message fairly quickly, this would not have worked had Alice and Bob chosen larger integers. If they had, the portion of my process where I found the prime factors of *n* would have taken much longer (as it is a nested for-loop), and it also could have resulted in multiple options for *p* and *q*.

Even though Bob and Alice used RSA, their method of encoding is still insecure because they encoded each character one at a time. By doing this, they mimic substitution ciphers, which can often be easily decoded because one can discern letter frequencies and compare findings to a dictionary. So, next time, Bob and Alice should really choose a different encoding method where each character is not encoded separately, and maybe then I won't intercept their message.