Aiden Chang, Eli Arbogast

CS231

Being Eve

Diffie Hellman:

g = 17 and p = 61. Alice sends Bob 46 and Bob sends Alice 5. To find the secret number k, we must obtain one of the secret numbers Alice and Bob has produced(a,b respectively). We know that 46 = 17^a mod 61 where a < 61. Using the function secretFinderDH() from the attached python code(being-eve.py), we find that a is 14. To computer k, B^a mod 61, where B is the number Bob sent to Alice. 5^14mod 61 = 12. 12 is the shared secret.

I would have failed to find Alice's secret a. My program is a brute program that guesses and checks the squares of 17 until 17^i mod 61 = 46. This is an inefficient linear time algorithm that can take years to complete if the integers involved were bigger.

RSA:

e_B = 31

p_B > 31

q_B > 31

q_B * p_B = 4661 = n_B

<u>To obtain p_B, q_B, and d_B:</u>

q and p are bigger than 31 and multiply to 4661. The ones place digits multiply to 1. Since the prime numbers over 31 only have the ones place at 1,3,9, or 7, the options of multiplying are 1*1, 3*7,9*9. Plug and check to see if the two prime numbers you picked multiply to 4661. Plugging and checking comes out to be 79 and 59.

Since e_B*d_Bmod(p_B-1)(q_B-1) = 1,

31d_B mod 4524 = 1.

Using secretFinderRSA() from the python function attached(being-eve.py) we find that d_B = 2335.

We only obtained p_B and q_B using this method because n_B was small. If large prime numbers were used for p_B and q_B, even writing a program will take a very long time to find p and q. Especially in my case where I plugged in and guess and checked.

Now we can find the secret message.

Secret message:

For each number y in the list, compute y^d_B mod n_B. Computing this using the function useKeyRSA() from the attached python function, the secret message is

“Dear Bob, Check this out.

https://www.schneier.com/blog/archives/2017/12/e-mail_tracking_1.html Yikes! Your friend, Alice”