

David Toledo

9/29/24

Diffie Hellman

- The shared secret between Alice and Bob is 65.

First, I wrote out what I knew. G was 7 and $P = 97$, and using what we saw from the lab, I set up the equations needed to find out the integers picked by each one. We know that $53 = 7^x \% 97$, and $82 = 7^y \% 97$. I wrote this code to solve this.

```
for x in range(1,98):
    if (7**x)%97 == 53:
        print("x is ", x)

for y in range(1,98):
    if (7**y)%97 == 82:
        print("y is ", y)
```

This gave me the numbers needed to get the A and B Alice and Bob sent each other, $X = 22$ and $Y = 41$. From here, we can solve for what the secrets where.

```
a = (82**22)%97
print("a is ", a)

b = (53**41)%97
print("b is ", b)
```

- Show precisely where in your process you would have failed if the integers involved were much larger.

It gets hard to pass this when the numbers become really big, as it becomes hard to look for a number efficiently. They get too large enough to check one by one for modding, making it take forever to find the answer.

$$\begin{array}{l} g=7 \\ p=97 \\ \\ X=22 \leftarrow \text{Python Code} \quad A = 7^x \% 97 = 53 \\ Y=41 \leftarrow \quad B = 7^y \% 97 = 82 \\ \\ \text{Secret} = 65 \leftarrow \text{Python Code} \quad A \rightarrow 82^{22} \% 97 \\ \quad \quad \quad \quad B \rightarrow 53^{41} \% 97 \end{array}$$

RSA

Alice's Message:

Dear Bob, check this out. <https://www.surveillancewatch.io/> See ya, Alice.

Work: First I looked up the factors of 162991. From there I saw that there were 4 factors, 1, 389, 419, and 162991. This meant that p_B and q_B were 389 and 419 respectively. To compute $\lambda(n_B)$, I looked up the least common multiple calculator and inputted 388 and 418, which resulted in 81092. From here, we were told from the public key that e was 13. From there I looked a greatest common factor calculator and confirmed that the GCF was 1. From here, I wrote a Python script to calculate $13 \cdot d_B \% 81092 = 1$

```
for dB in range(1,100000):  
    if 13*dB%81092 == 1:  
        print("dB is ",dB)
```

From here, I was told that d_B is 43665. Now I have all that I need to decrypt the message. I created this to apply my secret key to all of the numbers.

```
#Decrypting message  
  
message = [17645, 100861, 96754, 160977, 120780, 90338, 130962, 74096, 128123, 25052,  
119569, 39404, 6697, 82550, 126667, 151824, 80067, 75272, 72641, 43884, 5579, 29857,  
33449, 46274, 59283, 109287, 22623, 84902, 6161, 109039, 75094, 56614, 13649, 120780,  
133707, 66992, 128221]  
  
chars = []  
for num in message:  
    new = (num**43665) % 162991  
    chars.append(new)  
  
hexChar = []  
for num in chars:  
    new = hex(num)  
    hexChar.append(new)  
print(hexChar)
```

This gave me a new list, but the numbers were too big to be ASCII. So I did what I was told in class, converted them to hex, and then put those hex numbers into the Hex to ASCII [converter](#). This displayed the message I was looking for.

The way Alice encoded her message was by first turning the text into ASCII, and then taking those ASCII values and turning them into hexadecimal. From that, every 4 hexadecimal number was converted back into decimal. From there, Alice applied her key to the numbers created, and sent this as a message.

Hex to ASCII Text String Converter

Enter hex bytes with any prefix / postfix / delimiter and press the Convert button
(e.g. 45 78 61 6d 70 6c 65 21):

From

To

Hexadecimal

Text

Open File

Q

Paste hex numbers or drop file

4465 6172 2042 6f62 2c20 6368 6563 6b20 7468 6973 206f 7574 2e20 6874 7470 733a 2f2f 7777 772e 7375 7276 6569 6c6c 616e 6365 7761 7463 682e 696f 2f20 5365 6520 7961 2c20 416c 6963 652e

Character encoding

ASCII

Convert

Reset

Swap

Dear Bob, check this out.
<https://www.surveillancewatch.io/> See ya, Alice.

Bob:

Pick two prime numbers p_B and q_B and compute $162991 = p_B q_B$. For this exercise, make sure that $n_B > 128$.

$P_B = 389$, $q_B = 419$

Compute $\lambda(n_B) = 81092$ ---- $\text{lcm}(388, 418)$

Pick $1 < 13 < \lambda(n_B)$ such that $\text{gcd}(13, 81092) = 1$.

Find an integer $d_B = 43665$ such that $e_B d_B \bmod \lambda(n_B) = 1$. $13 * d_B \% 81092 = 1$

Share (e_B, n_B) with Alice. This ordered pair is your *RSA public key*.

Keep (d_B, n_B) secret. This ordered pair is your *RSA private key*.

$p_B = 389$, $q_B = 419$, $\lambda(n_B) = 81092$, $n_B = 162991$, $d_B = 43665$

Decrypt Alices message: $y^{43665 \% 162991}$

[17645, 100861, 96754, 160977, 120780, 90338, 130962, 74096, 128123, 25052, 119569, 39404, 6697, 82550, 126667, 151824, 80067, 75272, 72641, 43884, 5579, 29857, 33449, 46274, 59283, 109287, 22623, 84902, 6161, 109039, 75094, 56614, 13649, 120780, 133707, 66992, 128221]

[17509, 24946, 8258, 28514, 11296, 25448, 25955, 27424, 29800, 26995, 8303, 30068, 11808, 26740, 29808, 29498, 12079, 30583, 30510, 29557, 29302, 25961, 27756, 24942, 25445, 30561, 29795, 26670, 26991, 12064, 21349, 25888, 31073, 11296, 16748, 26979, 25902]

[4465 6172 2042 6f62 2c20 6368 6563 6b20 7468 6973 206f 7574 2e20 6874 7470 733a 2f2f 7777 772e 7375 7276 6569 6c6c 616e 6365 7761 7463 682e 696f 2f20 5365 6520 7961 2c20 416c 6963 652e]

This process would've failed in the same aspect because the numbers would become too inefficient to check individually. Looking for d_B , I had to take up the range until 100000, as the numbers were getting really big. If they got bigger, the numbers would get bigger and bigger, and it would take longer and longer to check for this number.

Alice's message would be insecure even if Bob's keys involved larger integers because there are still possible attacks that would be intercepted, such as the man-in-the-middle attacks where the eavesdropper still would know enough information to meddle with the conversation. Also, with enough time people could still probably break the code by checking the numbers individually if they were still a little bigger. Her encoding is very easy to break with modern tools, and so this encoding adds more of an inconvenience to an eavesdropper more than a security layer. Her message still is vulnerable regardless of the size of the key.