Lane Maitland

All code at end.

Diffie-Hellman

We are given the following:

- g = 11
- p = 59
- A = 57
- B = 44

Let:

- X: private integer of Alice
- Y: private integer of Bob
- K: shared key

We know that the following relationships hold:

- $0 < X < 59$
- $0 < Y < 59$
- $57 \equiv 11^X \mod 59$
- $44 \equiv 11^Y \mod 59$
- K $\equiv 44^X \mod 59 \equiv 57^Y \mod 59 \equiv 11^{XY} \mod 59$

Find K:

- I did this by looping over integers 0<i<59 (all possible values for X,Y) and checking if:
    - $57 \equiv 11^X \mod 59$
    - $44 \equiv 11^Y \mod 59$
- I found that:
    - X = 36
    - Y = 15
- I checked that:
    - $44^X \mod 59 \equiv 57^Y \mod 59 \equiv 11^{XY} \mod 59$
- This was true, so:
    - K = $36 \equiv 11^{XY} \mod 59$
- The shared secret key is the integer 36.

Comments:

- This method would not be practical if p was much larger because the loop could require many iterations and result in a long run-time. Even with a while-loop (skipping the appropriate calculation after X or Y has been found, terminating when X and Y are both found), it could be very inefficient.
- Without X or Y, we would not be able to calculate the secret key. Having both helps confirm that Alice and Bob have calculated the same shared key, but it seems that there

was only one possibility for X and Y. (I checked this by initializing X and Y as lists and appending values in the for-loop, but printing just returned lists of one value.)

Sources:
- https://www.math.ucla.edu/~baker/40/handouts/rev_DH/node1.html

RSA

We are given the following:
- public key of Bob: $(e_B, n_B) = (13,\ 5561)$
- encrypted data

Let:
- (d_B, n_B): private key of Bob
- p_B, q_B: prime numbers of Bob
- $\lambda(n_B) = lcm(p_B - 1,\ q_B - 1)$

We know that the following relationships hold:
- $n_B = p_B q_B$
- $1 < e_B < \lambda(n_B)$
- $gcd(e_B,\ \lambda(n_B)) = 1$
- $e_B d_B \, mod\, \lambda(n_B) \equiv 1$

Find d_B:
- Observe that the factors of 5561 are: 1, 67, 83, 5561
- Since 1 is not prime, we have that:
    - p_B, q_B= 67, 83
- So:
    - $\lambda(n_B)$ = 2706
- We can confirm that:
    - $1 < e_B = 13 < \lambda(n_B) = 2706$
    - $gcd(e_B,\ \lambda(n_B)) = gcd(13, 2706) = 1$
- I created a while-loop that found the smallest d_B such that:

  $e_B d_B \, mod\, \lambda(n_B) \equiv 13\, d_B \, mod\, 2706\ \equiv 1$
- I found that:
    - d_B= 1249

Decrypt:

- I created a for-loop that applied $(d_B, n_B) = (1249,\ 5561)$ to each integer in the encrypted data.
  - Applying the key is computing $y^{d_B}\ mod\ n_B \equiv\ y^{1249}\ mod\ 5561$ for y in the encrypted data.
- For each resulting integer, I used the chr() function to convert it to a character.
- I added each character to a string and got the message:
  - 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

Comments:

- This method would not be practical if n_B had more factors. This would mean that there are different possibilities for p_B, q_B.
- This method would also not be practical if d_B was large, as that would require many iterations of a while-loop.
- It is also possible that d_B is not the smallest d_B such that $e_B d_B\ mod\ \lambda(n_B)\ \equiv\ 1$. However, I do not know if this would necessarily change the result of the decryption.
- The encrypted message was formed by Alice finding the character code and applying the public key of Bob (e_B, n_B) to each character in the message. This means that Alice computed $x^{e_B}\ mod\ n_B$ for character code x.
- It is insecure to encrypt each character separately. Doing so essentially creates a substitution cipher, and people can figure out how the numbers and characters correspond without needing a decryption key (by observing the frequencies of numbers and their placement around other numbers).

Sources:

- https://www.integers.co/questions-answers/is-5561-a-prime-number.html
- https://www.calculator.net/lcm-calculator.html?numberinputs=66%2C82&x=64&y=26
- https://madformath.com/calculators/basic-math/factors-prime-numbers/relatively-prime-coprime-checker/relatively-prime-coprime-checker
- https://www.geeksforgeeks.org/python-ways-to-convert-list-of-ascii-value-to-string/

Code:

Users > lane-maitland > Desktop > CS 338 >

```python
# Diffie–Hellman

X = 0
Y = 0

for i in range(0,59):
    if (11 ** i % 59 == 57):
        X = i
    if (11 ** i % 59 == 44):
        Y = i

print("X: ", X)
print("Y: ", Y)

K_1 = 44 ** X % 59
K_2 = 57 ** Y % 59
K_3 = 11 ** (X * Y) % 59

if (K_1 == K_2 == K_3):
    print("key: ", K_3)
```

Users > lane-maitland > Desktop > CS 338 > cryptography.py > ...

```python
# RSA

found = False
d = 0
while (found == False):
    d += 1
    if (13 * d % 2706 == 1):
        found = True

print("d: ", d)

encrypted = [1516, 3860, 2891, 570, 3483, 4022, 3437, 299,
 570, 843, 3433, 5450, 653, 570, 3860, 482,
 3860, 4851, 570, 2187, 4022, 3075, 653, 3860,
 570, 3433, 1511, 2442, 4851, 570, 2187, 3860,
 570, 3433, 1511, 4022, 3411, 5139, 1511, 3433,
 4180, 570, 4169, 4022, 3411, 3075, 570, 3000,
 2442, 2458, 4759, 570, 2863, 2458, 3455, 1106,
 3860, 299, 570, 1511, 3433, 3433, 3000, 653,
 3269, 4951, 4951, 2187, 2187, 2187, 299, 653,
 1106, 1511, 4851, 3860, 3455, 3860, 3075, 299,
 1106, 4022, 3194, 4951, 3437, 2458, 4022, 5139,
 4951, 2442, 3075, 1106, 1511, 3455, 482, 3860,
 653, 4951, 2875, 3668, 2875, 2875, 4951, 3668,
 4063, 4951, 2442, 3455, 3075, 3433, 2442, 5139,
 653, 5077, 2442, 3075, 3860, 5077, 3411, 653,
 3860, 1165, 5077, 2713, 4022, 3075, 5077, 653,
 3433, 2442, 2458, 3409, 3455, 4851, 5139, 5077,
 2713, 2442, 3075, 5077, 3194, 4022, 3075, 3860,
 5077, 3433, 1511, 2442, 4851, 5077, 3000, 3075,
 3860, 482, 3455, 4022, 3411, 653, 2458, 2891,
 5077, 3075, 3860, 3000, 4022, 3075, 3433, 3860,
 1165, 299, 1511, 3433, 3194, 2458]

message = ""

for val in encrypted:
    char_code = val ** d % 5561
    message = message + chr(char_code)

print("message: ", message)
```