Ruben Boero

YOUR JOB FOR DIFFIE HELLMAN
Figure out the shared secret agreed upon by Alice and Bob. This will be an integer.
    The secret message is 6.

Show your work. Exactly how did you figure out the shared secret?
    On paper scan

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

    Computing the value of x and y in g^x mod p and g^y mod p respectively gets slow if g and p
    are large integers. It's also possible that there can be multiple values of x and y. This
    would take more time to figure out the secret message (g^(xy) mod p).

YOUR JOB FOR RSA
Figure out the encrypted message sent from Alice to Bob.

    Hi Bob. I'm walking from now on. Your pal, Alice.
    https://foundation.mozilla.org/en/privacynotincluded/articles/its-official-cars-are-the-worst-product-category-we-have-ever-reviewed-for-privacy/

Show your work. Exactly how did you figure out the message? (You should include an explanation
of how the message from Alice to Bob is encoded. That is, how does Alice's intended message
(whatever manner of message it may be) correspond to the integers in the plaintext that you
end up with after decrypting the encrypted message?)

    To begin, I found p and q using brute force (lines 44-47). By definition, p * q = n. n is
    given in Bob's public key, so I brute forced the values of p and q.

    Once I found p and q, I could brute force the value of d (Bob's private key). By definition
    ed = 1 mod (p - 1)(q - 1). e is in the public key, and I solved for p and q. I then used
    brute force to solve for d using the upper bound of n (lines 53-56).

    Now knowing d, I can decode Alice's message using the decryption function:

$m(c) = c^d \bmod n$

Found on wikipedia:

https://en.wikipedia.org/wiki/RSA_(cryptosystem)

This gives an integer. Turning this integer into binary reveals 2 ascii characters smushed back-to-back. Converting the ascii into English letters reveals the message.

Alice encrypted her message by taking 2 characters at a time, concatenating their binary ascii values together, then taking that binary number as an integer. That integer is then encoded using Alice's public key as per RSA.

$c(m) = m^e \bmod n$

Show precisely where in your process you would have failed if the integers involved were large (e.g., on the order of a 2000-bit value for n).

Brute forcing for d would have failed (taken way too much time) if n were large (lines 53-56).

If n were large, it would also have taken much longer to find p and q (lines 43-47).

Explain, briefly, why the message encoding Alice used would be insecure even if Bob's keys involved very large integers.

Alice's message would have been insecure because the 2 character strings that Alice encoded can be matched to English. From there, it's possible to figure out the message without knowing the encoding scheme.

$g=7, \ P=61$

DH

Ruben Bauer

$A \xrightarrow{30} B$

$\xleftarrow[17]{}$

$A \xrightarrow{g^x \bmod p = 30} B$

$\xleftarrow{g^y \bmod p = 17}$

x=Alice

y=Bob

$7^x \bmod 61 = 30$

$7^y \bmod 61 = 17$

$\downarrow$

$\boxed{x = 41, \ y = 23}$ (in PY)

shared secret = $g^{yx} \bmod p = 7^{23 \cdot 41} \bmod 61 = \boxed{6}$

where would my process fail if the ints were larger?

computing $g^x \bmod p$ and $g^y \bmod p$ gets very slow if g and p are large integers

- there can also be multiple answers for x and y. This would take more time to figure out the secret message (bc the secret message uses both x and y)

RSA

bobs public key

(n_bob), e_bob) = (170171, 17)

n_B = modulus = 170171

e_B = encryption exponent = 17

lcm of p and q

$p_B < n_B$, $q_B < n_B$

$p_B = 449$
$q_B = 379$
found in py code

$m(c) = c^d \mod n$

18537 → 2 ascii chars

1 0 0 1 0 0 0 | 0 1 1 0 1 0 0 1 (01)

H i

8 bits 8 bits