Fall 2023 CIS 3362 Homework #6: Public Key Encryption Solution
Check WebCourses for the due date

1) (10 pts) In the Diffie-Hellman Key Exchange, let the public keys be p = 79, g = 48, and the secret keys be a = 36 and b = 67, where a is Alice's secret key and b is Bob's secret key. What value does Alice send Bob? What value does Bob send Alice? What is the secret key they share? Use a program or calculator to quickly simplify the modular exponentiations that arise, but show what each calculation is.

Solution
Alice sends Bob 48^{36} mod 79. Using IDLE, pow(48, 36, 79) returns **10.**
Bob sends Alice 48^{67} mod 79. Using IDLE, pow(48, 67, 79) returns **75.**

Secret key = 10^{67} mod 79 or 75^{36} mod 79. Using IDLE, pow(10, 67, 79) returns **21** as does pow(75, 36, 79). This is the shared secret key.

2) (10 pts) In an RSA scheme, p = 47, q = 31 and e = 119. What is d? Show the work by hand, but for any complicated calculation, do it on a calculator or use a program. (So, show each step of the Extended Euclidean Algorithm, but feel free to use a calculator to quickly get quotients and remainders.)

Solution
n = pq = 47 x 31 = 1457, $\phi(n) = 46 \times 30 = 1380$

d = 119^{-1} mod 1380

1380 = 11 x 119 + 71
119 = 1 x 71 + 48
71 = 1 x 48 + 23
48 = 2 x 23 + 2
23 = 11 x 2 + 1

Take the last equation mod 1380

– 661 x 119 ≡ 1 (mod 1380)

it follows that d ≡ -661 ≡ **719** (mod 1380)

23 – 11 x 2 = 1
23 – 11 (48 – 2 x 23) = 1
23 – 11 x 48 + 22 x 23 = 1
23 x 23 – 11 x 48 = 1
23(71 – 48) – 11 x 48 = 1
23 x 71 – 23 x 48 – 11 x 48 = 1
23 x 71 – 34 x 48 = 1
23 x 71 – 34(119 – 71) = 1
23 x 71 – 34 x 119 + 34 x 71 = 1
57 x 71 – 34 x 119 = 1
57(1380 – 11 x 119) – 34 x 119 = 1
57 x 1380 – 627 x 119 – 34 x 119 = 1
57 x 1380 – 661 x 119 = 1

3) (40 pts) Alice and Bob both have their own RSA keys and have each sent a message to one another.

Here are Bob's Public Keys:

n = 475161435661642486778197616828073715814370607782529939019003664671198940828699604112043342909781631 9
e = 468589539452594594860263059045025103082428920 5

Here are Alice's Public Keys:

n = 502737807461227868384937706150119816811276509909221158894867100605414626276392654385763416519950455 1
e = 573233382278991937699015343101133403743093 09

You have intercepted two ciphertexts, one sent from Alice to Bob (using Bob's public key):

C = 104628457346696238773947394480738069161256512192752127852328241130543733762646756582533515306559171 1

And another one sent from Bob to Alice (using Alice's public key):

C = 332552387396355070498443772890437051983067840417292136208958713649030463848765062695653126080078016 4

Normally, you would be out of luck in cracking these messages, but Alice and Bob have poorly constructed their RSA keys. Determine, in text, what each of the ciphertexts decrypt to. (Note: a modified version of RSA2BigInt.java was used to create this problem. As such, this code should explain how to convert an integer plaintext to uppercase letters.)

Solution

Use Python or Java to deal with big integers. In this case, the weakness is that the two values of n share a common factor. The attached code in Java (**h6q3.java**) finds this common factor, thus factoring both Alice's and Bob's values of n. This can be then used to $\phi(n)$ for both of them. Finally, both secret values d can be determined by computing the appropriate modular inverse. Exponentiate accordingly to retrieve the corresponding plaintext messages, which are integers. To convert these messages back to letters, use the base 26 convention with 'a' = 0, ;b; = 1, etc. All of this work is included in **h6q3.java**. The output produced by this program is shown below:

Message from Alice to Bob, with punctuation/capitalization added and padding removed:

Bob, it's Alice. I am very interested in your offer. Let's meet at two am at the basement.

Message from Bob to Alice, with punctuation/capitalization added and padding removed:

Excellent! I look forward to it. Due to RSA, none will know about our collaboration.

Note: Looks like I had a typo generating the second message. I had meant "no one" instead of "none".

4) (40 pts) The following ciphertext below was created with the El Gamal cryptosystem with the following public elements:

q = 208827064597 (prime)
g = 148730885957 (primitive root)
Ya = 22100313146 (Alice's public exponent)

You also know that the plaintext was written in all lowercase letters and split into blocks of 8 characters and the value of each 8 character block is simply equal to its base 26 equivalent, treating A = 0, B = 1, …, Z = 25.

Use this information to decrypt the following ciphertext: (Note: This will also be given to you in a text file, for ease of processing and as mentioned, each line represents the encryption of one block of 8 characters.)

```
111537273770 135478365325
135874735585 91980775319
114008480727 189673465227
51859330405 168411804286
147104771103 105548487980
1785731042 1829176754
177464475649 64097517903
31371229910 160373635219
155078142943 88685005036
92163822772 83821102277
114922855196 109868919775
139345072195 37637914660
180092093087 72241715608
54004792799 102543868605
180382625017 101039780617
52988886207 106664942086
196270659674 1848943199
35121652348 107823603240
158294360726 126547219134
84120560054 82845406333
106816610946 160475344872
109119061605 199945687440
135206784648 11466088841
104712202700 173240884045
126757086234 179687780957
```

Solution

We must solve the discrete log problem. Namely, we must determine the value of a such that

$148730885957^a \equiv 22100313146 \pmod{208827064597}$.

While it's feasible to run a straight brute force to calculate this, it's possible this may take a couple hours, depending on the power of one's computer. (An estimate given by a student in class was 12 hours. When I ran a portion of the slow code on my desktop in my office, I approximated a search time of about 2 hours.)

Thus, to save some time, we can write code that implements the algorithm shown in class that finds the discrete log using roughly sqrt(q) operations, where q is the prime number used for the modulus. For this solution, we'll use the general idea with a modification. Since q is $< 10^{12}$, we'll calculate $148730885957^{1000000}$, $148730885957^{2000000}$, etc. until we exceed q (so this list will have 210,000 values on it.) We'll store these results in a HashMap, mapping each result to its corresponding exponent.

Then, we'll want to check if $148730885957^{1000000x}148730885957^{y} = 22100313146 \pmod{208827064597}$.

If so, then it follows that a = 1000000x + y. Checking this would require access to modular inverse, so an easier (but roughly equivalent technique is to find values x and y such that:

$$\frac{148730885957^{1000000x}}{148730885957^{y}} \equiv 22100313146 \pmod{208827064597}$$

Which would insinuate that

$$22100313146 \times 148730885957^{y} \equiv 148730885957^{1000000x} \ (mod\ 208827064597)$$

Thus, the strategy is as follows:

1) Create the HashMap described above, mapping each exponent to the corresponding modular exponentiation result.

2) For each possible value of y (up to one million), calculate $22100313146 \times 148730885957^{y}$ under mod 208827064597, and see if the result is stored in the HashMap.

The code that does this is in **h6q4p1.java**. This reveals that the corresponding value of a is:

a = 161672808561

Now, in the second program, **h6q4p2.java**, we'll read in the ciphertext from standard input, and decrypt it as follows:

1. For each block, calculate C_1^a mod q. This gives us the value of K.
2. Calculate K^{-1} mod q.
3. Multiply K^{-1} by C^2 mod q. This will be the plaintext block. Convert to text the same way as we converted the number from question 3. (In my code, I just hardcode 8 because I figured out that was the blocksize.)

Here is the output produced by the program, when the input given was piped in (note: I added 25 in my input to designate that there were 25 lines of input.)

```
didyouus
ethesqua
rerootal
gorithmt
obreakdi
scretelo
geitherw
aytogett
heprizeg
otonearr
oomtwofi
vefourin
hecthere
lookfors
omething
toputout
afireope
nthedoor
andslowl
yliftthe
extingui
sherthem
oneywill
beundern
eathitxx
```

Formatting nicely, we get:

Did you use the square root algorithm to break discrete log? Either way, to get the prize go to near room two five four in HEC. There, look for something to put out a fire. Open the door and slowly lift the extinguisher. The money will be underneath it.