Fall 2025 CIS 3362 Homework #6: Public Key Encryption Solutions

1) (10 pts) In the Diffie-Hellman Key Exchange, let the public keys be p = 97, g = 23, and the secret keys be a = 83 and b = 65, 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 to Bob $g^a \bmod p = 23^{83} \bmod 97$. By hand here is a table with some work:

exponent	$23^{exp} \bmod 97$
1	23
2	23 x 23 = 529 ≡ 44 (mod 97)
4	44 x 44 = 1936 ≡ (-4) (mod 97)
8	(-4)(-4) = 16 (mod 97)
16	16 x 16 = 256 ≡ 62 (mod 97)
32	62 x 62 = 3844 ≡ 61 (mod 97)
64	61 x 61 = 3721≡ 35 (mod 97)

$23^{83} = 23^{64} \times 23^{16} \times 23^{2} \times 23^{1} \equiv 35 \times 62 \times 44 \times 23 \equiv 2170 \times 1012 \equiv 36 \times 42 \equiv 1512 \equiv 57 \bmod 97$

Bob sends to Alice $g^b \bmod p = 23^{65} \bmod 97 \equiv (35*23) \equiv 805 \equiv 29 \bmod 97$

Shared Secret Key = $57^{65} \bmod 97$, as calculated by Bob,
Here is a similar chart for this exponentiation:

exponent	$57^{exp} \bmod 97$
1	57
2	57 x 57 = 3249 ≡ 48 (mod 97)
4	48 x 48 = 2304 ≡ 73 (mod 97)
8	73 x 73 = 5329 ≡ 91 (mod 97)
16	91 x 91 = 8281 ≡ 36 (mod 97)
32	36 x 36 = 1296 ≡ 35 (mod 97)
64	35 x 35 = 1225≡ 61 (mod 97)

$57^{65} = 57^{64} \times 57^{1} \equiv 61 \times 57 \equiv 3477 \equiv 82 \pmod{97}$

Thus, Alice sends Bob $23^{83} \equiv 57 \bmod 97$
Bob sends Alice $23^{65} \equiv 29 \bmod 97$
Their shared key is $57^{65} \equiv 82 \bmod 97$

Note: Full credit will be given if you just show the values for each computation and the answers they give (when put into IDLE).

2) (10 pts) In an RSA scheme, p = 41, q = 17 and e = 147. 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

$n = pq = 41 \times 17 = 697$

$\varphi(n) = (41 - 1)(17 - 1) = 40 \times 16 = 640$

We must calculate $d = e^{-1} \bmod \varphi(n)$. Let's calculate $147^{-1} \bmod 640$

$640 = 4 \times 147 + 52$
$147 = 2 \times 52 + 43$
$52 = 1 \times 43 + 9$
$43 = 4 \times 9 + 7$
$9 = 1 \times 7 + 2$
$7 = 3 \times 2 + 1$

$7 - 3 \times 2 = 1$
$7 - 3(9 - 7) = 1$
$7 - 3 \times 9 + 3 \times 7 = 1$
$4 \times 7 - 3 \times 9 = 1$
$4(43 - 4 \times 9) - 3 \times 9 = 1$
$4 \times 43 - 16 \times 9 - 3 \times 9 = 1$
$4 \times 43 - 19 \times 9 = 1$
$4 \times 43 - 19(52 - 43) = 1$
$4 \times 43 - 19 \times 52 + 19 \times 43 = 1$
$23 \times 43 - 19 \times 52 = 1$
$23(147 - 2 \times 52) - 19 \times 52 = 1$
$23 \times 147 - 46 \times 52 - 19 \times 52 = 1$
$23 \times 147 - 65 \times 52 = 1$
$23 \times 147 - 65(640 - 4 \times 147) = 1$
$23 \times 147 - 65 \times 640 + 260 \times 147 = 1$
$283 \times 147 - 65 \times 640 = 1$

Take this equation mod 640 to get

$283 \times 147 \equiv 1 \pmod{640}$, it follows that $147^{-1} \equiv 283 \pmod{640}$.

Thus, d = 283.

3) (50 pts) The following message was encrypted via RSA encryption. The public keys are as follows:

n = 153535199189155511054009430409868302266002806975797 7
e = 730215628455429163275499063151379643457490494895 59

Each integer in the ciphertext corresponds to a plaintext of 28 characters. In your write up, describe in detail what steps you took and which code (if you used any of the posted code) you used, or how you adapted it. Turn in attachments of any original code you wrote or anything that had non-trivial adaptations of the code posted for the class. Here is the ciphertext:

```
74209613410074397133739562490961655593632344033729 7
10759875070311261566551052167597394824929977674703 82
37136178901187016506391578582062362009111877953861 0
16527039554841755410947604318288947923580200640535 6
23788029530914035386955677082663380642981449721503 1
34452111653937533859814646710843688867699154630088 7
47059433656921674929477576688864353985428489141812 9
12828010520733511993604930575344540972828800762460 52
34048644521542217089640587131226549853498074777866 1
```

Solution

First, n must be factored. One could use Pollard-Rho, to do this, OR, one could recognize that an Easter Egg was provided for students. Namely that the prime number from question 3 divides evenly into n. In particular,

153535199189155511054009430409868302266002806975797 7%1234567890133 = 0

and

153535199189155511054009430409868302266002806975797 7//1234567890133

= 124363512461526246156156256154612542366 9

Thus,

124363512461526246156156256154612542366 9*1234567890133 =

153535199189155511054009430409868302266002806975797 7

This was all calculated in IDLE.

Now, continue to use Python to find phi(n):

153535199189031147541547904163712146009724737644417 6

Next, we must find $d = e^{-1} \bmod \varphi(n)$.

Use the Python code provided from class in the file:

https://www.cs.ucf.edu/~dmarino/ucf/cis3362/progs/rsa1.py

Here is the relevant code:

```
# Returns a list storing [x, y, gcd(a,b)] where ax + by = gcd(a,b).
def EEA(a,b):

    # End of algorithm, 1*a + 0*b = a
    if b == 0:
        return [1,0,a]

    # Recursive case.
    else:

        # Next quotient and remainder.
        q = a//b
        r = a%b

        # Algorithm runs on b, r.
        rec = EEA(b,r)

        # Here is how we put the solution back together!
        return [rec[1], rec[0]-q*rec[1], rec[2]]

# Returns the modular inverse of x mod n.
# Returns 0 if there is no modular inverse.
def modInv(x,n):

    # Call the Extended Euclidean.
    arr = EEA(n, x)

    # Indicates that there is no solution.
    if arr[2] != 1:
        return 0

    # Do the wrap around, if necessary.
    if arr[1] < 0:
        arr[1] += n

    # This is the modular inverse.
    return arr[1]
```

After loading these functions in IDLE, we run:

modInv(730215628455429163275499063151379643457490494895 59,
153535199189031147541547904163712146009724737644417 6)

which gives the answer:

88824721110976527557701484219281392827883401809370 3.

This is the value of d to use.

Some students may have tried to decrypt using this value of d assuming that the numbers converted to letters as shown in RSABigInt2.java, but this isn't the case. If you move onto question #4 and do that one, it tells you that the method of encoding is Radix-64, so the number that you receive has to be broken into separate 6-bit blocks. Using this knowledge, it's easier to make progress.

From here, I wrote a program to read in the ciphertext, use the value of d provided above to decrypt to a numerical plain text, then converted that to Radix-64 characters by breaking the number into blocks of 6 bits (just repeated division and mod by 64) and printing the corresponding characters. **The code I used is in the file h6_q3_decrypt.py.**

When I ran this program, piping in the ciphertext as input, I got the following output:

```
WithRadix+64/Icanusesomemore
charactersinmymessage/Thethe
meforHomework6isrecycling/As
such/youcanfindanotefrommeun
derneaththerecyclebinbytheAI
ThingsLab/It+sHECofcourseroo
m315intheyear2025/JUSTRYINGT
OUSEDIFFERENTCHARACTERSINTHE
CODEGoodluck/xxxxxxxxxxxxxxx
```

Cleaning it up we get:

With Radix-64, I can use some more characters in my message. The theme for Homework 6 is recycliing. As such, you can find a note from me underneath the recycle bin by the AI Things Lab. It's HEC of course, room 315 in the year 2025. JUST TRYING TO USE DIFFERENT CHARACTERS IN THE CODE. Good luck!

Several students had trouble finding the note. Congrats to Landon Craft for finally finding it. I have no idea how many students broke the code but couldn't find the note before Landon found it. It turns out that there are two recycling bins by Room 315 in HEC but some students gave up after not finding the note under the first bin. That lab should get an award for promoting recycling!!!

4) (30 pts) The following ciphertext below was created with the El Gamal cryptosystem with the following public elements:

q = 1234567890133 (prime)
g = 726288716355 (primitive root)
Ya = 268931642640 (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.)

```
299335298747 1172223606113
889598930003 557439201934
493347324365 308563064395
89330316409 284568396280
1011600488984 284687286118
78301339268 445083507637
184420679506 945520081269
113325085374 330613800716
801254124539 2492123523
189525982163 429945730554
884843344510 169440640565
518889437424 634600793624
2248173165 323732777084
575687437463 568722795104
891109133954 753816562536
1124597795350 970569441767
608617739348 63349455224
771764120123 778305272817
181728415187 585124549217
492606809251 383617280481
458984069941 197980928842
1082309772979 255983715074
775217121216 504345290258
1114180621864 18900995437
139742778128 205760018212
1098177806924 464716156995
159086739222 63106975262
251212528964 496548036082
1207655912195 764037801281
544740425341 289356820759
1073938227507 329586495737
989432039687 1210124857313
1190570442012 1222786515110
```

Solution

We must first solve the discrete log problem to find Alice's secret exponent. The key is to use the divde and conquer algorithm shown in class. A brute force search, while possible, is likely to take quite a while. This is the code from class with the algorithm:

https://www.cs.ucf.edu/courses/cis3362/fall2025/writtennotes/25-10-17/disclogdivconq.py

Just change the function call in this code to:

print(disclogfaster(726288716355, 268931642640, 1234567890133))

and this gives the answer that Alice's secret key, $X_A = 794703949047$.

From this point, a program was written (h6_q4_solve.py) to use this value to decrypt the given message. A tiny edit was made copying the ciphertext to a file and adding the first line with an integer representing the number of blocks of encrypted text.

I stored my plaintext in the file q4-mycipher.txt. The program output is stored in q4-myplain.txt. Here it is, nicely formatted:

My intention is for you to break this before number tres. Number three uses radix sixty four encoding. Also, to get your prize for this question, go near hec two four eight. Look for a blue recycle bin. There will be a note taped underneath it. Bring it to me for your prize. Yes you will have to tilt the whole container to get the note.

Congrats to Rohan Sangani for claiming the prize!