# Homework #4

Due date: **11 December 2023**

**Notes**:

- Note that there are five attached files: "RSA_Oracle_client.py" for Question 1, "RSA_OAEP.py" for Question 2, "ElGamal.py" for Questions 3 & 4 and "DSA.py" for Question 5.
- Print out your numerical results in integer format, without "-e". (We do not want to see results like 1.2312312341324523e+24).
- Winzip your programs and add a readme.txt document (**if necessary**) to explain the programs and how to use them.
- Name your **Winzip** file as "cs411_507_hw04_yourname.zip"
- Create a PDF document explaining your solutions briefly (a couple of sentences/equations for each question). Also, include your numerical answers (numbers that you are expected to find). Explanations must match source files. Please also add the same explanations as comments and explanatory output.

**1.** (**20 pts**) Consider a deterministic RSA Oracle that is implemented at the server "http://harpoon1.sabanciuniv.edu:9999". Connect to the server using the *RSA_Oracle_Get()* function, and it will send a ciphertext ***"C"*** **,** modulus ***"N"***, and public key **"e".**

- You are expected to find out the corresponding plaintext "$m$". You can query the RSA Oracle with any ciphertext $\underline{c} \neq c$ using the Python function *RSA_Oracle_Query()*, and it will send the corresponding plaintext $\underline{m}$. You can send as many queries as you want as long as $\underline{C} \neq C$.
- You should decode your answer into a Unicode string and check it using *RSA_Oracle_Checker().*
- You can use the Python code RSA_Oracle_client.py to communicate with the server.

**Important Note:** You have to find a mathematical way to find the message "m". Once you find it, code it then check your answer. Querying the server blindly won't get you the right answer.

For this question we are not allowed to use RSA_Oracle_Query() function with the given ciphertext. To decipher the message, we can pick an arbitrary random number and cipher it using the public key e and N. After that, we can multiply this new number with the given ciphertext.

Let's say

m = 65536.

m_c = m^e mod N

C_ = C*m_c mod N

We can decipher C_ using RSA_Oracle_Query() since C != C_ and get the message M_. After getting M_, we can calculate the inverse of m and use it to get rid of the m.

m_inv = m^-1 mod N
m_inv_pow = m_inv^e mod N
M = M_ * m_inv_pow mod N

The above calculation will give us the message m.

**M = Bravo! You found it. Your secret code is 32645**
In the digit form =
156053138494688764261663969923255791456060807758745999627667906341098899958264465653186776286254402261747119157

2. (**20 pts**) Consider the RSA OAEP implementation given in the file "RSA_OAEP.py", in which the random number R is an 8-bit unsigned integer. I used the following parameters for encryption:

   **ciphertext** (**c**) =
   155633174361451963459660128709513554675182231102646675371810749734360653 50566
   **public key** (**e**) = 65537
   **modulus** (**N**) =
   734200328912369016950504476555008613438247136051418228668850896212051316 80183

   I selected a random four-decimal digit PIN and encrypted it using RSA. Your mission is to find the randomly chosen PIN.

   Four decimal digits is not too big for brute force attack.
   Also, the randomness for each number is not too big (2^8).
   We can conduct a brute force attack by trying all possible combinations of the four digits with all possible randomness. We can encrypt the four digits with all possible randomness and compare the result with the given c. If the result matches, than we find the four digits.

   Four decimal digits is between 1000 to 9999. So, there are 9000 different values. Which is approximately 2^13. If we include the randomness, we need to try 2^21 different pairs which is not too big for a brute force attack.

   **The four digits are: 1308**
   **The random number is: 206**

3. (**20 pts**) Consider the ElGamal encryption algorithm implemented in the file "ElGamal.py", which contains a flaw. We used this implementation to encrypt a message using the following parameters:

   **q** = 20229678282835322453606583744403220194075962461239010550087309021811
   **p** =
   125150439093948037534504110226498542737215372510111077488268111684596806283513911544870413205950067362393321924922369439665230537444761277 2879

79638081511425065953301206216633715182811812047978317073494365584431393
55672347825267728879376289677517268609959671235059224994785463608330669
49445716325037358138003624765203096948104677201379927126871010448702216
48650048028640760669741530121255510609060541129204698690452233295770159
35824864428612446723942040465300185917923305042033306319809712618872063
79690413278828551849799932748592973092120274593593691383457761025429880
9205575162005025170878200786590751850006857921419
**g** =
22564831437414331634130076750679345428930229683374373122833819649423443
65449719628255630752397325376452002398784394008507857025386943645437696
55824087447134544253239858840674990793000248162416095913219379884242682
21939101049621388458734255909463417543341442928860029629015501605784824
52138075339294826241799645761655320983735381974177635207208471824667516
95667991397464334215955003732037881444580229687947056150451168946091620
04179026123230396712505675038461759906545129158781432012330509780462695
51126178155060158781645062181955781969136435905570787457855530003987887
049118699525033120811790739590564684316550493132
**public key (h)** =
12651261389333779943487931934773422247369566003549647139455822052906518
27674605003741290400826146165752452701594226002213036650208863340321329
79848926416072893065331590752192613664292834754982514402626203574735018
24937955593850701309595524998138852023345759936429351281324585455234984
89490586883187848396314164874056757696154989511633927620869557222556876
85599907930883941741601274620604045561100209252025573612167329896305069
36399163679682808070289756145961140222305243601505813448842198345190256
19777858430431159461562871537004523472161672182851052258466610762884570
310894027628303901161674783788320479747219000276

And the resulting ciphertext is
**r =**
38136774394448379903812816247692654840719898834948337653631552140717275
73627590213038823018054653614040833306533736593789523636716088751609591
51785286821705290541575145796194230921380378266117404213106755599686009
42963154830873754443624540928919604920987962346243921861126591249158725
46640723139762874453050592110272036917039293020539724872406856066252779
41948265167232013209242193986739266879595915531263480488821530060772558
43305317202103552015505297649368817612108108831029864641114090965723641
85502722477587178710137175828696000683028806920671859797982157383943866
111320227830105178421690303627627943337128795446
**t =**
68790858725328834966796372898277580443884935921922764850184204671271756
92447676225327570450845191312409829734608730732636786181351723791499758
73467997825997443956427111643147855992040692428669811529143452987980194
78640124849283241164105627138599989876596073170505753291429488326162640
96105332977657905567987590712241261025634719542322903245193802690474499
32300962686285307035975532477619822290316123173732108318081948470453314
13604021858729208684016357849233000803660656016265115503385857164467328

543722191419540194809031468192955271052196857743673492347620811165147289074687212410556494617517114100661282187862416 02

Can you find the message? (**Note:** The message is a byte object that contains a meaningful English text.)

There is an error in the number random number k generation in the encryption function. There are 65535 possible values for k, but the range of k supposed to be 1 < k < q-1. We can conduct a brute force attack by trying all possible values of k since it is small, 2^16 possible values for k.

After finding the k, we can find the message using the public key (h).

t = h^k * m mod p
we have h and k.
Find h^-k mod p and multiply it with t (in mod p). The result will be m.

**Message = Be yourself, everyone else is already taken.**
**K = 31659**

**4.** (**20 pts**) We encrypted two messages, $m_1$ and $m_2$, using the ElGamal encryption algorithm given in "ElGamal.py", however, it contains a flaw, and we lost $m_2$.

**q** = 1445431254694174381649371259143791311198736690037
**p** =
137248121434045436247980738953059412416367251619167172965225060439638326312552007992983578734870080149141102688002009860722627928048376753275218309927198296531391131491381377746970705292972549293385978940242862964757496679733959578043293370426396437630135799843979374589693726945392682404824784160383287430661
**g** =
127223641921850109909544249881449009944648689040286349526712184078921702602665543540563817762837809423359475544561229778960073396175252439333049143438367080170746166373310913545533812707513022571241268299810387846306038162098727078834162806032355796383642287190219288720676739470587659262303423658215573377024

$(message_1, r_1, t_1) =$ (b'Believe in the heart of the cards.',
981126369090898234738868042307346087836651513598202853843851849265867793188323428404467568452706851518435205925210307780631074614791855841297248385000267419660097063751812009739442913777532935355998701963457948398283879115798092238301956748210799021237004599484194930009559746053404002746439347954181179534 31,
765062002788709806228321620877063971849427311758810730722796538791253740267842312430822498385702091977887034189945986637702227749585904843662974734645479761571015367390566383404017099739109229529873329612584145068777452485994947010057901942620835406265751727713368885974020329234070572190289846977392942344 94)

$(message_2, r_2, t_2)$ = (b'????????????????????????????????????',
98112636909089823473886804230734608783665151359820285384385184926586779
31883234284044675684527068515184352059252103077806310746147918558412972
48385000267419660097063751812009739442913777532935355998701963457948398
28387911579809223830195674821079902123700459948419493000955974605340400
274643934795418117953431,
95801086901355834240081662719865802187550109851113545620170852280638597
49380166285757620063366674966331826060707996383796712218801355443439556
51964307083435544527207340562502675210978555861807927227967728935300895
00987302933561979841152407078582329739116130182358926512269862531407749
668332924957717479984854)

Can you recover $m_2$ using the given settings? If yes, demonstrate your work. (**Note:** $m_2$ is a byte object that contains a meaningful English text.)

Since r_1 == r_2, the same k was used to encrypt both messages. This gives us the following equations:

b is the private key
B = g^b mod p
r_1 = r_2 = g^k mod p
t_1 = B^k * m_1 = g^b*k * m_1 mod p
t_2 = B^k * m_2 = g^b*k * m_2 mod p

We can find g^b*k by multiplying t_1 with m_1 inverse in mod p.
Later, we can find m_2 by multiplying t_2 by g^b*k inverse in mod p.

**M2 = A person can change, at the moment when the person wishes to change.**

5. (**20 pts**) Consider the DSA scheme implemented in the file "DSA.py". The public parameters and public key are:

q = 18055003138821854609936213355788036599433881018536150254303463583193
p =
17695224245226022262215550436146815259393962370271749288321196346958913
35506375712221640003869912589713733824564565462318090744577539747691432
64541823312008430398287532100519638386733995377507645193811240740220035
33048362953579747694997421932628050174768037008419023891955638333683910
78329632006831350246795354984562936432868516805533133037843946010726267
22079113840299167310404286007959522483856834483390513263738796230245863
81484917048530867998300839452185045027743182645996068845915287513974737
09431107148527983017880233288432295348503295405569826328682916838056115
4757985319675247125962424242568733265799534941009

g =
47890739417772326639259461165485122364540071959307165458442555156719219

02088454647562920559586402554819251607533026386568443177012595965432651
51649487309428467188058704308016870979272958086439952207044001358870142
71007707855273217177840685312534890153131716384460348058478457205676914
12760307220603939165634874434595948570583948951567783902643539632274510
31700867667564432415210708332548490156210485764462112134840941155765304
18249730632155995395208828714498515133872706134004643148796528363523636
37833225350963794362275261801894957372518031031893668151623517523940210
995342229628030114190419396207343174070379971035

public key (beta):
18314081605332185106869037261386659325365184669318569898359418532687304
68186911958415037229987343935227988816813155415974234360530276380966386
58612174734034815855322536331991865794938293719845501829483638158455018
18002018688066945274182797974927581517692768509109442443956455724977667
48854242598561659704665374023326770662512666613356092618904914953512155
80425212764881853428583177337051045313795268854349501010366089241339590
14612382097254807376250471592757819220880767207174340624442369693937568
80954396658965471745598003472511293882525516878617801436300794663357187
223445935638034452125753926695866508095018852433

You are given two signatures for two different messages as follows:

(message1, $r_1$, $s_1$) = (b'The grass is greener where you water it.', 16472915699323317294511590995572362079752105364898027834238409547851, 959205426763570175260878135902895476834517438518783120550400260096)

(message2, $r_2$, $s_2$) = (b'Sometimes you win, sometimes you learn.' 14333708891393318283285930560430357966366571869986693261749924458661, 9968837339052130339793911929029326353764385041005751577854495398266)

Also, you discovered that $k_2 = 3k_1 \ mod \ q$. Show how you can find the secret key $a$.

K2 = 3k1 using this fact we can get the following equations.

h1 = H(m1)
r1 = (g^k1 mod p) mod q
s1 = k1^-1(h1 + ar1) mod q

h2 = H(m2)
r2 = (g^k2 mod p) mod q = (g^(3k1) mod p) mod q
s2 = k2^-1(h2 + ar2) mod q = (3k1)^-1(h2 + ar2) mod q

3*s2 = 3*(3k1)^-1(h2 + ar2) mod q
= (k1)^-1(h2 + ar2) mod q

Using these equations, we can solve for k1. The explanation and the equations are in the below.

s1*r2 = k1^-1*h1*r2 + k1^-1*ar1*r2 mod q
3*s2*r1 = k1^-1*h2*r1 + k1^-1*ar2*r1 mod q

=> subtracting the two equations above
s1*r2 - 3*s2*r1 = k1^-1(h1*r2 - h2*r1) mod q
we know h1, h2, r1, r2, s1, s2 and q so we can solve for k1.

After we find k1, we can find a by solving the equation below:

s1*h2 = k1^-1*h1*h2 + k1^-1*a*r1*h2 mod q
3*s2*h1 = k1^-1*h1*h2 + k1^-1*a*r2*h1 mod q

=> subtracting the two equations above
s1*h2 - s2*h1 = k1^-1*a*r1*h2 - k1^-1*a*r2*h1 mod q
= k1^-1*a(h2*r1 - h1*r2) mod q
we know s1, s2, h1, h2, r1, r2, q, and k1 so we can solve for “a”

**k1: 51400235354453527906658377827733856604754770842697713338906825164 53**