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.

Step 1: Query the values of "c", "n" and "e":

- c = 4970588410618684520616769777087491144540786951689424245402876626495743374781751403055244264807452811982080827774809116329570646626650089415989625985195382345756791379040942972448647260804401976140915490077534904793744539218120735676687229805163236359486929570288496228634431204890355618701453598386055547284026979306061551478674033661283559056482815427073542180699991714823601633143410848746507556919257847127789379293265206857245626727083488818651813072949776704633128868818170744467860137743287167769758656506344239292

660106423640203085405410451663921765943910891070890422286487391181480502398937005016878844119790

- n = 16939269462313198277725089002524968140769895904731797247921693446825361577176513265093962001473249149493514478809355121972226854971742591412020947781343582572258409348854687792672350790838706795045105779549770778632847428289027972956461589642862664604406205649788062899700494060455208957227204718515101331592572887762868780071870612248382350488623041960895952847347193036811161635348404536654480418700288618890111185617389009319953809419282210422869145631879305871459779170673590029863798266603861263916454445940018625253525716638055179599474296505004768081393212277916373461197482274715508530005125354616487101316387
- e = 65537

Step 2: Choose an arbitrary integer value (I selected 5) as r and raise it to the power e: 5^e
Step 3: Multiply 5^e value with c, then query the cipher.
Step 4: Divide the decrypted message, the message that is returned by the RSA_Oracle_Query, by r=5.
Step 5: Decode the decrypted message to see the actual one as string.

Result:

- **The message is: Bravo! You found it. Your secret code is 55301**
- **Received "Congrats" message from the server.**

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**) = 15563317436145196345966012870951355467518223110264667537181074973436065350566
public key (**e**) = 65537
modulus (**N**) = 73420032891236901695050447655500861343824713605141822866885089621205131680183

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

Attack:

- The possible values of R: R ∈ [1, 255] and R ∈ Z.
- The possible values of the pin: P ∈ [0000, 9999] and P ∈ Z.
- Based on these two information, encrypt all possible pin values with the help of each possible R values.

Required Time: Approximately 1 minute.
Result: The pin value is **1308** and the R value 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 =
12515043909394803753450411022649854273721537251011107748826811168459680
62835139115448704132059500673623933219249223694396652305374447612772879
79638081511425065953301206216633715182811812047978317073494365584431393
55672347825267728879376289677517268609959671235059224994785463608330669
49445716325037358138003624765203096948104677201379927126871010448702216
48650048028640760669741530121255510609060541129204698690452233295770159
35824864428612446723942040465300185917923305042033306319809712618872063
79690413278828551849799932748592973092120274593593691383457761025429880
92055751620050251708782007865907518500068579214 19
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
54372219141954019480903146819295527105219685774367349234762081116514728
907468721241055649461751711410066128218786241602

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

Attack:

- Inside of the ElGamal.py file, the k value is given as an integer between 1 and 2**16-1.
- Also r = g^k mod p.
- Hence, for each of the k values in range 1 to 2^{16}-1, if g^k mod p = r, then this k value is the one we are looking for.
- After finding the k value, focus on "t = h^k * m mod p" equation. Reorganize this equation as "m = (t * h^{-k}) mod p".
- Decode the found m value to retrieve the message as a string.

Result:

- The k value is: **31659**
- The message is: **Be yourself, everyone else is already taken.**

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 =
13724812143404543624798073895305941241636725161916717296522506043963832
63125520079929835787348700801491411026880020098607226279280483767532752
18309927198296531391131491381377746970705292972549293385978940242862964
75749667973395957804329337042639643763013579984397937458969372694539268
240482478416038328743066 1

g =
12722364192185010990954424988144900994464868904028634952671218407892170
26026655435405638177628378094233594755445612297789600733961752524393330
49143438367080170746166373310913545533812707513022571241268299810387846
30603816209872707883416280603235579638364228719021928872067673947058765
92623034236582155733770 24

$(message_1, r_1, t_1)$ = (b'Believe in the heart of the cards.',
98112636909089823473886804230734608783665151359820285384385184926586779
31883234284044675684527068515184352059252103077806310746147918558412972
48385000267419660097063751812009739442913777532935355998701963457948398
28387911579809223830195674821079902123700459948419493000955974605340400
274643934795418117953431,
76506200278870980622832162087706397184942731175881073072279653879125374
02678423124308224983857020919778870341899459866377022277495859048436629
74734645479761571015367390566383404017099739109229529873329612584145068
77745248599494701005790194262083540626575172771336888597402032923407057
219028984697739294234494)

$(message_2, r_2, t_2)$ = (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.)

Method:

- Observe that $r_1 = r_2$. Hence, k values are also the same since the $r = g^k \bmod p$ where g remains the same in r_1 and r_2 equations. In short, $r_1 = r_2 \rightarrow k_1 = k_2$.
- Assumption: Both messages encrypted using the same h and s values.
- $t_1 = (h^k * message_1) \bmod p$. Reorganize this equation as: $h^k = t_1 * (message_1^{-1}) \bmod p$.
- $t_2 = (h^k * message_2) \bmod p$. Reorganize this equation as: $message_2 = t_2 * h^{-k} \bmod p$.
- After finding $message_2$, decode the message to display it as a string.
- Observe that the assumption holds.

Result:

- The second message is: **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

3550637571222164000386991258971373382456456546231809074457753974769143264541823312008430398287532100519638386733995377507645193811240740220035330483629535797476949974219326280501747680370084190238919556383336839107832963200683135024679535498456293643286851680553313303784394601072626722079113840299167310404286007959522483856834483390513263738796230245863814849170485308679983008394521850450277431826459960688459152875139747370943110714852798301788023328843229534850329540556982632868291683805611547579853196752471259624242425687332657995349410 09

g =
4789073941777232663925946116548512236454007195930716545844255515671921902088454647562920559586402554819251607533026386568443177012595965432651516494873094284671880587043080168709792729580864399522070440013588701427100770785527321717784068531253489015313171638446034805847845720567691412760307220603939165634874434595948570583948951567783902643539632274510317008676675644324152107083325484901562104857644621121348409411557653041824973063215599539520882871449851513387270613400464314879652836352363637833225350963794362275261801894957372518031031893668151623517523940210995342229628030114190419396207343174070379971035

public key (beta):
1831408160533218510686903726138665932536518466931856989835941853268730468186911958415037229987343935227988816813155415974234360530276380966386586121747340348158553225363319918657949382937198455018294836381584550181800201868806694527418279797492758151769276850910944244395645572497766748854242598561659704665374023326770662512666613356092618904914953512155804252127648818534285831773370510453137952688543495010103660892413395901461238209725480737625047159275781922088076720717434062444236969393756880954396658965471745598003472511293882525516878617801436300794663357187223445935638034452125753926695866508095018852433

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, 95920542676357017526087813590289547683451743851878312055040026 0096)

(message2, r_2, s_2) = (b'Sometimes you win, sometimes you learn.' 14333708891393318283285930560430357966366571869986693261749924458661, 99688373390521303397939119290293263537643850410057515778544953 98266)

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

Main Objective: Find k_1 to find the private key (a).
Step 1: Analyzed the DSA.py function and observed "s = (k^{-1})(h+ar) mod q".
Step 2: Reorganized the given equation as:

- $a = ((s_1 * k_1 - h_1) * r_1^{-1}) \bmod q$.
- $a = ((s_2 * k_2 - h_2) * r_2^{-1}) \bmod q$.

Step 3: Recalled $k_2 = 3k_1 \bmod q$.

Step 4: Combined all previous info:

- $(s_1.k_1 - h_1) . (r_1^{-1}) \bmod q = (3.s_2.k_1 - h_2) . (r_2^{-1}) \bmod q$
- $s_1.k_1.r_1^{-1} - r_1^{-1}.h_1 \bmod q = 3.s_2.k_1.r_2^{-1} - r_2^{-1}.h_2 \bmod q$.
- $(s_1.r_1^{-1} – 3.s_2.r_2^{-1}) * k_1 \bmod q = r_1^{-1}.h_1 - r_2^{-1}.h_2 \bmod q$.
- $k_1 = (s_1.r_1^{-1} – 3.s_2.r_2^{-1})^{-1}.(r_1^{-1}.h_1 - r_2^{-1}.h_2) \bmod q$.

Step 5: Find the h_1 and h_2 values by using the “shake” method of SHAKE128.

Step 6: Compute $k_1 = (s_1.r_1^{-1} – 3.s_2.r_2^{-1})^{-1}.(r_1^{-1}.h_1 - r_2^{-1}.h_2) \bmod q$ using h_1 and h_2.

Step 7: Using k_1, compute $a = ((s_1 * k_1 - h_1) * r_1^{-1}) \bmod q$.

Result:

The private key (a) is:

2247688824790561241309795396345367052339061811694713858910365226453