Homework #3

Due date: 29 November 2019

Notes:

- Note that there are three attached files: "RSA_OAEP.py" for Question 1, "ElGamal.py" for Question 2, and "DSA.py" for Bonus Question.
- You are expected to submit your answer document as well as two Python codes for Questions 1, 2 and 3, respectively.
- 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"

1. (**30 pts**) Consider the RSA-OAEP implementation given the file "RSA_OAEP.py", in which the random number R is an 8-bit unsigned integer. Using the public key

(e, N) = (65537,
702122840264765512874978673446601730622426199353069976079859874283520529112 93)

I encrypted my 4 decimal digit PIN and the resulting ciphertext is

c =
604009437068235068302842801141398182887150160234171034652307805220758620907 39

What is my PIN? Submit your code used in finding my PIN.

2. (**30 pts**) Consider the ElGamal encryption algorithm implemented in the file "ElGamal.py", which contains flaw. I used this implementation to encrypt a message using the following parameters:

q = 133116579422373099821447968205529080913980370397 9
p =
157985265365233926063394088702502775477411699807450440916775405947257964574813502993749815770128973338289285516154109088043476314331444397215358170585840641049172791477662283893716386808139204949694492602287891654767148522867881937046157301612266431912023462991540765986938468014946969764702086638496649455657
g =
135065956040029542891335614580458248416002250295204395146754036299690682917789289716583464736425816867965184913947179997468650756414729019331183463002574881956749358833871584578559474520218159551812002168419391427229522879948629379275361929622066470148375436287416532348407065836711249965189758444892419490190
public key (h) =
653695313804348110910131692851440746542742641260193408761167219776465 67

45310844143947658061413114101871121703918315944733970349864572309970933
13100350016070503357404368252229384210531259359860240305661205857146822
25125302549720107383365077208839310065478286374555114097276440333388769
52331667104411748745458 9

And the resulting ciphertext is

r =
36032169644425073570328427142654913561401061701260122712492737824987810
62854993590551963255079610858746338241608699542316440867356686210833642
81601595244890540839091779710540890021439859180686924557245365290222297
11162933532847371564973218717503016150130093952097139288475679032477430
3386719979185998111726 3
t=
42244680577489180150438247901105682607063917920969521526593784819134087
33761717162443465848326282188055945272315168573124840030020549530341444
75930796914614370187694754794376852744464747852209669396707112060640762
36087750024913718835564780638938635976122622009741305316868203434730711
66386339806524917392564 5

Can you find my message?

3. (**20 pts**) In Kerberos, the Ticket Granting Server includes the identity of the server (S) in its response to client: $E_{K_{CG}}(S, K_{CS})$. What would happen if the identity of the server were not included? Does it lead to an attack?

Answer: If there are more than 1 server, or servers that are connected to each other, then you may want to include the identity of the server(S). This is important because unconstrained delegation/privilege may be granted to the user that allows a server to authenticate and access resources on another server on behalf of the user. In addition, the encryption $E_{K_{CG}}(S, K_{CS})$ must be done with respect to the ID number of the server because Cliff and Simon generate a shared key. If there are multiple servers, and encryption is done without the unique ID number of the server, then the user may be able to access multiple servers with the same key.

4. (**20 pts**) Alice and Bob wants to establish a secure channel to communicate securely. Suppose Alice and Bob have long term RSA public and private key pairs: (pk_A, sk_A) and (pk_B, sk_B), respectively. They can use both the RSA signature and encryption algorithms and they know each other's long term public keys. Show how Alice and Bob can achieve <u>forward secrecy</u>.

Answer: Forward secrecy is true only if the compromise of long term keys does not compromise past session keys, and this is achieved with station-to-station protocols. Alice and Bob can achieve forward secrecy with RSA signature and encryption algorithms by following the steps below. Alice can send a random R to Bob. Bob creates a new key pair, new public and secret key (pk'_B, sk'_B) used for encryption. A new signed S is created by signing the hash of the message R$||pk'_B$, with sk_B. Bob sends $sk_B||$S to Alice, who verifies

the signature S with respect to sk_B for R||pk'_B. Alice generates a random symmetric session key K, which is encrypted using pk'_B to produce X. The message M is encrypted using K to produce ciphertext C, and sends X||C to Bob. Bob can decrypt X using sk'_B to produce the key, K. Then, it is able to decipher C with key K, getting message M.

Bonus Question

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

q = 897434149680309024926610536586679400252190817513
p =
972230041992663135230491660533300290923805413007861389248731810884714387054537940463709143455924323680592712945441027227879573108377973046509430698205202875498266302306176257925262147992064864445546072751570317428081226670648766551387485679450518784599684348409721353547458938686602670097948760940573073602 71
g =
462149721005793561237198851171193251036131811560998097885323698431456173981903931327182010509863848021429387647707087272383149376926842244171487601439695456713666558346129313879250210049818171460576161508867009880801662561730986085868295719726529473739536216797593009764895897242447988019478770985237114257 9
public key (beta):
457202230925588203447699300286148036388590519071295012778809990627408521148896103778940395209730538471741449555526271742660619393235771846817282811568127366031229992622099530012382294391081176774238575412718410043094690662080833852542715896365421607679029218038602706993599110813469695221861143113902266779 95

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

(message1, r_1, s_1) = (b"He who laugh last didn't get the joke",
867552604169477346883674422144796797059303863627,
243861349833858115605937030382497401412336608822)
(message2, r_2, s_2) = (b"Ask me no questions, and I'll tell you no lies"
686145019080375810998084468514665120375929537329,
774583422188330317252601038183072854135396118762)

Also, you discovered that $k_2 = 2k_1 \bmod q$. Show how you can find the secret key a.