**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, 70212284026476551287497867344660173062242619935306997607985987428352052911293)

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

c = 60400943706823506830284280114139818288715016023417103465230780522075862090739

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

Exhaustive search on PIN and R will yield the PIN as follows: (PIN, R) = (5377, 157). See “RSA\_OAEP\_sol.py”

1. (**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 = 1331165794223730998214479682055290809139803703979

p = 157985265365233926063394088702502775477411699807450440916775405947257964574813502993749815770128973338289285516154109088043476314331444397215358170585840641049172791477662283893716386808139204949694492602287891654767148522867881937046157301612266431912023462991540765986938468014946969764702086638496649455657

g = 135065956040029542891335614580458248416002250295204395146754036299690682917789289716583464736425816867965184913947179997468650756414729019331183463002574881956749358833871584578559474520218159551812002168419391427229522879948629379275361929622066470148375436287416532348407065836711249965189758444892419490190

public key (h) = 65369531380434811091013169285144074654274264126019340876116721977646567453108441439476580614131141018711217039183159447339703498645723099709331310035001607050335740436825222938421053125935986024030566120585714682225125302549720107383365077208839310065478286374555114097276440333388769523316671044117487454589

And the resulting ciphertext is

r = 3603216964442507357032842714265491356140106170126012271249273782498781062854993590551963255079610858746338241608699542316440867356686210833642816015952448905408390917797105408900214398591806869245572453652902222971116293353284737156497321871750301615013009395209713928847567903247743033867199791859981117263

t= 42244680577489180150438247901105682607063917920969521526593784819134087337617171624434658483262821880559452723151685731248400300205495303414447593079691461437018769475479437685274446474785220966939670711206064076236087750024913718835564780638938635976122622009741305316868203434730711663863398065249173925645

Can you find my message?

Flaw is in the encryption function. The session key k is too small; it is only 16 bit. Therefore, an exhaustive search by checking r = gk mod p will yield the value of k used in the encryption. Then, computing the value h-kt mod p will give the message, which is

b’My favorite machine at the gym is the vending machine.’

See the file “ElGamal\_sol.py”

1. (**20 pts**) In Kerberos, the Ticket Granting Server includes the identity of the server () in its response to client: . What would happen if the identity of the server were not included? Does it lead to an attack?

Note that the client sends the identity of the server S in plaintext in its previous message to the Ticket Granting Service. Another malicious (or compromised) server, say S’, can intercept that message and changes the identity of the server S to S’. If S is not included in the response message, the client would believe it is using service S while it is S’ it is interacting with. This is men-in-the-middle attack.

1. (**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: () and (), 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 forward secrecy.

Alice can create a new pair of public/private key pairs (). She signs her new ephemeral public key with her long term private key, .

Bob receives the new ephemeral public key and verifies her signature. He then generates a random key and encrypts it using and sends to Alice.

Alice decrypts with .

They can use the common secret key to establish a secure channel.

**Bonus Question**

1. (**20 pts**) Consider the DSA scheme implemented in the file “DSA.py”. The public parameters and public key are:

q = 897434149680309024926610536586679400252190817513

p = 97223004199266313523049166053330029092380541300786138924873181088471438705453794046370914345592432368059271294544102722787957310837797304650943069820520287549826630230617625792526214799206486444554607275157031742808122667064876655138748567945051878459968434840972135354745893868660267009794876094057307360271

g = 4621497210057935612371988511711932510361318115609980978853236984314561739819039313271820105098638480214293876477070872723831493769268422441714876014396954567136665583461293138792502100498181714605761615088670098808016625617309860858682957197265294737395362167975930097648958972424479880194787709852371142579

public key (beta): 45720223092558820344769930028614803638859051907129501277880999062740852114889610377894039520973053847174144955552627174266061939323577184681728281156812736603122999262209953001238229439108117677423857541271841004309469066208083385254271589636542160767902921803860270699359911081346969522186114311390226677995

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

(message1, , ) = (b"He who laugh last didn't get the joke", 867552604169477346883674422144796797059303863627, 243861349833858115605937030382497401412336608822)

(message2, , ) = (b"Ask me no questions, and I'll tell you no lies" 686145019080375810998084468514665120375929537329, 774583422188330317252601038183072854135396118762)

Also, you discovered that . Show how you can find the secret key .

We can find a formula for computing the secret key as follows:

Applying the formula, we obtain the secret key

a = 253269165174290268846821928601654697793514680881

See the file “DSA\_sol.py”