# Homework #3

Due date: 10 November 2019

**Notes**:

- Note that there are four attached files: "rainbow_table.py" and "rainbowtable.txt"for Question 2, "DeterministicRSA.py" for Question 3, and "RSA_Oracle.exe" for Question 4.
- You are expected to submit your answer document as well as two Python codes for Questions 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_hw03_yourname.zip"

**1.** (**20 pts**) Consider GF($2^8$) used in AES with the irreducible polynomial p(x) = $x^8+x^4+x^3+x+1$.

**a.** (**10 pts**) Perform the following multiplication in GF($2^8$):

$(x^7+ x^6+ x^4+x^3+x^2+1) \times (x^7+x^5+ x^3+ x^2+x) = ?$

**b.** (**10 pts**) Show that the inverse of $(x^7+ x^6+ x^4+x^3+x^2+1)$ in GF($2^8$) is $(x^7+ x^6+x^5+x^4+x^3)$.

**2.** (**20 pts**) Consider ten digests in the attached file "rainbow_table.py", each of which is the hash of a six-character password. Your mission is to find those passwords using the rainbow table given in the attached file "rainbowtable.txt". Complete and submit the Python code in the file "rainbow_table.py" such that it finds and prints out the ten passwords corresponding to the digests.

**3.** (**30 pts**) Alice encrypts the private factors of the modulus using her private key. In order to increase security, she multiplies them with a random integer *k* (a process called blinding). Namely, she performs the following operations

$c_p = (kp)^e \bmod n$ and $c_q = (kq)^e \bmod n$,

where
n =
7473729494541494368852084160833887966786966146604539788038425546370721005202575292878010603718779143680279072437859979437027212668979951812104144364476251149631445677106574998958718100538368157559892086233484623658307080278702230379266872415713089871004732096589645883883977415506508347976001334807533323619602524830567712971581132341673584628698079736760920789626175805249978676215337039548212810503820392130214528748452439765014841123109184880811527461970066588991000432403773750048368279867220678747020767042484253902752403310560243017584585301433673020033953058809415002025715734068685302158511640316340598416777
e = 65537

**a.** (**10 pts**) Explain why this is not secure as anyone who obtains $c_p$ or $c_q$ can factor n.

**b.** (**20 pts**) Factor *n* assuming

$c_p$ =
6237092736280340368143892975246879700280574956823483459089826300838 54
9326698371192841125256658301422335506071280990036134969958430140091 82
1413942110999828019223701914534660496145184049436793233117527141710 46
9587852896572612908205944874924666105935993339272072516489985048899 80
3563277390008625395078803685108980714466378814566336644612044895653 93
8680103088252710048504048952616260337657469904426834625756755178823 71
2343160146527660412642787198955050511984140986234095815703702908475 14
6055296882216504049042640234735532803592978062118777287332266867117 93
527246849144732646792860895358114217072778019398298862559489 8212

$c_q$ =
6792216459178320663134922113672614244133286795167205940387994163295 12
1940464502516578964764558190390582955957319087527927606354995278787 00
2827810252216604393602551619997059706389437721444109986118687258571 55
8770491123542808098899449070164438070259725300867432155155552476715 44
5078689763334188428800272552596349007005222835623783668243124391007 97
9542657577232082178629086100963707200714469436928816142276093722080 93
7180495045103388174875710073753498922326189147660992582560179759354 31
3355942469090808651587693854760749192023182331797094543656070987815 65
408854411559147366625012913060391004535757931864005423266852 4224

and decrypt the following ciphertext

$c_m$ =
2105477800955456357154507258188617457189736593253101663074145074953 98
1395245071021806753010279680874168054072882966675195121143457746629 68
6736359549415401549852363809597210114934407644579644329802951418430 99
4896968996554627286997788295104796260715656641752826531876202868222 65
9539111193689690587665763970475954283298561326716629497355710903035 32
4921787171114135903504002506114489283933796611287777504398400559494 45
9519397397254887260526830796723748881723043560212833332272718498671 83
5837847760126707870313591817583090393736883523163338481442463241882 37
546682426764025888441963516716313575349019554430385874921591 4789

and print out the plaintext message. For the solution of this question complete and submit the attached file “DeterministicRSA.py”.

**4.** (**30 pts**) Consider the following security game. Suppose that an attacker wants to decrypt the ciphertext *c* encrypted using the RSA algorithm and obtain the plaintext *m*, where $c = m^e \bmod N$. She knows neither the private key *d* nor the factorization of the modulus *N*. However, she can query an oracle (e.g., a program) with a ciphertext $c' \neq c$, and receives the corresponding plaintext $m' = c'^d \bmod N$.

**a.** (**10 pts**) The attacker can decrypt *c* and recover *m*. Show how.

**b.** (**20 pts**) Consider the following RSA parameters and a challenge ciphertext

N:
413105080470194761547414333064752004340942254255543889977200661958494
534011949464940412937181797972315194912124170337875272269953823057328
093111008990493716790723567095060040847336735549070359918268959251096
858672312347266354823678845903826815732607877891116752803838696450255
241260806010800776952451894277162217528982318247301802639309883561496
368306259480703690662603612912378956692729173285461990259772640312588
196857444617991899814097065255670996261112542559432923146658846703096
162046912513465456692329753804229233254440617551891667821068692593634
1811732104917605118575775329334829543752265141266783630832854187

e: 65537

c:
286011333479246811807036978298499172055797623822007560823628330407176
989331751815848494737394310492374805419675958067102325270756298968875
428244016352736458046904203412669134207806910534413220489710408452767
455368567189168921003750345234473915004072109521598843246444074965165
007427386987929674642591506744871181356247565764232892236100024802851
129256309465367597958913617035234759551640734971193268573642824284719
009304086710729931481730871492097430958962565736552256356850187124726
045482360175673464358349365112652477377441584665739736701619949825887
312761535693506114283123294171040950993106172221548552046342629

The RSA oracle is implemented as a Windows exe file (RSA_Oracle.exe) provided in the assignment package. If you are unable to run the RSA oracle, you can send your query to Erkay Savaş (accessible via erkay.savas@sabanciuniv.edu)

I challenge you to find *m* corresponding to the ciphertext *c*. Note that *m* contains a meaningful message. Convert it to bytes to discover what it is.