## Homework #3

Due date: 26 November 2021

**Notes:**

- For Question 5, you can use a Python module for arithmetic in $GF(2^8)$.
- You are expected to submit your answer document as well as the Python codes you used.
- Zip 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. **(10 pts)** You are in a job interview, and you were given the following RSA parameters:

   **N**=
   54850090214586797322771539095341831801335980218725680233852488227557954
   79347601253294097226308258506404813974945781991532702821384186354911009
   56525038870603756062692199814547914931136384581391254255489944699369491
   84053598096412002568561253603284179050474481457454341400323202669056591
   77161474262484583073500890995340659704543618182825506616586863117372476
   06817922172862379659085526235765861964356873490884219476066897290853745
   79540101188932992302734875871705917047995941999319442231957991298097030
   78183143892520626455101414288073204323211682546090936226519524876589170
   17050428274312634323855991187203490466658877488298150081941270603917092
   05184515781002682993167829749424086319201353236499360965128782529483900
   12159315401233559314395029277689176390348772390678298859428873193576734
   33030624334940037307750441355075186017928404978264691170176937706161551
   66167821881261565592665850095889259064997895295099253226539533324191388
   13647990860225674264208568324536088861122145214508328541432792190255350
   52457936973864622557523397923414943924630749301754219076377009170985602
   73945426832561120092728167988591162147054797388781703708155609892045301
   98187016719122238686652561534165553128386248411282697990727192376902500
   80143673671335205433134989

   **C**=
   21000202784665228236404895052572788342254876258143969814982532821954936
   40367632414901866601611030793241685730477419473507311190940215876219605
   68992564251499575153695010352276351424008748971409946951098601517468716
   12968786725786196380163877362337513482433638170770214772432995708021921
   92093103420652939359741759068556271288862908012816502348800235533259918
   74310497676417671135535823121979117339520847820264954549100267870376726
   28893171835588445525167559613346276025954740271946011695869017100396289
   86085787895096899294580498390533337572059767829455424943511035813633079
   12166755386987973685164853834583531209150315638560160174994069597269429
   85803728450806902572479261565515904585380720764427396714868810685366733
   37462683356233212594721433039278161283184382964225806397802344898418328
   97953928274712924887560685975172164724444045948993850479815287161644772
   54952160524473824260276330759495551998024615539930721545947311060583151
   12677901229368375168009859683369385012524321503008887679722582310123171
   82800441928577419507189904989872944869947373385462904190716225171693307

99305887915917505982513696010753820504785696894665705120622810313910372
98369888256

**e** = 2^4+1

You are asked to retrieve the plaintext “M” using only these given parameters, the plaintext is a 224-bit number.
(**Hint**: M << N)
Show your work.

2. (**20 pts**) Alice encrypts the private factors of the modulus using her public 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 =
1801696747726890556388586992423725831869469433661682966355995166715425415350566907236623817747326344055800049299153637253721328398852948471122989344150707180374979949879010570248344453600398813266058695045156892148530402985920594296133750386988236160562312199332614510747266321550923690859424172665179765204844084148761662222708803978630071030127595680427734334141473748902677036830357336916234986094286090582579482015183896406921446654729571774920292192188350170439477757254225345384297979322548882895223302425219597319124605217762070912435239286501893859858247497583979408401927872726727810561146280251924854860423 1
e = 65537

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

b. (**10 pts**) Factor *n* assuming
$c_p$ =
98678201605000482554316389279502944428888574528020173564535054163100918080066051458561454637430741425731211995718833036871346839153552932830570543392241865757084971676229490632890028916331940693205684071065372238648211840804518908590882684748266632691167166373245039915433304625678767352956091642382945898541786449506952598537109083168058743960695159818352236303890158565093793642693347787163574152394398815883714001963553529898966222975063782158757003033953495543260999659385753480879632777368126655811080085382675645814611516277504237808251898719871924452793124852940793651748005343991400026276451587417088556708 9
$c_q$ =
140485344653232305453253541357793353734644094796221306042212463008486
867889142210276208096483328329570105315597285881500662659086635261398

44306737776707609375722715383010693830027879879337369996602006590945672203979052485220246619624725419251173989782594643346471157752575503977729450420254475672606438870084069644514973475243788574248080622778361942012108616350667689291762507592733743801333017344651751499710647047542569741531303356891214062815518816727003969643882074715733945650194347254192244884148448163326582044881804918361223903654789076948440449087391682096234090950225916257119515328501420323582955047875300

and decrypt the following ciphertext

$c_m$ =

5310140768263889112686766982663485232724147322779548078336689719623100738344476071783487189533703110721280321221697563144942842042881171904224385107645132790162181233375357861709728095507173847988163917175814859755934121945906829725954188639886204218206679347637378077917048385488686117395519692997296468695861765637450771729243711977287157170827507271298730049110587320022257849845106129870227608995699650687879819478556102994986257728139323444273420364374072371315966355286935606776340869423866064535925527997277884492246490694288615200901115911705053893536184781927590935953156270349338459654278881122216658517446

and print out the plaintext message.

**3.** (**16 pts**) Consider the Geffe generator of three LFRSs ($LFSR_1$, $LFSR_2$, and $LFSR_3$) with the following connection polynomials:

$C_1(x) = x^7 + x^5 + 1$
$C_2(x) = x^{13} + x^7 + x^3 + 1$
$C_3(x) = x^{11} + x^2 + 1$

You also observed the following output sequence of the Geffe generator:

z = [0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0]

Can you find the initial states of $LFSR_1$, $LFSR_2$, and $LFSR_3$?

**4.** (**5 pts**) We challenge you to get the plaintext of a ciphertext C that was calculated using an RSA setting, however, we lost the decryption keys, we only have the following:

N = 237540380304900134239
C = 226131284405640469226
e = 2^16+1

(RSA Encryption: $m^e$ mod N | Decryption: $C^d$ mod N)
Can you retrieve the message using only these information? If yes, show how.

- Once you find the message, send it to your TA “Anes”, the first one to solve it and send the message to the TA would receive an extra 5 pts.
- You are not allowed to use external tool (including online tools).

5. (**14 pts**) Consider GF($2^8$) used in AES with the irreducible polynomial p(x) = $x^8+x^4+x^3+x+1$. You are expected to query the server using *get_poly()* function which will send you two binary polynomials a(x) and b(x) in GF($2^8$). Polynomials are expressed as bit strings of their coefficients. For example, p(x) is expressed as '100011011'. You can use the Python code “**client.py**” given in the assignment package to communicate with the server.

   a. (**7 pts**) You are expected to perform c(x) = a(x)×b(x) in GF($2^8$) and return c(x) as bit string using *check_mult()* function.

   b. (**7 pts**) You are expected to compute the multiplicative inverse of a(x) in GF($2^8$) and return $a^{-1}(x)$ using *check_inv()* function.

6. (**35 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.