Homework #3

Due date: 24 November 2023

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.
- Brute-forcing the solutions (trying to query all possible solutions) in a server-related question would not be considered a valid answer and will result in a score of 0 for the respective question.
- Do not submit .ipynb files, **only .py** scripts will be considered. You can work on Colab but please, submit a Python file in the end.
- 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. **(15 pts)** You are in a job interview, and you were given the following RSA parameters:
   N=
   22365785976614402332397512075124630827034962474834967508376992591878950
   34976146624899681362879032402174196764503614296944218715118439577009702
   18235759498443614833318092931765347216940713346359968279518075329167958
   73629655454372863524613892925120878044667556713089187256277620400880942
   67449215491946365907708865695556836507394680104113645504915425469409338
   66887184619665711809176911780017788816413701783562918995081107754493978
   73342307750859548327215761131604445299914396581941939357225594068024701
   56301977446603456884493329985847799775443583165975769943149863223322845
   80410046513600329892486466122703108374316422340853237281774848941313615
   83681425555600156607217562397349329759129433534908756960805661036304391
   08934583450923383815434902213347700326633618836715343980988824339498954
   40210549331384333277774764351905291047781299766526170463628876778618958
   16432682335827401878705944583782205741084481506390559086053735656140311
   25387128925135332916959736300899479885278863339911134505004369987817266
   35149401000636540614535711159426410828179415759829439555704024037537581
   91438493616361452856840835486794570579705182218479249220132822428778059
   37507851590516843339593375327842430003170696068876068797437618214033096
   77083277828514944243054119009100504712199688593975568221513306723156884
   00023275713385475636673375705418460899832831371793346707182198279244735
   32685861642794873504118832531828344030201577499982915225832296328049335
   69509872827094895016940943886412321094069042912067184654291452447271607
   12334951148817109453889551067078695519930282020132174315707008087811410
   28641836700217121954352851945714621142371128880855259924062736762516185
   10314762959113098403437875872000055298048764042889369624300372225884514
   93743068203434317795857684578223380732309464862655712412606898281225998
   99301541810972769459376914733384588437545684656515385087580733584751932
   3533

C=
1099690731774404820118023919118480187887002635995425116380878699182398839204072169495677454061351187067291518696767449734569538264543011182408380035557053640855074823732249555664059589795090426697825592030049420606332644364403807706086785967885357313546368957601643718951818679943388857540103021087567684524465658391113407839387642187763991165704754307039546498784217046279934376752539321519160977407319028580673401739027081025769770474600551138459035828224316870549752867536482720149807798871390730269607077902128491358540075672275189364330649762937864320632331055818924807183171780835995553044898648188674931617535642437572787359717346627772786035759513 6

e = 2^4+1

You are asked to retrieve the plaintext "M" using only these given parameters, the plaintext is a 128-bit number and M << N (which means that M is too small than N).
Show your work.

Result:

- **Since $m^e < n$ (which is checked in the code file), we do not need modulo reduction for m. Meaning that we can directly find $m = c^{(1/e)}$: The plaintext M is: 340282366920938463463374607431768211456**
- **Possible "utf-8" decoding of the plaintext:** 😊 (smiling face)

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 =
1918361159411070404794426809840967974777769353357491434201327588862411634257797931230340362144109492849084860870935003922754310066663633684520457174499251412444151639165560973036591551350646400669817607461535092946799514695463580889979008650605522131698832134960609074526419504670773712974549258713219266086529801320700401681765125510257808966626173920533488804958664868638563235912259627125244377463544535152004607490824718881706644752561368476339305633719575522238966418639485495146262681769880721615070148094756692833171503752074705662188239798689102120997485064799575921586026671319271818461255092316143431940961316583546473099237449755409373180408918291605780136333109623395414361855094140335198046844627244986704201974351462960689505220298363091032183218874210416417719748655050226578564389603910038870248018907324637618899318852815946863999194819512645109688500446529459662402618430236765623524816799471564284544594738244329827946603609406266452608971474023507335797934547214673666130099218439298941865173438837614888353345837257745851339829795154457992643490774513386616054332568209918260966112425882868622906613937397970188743 4

97372590948593901543975084744946005962187441191496287970898124145277567
15416006852063181232015540335657245930749321809072404240556207462634556
07699700524768075354406018616956119182772903779733007050892970918375262
17352732496319085314920708082969122976857702438125484641488846223006033
33609697539815317426726928647381987950131331715044601820176825782919092
77355683737362686615312733563660679559873862037089763914089959724933586
05228390683894692756825505389326726472693267858287411509970899277244713
14364367502593122376235668982872064691932255585246865387923606731047553
13300203837226598476214001721836367864196351049976579073142410364805784
05353865022554985486575064306462132741635256070685397034325118442921285
4679
e = 65537

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

Result:

- $C_p = (k*p)^e \bmod n = (k*p)^e - j*n$ (where $j \in \mathbb{Z}$) $= (k*p)^e - (j*p*q)$.
- $C_p = (k*p)^e - (j*p*q) = p * [(k^e * p^{e-1})-(j*q)]$.
- Due to this final equation, $p = \gcd(C_p, n)$. Since we can find p easily, the system is not secure.
- Same rule applies for the q value too.

b. (**10 pts**) Factor *n* assuming

c_p =
8519815578170286631805017896096474075672015420898414555627327356398545
828618487300888192159283842112619538927550062944039231761324497687980
388057286117564517137213293019868685782639869702929608521619592268535
229492028029634820515189201256121144377516210646924865432305711831525
008988947515797126410611036738460263123054867022095937573549240340925
254091916638534390288411432539369888827553476153519678858720450565692
176315412675291771897611394319368108029911640959591350150421488906242
934415514983630107918903946261995406986686430038313043890273679453834
690423843286897558114707250513165137877470596586573123707812665592592
403567834932132278135356380337936297100779430354167187083997494595094
626591016495726463422933061481013504294768979677472371131930976861335
597522841734695274312040622145269057593028252460451835507800545385424
502204384514998189188158541525373820209794737665646233760704396689189
013105691772177688370635231713069776525934039464735163426253774229903
564605724843628061985053109841452458158235245878493737946522700803668
031486833088641375203452949089151179418388706769255178293828545447152
214805300142824095390634151626234438344708486656313860224440149446722
221773277601414826225728090233257030707578815037615238171938849710818
800602540059212528045709993461890163651551768045405513467695102682020
172652244680414719946537811168638190661135178875524417373837188607629
650148437360728889262240823128194832387029814711445604226771482932408
669331982883441090083879922610487695221496282020974515170482237913623
702763753281683261062144500298216845651293952563391167113383475580475

0045792900310469183134697466636763872767179461856691128995322469285487447938786325053015482489530254757378317395587115328986426716585076998064210883258828464744941273484010073386037552987182465771854300536973027299524764873622012478186053079729537911008670556589

c_q =

48184029866436360449528431298297059227259882990977050309184010413342970838740266996058956488274532679943325333991382838566176890291559095104716316095299468208686023751987393867864068468367547625334029861164445147106797073587099817197225492678887454649900791296059221304667885027123500190456937567035939888018706134687962954306671446207514281280469337798166154179590794129781597759427113306666770914102744529785750216352939413818340423714416966352713317477353615913794502217793489063586151927663074640502178510008519948143223528672871083868003818332558082307310010488926091283568931401099226922428211853247263006316903280948604310417097512297233615064888507433697972427343810795543329084470455477460112439927375463426420043480598247171634675156447454315414316801014784140367822320078056312242170065231457405237354388368298826262605556540651456554478549992093789414176162932930526223274916046573578307314813262058723663018829062641859943895102537839023424853860359930589099649128536560695036098415508010397772794339193851694305406644406824077791484608018638977828675990854391686495435052212013748187056778771806771450589696015007435284599020857754622733268322061401137815369720432446722819174683832546110910150059033851129227475444031660253620081707372012028792045041226482026188689672685041401213503656968596845286286540158334194582157869413208570702389619786521435670095228260805084456337469970826839834750606181489164187333781839672351296542868270334289349306453819429978632195805652205491116546222903842938030197845456214944198197300535981366429694463836000570392230389623902752641047748855524466240277207080750848483816984165583496784921010360726812834033106684166054776249902473870807852139711090714575294241686786187520826514025530392278498095198341712746819529749483372585680183346698630659083177489648149773761185798376228660955037681976614I3

and decrypt the following ciphertext

c_m =

172848288211416685603282354932360557971304059670763682199383435963868500148706386534697210638390065530228267133581948041498904505178993814709246745819492418376689358753522918376523234014124949810716237794880580694971790977763648689436017132643101332471248541108959265828346683832844081458155545765675070124650186739032982423587127281455778587634781124790027712594935258076846574264513386531287376117948606657102686393038586248002979140492947693573131139007314381474781938408307716833963251800718979780571443537325408508047240097427436806820452665019208185532305920139291552495894763989335439835817719740957265732107570025545100042355203025131498999148945291892978898093833012754835506941859064119833009544011419051103543206791596224369984223100745495349443635647092809400249940376709338464949084207198123184305476857855749352971

8212248374000615548213643423991321737723223060477210985847294810433191787893419222320423705481960357356943448874757935527228486703666367960398985790104880986620154010437391292643620797296252589898269633766328550214687520692980380923565067519637974711276815485577261639719591833080483985415833965627198502007168941289216168432917922312247707712961888596081170335361809902030483653605719840594299348495489300236771074325225105323450300637254009094683227479141830186791576095530102229621203611688982459451048429430253064809159276438419405955765763457844129271878734421416104191404689553066831980595000319096176966461932116958242960227343134139123567988440293552907917437570229383541482629782079480862663978043250398329722219508388704999089282906664754858575531462514889313482491646255044031348119733602661865586633091791695502190691818488605967807302727318082842483243701340921660030815554124647719995117645208273368037359335651906737986038473902686977967049445908642512694650162097605982512146831017115578920338711896567958 8

Result:

- **Find the p value by finding gcd(c_p, n):**
  348375397960405498713257599300924728067748083077651791167490636714011869880499871551089391881129421614419015172537741912903661133931128082666796637621282189548526874127067371876551272490934420164918803298576766154181661322659737958291226353292398847291918804694400491065678617090286344183861503211453337865932976128950630533279408703207015063259339012277026422481160921067116335029011968965768999795296038389482132161352725138507140962790484668740965416244493611496774878387754868289247004968965713980508241917234475778967209922561117319659136145363455975530730427561621837535414109969617015252876273274461817006082206675864548099634543492121154484646533760460328497741795958006077857004574398660975871741055516608620338782772370415761347760939210434344105682460380120505099126996615114944291985543585557633944902132740238508060814779284746633322484038677656603613841977357271474071500046367306080742491700183648163748816811 9
- **Find the q value by finding gcd(c_q, n):**
  5506591942606409786007683694164320552666853760563997278112673956252642062580230341282514609867273797940285064713708705697045430216001808829496025459965762277595065815015312928959182173836874784924700204423500487879545871499053277509540470387842362515320251346826624958017594645145534876296220713938374221682083338460760448977511677765195762206864734020500176548242446545289154236269695091850232180279774486799056250625062409745209690007982157634073337918524689046502794950964394031916498102270905968246681208683669797493641971374341341036682069179230118965760408474627770213910582526956588655421840245359419231811191941978620018710013661856005725155989723730148659469997175814878286107665317557624508522339262171444938321292609872613647002641849742438288664089486939747792445254182820234942822660616550075433187549789 92

019046633414923988970537763104184380773629868368290337444206944
5140251407366732295733356931580544159902241

- **Find the phi(n) value by finding (p-1)*(q-1):**
  191836115941107040479442680984096797477776935335749143420132758
  886241163425779793123034036214410949284908486087093500392275431
  006666363368452045717449925141244415163916556097303659155135064
  640066981760746153509294679951469546358088997900865060552213169
  883213496060907452641950467077371297454925871321926608652980132
  070040168176512551025780896662617392053348880495866486863856323
  591225962712524437746354453515200460749082471888170664475256136
  847633930563371957552223896641863948549514626268176988072161507
  014809475669283317150375207470566218823979868910212099748506479
  957592158602667131927181846125509231614343194096131658354647309
  923744975540937318040891829160578013633310962339541436185509414
  033519804684462724498670420197435146296068950522029836309103218
  321887421041641771974865505022657856438960391003887024801890732
  463761889931885281594686399919481951264510968850044652945966240
  261843023676562352481679947156428454459473734529523572555713209
  224048389161469068907387666388302823591563379082314230077603618
  583258596975808694489831561383723885722159099892668367787437532
  976875763702453475428966230133995639299400100795654857422926192
  855523634247602883583472981135286937086336349725211922498403068
  606367272732234658494602652156012126204659743263206218751424431
  581427812881393123372589274707123193940884320571413376748507255
  469981363570153986797856010432462253273038488277240887487216046
  071064806464129037000794940780997702529179903536757281174095506
  675988424988998980738273173389615105791590333670944726390599625
  539256130577014558364601650757551768244600185744892750666368364
  918662335031094458521293319936139051630535554747255326973140989
  056085501919040467956378069552273464438288037724480514078296682
  311643288194514840824238083499491997967468737755030342820404052
  386539499018229543478907277521888034862544611867012721331671333
  19984483122247564784320
- **Find the private key by finding e^{-1} mod phi(n):**
  189670030740435370888909278723873845097693090604552195648173434
  511066552733574213879079549637383253441784261912894968413541527
  731494580880809118610153086033495801903906206958465984135301994
  041265547967576613347296448980200073170347937760079853038768280
  039101345869640359016135380246462730384162102080456512517923518
  283448933843988644106650056625228789124934730144661195192292875
  776136666400406579377306399811166886722893890946149420113877228
  120239495227349630491347663620593836674899663370295623786744116
  606475877671278073475942785273513881946647291846972144397881721
  497964494880403774165518746408816724612273920805759236254422413
  920821874362972296520983076035185827080849755507808817011954369
  304819915225584496686457225956836675077381933670689340423287017
  068271956623514379642878986358136031915945442358955514382533741

11500952411487206603126317432263716673156411261693918514640240582032504394723912833598449022529402571998290395211161528438208673101703575836109766989637333052867937596368882305743574375515622227760361691069131723477856754438901900824260715675461987258837454596427404180611518264207993218242646487222875700438581791230082800446071568804733660006749145768624344192051142014631465639999969744160717100988400524517627815864835717573238562088665781439153441987930810341741662263243050166289760083422453229511013150407274143259353557037868673049766289386033691468615452550677124046362293314021912064601349326534298595006593621234169807202166748514598115499485613952617437493290322796094123069306931864545492611805258614753640179039446308108943399471772532964082795420240967712631992316162247130631747720848285636973651339879255790122019390672753054655801728281238396249417021169523675771127374689328379355388388475359745766501820455315795201372863758460855089366135495565562590386477530242440610975703531293968761733601393912938881171393

- **Find the message value by finding $c_m^{private\_key}$ mod n:** 638430993074629687195837936481159006693552285988028013368030841243284972629648604448109734668091466074510200197045977080275481509858813617320605969234825499359653656530495654596514919349064617114149002930871172405935648860559671540513381615320335361620724965447066373540766149046617848968335576132622105768790894594492599420083997078646795696149541761157224739200181378368968266501730694986378199803088188901855697322097918673080384564212532124778222986091542091264548448179852908954422806679427419346552561737
- **As the final step, decode the message and reverse it:** I am free. Every single thing that I've done I decided to do. My actions are governed by nothing but my own free will. Do you wanna know what I hate more than everything else in this world? Anyone who isn't free.

3. (**15 pts**) Consider the combining function given in the following table, that is used to combine the outputs of four **maximum-length** LFSR sequences:

$F(x_1, x_2, x_3, x_4) = x_4 \oplus x_1x_4 \oplus x_1x_2x_4 \oplus x_1x_2x_3x_4$.

Analyze the function F in terms of three criteria:

- Nonlinearity degree
- Balance
- Correlation

Is this a good combining function? Explain your answer.

Short answer: No, it is not.

Reminder: No code file exists for this question since it is not required.

- **Nonlinearity degree:** 4 (due to $x_1x_2x_3x_4$).

- **Balance & Correlation:**
  - To find the balance and correlation, start with finding the truth table of the $F(x_1, x_2, x_3, x_4)$.

X_1	X_2	X_3	X_4	$F(x_1, x_2, x_3, x_4)$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

  - **Balance:** The # of 0s should be equal to the # of 1s in the output sequence. In our case: # of 0s = 11, # of 1s = 5. So, the combiner function is not balanced.
  - **Correlation:** Analyze the x_4 input of the combiner function. 13 out of 16 values of the truth table matches with the output of the function. Hence, the correlation of $P(x_4 = F(x_1, x_2, x_3, x_4)) = 13/16$, which is very high.
- **Result:** Since the function is not balanced and the correlation of x_4 and the output of the function is high, it is not a good combiner function.

4. (**15 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 = 9244432371785620259
C = 655985469758642450
e = 2^16+1

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

- You are not allowed to use external tools (including online tools).

Result:

- **Find the p and q values s.t. p*q = n:**
  p = 2485770689, q = 3718940131.
- **To find d = e^{-1} mod phi(n), first find phi(n):**
  phi(n) = (p-1) * (q-1) = 9244432365580909440

- **Now find the d value:**
  d = 4032669742276769153.
- **As a final step, find m = c^d mod n and decode it:**
  m = 71933981384993,
  The message is: Aloha!

5. (**15 pts**) Consider GF(2^8) used in AES with the irreducible polynomial p(x) = $x^8+x^7+x^6+x+1$. 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 '111000011'. You can use the Python code "**client.py**" given in the assignment package to communicate with the server.

a. (**7.5 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.

Result:

- a(x) = 01101101, b(x) = 00011001
- c(x) = $(1 + x^2 + x^3 + x^5 + x^6) * (1 + x^3 + x^4) \bmod (x^8+x^7+x^6+x+1)$ = 10011100 = $x^2 + x^3 + x^4 + x^7$.
- Received the "Congrats" message.

b. (**7.5 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.

Result:

- $a^{-1}(x)$ = 01101110 = $x + x^2 + x^3 + x^5 + x^6$.
- Received the "Congrats" message.

6. (**20 pts**) We want to perform modular multiplication for the three values given below (i.e., $a_i \times b_i \bmod q_i = r_i$).

$$a_1 = 2700926558$$
$$b_1 = 967358719$$
$$q_1 = 3736942861$$

$$a_2 = 1759062776$$
$$b_2 = 1106845162$$
$$q_2 = 3105999989$$

$$a_3 = 2333074535$$
$$b_3 = 2468838480$$
$$q_3 = 2681377229$$

However, instead of performing three different modular multiplications to calculate the results r_1, r_2, r_3 for these values, we want to perform only one multiplication operation modulo Q where $Q = \prod_{i=1}^{3} q_i$ and get a result R. Utilizing the Chinese Remainder Theorem techniques discussed in the lectures, show that you can reconstruct the results of the three operations (r_1, r_2, r_3) from R.