**Homework #3**

Due date: 23 November 2022

**Notes:**

* For Question 5, you can use a Python module for arithmetic in GF(28).
* You are expected to submit your answer document as well as the Python codes you used.
* 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. **(20 pts)** You are in a job interview, and you were given the following RSA parameters:

**N**= 14160376831985083549234691952615806088754482769313863972472612104960426897223528465771355471579374406630645418210046457006222595114565610230771399642073008939064571341163391040192895053667746239198745313836608357861688352973345896714893493445912599660303023771086463444245877244662588475490852139709576652071347040782452020420571773050781132633528534621842646272347587344930762951115906409730066558110915658602904086152356440870165207911781076319159782794658036524692436267278201087720505491720755129001866266643708208179209558133605991939687875289929389527569467618218808022773618095749432994955099002138061774558092646935053922768729867596344560354854024333693063848315568843274016677634235943265653870168403483754828778660331273571374962669367723145859407420437087211873179386669565722950274706684709564825917176846099470175848587712359262011453691289856193406245598805109425265282471927598683177731795099636499428783962103265346384863380645563783957116490551095371939941953588841421901208018208833231980963252778549870359034986607401895073749621109530739274084626457126786909062587418357842878093562018076560481599295590002481431272398795041334444497078057214818284076399718097402714878381631253086537958945383846515538826876788531288766825879748890645117613533585994704646456047742249832303552280560598950442935275040981664802396857172617709467797511943183885432902060866416575711977772297223391851636143412166880420086696224346267661926864577871074351112573238283297717693120212428566507110945526676138784819788038239839295288788351687768791008155514660231443432484340203637725314815951740765116548061295612618198058823576843639508119932329130553908867479055715564386028123392718092251110077019826654081948597242333738564418061676295146137851144050784468238754208055080766136851080662924281661590293015860041945804754733697510953608299395441403

**C**= 6964000134802147490350421365650336249614250610865941864507560530973065829363013742586720564552761485372116221740627172129123876523265935559212398148078283338483855884374390808303815665061687242251596246200216973015491595396418311911498012690876633098536884830950045033940873472961361944443087691593107828862821646570032433642983137001852284845829650930647062230124067089431359423080345777147816524499368041933760389488812347842387454340351987432497450711320078300294448658548685160053953562838653141801302708015723513455630185668900982623299071334073972206896315094654233704852582483630519657086758386996034282108495983647800489173557906086552456716216042967828046145491119906240052603113568412111123995516281289156205173721081926194326415619996714900950517971362831818945544753717887534879164670744808319717216154781106605434942135917537990413956682184006665259542413721048288165850192732549204068618353559752841030622793211811546498498416331840835131156082144938676253315415796019426308810557107725216135397364965385028974827442762669961840515952862589707634652185583315205043783377926870982096766350787995676580790658222090464434721501852708698323447101795557300364436799578803807748130751797841330640652950406091265871346916787635907157418424516138771191157069377784992022861611028548265664755868362046368995908027874449613065707583578153986732928925323015151853338913936042198087258685083559185030411159644723900086245429733121433845958021799786535702133974611668238336

**e** = 2^4+1

You are asked to retrieve the plaintext “M” using only these given parameters, the plaintext is a 289-bit number and M << N (which means that M is too small than N).

Show your work.

**A:** q1.py shows the solution of this question. To solve the question, we would need decryption key normally, however, since M << N (too small than N), we will assume that M17 does not exceed N. Therefore, We will directly take 17th root of C and find M.

A:

497323236409785261806112890034063353149283695392257227218669542815848464507688615673856

1. (**25 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:

cp = (kp)e mod n and cq = (kq)e mod n,

where:

n = 735129733350600300814682820983363975841858401155438497255664794289823750303771869563783060904740656108396832861631223213759233199110153841037374050058977286914202841273524024666932250131821113376668815406723621016490979772825337454765842227937701790973178644581429373241485573097140173355071014592912476993973876358900848501647208271118220494126859489227794428441209121779400263427064528800044153594488608641089810344852811969047683323752216047087315762593265636128466350649798266807305401826019476523383574373865654916089798354741168921810503521914845585839161180821237530439632455643648718893847945602507596343025039916399868955525098511955168845334399609780438491582190893008973851423386317328091461841967101793387569401290989098810214189082780241178398539863285659421612567764560272257730862326168958364031351936876932865703535175597367552038394861003656869677510780585595075450570568618061324321406580734938998169197647023586630354046716320331115546825940635140424868018161440043319003351900977331129699978699948137567053016132949764669941675005253460547841482625174892031076638216262945295769084071268327053819352242156952428409005165677559548048829999259603172407376651982303486619116704252671413990026284868294201399181512043

e = 65537

* 1. (**10 pts**) Explain why this is not secure as anyone who obtains cp or cq can factor n.

Cp = (k\*p)e + n \* y = (k\*p)e + p\* q \* y

Then p divides cp and p divides n. Therefore, we can find p from this logic, since we know p divides cp and q divides cq, and both divides n. We can find egcd(cp,n) to find p.

* 1. (**10 pts**) Factor *n* assuming   
     cp = 591585661179913908292670964497762439966067617632980318594933696993454439728774695673398116279819100780981035776904828340000286556561821690365937795767595578460020372779099849512212679341203336712606691026980784826758612415894186878559418781933556513172143678225888892413399407357572102187313811291121304928928848821740914239643286098598943861830177312939795889893467307582185899931290683200944769576636703683519903565708396786917226998954212438208550716658336081133840312436400158765503544052687056167181841283624622810770703450396047554977193850903901459806510612970231465678503924692665502872536500180880428444652182260677241127037607744405052492742074626162505690986835662409717998630436892609085891138004339602274790886677062920378513594058185353516270486167595057731308599477294190461449818277150803142015005367485865449897543408948648598768730426058899226212982674326317239306732825540115470519539729209054092010428475270982133241739755670905018636775489391024836583829979747436647847343118308007690487133027090643538988141634837671142574166059246803646505989258216228280398544311438792645908159200659488745926355055587928710053702116250352855728835609501615125001498450195370742144016227930442188681846248774022591908284395554

cq = 309411729251034251013395318719711706160811892862394890384299153591046151499917121473269348428595368341405902181095462992704329677547559777979434641542836057927150208356368122522229704600457038948359531454800468151991865398538394737136726928964793436886497125779317949812565735742165451557954880576255424560841686766527870866834024733826068596435927601933868356893971916253250737903827232908056159513671751051011444556858915046438167636879877207093812917634178160773097343436768913865449127177896443942872076146021042760369486915412609972874901144604578896888158193243455010603320402839406279239012820373216932302406828263523610721896771030364051463816982602691512661678520212734151948259164880771727519256923996762844291396810356605766709626469613406423888478179326659617558931774201308469110659393420333748745481493594465373505620910145316327984388627313690258503451934907257473375836215663070727820311136494801887688477512058462477531334799901621825205074964487431978969359431368816027376768643419649233036868068405017555516631026700926970709260767677416518151837286394000000084320497258684909083576218584531666178186869180867507209104151471590432167256468837661447181958281610404744566801864938311314315544802024261994226882434417

and decrypt the following ciphertext

cm = 410549325685366799284798358668331291049211237703447489321746464354090480951066742200309912524869634695769764148842963115769175527179769353501136785040366302701836605559269222653839509896223034315989474752525422032311356077141993653194048634554747064603617789881601934904375748720866525993835837380575119329831703593942313434602973137707915466599022298281760026130881260276637144571320608488322550304670024203670226128476075072338279764675774256431888736850559740563151465682499215963629918224101757068211315022934434916282345252224165558403932898552995551505591198395331625457165173901358070740812251352212374898226752191458989544999183050305396079595170576363831967410596235893058231852520802407515041647925479992058184613103289709752618729186509661786057001647083804585301703801272797236111687348907977621232083642431914363895266519463459112358271172160273751026080398345460929751476512586349360905588246751625224080678692961596943056437462453579785432258654674275520068227719175943507656108554707552928863240951458193510571735005767226590398204791730466993980158501690130104327829589781448607958503898902133445000532521723138230966914652700465728532524467483612403840506027543892612439531633794667013713638856396514285792898762678

**A:** We find p and q from logic above. Then, phi(n) = (p-1) \* (q-1). After finding phi, we can find d such that d = e^-1 mod(phi(n)). Then, ptext = cm^d mod(n) :

245478921987395749920533611168546289439886253096741178542478990993042724129658131888054551898212305170185993107821969409131036912411996765029383999031362442343139947199428706920360327036588114050940006175150257669311105571047025434854405189809998459978053542620246340174367407634520393456142088901495721858291489481826980674745399239537730420385058955895154527547682777055940055537057851908997477448911471300665288980461871251088202072422801565609087805457126289069586423729197715529274493284394414778196769596004120962017347581897777994018266065165507553177495724674778993974994141411745689634954009043495964390483273864733005277288435533935494133796440199264309265068575922460911566036541493642999623513728969291323967309789741605571045421329086002588913420865502539964140905265130326764056376576833640803742257557614840718987665592036730804512269630519250702045188204869033169566916708951131410435941953757978697

1. (**20 pts**) Consider the combining function given in the following table, that is used to combine the outputs of four **maximum-length** LFSR sequences:

F(x1, x2, x3, x4) = x1⊕x1x2 ⊕ x2x3 ⊕ x2x3x4 ⊕x1x2x3x4 .

* 1. (**5 pts**) The lengths of LFSRs are 60, 95, and 97, and 75 respectively. Compute the linear complexity and the period of the output sequence.

// we know that expected linear complexity is E(n) = n/2 + 2/9 . then linear complexity is calculated like:

E(F) = 60/2 + 2/9 = 30,2222222222= Linear Complexity

Maximum frequency LFSR is 60. Then, max period is 2^n -1:

Period = 260 - 1

* 1. (**15 pts**) Analyze the function F in terms of three criteria:
* Nonlinearity degree -> 4
* Balance 🡪 7 1s 9 0s (not fully balanced)
* Correlation 🡪

P(y = 0 | x1 = 0) = 7/8 && P(y = 1 | x1 = 1) = 6/8

Is this a good combining function? Explain your answer.

**Answer:**

We calculate nonlinearity degree 4, which is acceptable (x1\*x2\*x3\*x4). To find balance, we write all input-output combinations, and count 1 and 0s. When I did that I found 7 bits of 1, 9 bits of 0. We see that output is not balanced.

Also, again from input/output bits, I saw that output is highly correlated with x. Whenever x1 is equal to 0, y is 0 with 7/8 probability, and whenever x1 is equal to 1, output is equal to 1 with 6/8 probability.

Then it is not a good combining function.

1. (**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 = 15220196297956469159

C = 6092243189299681137

e = 2^16+1

(RSA Encryption: me mod N | Decryption: Cd 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).

**Answer:** N is too small. We can factorize N into 2 primes easily. Then with that factorization we can calculate phi. From phi value we can calculate d (d = e^-1 mod(phi(n))). Using that d,

Ptext = C^d mod(n)

1. (**20 pts**) Consider GF(28) used in AES with the irreducible polynomial p(x) = x8+x4+x3+x2+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(28). Polynomials are expressed as bit strings of their coefficients. For example, p(x) is expressed as '100011111'. You can use the Python code “**client.py**” given in the assignment package to communicate with the server.
   1. (**10 pts**) You are expected to perform c(x) = a(x)×b(x) in GF(28) and return c(x) as bit string using *check\_mult()* function.

Here, I used library called galois. Galois is a library that works with numpy arrays and converts them into Galois field with specified irreducible polynomial. Trivially, I can multiply 2 polynomials with this field.

* 1. (**10 pts**) You are expected to compute the multiplicative inverse of a(x) in GF(28) and return a-1(x) using *check\_inv()* function.

Multiplicative inverse is a(x)\*a-1(x) = 1. With respect to that, I calculated results in q5.py