**Homework #3**

Due date: 24 November 2023

**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.
* 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**= 22365785976614402332397512075124630827034962474834967508376992591878950349761466248996813628790324021741967645036142969442187151184395770097021823575949844361483331809293176534721694071334635996827951807532916795873629655454372863524613892925120878044667556713089187256277620400880942674492154919463659077088656955568365073946801041136455049154254694093386688718461966571180917691178001778881641370178356291899508110775449397873342307750859548327215761131604445299914396581941939357225594068024701563019774466034568844933299858477997754435831659757699431498632233228458041004651360032989248646612270310837431642234085323728177484894131361583681425555600156607217562397349329759129433534908756960805661036304391089345834509233838154349022133477003266336188367153439809888243394989544021054933138433327777476435190529104778129976652617046362887677861895816432682335827401878705944583782205741084481506390559086053735656140311253871289251353329169597363008994798852788633399111345050043699878172663514940100063654061453571115942641082817941575982943955570402403753758191438493616361452856840835486794570579705182218479249220132822428778059375078515905168433395933753278424300031706960688760687974376182140330967708327782851494424305411900910050471219968859397556822151330672315688400023275713385475636673375705418460899832831371793346707182198279244735326858616427948735041188325318283440302015774999829152258322963280493356950987282709489501694094388641232109406904291206718465429145244727160712334951148817109453889551067078695519930282020132174315707008087811410286418367002171219543528519457146211423711288808552599240627367625161851031476295911309840343787587200005529804876404288936962430037222588451493743068203434317795857684578223380732309464862655712412606898281225998993015418109727694593769147333845884375456846565153850875807335847519323533

**C**= 10996907317744048201180239191184801878870026359954251163808786991823988392040721694956774540613511870672915186967674497345695382645430111824083800355570536408550748237322495556640595897950904266978255920300494206063326443644038077060867859678853573135463689576016437189518186799433888575401030210875676845244656583911134078393876421877639911657047543070395464987842170462799343767525393215191609774073190285806734017390270810257697704746005511384590358282243168705497528675364827201498077988713907302696070779021284913585400756722751893643306497629378643206323310558189248071831717808359955530448986481886749316175356424375727873597173466277727860357595136

**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.

1. (**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:

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

where:

n = 19183611594110704047944268098409679747777693533574914342013275888624116342577979312303403621441094928490848608709350039227543100666636336845204571744992514124441516391655609730365915513506464006698176074615350929467995146954635808899790086506055221316988321349606090745264195046707737129745492587132192660865298013207004016817651255102578089666261739205334888049586648686385632359122596271252443774635445351520046074908247188817066447525613684763393056337195755222389664186394854951462626817698807216150701480947566928331715037520747056621882397986891021209974850647995759215860266713192718184612550923161434319409613165835464730992374497554093731804089182916057801363331096233954143618550941403351980468446272449867042019743514629606895052202983630910321832188742104164177197486550502265785643896039100388702480189073246376188993188528159468639991948195126451096885004465294596624026184302367656235248167994715642845445947382443298279466036094062664526089714740235073357979345472146736661300992184392989418651734388376148883533458372577458513398297951544579926434907745133866160543325682099182609661124258828686229066139373979701887434973725909485939015439750847449460059621874411914962879708981241452775671541600685206318123201554033565724593074932180907240424055620746263455607699700524768075354406018616956119182772903779733007050892970918375262173527324963190853149207080829691229768577024381254846414888462230060333360969753981531742672692864738198795013133171504460182017682578291909277355683737362686615312733563660679559873862037089763914089959724933586052283906838946927568255053893267264726932678582874115099708992772447131436436750259312237623566898287206469193225558524686538792360673104755313300203837226598476214001721836367864196351049976579073142410364805784053538650225549854865750643064621327416352560706853970343251184429212854679

e = 65537

* 1. (**10 pts**) Explain why this is not secure as anyone who obtains cp or cq can factor n.
  2. (**10 pts**) Factor *n* assuming   
     cp = 8519815578170286631805017896096474075672015420898414556273273563985458286184873008881921592838421126195389275500629440392317613244976879803880572861175645171372132930198686857826398697029296085216195922685352294920280296348205151892012561211443775162106469248654323057118315250089889475157971264106110367384602631230548670220959375735492403409252540919166385343902884114325393698888275534761535196788587204505656921763154126752917718976113943193681080299116409595913501504214889062429344155149836301079189039462619954069866864300383130438902736794538346904238432868975581147072505131651378774705965865731237078126655925924035678349321322781353563803379362971007794303541671870839974945950946265910164957264634229330614810135042947689796774723711319309768613355975228417346952743120406221452690575930282524604518355078005453854245022043845149981891881585415253738202097947376656462337607043966891890131056917721776883706352317130697765259340394647351634262537742299035646057248436280619850531098414524581582352458784937379465227008036680314868330886413752034529490891511794183887067692551782938285454471522148053001428240953906341516262344383447084866563138602244401494467222217732776014148262257280902332570307075788150376152381719388497108188006025400592125280457099934618901636515517680454055134676951026820201726522446804147199465378111686381906611351788755244173738371886076296501484373607288892622408231281948323870298147114456042267714829324086693319828834410900838799226104876952214962820209745151704822379136237027637532816832610621445002982168456512939525633911671133834755804750045792900310469183134697466636763872767179461856691128995322469285487447938786325053015482489530254757378317395587115328986426716585076998064210883258828464744941273484010073386037552987182465771854300536973027299524764873622012478186053079729537911008670556589

cq = 4818402986643636044952843129829705922725988299097705030918401041334297083874026699605895648827453267994332533399138283856617689029155909510471631609529946820868602375198739386786406846836754762533402986116444514710679707358709981719722549267888745464990079129605922130466788502712350019045693756703593988801870613468796295430667144620751428128046933779816615417959079412978159775942711330666677091410274452978575021635293941381834042371441696635271331747735361591379450221779348906358615192766307464050217851000851994814322352867287108386800381833255808230731001048892609128356893140109922692242821185324726300631690328094860431041709751229723361506488850743369797242734381079554332908447045547746011243992737546342642004348059824717163467515644745431541431680101478414036782232007805631224217006523145740523735438836829882626260555654065145655447854999209378941417616293293052622327491604657357830731481326205872366301882906264185994389510253783902342485386035993058909964912853656069503609841550801039777279433919385169430540664440682407779148460801863897782867599085439168649543505221201374818705677877180677145058969601500743528459902085775462273326832206140113781536972043244672281917468383254611091015005903385112922747544403166025362008170737201202879204504122648202618868967268504140121350365696859684528628654015833419458215786941320857070238961978652143567009522826080508445633746997082683983475060618148916418733378183967235129654286827033428934930645381942997863219580565220549111654622290384293803019784545621494419819730053598136642969446383600057039223038962390275264104774885552446624027720708075084848381698416558349678492101036072681283403310668416605477624990247387080785213971109071457529424168678618752082651402553039227849809519834171274681952974948337258568018334669863065908317748964814977376118579837622866095503768197661413

and decrypt the following ciphertext

cm = 17284828821141668560328235493236055797130405967076368219938343596386850014870638653469721063839006553022826713358194804149890450517899381470924674581949241837668935875352291837652323401412494981071623779488058069497179097776364868943601713264310133247124854110895926582834668383284408145815554576567507012465018673903298242358712728145577858763478112479002771259493525807684657426451338653128737611794860665710268639303858624800297914049294769357313113900731438147478193840830771683396325180071897978057144353732540850804724009742743680682045266501920818553230592013929155249589476398933543983581771974095726573210757002554510004235520302513149899914894529189297889809383301275483550694185906411983300954401141905110354320679159622436998422310074549534944363564709280940024994037670933846494908420719812318430547685785574935297182122483740006155482136434239913217377232230604772109858472948104331917878934192223204237054819603573569434488747579355272284867036663679603989857901048809866201540104373912926436207972962525898982696337663285502146875206929803809235650675196379747112768154855772616397195918330804839854158339656271985020071689412892161684329179223122477077129618885960811703353618099020304836536057198405942993484954893002367710743252251053234503006372540090946832274791418301867915760955301022296212036116889824594510484294302530648091592764384194059557657634578441292718787344214161041914046895530668319805950003190961769664619321169582429602273431341391235679884402935529079174375702293835414826297820794808626639780432503983297222195083887049990892829066647548585755314625148893134824916462550440313481197336026618655866330917916955021906918184886059678073027273180828424832437013409216600308155541246477199951176452082733680373593356519067379860384739026869779670494459086425126946501620976059825121468310171155789203387118965679588

1. (**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(x1, x2, x3, x4) = x4⊕x1x4 ⊕ x1x2x4 ⊕x1x2x3x4.

Analyze the function F in terms of three criteria:

* Nonlinearity degree
* Balance
* Correlation

Is this a good combining function? Explain your answer.

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 = 9244432371785620259

C = 655985469758642450

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).

1. (**15 pts**) Consider GF(28) used in AES with the irreducible polynomial p(x) = x8+x7+x6+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(28). 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.
   1. (**7.5 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.
   2. (**7.5 pts**) You are expected to compute the multiplicative inverse of a(x) in GF(28) and return a-1(x) using *check\_inv()* function.
2. (**20 pts**) We want to perform modular multiplication for the three values given below (i.e., ).

However, instead of performing three different modular multiplications to calculate the results for these values, we want to perform only one multiplication operation modulo where and get a result . Utilizing the Chinese Remainder Theorem techniques discussed in the lectures, show that you can reconstruct the results of the three operations () from .