**Q1)** Since we know the length of the message (129-bits) we can implement a brute force attack to find the cipher text starting from the shortest message. Q1.py do this for us.

ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Luckly, the first message we try is the actual cipher text. **So the answer is: 2^128**

**Q2)**

1. If we know cp and cq, since n is public (known by everybody) we can easily get p and q only by checking gcd (cp,n) and gcd(cq,n).

By calculating these two we can easily find the primes and the remining part is just classical RSA decryption. Calculate , get the inverse modulus and calculate the power to obtain the message.

1. Following code exactly does the what I mentioned above:

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Answers:**

**Decryption key = 18967003074043537088890927872387384509769309060455219564817343451106655273357421387907954963738325344178426191289496841354152773149458088080911861015308603349580190390620695846598413530199404126554796757661334729644898020007317034793776007985303876828003910134586964035901613538024646273038416210208045651251792351828344893384398864410665005662522878912493473014466119519229287577613666640040657937730639981116688672289389094614942011387722812023949522734963049134766362059383667489966337029562378674411660647587767127807347594278527351388194664729184697214439788172149796449488040377416551874640881672461227392080575923625442241392082187436297229652098307603518582708084975550780881701195436930481991522558449668645722595683667507738193367068934042328701706827195662351437964287898635813603191594544235895551438253374111500952411487206603126317432263716673156411261693918514640240582032504394723912833598449022529402571998290395211161528438208673101703575836109766989637333052867937596368882305743574375515622227760361691069131723477856754438901900824260715675461987258837454596427404180611518264207993218242646487222875700438581791230082800446071568804733660006749145768624344192051142014631465639999969744160717100988400524517627815864835717573238562088665781439153441987930810341741662263243050166289760083422453229511013150407274143259353557037868673049766289386033691468615452550677124046362293314021912064601349326534298595006593621234169807202166748514598115499485613952617437493290322796094123069306931864545492611805258614753640179039446308108943399471772532964082795420240967712631992316162247130631747720848285636973651339879255790122019390672753054655801728281238396249417021169523675771127374689328379355388388475359745766501820455315795201372863758460855089366135495565562590386477530242440610975703531293968761733601393912938881171393**

**Message = 638430993074629687195837936481159006693552285988028013368030841243284972629648604448109734668091466074510200197045977080275481509858813617320605969234825499359653656530495654596514919349064617114149002930871172405935648860559671540513381615320335361620724965447066373540766149046617848968335576132622105768790894594492599420083997078646795696149541761157224739200181378368968266501730694986378199803088188901855697322097918673080384564212532124778222986091542091264548448179852908954422806679427419346552561737**

**Q3)**

Let’s simulate the F(x1,x2,x3,x4) by using following python program. (Q3.py)

**metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu**

The resulting table looks like that:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

By using this table, we can analyze the function.

**Nonlinearity degree:** The highest degree term is 4. So it is 4.

**Balance:** The result is not balanced, function is more likely to produce 0 than 1. That’s why it is not well balanced.

**Correlation:** There is a strong correlation between x4 and F. Outputs are almost identical to input value of x4.

**That is not a good combining function** due to the 2 reasons. **Firstly,** it does not produce balanced outputs. P(F = 0) = 11/16 and that is far away from 1 / 2. **Second reason is** output is highly correlated with x4, that makes correlation attacks possible.

**Q4)**

* To find the decryption key, we need to find phi(N). Since N is multiplication of 2 prime numbers, it is (p-1) \* (q -1).
* My strategy to find the primes is to check numbers in form of 6k + 1 or 6k – 1. That’s because every prime number larger than 3 give us reminder 1 or 5. (That is very simple to prove. In mod 6 there are 6 different possible values which are 0,1,2,3,4,5. If the reminder were 0,2 or 4 that means the number is even. Also if the reminder is 3 that means the number is divisible by 3. That’s why only possible reminders are 1 and 5)
* Thanks to that number theory trick, we can search 6x times faster than looking for all the numbers. Also, it is more efficient than checking the odd number 33% percent since we always pass the 3-divisible numbers.
* The second trick is to look only for the numbers smaller than the square root of the numbers.
* Since we expect large primes, we will start to search the numbers from top to down.
* The implementation below (Q4.py) does that for us.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Answer:**

**########################**

**Number 1 is: 2485770689**

**Number 2 is: 3718940131**

**########################**

**Decryption key is: 4032669742276769153**

**Q5)**

**metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

I used galois library of python (Q5.py), basically turns the binary strings into field elements and turns the operations into field operations. I obtained the two binary polynomials which are correct:

**Answer:**

**a \* b = 00110010**

**a ^-1 = 11001101**

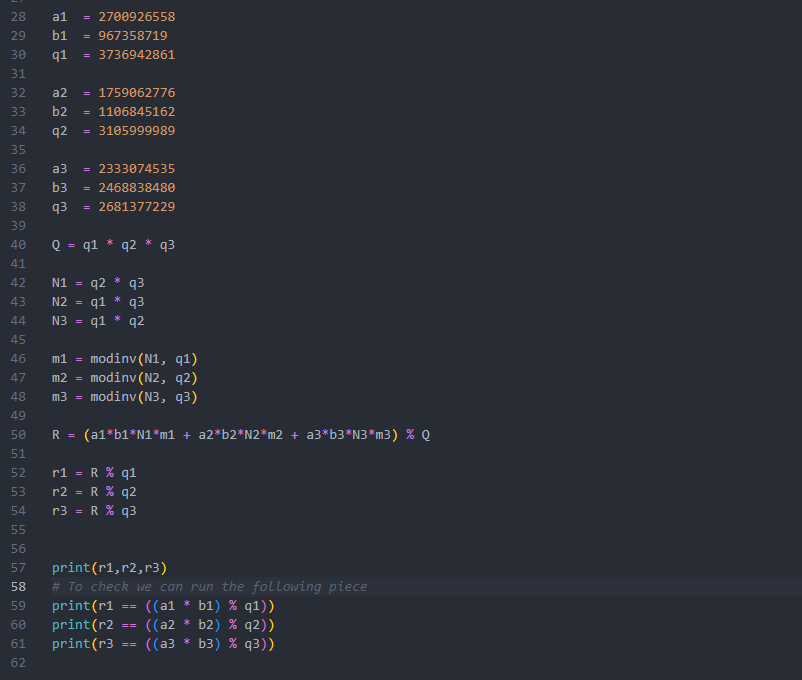
**Q6)**

By using Gauss algorithm we can built a useful R. The principle behind that is as follows:

where is the individual modulus and . Every term will hold the following equations:

So, whenever we want to extract a specific from R, all we need to calculate the of the R.

Following python program written based on this principle.



Answer:

**R** = 17531516279242048504396112056

**R1** = 1643182479 **R2** = 363289399 **R3** = 2376063578