Homework #2

Due date: 20/10/2017

Notes:

- Computer programs and other soft material must be submitted through sucourse.
- Winzip your programs and add a readme.txt document to explain the programs and how to use them.
- Name your winzip file as "cs411_507_hw02_yourname.zip"

1. (**20 pts**) Consider the following modulus:

n=93923989829536838645425792878391884185852090015047096928325495680735077974573254200941142654635659889069058397106827931666572379715390977199861730622956934976456672365327932404671265952251676873946441183498231232238528978514016592480741429133646590466736838628474537927484115146306435052269979805285188280829

Consider also the following number:

y=56944778095026064508803320557324194828217604358002584860993983127246257172572625621288924208876440674553832158479040031724187898214806257072823966168352349606611537362177018477860328420295741255602485512749523176875861988864327723267007959112818744831884377494800464304976689851182957989115066715277845807162

The integer y is a quadratic residue modulus n, and its square roots are known:

{
11539293044780349664317193739835099161008802876501927386196632728052884864570713532203427639431106901813684612368583348366278480903942058202903897004649103055743285751895970690426609694759832511976035062765586527404034628524567695175048634551582051994897593554954834163440853023500476963332472457537871514030,

65462078929074573896561682473134429960343753900612807859205872292206976673766764795638688642515250572910065281537469802520465145301193070320295080324024966677495122426689606204830245523869232443939539387029183651868429281477875679397889633306037656227303018935833418527282673149878942564121394686954099450002,

284619109004622647488641104052574542255083361144342890691196233885281013008064894053024540121204093161589931155693581291461072344141979068795666502989319682989615499386383261998410204283824444300069017964690475803

70099697036140913082851795827608934239433819692641119400201441996427492488148585118331088830827,

82384696784756488981108599138556785024843287138545169542128862952682193110002540668737715015204552987255373784738244583300293898811448918996957833618307831920713386613431961714244656257491844361970406120732644704834494349989448897305692794582064538471839245073519703764043262122805958088937507347747316766799

}

Factor n.

We can find such couples from x1,x2,x3,x4 such that a, -a, b, -b (mod n). So if we add n to either of them we might get a + (-a+n) which would be equal to n. So we check every option to determine a and -a. Then compute p and q using the properties p = gcd(a-b,n), q = gcd(a+b,n). n = p *q. So we have factorized n.

p=8799871096643187569636362221703137117970705734262560672209772353211410091242059304273666872097572116972793282600477392267607320573024412287806194663768831

q=10673337006648339312925067672751218131836069322217857757055585879216897770517306431254639065397004795737064426817789349478902907246196721814345105935705859

(Q1.py)

2. (**20 pts**) Consider the group Z_{46}^*.

   a. How many elements are there in the group Z_{46}^*? List the elements of Z_{46}^*. (**5 pts**)

      # of elements in Z46*: 22

      Elements in Z46*: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45]

   b. Find all the generators in the group Z_{46}^*. (**5 pts**)

      Generators : [5, 7, 11, 15, 17, 19, 21, 33, 37, 43]

   c. How many elements are there in the group Q_{46}^*? The group Q_{46}^* consists of all residues in Z_{46}^*, which are relatively prime to 46. List the elements of Q_{46}^*. (**5 pts**)

      # of elements in Q46*: 11

      Elements in Q46*: [1, 3, 9, 13, 25, 27, 29, 31, 35, 39, 41]

d. Find the generators in Q_{46}^{*}. (**5 pts**)

# of generators in Q46*: 10

Generators : [3, 9, 13, 25, 27, 29, 31, 35, 39, 41]

(Q2.py)

3. (**30 pts**) Consider the following numbers:

p=
12997110888321186459316436989724265349460712629389513072528028592113499927549315535209275673470089822303733552137264381101263928058260190003283395786606613

q=
12208677561088985820124024567796306546470708824672729385334541873189398794633650052335324309711480763919028490727764288604694220087933148985261608483250761

n=p×q

m=
104508383527831908823036869386648261753406819764116432937577039021334848453871469297253914394880121355608067237189088530241064709118310089323613713098921642285063603825756456138538154719968501891170428872546150841926912027053577225264111903314505091217903294998448178692295392732881494063202944747727969714684

e = 67

a. Compute $c \equiv m^e \bmod n$ (Hint: you can use `pow(m, e, n)` function of Python) . (**5 pts**)

c=112616469767608584811609166425488271062586447252115125537564863193044538537395472094406209420709926485510934377621008687907559465706333197400364258608745396555196216103894265082687542702158480984946698751962784029154848419122472663111880510768289690233009425382958375171148628817677521343074095737592803335878

b. Compute the number d such that $e \times d \equiv 1 \bmod \phi(n)$, compute $m' \equiv c^d \bmod n$, and show that $m = m'$. (**5 pts**)

d=7578628587999150073236717561117463915333666664481088085953355611
110635318481906968530859889446198465071167276595240674198810613678
202348785279699393207600433392749574648026730955909747733959724543
286400914612288252757277678942885970575343295875556471182611301357
9635422399855564886228068362942954287317026923

m'=104508383527831908823036869386648261753406819764116432937577039
021334848453871469297253914394880121355608067237189088530241064709
118310089323613713098921642285063603825756456138538154719968501891
170428872546150841926912027053577225264111903314505091217903294998
448178692295392732881494063202944747727969714684

m=104508383527831908823036869386648261753406819764116432937577039
021334848453871469297253914394880121355608067237189088530241064709
118310089323613713098921642285063603825756456138538154719968501891
170428872546150841926912027053577225264111903314505091217903294998
448178692295392732881494063202944747727969714684

m=m’

c. Compute c^d mod n using Chinese Remainder theorem and p and q. You are allowed to perform exponentiation operation only in mod p and mod q. List also the following values: c_p = c mod p, c_q = c mod q, d_p = d mod (p-1), d_q = d mod(q-1), p^{-1} mod q, q^{-1} mod p, ${c_p}^{d_p}$ mod p, and ${c_q}^{d_q}$ mod q. (**15 pts**)

CRT=104508383527831908823036869386648261753406819764116432937577039
902133484845387146929725391439488012135560806723718908853024106470
911831008932361371309892164228506360382575645613853815471996850189
117042887254615084192691202705357722526411190331450509121790329499
8448178692295392732881494063202944747727969714684

cp:
781151220173998169560118163666934108417635156410906391585072586937
012007777347033782437467383754598357919073480515977646119129506122
4280922787521081025538

cq:
414099223340511187283012780558372789084422184715003136337227384609
480589079542153076732115059295342421606521303114594809556351211213
0675224752020408206822

dp:
329777440449940551952805117649720165583331514477047346616382815023
775371296027409102324905147748547730094731919900738027942517577597
6466120236085498094215

dq:
1093314408455730073443942498608624466848123178328901138985182854315468548773162691253909639675654993783793596184575906442211422694441774535993576879097083

p':
8706114279655132029376590928695201071142108092857187630065448441095659251607223169125680908408448451959138384540652167908663667864283898750424266963666954

q':
3728757937489288621512116061599042787876890623154900962291439413032962675674312472045367736398277861072304371603553616800659716640669275328475542942016972

cp^dp:
1179234833092803522820899107094638250619428836081357445595824826095652209778390289698602070888336486037874421839672513698478948255274319805011279065232950 7

cq^dq:
2185715783289934948031269796497771409995671449447628856583293947520136530429843834953363721243428807803221996057231855934964172118007401489722093005810481

d. Measure the execution times of the exponentiation methods in (b) and (c) using the Python function as follows:

```
import timeit
import time

…

iter = 100
t1 = time.clock()
for i in range(0, iter):
    pow(c, d, n)
t2 = time.clock()
print t2-t1
```

Which one is faster? (Warning: In exponentiation with CRT, you should not include the execution times of values that can be precomputed, such as p^{-1} mod q and q^{-1} mod p)

Exponentiation time in part b: 0.497464098765

Exponentiation time in part c: 0.345819259259

Chinese Remainder Theorem is faster.

(Q3.py)

4. (**15 pts**) Solve the following equations of the form ax ≡ b mod n and find all solutions for x if a solution exists. In case there is no solution, your answer must be "NO SOLUTION", and explain why there is no solution.

   a. n = 1200320707477907914300088804988
      a = 7211941535834517096225500817
      b = 102092299425228521972149597163 (**5 pts**)

      There is exactly one solution. gcd(a,n) = 1
      a': 4308652807136477402944414285
      x: 83912145255473489949903796379

   b. n = 1200320707477907914300088804988
      a = 44575693167043501900449109190
      b = 84664078284205068314514580089 (**5 pts**)

      There is NO solution. Because d doesn't divide b
      Remainder: 1

   c. n = 1200320707477907914300088804988
      a = 404
      b = 2124884389680246530198080982220 (**5 pts**)

      There are exactly 4 solutions. Because gcd(a,n) = 4
      x1 : 29622559878710622345195056909
      x2 : 59630577565658320202697258156
      x3 : 89638595252606018060199459403
      x4 : 119646612939553715917701660650

(Q4.py)

5. (**15 pts**) Consider the following two polynomials over GF(2) :

   $p_1(x) = x^6 + x + 1$
   $p_2(x) = x^6 + x^2 + 1$

a. Are they irreducible over GF(2)? Explain your answer. (**5 pts**)
$p_1(x) = x^6 + x + 1$ is irreducible since it divides x^64 – 1
$p_2(x) = x^6 + x^2 + 1$ is not irreducible since it doesn't divide x^64 – 1

b. Are they primitive over GF(2)? You need to show whether the roots of these polynomials generate all nonzero elements of the field GF(2^6) which has 64 elements. (**10 pts**)