Name: Mary Precious

1) **Public key:**

-----BEGIN RSA PUBLIC KEY-----
MIIBigKCAYEAwThtE01uwKTJnm/myEQwr0hxIc4hUrx3XJemxPRCUsxdZEb9gg+h
q2GcgCFCqX30kIQRBzXluZYD0tIOh40ZK/xDspaogDoKN7u8b9ekflw6GuKg1fQ8
7K5yx7yAd+mxUYRsjXJ12aQyE8ppG7MA2KxXkPVW+awjbvddfJ8Eo1JteM5pteZH
3tkmODw2YB1N7ufNKA3fFZJM5KZY/OtS+WWxv7J5eGH9lFFBSYkfRgG4HjcPqdME
RzhGqDsRTWORaubCj8fTrk2dGNO7sDB4rB90NNgkSI8kIbEjcbfTt0c/WnvKI5Ic
OD5sUEU12Lo+UTwWSreZyNSnZJdpBTf6Omvmejh5SApXFArqtqhd5V1aQYhPUrU/
YeWFtdMyWtha1RGng+qf3KdvDwlpxvP2n3NdXt7LFtqoz2x5fhyOLle+m/c+bFIp
BJrNWfR3W6AsCSVkNBAoz5JjxVaag8agyhTP8kgwSpKsvtyG7kS2KuWEjz63uhLt
VO50dlw2y3c7AgMBAAE=
-----END RSA PUBLIC KEY-----

Private key:

-----BEGIN RSA PRIVATE KEY-----
MIIG4gIBAAKCAYEAwThtE01uwKTJnm/myEQwr0hxIc4hUrx3XJemxPRCUsxdZEb9
gg+hq2GcgCFCqX30kIQRBzXluZYD0tIOh40ZK/xDspaogDoKN7u8b9ekflw6GuKg
1fQ87K5yx7yAd+mxUYRsjXJ12aQyE8ppG7MA2KxXkPVW+awjbvddfJ8Eo1JteM5p
teZH3tkmODw2YB1N7ufNKA3fFZJM5KZY/OtS+WWxv7J5eGH9lFFBSYkfRgG4HjcP
qdMERzhGqDsRTWORaubCj8fTrk2dGNO7sDB4rB90NNgkSI8kIbEjcbfTt0c/WnvK
I5IcOD5sUEU12Lo+UTwWSreZyNSnZJdpBTf6Omvmejh5SApXFArqtqhd5V1aQYhP
UrU/YeWFtdMyWtha1RGng+qf3KdvDwlpxvP2n3NdXt7LFtqoz2x5fhyOLle+m/c+
bFIpBJrNWfR3W6AsCSVkNBAoz5JjxVaag8agyhTP8kgwSpKsvtyG7kS2KuWEjz63
uhLtVO50dlw2y3c7AgMBAAECggGAOwh5UtdqihMOwohf0zn6QeN5JSHsTnHkafav
bidOiCuGMRf7ZhxOgaUOApum2U/xQ6IF7dUKvdvWiTfCVqnGUgj9yfd7vcl9y69h
EC6rRCv7WLF7AENhxdI6oUm3Wmuc0FIMNF8PIJKo9iD2Yg8bqBdnnZNHonp0TPfo
qSWQbjHF3o36ACg+NxjTM4U5+72YTPCnXqB2Uwni0kz8jTF8pVPrrazireQKbKEy
uDvbwTDPRvP9ph4hhXztIYMCc0mC4M5g8hkdn/z2GOBEYBfHhgJQPbUIjGr5csy+
SReajJCaLENXsU7XjKY61g2hu8dGfhVrwFJb4rMaIznPB3JMQ+JVnxSKHDVjYCEg
oOmNZ23PscT4L1GTJII+xMq6A0XJoTLvf35Ho4s31ZGcenme6cfI/VMA9SbCYwB+
WWU/xYDzIPNkkcNeLILF5svo0HH5I43F6UHpdnBk4BLPCmTluU9S0W2ocuq+KsFF
KHxCrlsV2ZEPsY+hkLAE7XhncA7hAoHBAPQy/HOddaKqzulyQzaPXOZjWCvGPb+2
w9l0S80IF8v5MzJNKdTOhCn3OQLvjIfyv/+K4uJXy79FT4xp0kZ+DbXvdAQlLwOv
DCxb92t/Wev1Sb7u3t9DrxZvyfbJh8TYNuXYuGMGEkG+wnrqGSAoQEIDsaJQHLv2
2wBXJThE8q/Ppnsu1oLsFq2EnwzIljhmtaNDXcUzCS0hNQ2frotpx6Zf07JEpoGr
jZe3F0JdHzOwLrav1MayuWeIZYS113u4SwKBwQDKjsbBh500GAulZax7KV3F1tPc
ttciogPFScetM/BJG9FBvwLZMkcx/YM04hXEmeXJMo1A1zeD6laTEwFE07pyrmsa
uOvTQ6hsP2oeoHBeRFc+aCpR5edjuKnLkuS4f3HfsZ5OoO4V0NZm66y0X1odKxsy
hwGcZH5tlYxm1VrksJB9eDW6OMYoE2DFQNowzh9ZVDEfblRrqLE2EBkF4KZ7QpS0

geJUEX24Kg2Fm3oJlXPbgz+i2Y8sA3nSoObAxtECgcBafiJrPaiAFUgD3mHl8A5/
GcVsw1PSnYnKJXyJiDw5TCmatFmSEM2NN54a5mZvYpefvPoC4oIQaUDnSbewg7nU
991IQ/c+KDocnh/75/+MnyMq7B4PImfZqqdvGpODcJQ3bK43k+JkFqq/Hc0dy4y0
aeYDvihHx1y0rZaPWY2NdSe+ckbv0Uk+F+1QsWxpjjY1QBuZniYCYVHlsIzgAN0J
XG14nFQPrWEVHbYy04tifiFxdYsl7skiAruMCE7H/C0CgcBxOE0wb7UXITmhQauf
TAf2RZY2kJy/5v9kj1DlJ1rwAnR841+cN9ZDwwhLzvOL6NngFDmQPLvzKFEr9DuJ
VS+qWoPTc/mdJPxHRUrzw4oLpvd9EoxVKsSjoNyHxZvC5Lmp54YtPRbXatvsu17V
k1azZxzUqVHlMObaKfVIpYkguvCIsWCWrpVUYUB+ATn0fcJYFA9BGlJluu0S4vyZ
pHBqBVVJcDAp/XzgK+FTBQy5vqf70ukHBcRZPbgiW32f04ECgcABmxeFS5W5Lgx3
qgJ8kOD+zfmZ9wqTSPFCeLPOqC1Hk/i1F1iTFyoO4F7Yl0FcVlQXzEh4iMtgkU+P
+LkZwaDfEgCg59cHlTDpIQJh0kTQ80ugqbOWoyWn8T8CtwpmJCpCUURm6QRZ1Qtg
XqsKyWIB9S0ABY1VujW48GArRadZd6jL8VjCRF7InxQv7lVLfGDtOWhuUJULqtws
wefN4XQeAygC/3BKJdnpXeaDKY8RGskjjDkOLZUkCBU9wwR96SA=
-----END RSA PRIVATE KEY-----

Private key

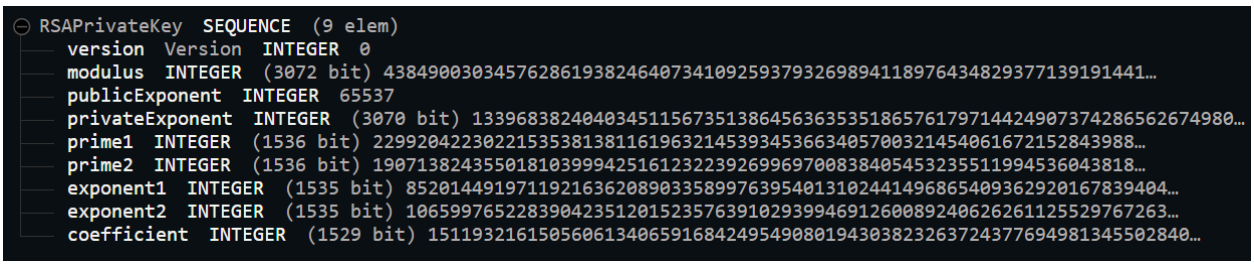
We expect to see a sequence containing:

- Version Version,
- Modulus INTEGER, -- n
- Public exponent INTEGER, -- e
- Private exponent INTEGER, -- d
- Prime1 INTEGER, -- p
- Prime2 INTEGER, -- q
- Exponent1 INTEGER, -- d mod (p-1)
- Exponent2 INTEGER, -- d mod (q-1)
- Coefficient INTEGER, -- (inverse of q) mod p
- Other prime information optional

Decoding RSA Private Key

Using [Lapo Luchini's ASN.1 decoder]

- I copied the private RSA key generated including the human readable text and pasted it in the window of the web app that says ‘private key here’. Then I clicked on the decode button.
- The decoded output looks like:



Names and values of the integers:

- Version: 0
- Moculus (n):
  438490030345762861938246407341092593793269894118976434829377139191441041734397042534046096618178303073689458270359419526614059242006209600217305398774134356060385751534761113214089858417594504862573274918352194546955830259767312788703659150914979086571654713556559967145732463963641903733706550936094505289564091025014080285482456880332123453060153087229082900058457623995820202411534336317231999540219444749074195002936817909891813039214273601159346451628046557591771713207829478467623712257334445439799102390936706187333807785300673089173764684322591241213534448292329433228598997986909594071927136598435740178330540575582781729263491411632253709152595545814341395165301608408174233171428033784216319884817500200168590748606300672401075883105648923113606772682784320062313379584121247014668522787936686972001421465288532485271810666132057455603242647738119564506963079206310177690444016648580862051706117424153524420709356 3

- Public exponent (e): 65537

- Private exponent (d):
  133968382404034511567351386456363535186576179714424907374286562674980911220346246893498405367743170460113890213885387753198854207587932539254941574982902668895388923859507175639497096832254982236954768812886857674499848776894287257094669685502397519728157259708912526086928012053404974876176911811547984793520328891982497361127534585270764726805063479646427619632734134996541006040651113357033985791138043276480653776398124799880430481776514035064369653797829870366932961422442042229012459027435395048828526899977097802088235984016759567825145700599834412844418902276131227872699161761051546575169735697454903140816677008538645401601106643076908368950536148576490718843407529790759799522892409033188921821641326520906835625647726612172821029938231411733403094768496732923552556565481117947698311578158018867672756388036163111452981357081753795465394533685431671266517175069632150986066728628907612738930282443552675738000 1505

- Prime1 (p):
  229920422302215353813811619632145393453663405700321454061672152843988485345111902653042307343410340307624012904650259742685850953652461463445218539943070836498133845675078280543080932195302631384166757702082721426052474512772482032361551395586583960672239206905929931798098965075307071448383848625707804619099261692419978108804882695650052204091607 04

549106637005954637899254391481281177959706310005780090085569233644290975764001075887113583655640335835954830193 39

- Prime2 (q):
  1907138243550181039994251612322392699697008384054532355119945360438189863358534729896675711217664023712795729696164911324256635843391131301574572248499676618705570015945771738078046130397995577505574637190107428313145457983427206700133804829345563874427937255298323865049283820039698868673895974007410002580418678535911552729079677056826708744036092071482357457131801567797626969806321681620897857535325249862099492489806636430737847229704431054195809756978923217
- Exponent1 (d mod (p-1):
  8520144919711921636208903358997639540131024414968654093629201678394043601463887068116919413677866738958079828802563754994687850009008161956193566170342582564343315342577400737399123425385842662611767211731969685144438097589441839934590593394900251871287976835859765206903934366569888058952118875938690728869866898793218469491181136070551242547826065744230034733457655004368404893638626850930427770578074489496534908299901390123596273775737257745719152722874194379

- Exponent2 (d mod (q-1):
  1065997652283904235120152357639102939946912600892406262611255297672639441453680275654592438673199391437617424816972290944507210952822129817344091591117081250231509541543334457013213693772058135025774907449959798495005026425513914824287067435320302971573984124022890883386260660324614324080476026059164185338448464716534354632827970916362909420808552798610582089043626577834851628209182230513095355558448426887840893066307531710801361364092538846411964279989195649

- Coefficient (inverse of q) mod p:
  1511932161505606134065916842495490801943038232637243776949813455028408347931367809497668004527976190761892126215912448875951453282800280506977366539338003014407606008937342783365875739545524729453812341214406049132386882758579780864394903967927096468914572088854209384471669018230927389461093925475491739353790696377294832463037138191703720794929458392428281738380268506342211096152365475256896529912678799028884266953577226512311157126284854761684082756248604 8

Public Key

We expect to see a sequence containing:

- String "ssh-rsa"
- Modulus INTEGER, -- n

- Public Exponent INTEGER -- e

When we went through the computation of RSA, these are the only two numbers that each party sends over to one another through the internet. WIth these numbers available, they are able to use their private keys to decrypt the message they receive. The string “ssh-rsa” entry helps specify that the data contains an rsa public key.

Decoding RSA Public Key

First I convert the rsa public key from the OpenSSH-specific format to the PKCS#1 PEM format using the command:
ssh-keygen -f id_rsa.pub -e -m pem > id_rsa.pub.pem

I then use the cat command to look at the content of the generated .pub.pem file. I paste the output in the ASN1 decoder and get the following output:

Names and values of the integers:

- Modulus(n):
438490030345762861938246407341092593793269894118976434829377139191441041734397042534046096618178303073689458270359419526614059242006209600217305398774134356060385751534761113214089858417594504862573274918352194546955830259767312788703659150914979086571654713556559967145732463963641903733706550936094505289564091025014080285482456880332123453060153087229082900058457623995820202411534336317231999540219444749074195002936817909891813039214273601159346451628046557591771713207829478467623712257334445439799102390936706187333807785300673089173764684322591241213534448292329433228598997986909594071927136598435740178330540575582781729263491411632253709152595545814341395165301608408174233171428033784216319884817500200168590748606300672401075883105648923113606772682784320062313379584121247014668522787936686972001421465288532485271810666132057455603242647738119564506963079206310177690444016648580862051706117424153524420709356 3

- Exponent(e): 65537

Sanity Checks

- First we want to verify that the values of p and q from the private key satisfy p.q = n. This can be done using python since p, q, n are very large integers. I create variables for p, q, and n. I then compute (p*q) - n, which gives 0. Hence p, q, n are valid numbers for rsa.

```
>>> p = 22992042230221535381381161963214539345366340570032145406167215284398848534511190265304230734341034030762401290465025974268585
0953652461463445218539943070836498133845675078280543080932195302631384166757702082721426052474512772482032361551395586583960672239206
9059299317980989650753070714483838486257078046190992616924199781088048826956500522040916070454910663700595463789925439148128117795970
631000578009008556923364429097576400107588711358365564033583595483019339
>>> q = 19071382435501810399942516123223926996970083840545323551199453604381898633585347298966757112176640237127957296961649113242566
3584339113130157457224849967661870557001594577173807804613039799557750557463719010742831314545798342720670013380482934556387442793725
5298323865049283820039698868673895974007410002580418678535911552729079677056826708744036092071482357457131801567797626969806321681620
897857535325249862099492489806636430737847229704431054195809756978923217
>>> p*q
4384900303457628619382464073410925937932698941189764348293771391914410417343970425340460966181783030736894582703594195266140592420062
0960021730539877413435606038575153476111321408985841759450486257327491835219454695583025976731278870365915091497908657165471355655996
7145732463963641903733706550936094505289564091025014080285482456880332123453060153087229082900058457623995820202411534336317231999540
2194447490741950029368179098918130392142736011593464516280465575917717132078294784676237122573344454397991023909367061873338077853006
7308917376468432259124121353444829232943322859899798690959407192713659843574017833054057558278172926349141163225370915259554581434139
5165301608408174233171428033784216319884817500200168590748606300672401075883105648923113606772682784320062313379584121247014668522787
9366869720014214652885324852718106661320574556032426477381195645069630792063101776904440166485808620517061174241535244207093563
1391914410417343970425340460966181783030736894582703594195266140592420062096002173053987741343560603857515347611132140898584175945048
6257327491835219454695583025976731278870365915091497908657165471355655996714573246396364190373370655093609450528956409102501408028548
2456880332123453060153087229082900058457623995820202411534336317231999540219444749074195002936817909891813039214273601159346451628046
5575917717132078294784676237122573344454397991023909367061873338077853006730891737646843225912412135344482923294332285989979869095940
7192713659843574017833054057558278172926349141163225370915259554581434139516530160840817423317142803378421631988481750020016859074860
6300672401075883105648923113606772682784320062313379584121247014668522787936686972001421465288532485271810666132057455603242647738119
5645069630792063101776904440166485808620517061174241535244207093563
>>> (p*q) - k
0
```

- We equally want to verify that e*d mod(λ) = 1. We do this similarly using python with the integers from the private key, and indeed the answer is 1.

```
>>> lamb = math.lcm(p-1, q-1)
5626749809112203462468934984053677431704601138902138853877531988542075879325392549415749829026688953889238595071756394970968322549822
3695476881288685767449984877689428725709466968550239751972815725970891252608692801205340497487617691181154798479352032889198249736112
7534585270764726805063479646427619632734134996541006040651113357033985791138043276480653776398124799880430481776514035064369653797829
8703669329614224420422290124590274353950488285268999770978020882359840167595678251457005998344128444189022761312278726991617610515465
7516973569745490314081667700853864540160110664307690836895053614857649071884340752979075979952289240903318892182164132652090683562564
7726612172821029938231411733403094768496732923552556565481117947698311578158018867672756388036163111452981357081753795465394533685431
6712665171750696321509860667286289076127389302824435526757380001505
5626749809112203462468934984053677431704601138902138853877531988542075879325392549415749829026688953889238595071756394970968322549822
3695476881288685767449984877689428725709466968550239751972815725970891252608692801205340497487617691181154798479352032889198249736112
7534585270764726805063479646427619632734134996541006040651113357033985791138043276480653776398124799880430481776514035064369653797829
8703669329614224420422290124590274353950488285268999770978020882359840167595678251457005998344128444189022761312278726991617610515465
7516973569745490314081667700853864540160110664307690836895053614857649071884340752979075979952289240903318892182164132652090683562564
7726612172821029938231411733403094768496732923552556565481117947698311578158018867672756388036163111452981357081753795465394533685431
6712665171750696321509860667286289076127389302824435526757380001505
>>> e = 65537
>>> e*d % lamb
1
```

- We can also verify that gcd(e, λ) = 1

```
>>> math.gcd(e,k)
1
```

Note: It happened that I named the modulus k instead of n when I was running the checks in python.

- The values of e and n are the same for the public and private keys.