Private Key

-----BEGIN RSA PRIVATE KEY-----
MIIG5AIBAAKCAYEApAK3BJWNIjUbfCt/Nd0c6Io2ZxljRB+eK01eZptqmNVo7ejMi1A7oWDOJagDcQf4QveHQ/j7Otv3
r77vU6GXj3nThUAHDZiJvDyN5HqnW6X9A2RMzlY3civWj/zWoSgAIbhJ3JqiS0UKpPA9AsYt6KMidMHUqZWh97UH
Gpk/rIvGIMx+65rqMfhP7/oaFHGoZJAICW3xUQImBmUrO/9cboKkrmNlDzQ02eXPkLvb8Pu5g5YxrknUmd7uNgRrPu1
AvslZSKVnV31OdYQ9c3tTeDAI17+2NdCedABqVutqFEPSD2gMgsPWpmlidHjE8NcFkd9Zb9qN0aZuCsZwNMeMaU
UBfuP3uVVzRsSwT6n9Yg0nB2PBJAM3w/OC/2nBd3fW0TIfDVNoPKQrPRnZp0cyVvm/H981cuv8NAJKohIFAGvY5b
Cw148LyL5w3E+9tWLvwVQOHE02jL5+hmi3IFzUgCaedZ/U0EaPvhLynlz0wjiFqfwqpjcZd/TaTpxunNgBAgMBAAECg
gGACQ8tVM998DnGvuEYMb14709TGfbJCOZGMlYftMM47CA9n4aQZdGUuaFDQck6B3BNQkWx7ws9tU6ZK9f8G
D6AxKgSCvYybYnTcdZWcBxmSiWtCBbuNyeEAIZzX5izsksnVrV7Pvg7prS1lc6/T/vqZUM9j/105bkiLTCOp+V3IuzFkOI
60IDKFuCEadkCv8ZXatabdR3hinv35bqI3NOKI4CL4CYVWvWjRbXv9SqrR3BgGuic7eeeGrG5rronwl4QcspAcWYSR
cjTd898W2m9Quc1XYMnx5PWtEKha5BlHJeKvhlYlgXyR0gdiNRfg7iS0LtGwpgtapbMZSXfB0FUm1q5UCCXd1aCHF
mFo2cWFA7gpMcuCLHkOkywTln4pKBKJii0p6fs0jf5vps054kVoUJyBA79Kp5Mzg2h467gQ32gJLbg+szErsi10x6M0/3
kIRXnig8sD5+fhtkh+ei9SYwDuBwrhL2tHQr7FitNPwwFwtiSMIfR0M357CTJGeLBAoHBANqZCilC40y/5GlssjhJlyGjb2j
ry3tgzuI2UmiqovqK4dQDvfHTxetOy82QkXm3TVkElXYjl8xzsq6GUevAuNKE97RTSOawiy5q9iebRFPeE4CmnxPbuw
UKP2NOKoQ4PbnykMEvpOWljIPyWytVU/64ZL/yU3NoX8/x1h/GYOORMc1ZXu/RiQiexSE1wQg3eiJb26AeC4K2Utl7
5X5SXDTJYT+FQKeG15B1TXcTf5tsD3qNxFCzATAPtYZhi4J6TQKBwQDAEqliP3Z5aC5Q53GRUoBS1rFtF/93tKub+
7CG4EUxJ5a6ZT+eVilBBC8NvOWrKpKB+IDNYPeFm3d5gkxQWyHIb6e3MfMADA2D3kEFfusvNmGfpOPXBOla/gMu
wf1zCoEoBM1aPglXOJyPgTMLyVXVkAr+Q1r0PKaMoCibIlhrE4k6WuCjKVJpLuclb2sZmcV2qGICo9g+GlWmFuidCj5
tN4cISUdvql9yWOXZnGF3nFDtEvVjzOj/NiKm8UCLhoUCgcEAmajCT8bI6IZrFQBnNtHj/q1X/zTg0FV4nPujaheogN1n
aRVIS9lFYub8s/9Pg3OqV0y1k1nptroNfEjHBSVvxevaA1QYBrFK5NIjPiwSl7Gfm+uEHv1yNVTCwheMI201VqR/xIytK
RB5+bY/B/LvHDqZ96NrrdBMqWuhzKwyJ0VE8MhFx+ZOoVSZ1YJnslLs84IGnEMCbFuIWWOSKn3Sevij93a/wAM1P
EJ4tZ9rL+GB7+//c4qg5xvjyhZ/brRxAoHBAKRSUt9e8/jCDH3iw+JT9rzoODrhAQpkYzy7gzagwBFhkS4+WMMZa4nY
DnluVf6uGVFgTLFod9qyOvCVbLyCeulTMOUlsoMjzJw4lOAiyQpHOBXgd5wI86uJDSf/htnY7s5R7W17DXq1UBvlqMo
4oVpoNT96WGIDUWmVy6eNJp/Uwt5NyUlNnNen7NC2FaZ9S9ZMEC26BtJ6rTO+/Zr8nJPLoZaROrBpzeaB74LQqwl
tlX59bbrIH73XjqXMWDAGEQKBwB298ovgqKBdgKUzU3tSg4nGsvBuJSDQ+UhPBHNU9lgETPBFoeGYt9GYl1j/V1Z
oEflOZ61OxAhWPdBZwEeykFhz4FydDKMar+0NcMfrO+qt6NNIb6ZNHijyvICVs9eZwd1Jl3yc239HGQfh4V5zaevIxxS
Hdf9lifUT5NyPmTzMonpVIm0nv2+FrUV9/cs/2g5SsO1hMqNHT4AlfXKRciyNTtaY8sO6P0b0fbPzBCJ7FKBNriPwbwO
fVmBvWsMDTw==
-----END RSA PRIVATE KEY-----

This data should contain nine integers: the version, the modulus (*n*), the private exponent (*e*), the public exponent (*d*), the first large prime (*p*), the second large prime (*q*), *d* mod (*p* - 1), *d* mod (*q* - 1), and the coefficient (who's definition seems unclear to me).

Decoding was not particularly hard given the tools provided to us. I used the updated ASN.1 decoder from the assignment outline. I then popped the hex provided into a public hex to decimal conversion.

```
        identification: offset - 0, type - sequence, length - 4+1764 bytes
        version: offset - 4, type - integer, length - 2+1 bytes
0
        n: offset 7, type - integer, length - 4+385 bytes
372201951123833576042973021030166415480538737149366759562173977773096258895597708082250628024
285839590037324455293210055392651146862508348261753752020341387599920230020728736416747158479
103439612384968655202138470423002631183233929139016330873455352259916646717334021455153442217
261267024761972237478366000376569961917217424165292504516705448169193345482538251567384192846
983808343295086807537194652243192384444921623638975653240419039249906217490284576033997427237
937201823103137590187693497726281712131501381155073493037426634208155369708411271466977264531
545909327892604774771022226315296546826746949905182407539497976536079850622183103991449735413
760899284185833849786029572860351497641800830428853785304749590172367740515084148026432170902
```

28254721681512344598249968360089402996449908051805010750135889055884178105862504179624014559376535577836947464284283481925790148677854399089976050501049477134915751196917527822233 61
e: offset 396, type - integer, length - 2+3 bytes
65537
d: offset 401, type - integer, length - 4+384 bytes
20558937135790126879099780828069677038002200718383625579673616423373185486092798011165406922012218125882098883503386184911140228445645669174779873694457113128640481448266445303090234732965415740678297549961575007445126715059940290576059922147213152496771262967739584626511061855539401965754281390659805681086457775945501427826698794060614213686505026330878564332727021043167697425923191696954157874805382553850353483949114116678064557762556426252572289931535936725843880293844522718929172443963396291209357805563954942689165366584258757826857922676429396386855550129203406959430332453562467429519022677977660091482421708923118442277479610490628196624163820904316280039812303737923837350200058956727552504831869229069667438527730068023467304332364031337437336168470040760802232044402495822291009900497647445643860030350915686942210606253281206632100482523304726312175043046918518110830680419818756778741400502810365441670419 3
p: offset 789, type - integer, length - 3+193 bytes
20581602385449674888735285688095311528098710714108775154587829860651857645578676771790786375001694430743434904364913154107540130606783516361163418968824474944713059448365228598464389506665361130871441368887680817121213741301999150045615822525019581370835801302337480140555572192156623002466873972251059096456680519248300359931898007201596439651094536125564357591909214356343794988738804488161455876853875997152429875119026790089424217888554653880640196502588278537 3
q: offset 985, type - integer, length - 3+193 bytes
18084206669300183471651959505322921539882735487453005258831387094762533410406009999597757943965124464365663646667807437120364232530540368575633123873711699396098161427538448793903696974524517656681562541193324033397014779698582605752350222670383405561941057781535153585099361876900156455417089061395279860855188426673509942565864994907415041293693104736103009534930678777106459326396830112072826675820887952129916651361466512991468872773618295667116428282089147763 7
d mod (p - 1): offset 1181, type - integer, length - 3+193 bytes
14467449817551850447445826648750702831628719733548423849452250576323447487930764019147320547531593741481736411100337506804189369925283473941835549156290430012863591904836799839389314368876510315958703648044885787923464220734859649424011317455522866084679855568550330883548439808783913033517615242590624997414687247825361261292745737457667344886806278746852904855945094938249324008441326962793871390739648739442118795290955127116584446491141428429335071371610492017 3
d mod (q - 1): offset 1377, type - integer, length - 3+193 bytes
15471341372573091336017548339784328927142609721386622806234008447581453579728156135579094136170965995819980001241598293856364828868003378020028350235001107187396916473430668888972527103225133855412741748367016959128163736729068372557139149919979222495554004142870426185743964041744315642042433273281583286431641220578231319381293825577836237933968592312470113988271202030616234359853930945418993942239568030303747209263018524506273950091140274141147597418518297773 3
coefficient: offset 1573, type - integer, length - 3+192 bytes
28002917707971293041775711884795070532985096741882439176072095351630385601658083046618910221276479796325536482989822623181064025623553015928132671203511186190462300799222943839915116803175750877814827148940222534032771970297655996177547682199251185212720580078794404654046071088829428039698751636372694926887357680540089104830725019730128814941031569855534267826361433320921680640802890702487063670792804575264775395085356927861565574564050264794194920350331783 9

Public Key

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCkArcElY0iNRt8K3813RzoijZnGWNEH54rTV5mm2qY1Wjt6MyLUDuhY
M4lqANxB/hC94dD+Ps62/evvu9ToZePedOFQAcNmIm8Pl3keqdbpf0DZEzOVjdyK9aP/NahKAAhuEncmqJLRQqk8D
0Cxi3looyJ0wdSplaH3tQcamT+si8YgzH7rmuox+E/v+hoUcahkkCUJbfFRAiYGZSs7/1xugqSuY2UPNDTZ5c+Qu9vw+
7mDljGuSdSZ3u42BGs+7UC+yVIlpWdXfU51hD1ze1N4MAjXv7Y10J50AGpW62oUQ9IPaAyCw9amaWJ0eMTw1wW
R31lv2o3Rpm4KxnA0x4xpRQF+4/e5VXNGxLBPqf1iDScHY8EkAzfD84L/acF3d9bRMh8NU2g8pCs9GdmnRzJW+b8f
3zVy6/w0AkqiEgUAa9jlsLDXjwvIvnDcT721Yu/BVA4cTTaMvn6GaLcgXNSAJp51n9TQRo++EvKeXPTCOIWp/CqmNxl
39NpOnG6c2AE= Sophia@Sophias-MacBook-Air-2.local

This data should contain an identification string and only two integers: the modulus (*n*) and the public exponent (*e*).

In order to get the data into a legible format, I used base64 and hexdump:

00 00 00 07 73 73 68 2d 72 73 61 00 00 00 03 01 00 01 00 00 01 81 00 a4 02 b7 04 95 8d 22 35 1b 7c 2b 7f 35 dd
1c e8 8a 36 67 19 63 44 1f 9e 2b 4d 5e 66 9b 6a 98 d5 68 ed e8 cc 8b 50 3b a1 60 ce 25 a8 03 71 07 f8 42 f7 87 43
f8 fb 3a db f7 af be ef 53 a1 97 8f 79 d3 85 40 07 0d 98 89 bc 3c 8d e4 7a a7 5b a5 fd 03 64 4c ce 56 37 72 2b d6 8f
fc d6 a1 28 00 21 b8 49 dc 9a a2 4b 45 0a a4 f0 3d 02 c6 2d e8 a3 22 74 c1 d4 a9 95 a1 f7 b5 07 1a 99 3f ac 8b c6
20 cc 7e eb 9a ea 31 f8 4f ef fa 1a 14 71 a8 64 90 25 09 6d f1 51 02 26 06 65 2b 3b ff 5c 6e 82 a4 ae 63 65 0f 34 34
d9 e5 cf 90 bb db f0 fb b9 83 96 31 ae 49 d4 99 de ee 36 04 6b 3e ed 40 be c9 59 48 a5 67 57 7d 4e 75 84 3d 73 7b
53 78 30 08 d7 bf b6 35 d0 9e 74 00 6a 56 eb 6a 14 43 d2 0f 68 0c 82 c3 d6 a6 69 62 74 78 c4 f0 d7 05 91 df 59 6f
da 8d d1 a6 6e 0a c6 70 34 c7 8c 69 45 01 7e e3 f7 b9 55 73 46 c4 b0 4f a9 fd 62 0d 27 07 63 c1 24 03 37 c3 f3 82 ff
69 c1 77 77 d6 d1 32 1f 0d 53 68 3c a4 2b 3d 19 d9 a7 47 32 56 f9 bf 1f df 35 72 eb fc 34 02 4a a2 12 05 00 6b d8
e5 b0 b0 d7 8f 0b c8 be 70 dc 4f bd b5 62 ef c1 54 0e 1c 4d 36 8c be 7e 86 68 b7 20 5c d4 80 26 9e 75 9f d4 d0 46
8f be 12 f2 9e 5c f4 c2 38 85 a9 fc 2a a6 37 19 77 f4 da 4e 9c 6e 9c d8 01

Offset 0: value 7, byte length of the identifying string
Offset 4: value “ssh-rsa”, identifying string
Offset 11: value 3, byte length of the exponent
Offset 15: value 65537, public exponent
Offset 18: value 385, byte length of modulus
Offset 22: value (very long number), RSA modulus

Sanity Check

I chucked my numbers into my Python shell:

```
>>> p = 20581602385449674888735285688095311528098710714108775154587829860651857645578676771790786375001694430743434904364913154107540130606783516361163418968824474944713059448365228598464389506665361130871441368887680817121213741301999150045615822525019581370835801302337480140555572192156623002466873972251059096456680519248300359931898007201596439651094536125564357591909214356343794988738804488161455876853875997152429875119026790089424217888554653880640196502588278530
>>> q = 18084206669300183471651959505322921539882735487453005258831387094762533410406009999597757943965124464365663646667807437120364232530540368575633123873711699396098161427538448793903697452451765668156254119332403339701477969858260575235022267038340556194105778153515358509936187690015645541708906139527986085518842667359942565864994907415041293693104736103009534930678777106459326396830112072826675820887952129916651361466512991468872773618295667116428282089147763720
>>> phiN = (p-1)*(q-1)
>>> e = 65537
>>> d = 20558937135790126879099780828069677038002200718383625579673616423373185486092798011165406922012218125882098883503386184911140228445645669174779873694457113128640481448266445303090234732965415740678297549961575007445126715059940290576059922147213152496771262967739584626511061855539401965754281390659805681086457775945501427826698794060614213686505026330878564332727021043167697425923191696954157874805382553850353483949114116678064557762556426252572289931535936725843880293844522718929172443963396291209357805563954942689165366584258757826857922676429396386855550129203406959430332453562467429519022677977660091482421708923118442277479610490628196624163820904316280039812303737923837350200058956727552504831869229069667438527730068023467304332364031337437336168470040760802232044402495822291009900497647445643860030350915686942210606253281206632100482523304726312175043046918518110830680419818756778741400502810365441670419 3
>>> n = p*q
>>> (e * d) % phiN
1
```

It appears I am sane.