Authors: Avery “Alberto” Hall, Brian “Viejo” Rhee, Robbie “Roberto” Young “Joven”

Private Key

-----BEGIN RSA PRIVATE KEY-----
MIIG4gIBAAKCAYEAolGKwfT+dOuxpNlJuW/1vQmInfn7VWqAjPzM/Vo59x84vu6C
w8Hiem2kVpiSEY6pxPZmznIuFLCwRtoMIVexOfneKnqr8kIyI1P4v37i/8CppL+R
NLQiPU72nc2ANA+zUg2unFqIzQ6dTZZQf8RsbbUUANhJeo745mhwcjmfg5V7n/fV
Uj2+sQxgMkeSDCCWtQIu4upy7DWmcDT0A9EQ3+1BGuRMkBp5AIJihNJ3ZJLXi9FM
UzLKXMjhsT1ilTykMHdBDbuv+AwwE+6IbjFjh9atb+sHBX4c9v1FTvQgvGRIPAxt
0bzLp95OuieKRLDei1ZwD1vOYIn3+DxQDjYk8C3dWkFPD3jme5ZouIfQoVX8pzNL
JHNTsi0h7j8ezwiyz6oYOKzFWaOD9NJjyJANTdyUylH16FEOVE/Aw2YHW1Qt1Scu
sx0ApcTRmpLDKz6Hzw+KCE4CB1+2gktrFAzvvPtJOgoXmxLEfDeSfSGpDr1itsKO
ukD7zV6mR6GsWy/pAgMBAAECggGAXQ3IMYdFNTxFnA7hgAiaZWQIySAdoQssZScs
UXYfUve4LWVI2/bT4oHmu6o92wjhdMQZjfLRThU5CXVJjPfb5zPC1LF8z6Z5Xd1u
9kFE23/VAVi9Yux90I3rPe4ouC/kCn9Dgek5Ee116pvozEaMLpynuhBEl5lNQbHP
/Pvh4aqZs8x9o+c34gIJR+nV5VPtchmtTXNArP/YyezhkyqQtP8r2sHUzqwOgGc/
GYwa3DJ+lW5Ihk3Qyz5yfIyHbqzTPgu8B2fmvrroAV2Mog5ZYiY7ZvS0xZIUtNw/
/1rmNXGGLQF6Tp7sB0cX+zF6h/K0RwQnkbJ8of1B/eaacsihOhwm1rp9xiXXRT7q
nKnsM+4qiSztq0ArOU0Ragz2NOEzIlDQaKNlCjOhTHUlbqX5SSKuZxSCTYNyyr7S
a1pwR+6aoE9u+aZuypALvch9bLzdKiI5fU4/3OFrtS1tX4GxinaL2y3OnNt+OBt6
CTd0WFfJpa4pvNoG0PTM1u8bXP79AoHBANIxdjueOymdQrJW38IovoD3cbMBBc05
MLvVZ9jLyTpipNiXQBfiBN0zM8Wcq9QRAkDKBqgAWSYAgvLytHsgTBRTKkfiLuxN
+qeVNGO6JzDFAhfSUWVvRDkgjodnA+uLZk2aluHbutUwl0wX1J/FdxqoupzPLGzM
/L2WI4D+ep5b2GoAvL72tkMYDTo5w4Uj6hyafXYfFk88salpPVDghnUm9O5KIcbH
mJZpGssf7//r6aTYmeVS0GUyDrS9QmQuzwKBwQDFsS77byi7sqTvdwHfqyng1VfC
/cC+/fScxt9BmozicYXaogLVKaC4yAql39hi651LBAQ/YWkxOua9x6H70bdX+mQD
MDIZ1d5yAEPc2aXwPXrMtNOra6rWTqFgPvGF+XyI5uWl93PVbQwt0apav0M/nFL1
0NYMAqOevZzj58RShPWMqO1DCVz1tdINgRpgsK7caesAt80m30d+0Xe4LrNKZ4gd
0FCSCuyISiZqf+4xaSy38jt00uO2bRQ2iDHDo8cCgcBP9OTG9QAIanwOnyFrGFiv
cnFb6I2ubYzuRtXEEHExtoiCo7aL3zx+cJXHHDv45ad3weIaUleXLyahCFZDaBvI
t4Ij6hRza45n5UgSj6OM18ReD3FLJrJLEOlm9Hb3SQXXi3DN5eoCYwOKvJ3zUQhn
qbujHuQFV0zlSZGj5in6x3FhkXo7Ilpe121gssx/fxP1YYdMQbMY3Tuv+AqNE3vS
YMlXQ/fpR8/xl3XLwxBmx/UO8h5Ir+H6PLp/Ymc4+N0CgcBNLVPLIuVDCYXsWohn
ulmaBY/Cdu/1YA1L2zOzV9OiX3FDHjZ+40Z0fNvzT8UZMSRx8bp6x9uaNYV5F2N8
fK3X8c9zkzAr99tkpAfaQznN5SJ4oit0Y48JJR9JBKmjhPeCNqn2tfkJnA3CIXUh
AkQNpEia4JF/lNQtl5aVPpjdIjz+UcB2iEQ7RGUhkoIqty88qzjsSdUUVNyDfOEB
DXI/HNX91aJusUcQGeqPbywdLBMiKLXfcKknQcQ/WcIbv2cCgcA/PiNswqbdYTrZ
32SJyfOZFwbRLHmIGWufUab7ai9gZMrKrv3POa2Y1sFAkMm+EG5QOvdvt1Nmov5u
Xbk8sPaecKcofJwySMaN7aLB29T2aAXK2KXCE8t1/ppgMCKGQ2rFDj5bbRe0bxwb

vd4B8UioFcYHj1g4SBT0qc+eOgmGDvXudLdPWT2OerTnmoV7P7ctgNun/nUqYauG
ZeGGcE3LzwOuZRtyHDXxOyXoj+sQ0JhnBjZwvb9xuVSs1u0n8FU=
-----END RSA PRIVATE KEY-----

What is to be expected in the private key file:

version, modulus, publicExponent, privateExponent, prime1, prime2, exponent1, exponent2, coefficient, otherPrimeInfos (optional)
(https://datatracker.ietf.org/doc/html/rfc8017#appendix-A.1.2)

How we decoded the private key:

After removing the hyphenated header and footer, we copied the private key text into an online decoder (https://lapo.it/asn1js/ and https://holtstrom.com/michael/tools/asn1decoder.php).

Decoded private key:

Version: 0x00

This corresponds to the RSA version being used, and is represented by the bytes starting at byte offset 4, starting with the value 02 to represent that the value is of integer type, 01 to represent that the length of the value is 1 bit (check bit), and then 00 to represent the value itself.

Modulus (n):

0x00a2518ac1f4fe74ebb1a4d949b96ff5bd09889df9fb556a808cfcccfd5a39
f71f38beee82c3c1e27a6da4569892118ea9c4f666ce722e14b0b046da0c2157
b139f9de2a7aabf242322353f8bf7ee2ffc0a9a4bf9134b4223d4ef69dcd8034
0fb3520dae9c5a88cd0e9d4d96507fc46c6db51400d8497a8ef8e6687072399f
83957b9ff7d5523dbeb10c603247920c2096b5022ee2ea72ec35a67034f403d1
10dfed411ae44c901a7900826284d2776492d78bd14c5332ca5cc8e1b13d6295
3ca43077410dbbaff80c3013ee886e316387d6ad6feb07057e1cf6fd454ef420
bc64483c0c6dd1bccba7de4eba278a44b0de8b56700f5bce6089f7f83c500e36
24f02ddd5a414f0f78e67b9668b887d0a155fca7334b247353b22d21ee3f1ecf
08b2cfaa1838acc559a383f4d263c8900d4ddc94ca51f5e8510e544fc0c36607
5b542dd5272eb31d00a5c4d19a92c32b3e87cf0f8a084e02075fb6824b6b140c
efbcfb493a0a179b12c47c37927d21a90ebd62b6c28eba40fbcd5ea647a1ac5b
2fe9

This corresponds to the RSA modulus n, represented starting at byte offset 7, starting with 02 to represent that the value is of integer type, 82 01 80 to represent the length of the value, and then the value itself.

publicExponent (e):
0x010001
This corresponds to the RSA public exponent e , represented starting at byte offset 396, starting with 02 to represent that the value is of integer type, 03 to represent the length of the value, and then the value itself.

privateExponent (d):
0x5d0dc8318745353c459c0ee180089a656408c9201da10b2c65272c51761f52f7b82d6548dbf6d3e281e6bbaa3ddb08e174c4198df2d14e15390975498cf7dbe733c2d4b17ccfa6795ddd6ef64144db7fd50158bd62ec7dd08deb3dee28b82fe40a7f4381e93911ed75ea9be8cc468c2e9ca7ba104497994d41b1cffcfbe1e1aa99b3cc7da3e737e2020947e9d5e553ed7219ad4d7340acffd8c9ece1932a90b4ff2bdac1d4ceac0e80673f198c1adc327e956e48864dd0cb3e727c8c876eacd33e0bbc0767e6bebae8015d8ca20e5962263b66f4b4c59214b4dc3fff5ae63571862d017a4e9eec074717fb317a87f2b447042791b27ca1fd41fde69a72c8a13a1c26d6ba7dc625d7453eea9ca9ec33ee2a892cedab402b394d116a0cf634e1332250d068a3650a33a14c75256ea5f94922ae6714824d8372cabed26b5a7047ee9aa04f6ef9a66eca900bbdc87d6cbcdd2a22397d4e3fdce16bb52d6d5f81b18a768bdb2dce9cdb7e381b7a0937745857c9a5ae29bcda06d0f4ccd6ef1b5cfefd
This corresponds to the RSA private exponent d, represented starting at byte offset 401, starting with 02 to represent that the value is of integer type, 82 01 80 to represent the length of the value, and then the value itself.

prime1 (p):
0x00d231763b9e3b299d42b256dfc228be80f771b30105cd3930bbd567d8cbc93a62a4d8974017e204dd3333c59cabd4110240ca06a80059260082f2f2b47b204c14532a47e22eec4dfaa7953463ba2730c50217d251656f4439208e876703eb8b664d9a96e1dbbad530974c17d49fc5771aa8ba9ccf2c6cccfcbd962380fe7a9e5bd86a00bcbef6b643180d3a39c38523ea1c9a7d761f164f3cb1a9693d50e0867526f4ee4a21c6c79896691acb1fefffebe9a4d899e552d065320eb4bd42642ecf
This corresponds to the prime factor p of the modulus n, represented starting at byte offset 789, starting with 02 to represent that the value is of integer type, 02 81 C1 to represent the length of the value, and then the value itself.

prime2 (q):
0x00c5b12efb6f28bbb2a4ef7701dfab29e0d557c2fdc0befdf49cc6df419a8c
e27185daa202d529a0b8c80aa5dfd862eb9d4b04043f6169313ae6bdc7a1fbd1
b757fa6403303219d5de720043dcd9a5f03d7accb4d3ab6baad64ea1603ef185
f97c88e6e5a5f773d56d0c2dd1aa5abf433f9c52f5d0d60c02a39ebd9ce3e7c4
5284f58ca8ed43095cf5b5d20d811a60b0aedc69eb00b7cd26df477ed177b82e
b34a67881dd050920aec884a266a7fee31692cb7f23b74d2e3b66d14368831c3
a3c7
This corresponds to the prime factor q of the modulus n, represented starting at byte offset 985, starting with 02 to represent that the value is of integer type, 81 C1 to represent the length of the value, and then the value itself.

exponent1:
0x4ff4e4c6f500086a7c0e9f216b1858af72715be88dae6d8cee46d5c4107131
b68882a3b68bdf3c7e7095c71c3bf8e5a777c1e21a5257972f26a1085643681b
c8b78223ea14736b8e67e548128fa38cd7c45e0f714b26b24b10e966f476f749
05d78b70cde5ea0263038abc9df3510867a9bba31ee405574ce54991a3e629fa
c77161917a3b225a5ed76d60b2cc7f7f13f561874c41b318dd3baff80a8d137b
d260c95743f7e947cff19775cbc31066c7f50ef21e48afe1fa3cba7f626738f8
dd
This corresponds to the private exponent d mod (p - 1), with p being prime1, represented starting at byte offset 1181, starting with 02 to represent that the value is of integer type, 81 C0 to represent the length of the value, and then the value itself.

exponent2:
0x4d2d53cb22e5430985ec5a8867ba599a058fc276eff5600d4bdb33b357d3a2
5f71431e367ee346747cdbf34fc519312471f1ba7ac7db9a35857917637c7cad
d7f1cf7393302bf7db64a407da4339cde52278a22b74638f09251f4904a9a384
f78236a9f6b5f9099c0dc221752102440da4489ae0917f94d42d9796953e98dd
223cfe51c07688443b44652192822ab72f3cab38ec49d51454dc837ce1010d72
3f1cd5fdd5a26eb1471019ea8f6f2c1d2c132228b5df70a92741c43f59c21bbf
67
This corresponds to the private exponent d mod (q - 1), with q being prime2, represented starting at byte offset 1376, starting with 02 to represent that the value is of integer type, 81 C0 to represent the length of the value, and then the value itself.

coefficient:
0x3f3e236cc2a6dd613ad9df6489c9f3991706d12c7988196b9f51a6fb6a2f60

64cacaaefdcf39ad98d6c14090c9be106e503af76fb75366a2fe6e5db93cb0f6
9e70a7287c9c3248c68deda2c1dbd4f66805cad8a5c213cb75fe9a6030228643
6ac50e3e5b6d17b46f1c1bbdde01f148a815c6078f58384814f4a9cf9e3a0986
0ef5ee74b74f593d8e7ab4e79a857b3fb72d80dba7fe752a61ab8665e186704d
cbcf03ae651b721c35f13b25e88feb10d09867063670bdbf71b954acd6ed27f0
55
This corresponds to the CRT coefficient q^(-1) mod p, with q being prime2 and p being prime1, and is represented starting at the byte offset 1571, starting with a 02 to represent the value being of the integer type, 81 C0 to represent the length, and then the value itself.

otherPrimeInfos: This is not provided in the private key, as per the documentation is left blank when the version provided is 0.

Each element of the sequence starts with the value type of the value itself, which for us was always 02, then the length of the value, and then the value itself. To interpret the value, knowing this one can start at the value, and keep on reading until the length is over.

Public Key

-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAuQrajHZOynw/G3+0yv2ShxqavSI4KATXibXaJnGLskRHzziEKbUI
eJ5sjiSSRjOJntU8/+DaZTo7CPqHSLZQeo+wmRQ5y29E+IRFFQXCADqtI+BhQRpd
msYptE0bW5vuZXGvd08K1B0agmhc1EsbazClEJ6BeK+jv1u3Xdptyx6Kdb1IPSiX
i/r8eAyAt8SyjPgzu+IlTMXxxtsYSr184ZNKtoGdaBNL9sxDFP1AVDKSkCD8V41O
FAVon2XW189BHyaigSDO8WNB4Bq9ySnifH+nXdle61Q9IJLC7gVfp3VSK7hy5uqu
yh8V4o+nXdGB32GtZSazpQihxZSJmUkiLQIDAQAB
-----END RSA PUBLIC KEY-----

Found using instructions from
https://www.thedigitalcatonline.com/blog/2018/04/25/rsa-keys/

What is to be expected in the public key file:
We expect to find an integer representing the RSA modulus n and an integer representing the RSA public exponent e.

How we decoded the public key:

After removing the hyphenated header and footer, we copied the public key text into an online decoder (https://lapo.it/asn1js/ and https://holtstrom.com/michael/tools/asn1decoder.php).

Decoded public key:

modulus (n):
0x00b90ada8c764eca7c3f1b7fb4cafd92871a9abd22382804d789b5da26718b
b24447cf388429b508789e6c8e24924633899ed53cffe0da653a3b08fa8748b6
507a8fb0991439cb6f44f884451505c2003aad23e061411a5d9ac629b44d1b5b
9bee6571af774f0ad41d1a82685cd44b1b6b30a5109e8178afa3bf5bb75dda6d
cb1e8a75bd483d28978bfafc780c80b7c4b28cf833bbe2254cc5f1c6db184abd
7ce1934ab6819d68134bf6cc4314fd405432929020fc578d4e1405689f65d6d7
cf411f26a28120cef16341e01abdc929e27c7fa75dd95eeb543d2092c2ee055f
a775522bb872e6eaaeca1f15e28fa75dd181df61ad6526b3a508a1c594899949
222d
This corresponds to the RSA modulus n, represented starting at byte offset 4, starting with 02 to represent that the value is of integer type, 82 01 01 to represent the length of the value, and then the value itself.

publicExponent (e):
0x010001
This corresponds to the RSA public exponent e, represented starting at byte offset 265, starting with 02 to represent that the value is of integer type, 03 to represent the length of the value, and then the value itself.

Sanity Check
(Please excuse the annoying copy paste)
Using two methods we used in the previous assignment (taken from https://stackoverflow.com/questions/4798654/modular-multiplicative-inverse-function-in-python
), when calling modinv with d as the first input variable the value of e was printed, and when calling modinv with e as the first input variable the value of d was instead printed. This is a key relationship of RSA, and works as shown below.

```
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
```

```
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b //a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('DNE')
    else:
        return x % m

n =
368361979461904531651536465778248123107890088154635793626783027230186102564507073764417618012527018095902012949176022808964527282153533032946729993191793118668456470765648657915477114736570515750475404487374090451559314850927205237427234676317197425824145522938685212289243748535322773514332120798550260507401079936511692831945071491010178932480255484090523619333174778393016533157165225013870122301807193126515986149166282078461534678453047760257704314975274674230101401730884582805680949453634959543956629472476608253824848437216893455069797315706423854236855390547215855287130594018164297022150018956275636497660742774851076810677874075816462938625643331647257820050215350745944477892639186334681056779410636107539444529228019963054047296450274728803421285759270982573316408473761917174476922032018267428023342037283766183251843221677239492320498050018944192961154939844811664343697565348970187445714676822284002654756452 1
e = 65537
p =
197902854272825640925704314308817845576233423622324934157442301611974506095378443856947197487064686076621050100114075952465051007083534723771620187380758671077680616988037492974262316118702685678353595670112186705919035699980881876805650574012099887822186016695582958883046244785033291171693816659088888516111229361792691884296380069651573943969976380433100172919349143362692966809494546474356585289083539409592993931124437231277327430121589912572299327782388910 3
q =
186132726996492292249036948073502679370256263816777016168751496696597237234480485027792565995474221231095473953766369263607352152890112640364986984164003784375831602445730844954813354935969553829729974050432884089058025889374579003646329164192465333443176409942036365165301587903118946358998210067602854002727488567745535961534148761312107713513729810579929064378753073718907408162965030427724766125530308236895769724144872037530153471629664164589788814395019360 7
d =
211174266908207808698595855101004932073279803806366196230402141020573447967577021658039494123146506505960518920815605733488048804763574615573517136490972416520325588036318197759164909558992490459757258067887330093771991672859392831154594107510466217337366242
```

```
616069428123337608926569905911271069937933256595870515499560260789920324106821237357099282524265148891495898646550864765968046365701758004867497719653879977960698939285745125950290056874752311042890825713277546102820719907735844408242006112917305193694186784152448634883856858497555541721773172357981958396326226737120261475728217521095706139192055844740304584478234008430472377169678170985428282268872718091864256619997752399487775243455131776556380625921775985156562404660137436700617445955265624074625419410576813089584310398621595033790507267226420795598398872465080131890517232660568757624687783973561296402317470737887511523099322427832012133448787992645291186 9

test_int = modinv(d, (p-1)*(q-1))
print(test_int)
```