

SSH KEY FILE FORMATS

Zitian Yue

April 19, 2022

1. The content of id-rsa-homework is displayed in text as below:

```
-----BEGIN RSA PRIVATE KEY-----
MIIG5AIBAAKCAIEAurCOHSc2ajC53lXjkL1++xb1lM8P1LsItYTYv+sJ36qAeSmk
vqPi33MKYRthZCbZabrn+Y0y6GHZlR1nLMiatHXjcPgfro3bLFG5ZapTrfEDH8CM
OWOTABit1aQndH8yCMJPdN1xtD1u7T/yKV1Kh/ymLMiaY5YGT5CkXQfDtiim5IgC
umgw/GEaGdCJgj+LsPZyz7iKSd6IdqJs17Tc048WvGhm6NI5cbTvcNuoIR6lPrt1
Mtlwd8rGRu78YRZBP5ocYprb31ItXqdtAcuX19VioQUlXFRDGSsLV6vIpxv0xvN9
wix7HAifnIOFpACb7Uk+kZgHOiQ9jyWC8/MF3yofIXwWGLShe1kw8MaOxULQBYPH
hBvcEnaKy7iIK6NRJqw9+f8bXS4jN4jVg7e+Cq9FB9Bh0liWozjsXfzUDFXb489M
LhwYatDkfc9wXDMIBDisSDwaxU9/411Z774eEihtBfbqRfVgWfHMcAoBF+wun82U
i2E/HnStXbyhl5+7AgMBAAECggGBAIZkvsmx0sA693KQjdNCNsJ2TC+lp5XSvoHQ
6p+tyVvT18xhKBF81PNaU1x4Y3G/bb9pJ1lERiN3XTS1B/L7LjM+sk4+sD/uKpsK
8dw3+ak72K3g5JDj14z4N xuWojia4Dnh92T/jLzdhJ8R63fPirm+LLomZufNthlM
3KWD02WseggFyUD7hFgKZsmyj2rN5/eqUhgq1twTB/uvDX9y2R+e8BUNaxerTmMG
h1/NIILHEFifAgjK8uWpeXbS6fsZi1gOgf772no0r859vrJ/JnWetNomHAH/u5Q
0Qk9yGrnUu/mVa3hNUoK0CCoubGc29Pue6RhKV0xAeQhiI7U1lStETH4IXkr56TP
xhsuZSUX4PndCSUukZN4thZD7nczdtzj0gDdd7FR+OI5Catu0Pj5QPX+sDDlUBB
M+nTaSefJHLJP3L2pQY12bFdNiJhWw0osdJE4vBoGEQipprL0EDzZR4k87JGcd9
ZutDvQM+erpkHOMvaWvKMFobZwnvUQKBwQDeBCEDQls99wpmpGTax7Kvin0ko0nV
o8obLnrFojB533rCISn0Jn95ntpGAvxBNGZ/mLcBo+9w13pLfHHYL38xX04TezTI
RZ7MYOSBKS1ME2JBC5GLwL384UzjT0zeTPWpPb6rgwCT7+UieApMfxTp5ktIMSP
Cp/E15KMKem3x4CUhweY9P8ki7o3J0yrkmHnHd27DXce5JEsSQz9SYYSYaVwdeSI
rJqsNz2ZuTwDUDp6h0y2GMRGqwnrUaDZ1u0CgcEA1t96YNfyYbBfj47MAZuf+6I
GaFK7H6UEUXa9ArHeoam/6jldB6Zf8sHucEsHhkn9Mp1YTJNFxmhc9lFHJpKev6o
WiX33rn0PuJ1gIvqAVMUD9yn5RB4ROGTyFfNqCXPLMUjpyhWf7nr1TaBVf8I+Kg
cx1siubz3pi4MAF022hnsV10iNS3zw+3ju69IwwAW1GjQdKkTs+CJAAtDctIRYXC
TB5WWDUwYZPHYDm4i/ygyF9XzQ4nt42kYuh/HJRHAoHATSHERQQS7icodTXXP8N2
5McnmVWC+lJleQ2iR70aAHMOcZncma1sEXoUCalpXxdT4DX0aQQ3uZzeLSVHuYen
HTEGu1zDjhwOmKOVOnHjAr3KGfJwHMH9YJ+k8UMDzid0Ha+q2oGAhszvcgec0rBV
23DIfszbfiJaBQIY8QB6r26E2Xx106ZKgJrwBcYzJdDokLmSICyD9STpmWCYvefqw
S93E6H4Jtzfdv5EcjJ2cysQUU39uc8wE7Ng9Qfy5wjf1AoHBAmmH0w9l0rcQnjyh
PZ30d26rNlN1KAlilsjay/LoeckHqaFATq9cCf2Jbji3Und7S14RCduwCXlNk7l4
QxfEH8hlrg8seUBrLnuPfy4w8LRFEREYo7fxbAixay7Rgk0b+qZ4sW16H6STd9yv
Xbit9MkpoZL66RNmr4Giws0n/sZLjJrbYzYlRbUmweh1X6ufW0+KOW6uX2HLTeX/
HuW/Vpjr4MCwZJTpqkFXHmXppk1Bfzp/97TCnWAoQ0Q1x1JNbwKBwHNuk4kS3tQ
YFHwHc4yFQDqUGdErY5boZyRZwZn1VGGIwIzVlKCYuwyUXwJ4V8B/NqF5v3h3UEB
xm0eIq2xB0ByRlxvSRatNvnj1tooB8eoa1NZUbZ6M5d7VcmZ5WN6zmAvVh3ToBWO
ANn9H3ulCxZ3ephod+z+NuZjjzWTFY7vgYZpq4Id7omqSz2CaTsNYZcWWRnz8Tud
tBYxAQE+tVrrPTus/u0QS+FfB8ywkV2EgMuafoisDc+KCUlJ+EvJ7w==
-----END RSA PRIVATE KEY-----
```

The content of id-rsa-homework.pub is:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC6sI4dJzZqMLneVeOQvX77FvWUzw/Uuwi
3JNi/6wnfqoB5KaS+o+LfcwphG2FkJtlpuuf5jTL0YdmVHWesyJq0deNw+B+ujdsUblqlOt8QMf
wIw5Y5MAGK3VpCd0fzIwK903XG0PW7tP/IpXUqH/KYsyJpJlgZPkKRdB8O2KKbkiAK6aDD8
YRoZ0ImCP4uw9nLPuIqwPoh2omzXtNw7jxa8aGbo0jlxT09w26ghHqU9G2Uy2XB3ysZG7vxhFkE
/mhximtvfUi1ep20By5fX1WKipSVcVEMZKwtXq8im/HTG833CLHscCJ+cg4WkAJvtST6RmAc6
JD2PJYLz8wXfKh8hfBYYtKF7WTDwxo7FQtAFg8eEG9wSdorLuIgro1EmrD35/xtDLiM3iNWDt7
4Kr0UH0GHSWJY7OOxd/NQMVDvjz0wuFZhq0OR9z3BcOYgEOKxIPBrFT3/jXVnvvh4SKG0F9
upF9UbAWExwCgEX7C6fzZSLYT8edK1dvKGXn7s= james@LAPTOP-NID7KFEL
```

In my private key file, I expect to see this list of terms: version; modulus; publicExponent; privateExponent; prime1; prime2; exponent1; exponent2; coefficient; otherPrimeInfos.

ASN.1 JavaScript decoder

SEQUENCE (9 elem)

```
INTEGER 0
INTEGER (3072 bit) 4236690303636021287646059910578410209594835963711772386031370592475...
INTEGER 65337
INTEGER (3072 bit) 30498664064203666040082066378351987011238571350444725788449703038006191...
INTEGER (1536 bit) 209416962300244710504691674010724863799710240343510007891973220901722...
INTEGER (1536 bit) 202308972600698121813033260152694938523623128612312517188524352636642...
INTEGER (1536 bit) 726218670704741067388540328732996969690190465422639873062166930312...
INTEGER (1536 bit) 183635794021434407073842323853426071045023745604230611489021195913616...
INTEGER (1536 bit) 1086824183971494059203894925260930740663414451365098965338546280167...
```

OKCvHcEzjTfJwCzIYUvMhKqBzSxLk3EoGChPAXQrVdWUQUqXZDzSLStyRtPHIHIEICQZDJPBKA0V
OnJkr3K6GrVHEH9F7jKhSMUdzidOHArq2oGhgzvzeqHwV23Dfrszbf1A8V188RCdmXCLnKft74
gJrwBeVyzJokLacSIcyD9StPtMcVwfvcS93EH6HJ4jZfdv5EcJzyScyQLl39ucOw7Er9qdy9vFlXBH
AMMH09rl0rcOnjrpb230d26N1N1KAlisjayLoeckHgqFAtg9c4FtJbji3UnD7r13QRcdmXCLnKft74
QxFEHShlrq8seELBrLnufFY4+8LFREYo7rfxbaiay7RgkObq+z4sV16H6S7d9rvYbi+9MqpZL66RNm
r4Giwo/n/zLj7ryTYrLBumehLX6urFW0+K0W6ZXLHTE/HuH/Vpj4MCwZJTtpkgXhmXpkLBFzpf/
977CNtAoq0Q1xLJNwbKBwHukkl483oQYFHhc4yFDQUGdeRYSbozRZwnZlnVGWTiZxLKCYuryYJw
4V8B+Nafsv3n3EBEmoeIQzxB0SyBrlxvSRatNvnjjioob8SeouLNZu2EM5d47CmZ5ZWm6waZh3pJG8W
UwGdSLcl7c5ebhdzcrNofLzRTFXz7ozTzoLdT7moSz2CaLNYZeTRzpZtdtRvRtADeAtvrrTtu

☒ with hex dump decode clear

选择文件 id_rsa_homework

Examples: PKCS#7/CMS attached signature (DER) ▼ load example

Instructions

This page contains a JavaScript generic ASN.1 parser that can decode any valid ASN.1 DER or BER structure whether Base64-encoded (are recognized) or Hex-encoded.

This tool can be used online at the address <http://lapo.it/asn1.js/> or offline, unpacking [the ZIP file](#) in a directory and opening index.html

On the left of the page will be printed a tree representing the hierarchical structure, on the right side an hex dump will be shown.

Hovering on the tree highlights ancestry (the hovered node and all its ancestors get colored) and the position of the hovered node given by offset and length in bytes.

base64 are recognized) or Hex-encoded.

30	82	06	E4	02	01	00	02	82	01	81	00	BA	B0	8E	1D
27	36	6A	30	B9	D6	55	83	90	B0	7E	1F	F5	94		
0F	DA	48	ED	B7	21	48	ED	0F	09	DF	0F	0A	79	39	
82	A3	ED	7F	03	61	1B		61	64	26	D9	B9	EA	F7	99
8D	32	E8	61	D9	15	ED	67	2	C8	9A	64	75	E3	70	F8
... skipping 28 bytes ...															
EA	F5	3F	46	0C	58	47	0	0A	01	17	EC	2E	9F	03	00
8B	61	3F	1E	74	AD	5B	CB	81	9F	9F	08	CD	91	04	
01	02	82	01	81	00	86	64	56	C9	B1	D2	CC	3A	F7	72
90	8D	42	36	CF	6A	2F	2F	45	AF	47	90	DD			
82	01	81	00	86	64	2F	DC	61	68	21	7C	DA	63	54	
5C	78	63	71	BF	6F	BF	69	27	59	44	23	77	5D	30	
B5	07	F2	F6	2E	33	3E	42	46	30	37	36	FE	2A	9E	0A
... skipping 28 bytes ...															
66	EB	43	BD	03	3E	7A	BA	64	1C	E3	2F	69	6B	CA	30
3A	5F	16	67	AA	64	5F	02	81	00	DE	6C	21	43	42	58
6D	1A	66	49	64	FF	DB	CB	82	AF	73	AA	43	AB		
AC	1A	2E	74	AD	30	7D	F4	7A	02	C1	29	94			
7F	79	9E	DA	42	ED	FC	41	34	66	7F	9B	87	01	A3	EF
70	7F	74	AB	7C	71	D8	7F	31	3F	5E	43	7B	34	38	
... skipping 96 bytes ...															
AC	9A	AC	37	3D	99	B9	35	9D	50	3A	7A	87	4C	B6	18
C4	46	A9	69	ED	51	AD	09	D6	ED	02	81	C1	00	DE	0F
74	60	DF	7F	62	30	5B	16	3E	3B	30	OC	DA	7F	BB	58
19	A1	4A	EC	7E	94	11	45	DA	F4	0A	CF	7A	6E	BC	FF
AC	E5	00	BE	99	7F	C3	07	89	C1	2C	1E	19	27	P4	CA
75	61	32	4D	17	19	A1	73	D9	45	12	9A	7A	FE	48	AS
... skipping 96 bytes ...															
4C	1E	56	58												

The second integer corresponds to the modulus. Its decimal value is: 423669303636023128764640599105784110295948335963711772386031370592475... According to the decoded base64 data displayed, this integer starts at the offset of 7 and has length of 4+385. The first tag 02 indicates the type of Integer. The second tag 82 01 81 indicates the number of bytes of data needed for reading. The third tag 00 BA B0 8E 1D 27 36 6A 30 B9 DE 55 E3 90 BD 7E FB 16 F5 94 CF 0F D4 BB 08 B7 24 D8 BF EB 09 DF AA 80 79 29 A4 BE A3 E2 DF 73 0A 61 1B 61 64 26 D9 69 BA E7 F9 8D 32 E8 61 D9 95 1D 67 2C C8 9A B4 75 E3 70 F8 ... skipping 288 bytes ... EA 45 F5 46 C0 58 4C 70 0A 01 17 EC 2E 9F CD 94 8B 61 3F 1E 74 AD 5D BC A1 97 9F BB represents the value of this integer.

The fourth integer corresponds to the privateExponent. Its decimal value is: 304986044630366040082063783581987011238571350444725788449703038006191... According to the decoded base64 data displayed, this integer starts at the offset of 401 and has length of 4+385. The first tag 02 indicates the type of Integer. The second tag 82 01 81 indicates the number of bytes of data needed for reading. The third tag 00 86 64 56 C9 B1 D2 C0 3A F7 72 90 8D D3 42 36 C8 F6 4C 2F A5 A7 95 D2 56 81 D0 EA 9F AD C9 5B D3 D7 CC 61 28 11 7C D4 F3 5A 53 5C 78 63 71 BF 6D BF 69 27 59 44 46 23 77 5D 34 B5 07 F2 FB 2E 33 3E B2 4E 3E B0 3F EE 2A 9B 0A ... skipping 288 bytes ... 66 EB 43 BD 03 3E 7A BA 64 1C E3 2F 69 6B CA 30 5A 1B 67 09 EF 51 represents the value of this integer.

The fifth integer corresponds to the prime1. Its decimal value is: 209416962300244710504691674010724863799710240354310007891973220901722... According to the decoded base64 data displayed, this integer starts at the offset of 790 and has length of 3+193. The first tag 02 indicates the type of Integer. The second tag 81 C1 indicates the number of bytes of data needed for reading. The third tag 00 DE 6C 21 03 42 5B 3D F7 0A 66 A4 64 DA C7 B2 AF 8A 73 A4 A3 49 D5 A3 CA 1B 2E 7A C5 A2 30 79 DF 7A C2 21 29 F4 26 7F 79 9E DA 46 02 FC 41 34 66 7F 98 B7 01 A3 EF 70 97 7A 4B 7C 71 D8 2F 7F 31 5F 4E 13 7B 34 C8 ... skipping 96 bytes ... AC 9A AC 37 3D 99 B9 35 9D 50 3A 7A 87 4C B6 18 C4 46 A9 69 EB 51 A0 D9 D6 ED represents the value of this integer.

The sixth integer corresponds to the prime2. Its decimal value is: 202308972006098121813033260152694938523623128612212517188524352636642... According to the decoded base64 data displayed, this integer starts at the offset of 986 and has length of 3+193. The first tag 02 indicates the type of Integer. The second tag 81 C1 indicates the number of bytes of data needed for reading. The third tag 00 D6 DF 7A 60 D7 F2 63 20 5B 16 3E 3B 30 0C D4 7F EE 88 19 A1 4A EC 7E 94 11 45 DA F4 0A C7 7A 86 8C FF A8 E5 0D BE 99 7F CB 07 B9 C1 2C 1E 19 27 F4 CA 75 61 32 4D 17 19 A1 73 D9 45 1C 9A 4A 7A FE A8 ... skipping 96 bytes ... 4C 1E 56 58 35 30 61 93 C7 C8 39 B8 8B FC A0 C8 5F 57 CD 0E 27 B7 8D A4 61 48 7F 1C 94 47 represents the value of this integer.

The seventh integer corresponds to the exponent1. Its decimal value is: 726218670704741067738854032873299659669501904654226398730621693308122... According to the decoded base64 data displayed, this integer starts at the offset of 1182 and has length of 3+192. The first tag 02 indicates the type of Integer. The second tag 81 C0 indicates the number of bytes of data needed for reading. The third tag 4D 21 C4 45 04 12 EE 27 28 75 35 D7 3F C3 76 E4 C7 27 99 55 82 FA 52 65 79 0D A2 47 BD 1A 00 79 8E 71 99 DC 99 AD 6C 11 7A 14 09 A9 69 5F 17 53 E0 35 CE 69 04 37 B9 9C DE 2D 25 47 B9 87 A7 1D 31 06 BB 5C C3 8E 1C 0E 98 A3 95 3A 71 E3 02 ... skipping 96 bytes ... 24 14 53 7F 6E 0B CC 04 EC D8 3D 41 FC B9 C2 31 75 represents the value of this integer.

The eighth integer corresponds to the exponent2. Its decimal value is: 183626794021434407678842233855426071045023754604230611486021195913616... According to the decoded base64 data displayed, this integer starts at the offset of 1377 and has length of 3+193. The first tag 02 indicates the type of Integer. The second tag 81 C1 indicates the number of bytes of data needed for reading. The third tag 00 C3 07 D3 0F 65 D2 B7 10 9E 3C A1 3D 9D CE 77 6E AB 36 53 75 28 09 62 96 C8 DA CB F2 E8 79 C9 07 A9 A1 40 4E AF 5C 09 FD 89 6E 38 B7 52 70 FB 4A 5E 11 09 DB B0 09 72 E7 2B B9 78 43 17 C4 1F C8 65 AE AF 2C 79 40 6B 2E 7B 8F 7D ... skipping 96 bytes ... 65 E9 A6 4D 41 7F 3A 7F F7 B4 C2 9D 60 28 AB 44 35 C6 52 4D 6F represents the value of this integer.

The ninth integer corresponds to the coefficient. Its decimal value is: 108682418397149405920388492526093087406631444513650898965338546280167... According to the decoded base64 data displayed, this integer starts at the offset of 1573 and has length of 3+192. The first tag 02 indicates the type of Integer. The second tag 81 C1 indicates the number of bytes of data needed for reading. The third tag 73 6E 92 4E 24 4B 7B 50 60 51 F0 1D CE 32 15 00 EA 50 67 44 AD 8E 5B A1 9C 91 67 06 67 95 51 86 21 62 33 54 B2 82 62 EC 32 51 7C 09 E1 5F 01 FC DA 85 E6 FD E1 DD 41 01 C6 6D 1E 22 AD B1 07 40 72 46 5C 6F 49 16 AD 36 ... skipping 96 bytes ... E1 5F 07 CC B0 91 5D 84 80 CB 9A 7E 88 AC 0D CF 8A 09 49 49 F8 4B C9 EF represents the value of this integer.

Since the version number is 0, it means that the term of otherPrimeInfos does not appear in my RSA private key file.

Public Key

In my public key file, I expect to see this list of terms: modulus; publicExponent.

I find that the three decoders do not work if I take my .pub file as the input directly. After looking it up through almighty internet, I learn about a command line which converts my OpenSSH public key format into a PEM format according to the blog at: <https://blog.oddbit.com/post/2011-05-08-converting-openssh-public-keys/>. My result in the terminal is displayed as below:

```
PS C:\Users\james\Desktop> ssh-keygen -f id_rsa_homework.pub -e -m pem
-----BEGIN RSA PUBLIC KEY-----
MIIBigKCAyEAurCOHSc2ajC53lXjkL1++xb1lM8P1LsItyTYv+sJ36qAeSmkvqPi
33MKYRthZCbZabrn+Y0y6GHZlR1nLMiatHXjcPgfro3bLFG5ZapTrfEDH8CMOWOT
ABit1aQndH8yCMJPDn1xtD1u7T/yKV1Kh/ymLMiaY5YGT5CkXQfDtiim5IgCumgw
/GEaGdCJgj+LsPZyz7iKsD6IdqJs17TcO48WvGhm6NI5cbTvcNuoIR6lPRTLmtlw
d8rGRu78YRZBP5ocYprb31ItXqdtAcuX19VioqULXFRDGSsLV6vIpxv0xvN9wix7
HAifnIOFpACb7Uk+kZgHOiQ9jyWC8/MF3yofIXwWGLShe1kw8MaOxULQBYPHhBvc
EnaKy7iIK6NRJqw9+f8bXS4jN4jVg7e+Cq9FB9Bh0liWozjsXfzUDFXb489MLhWY
atDkfc9wXDmIBDisSDwaxU9/411Z774eEihtBfbqRfVGwFhMcAoBF+wun82Ui2E/
HnStXbyh15+7AgMBAE=
-----END RSA PUBLIC KEY-----
PS C:\Users\james\Desktop>
```

I then copy the translated text to the Lapo Luchini's ASN.1 decoder and derive the result similar to private key file's as following:

ASN.1 JavaScript decoder

SEQUENCE (2 elem)
INTEGER (3072 bit) 423669303636023128764640599105784110295948335963711772386031370592475...
INTEGER 65537

30 82 01 8A 02 82 01 81 00 BA B0 8E 1D 27 36 6A 30 B9 DE 55 E3 90 BD 7E FB 16 F5 94 CF 0F D4 BB 08 B7 24 D8 BF EB 09 DF AA 80 79 29 A4 BE A3 E2 DF 73 0A 61 1B 61 64 26 D9 69 BA E7 F9 8D 32 E8 61 D9 95 1D 67 2C C8 9A B4 75 E3 70 F8 1F AE 8D ... skipping 288 bytes ... 46 C0 58 4C 70 0A 01 17 EC 2E 9F CD 94 8B 61 3F 1E 74 AD 5D BC A1 97 9F BB 02 03 01 00 01

☒ with hex dump

选择文件 未选择任何文件

Examples: PKCS#7/CMS attached signature (DER)

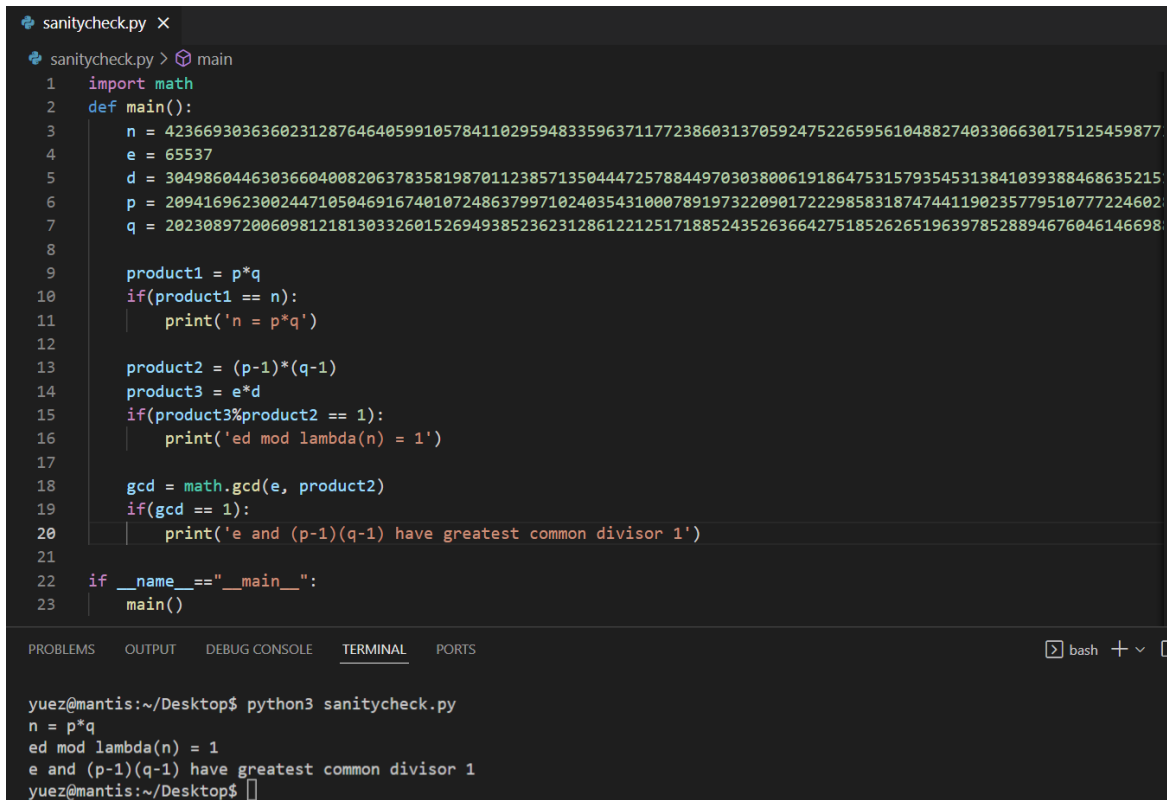
The first integer corresponds to the modulus. Its decimal value is: 423669303636023128764640599105784110295948335963711772386031370592475... According to the decoded base64 data displayed, this integer starts at the offset of 4 and has length of 4+385. The first tag 02 indicates the type of Integer. The second tag 82 01 81 indicates the number of bytes of data needed for reading. The third tag 00 BA B0 8E 1D 27 36 6A 30 B9 DE 55 E3 90 BD 7E FB 16 F5 94 CF 0F D4 BB 08 B7 24 D8 BF EB 09 DF AA 80 79 29 A4 BE A3 E2 DF 73 0A 61 1B 61 64 26 D9 69 BA E7 F9 8D 32 E8 61 D9 95 1D 67 2C C8 9A B4 75 E3 70 F8 1F AE 8D ... skipping 288 bytes ... 46 C0 58 4C 70 0A 01 17 EC 2E 9F CD 94 8B 61 3F 1E 74 AD 5D BC A1 97 9F BB represents the value of this integer.

The second integer corresponds to the publicExponent. Its decimal value is 65537. According to the decoded base64 data displayed, this integer starts at the offset of 393 and has length of 2+3. The first tag 02 indicates the type of Integer. The second tag 03 indicates the number of bytes of data needed for reading. The third tag 01 00 01 represents the value of this integer.

I note that these two values contained in the public key file are the same as corresponding values contained in the private key file. This makes sense to me since they belong to this one pair of keys I created. These two values have to be the same so that I can use my private and public keys for encryption and decryption appropriately.

Sanity check

To examine the integers I found in my files, I extract the values and write a python program to execute complicated mathematical calculations for me. According to the result generated by my program, I conclude that: $n = p \times q$, $e \times d \bmod \lambda(n) = 1$, e and $(p-1)(q-1)$ have greatest common divisor 1. These relationships match my expectations of these integers I found in my RSA key pair. Results and codes of my program is displayed below:



```
sanitycheck.py X
sanitycheck.py > main
1 import math
2 def main():
3     n = 42366930363602312876464059910578411029594833596371177238603137059247522659561048827403306630175125459877
4     e = 65537
5     d = 30498604463036604008206378358198701123857135044472578844970303800619186475315793545313841039388468635215
6     p = 20941696230024471050469167401072486379971024035431000789197322090172229858318747441190235779510777224602
7     q = 20230897200609812181303326015269493852362312861221251718852435263664275185262651963978528894676046146698
8
9     product1 = p*q
10    if(product1 == n):
11        print('n = p*q')
12
13    product2 = (p-1)*(q-1)
14    product3 = e*d
15    if(product3%product2 == 1):
16        print('ed mod lambda(n) = 1')
17
18    gcd = math.gcd(e, product2)
19    if(gcd == 1):
20        print('e and (p-1)(q-1) have greatest common divisor 1')
21
22    if __name__ == "__main__":
23        main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash + -

yuez@mantis:~/Desktop$ python3 sanitycheck.py
n = p*q
ed mod lambda(n) = 1
e and (p-1)(q-1) have greatest common divisor 1
yuez@mantis:~/Desktop$
```