ASN.1 to DER to PEM

Once we have created our key, we want to save in `PEM` format.

1. Compute the `DER` serialization of your key data
2. Compute the base64 of that value

$$output = base64\_encode(DER\_serialize(keydata))$$

Parsing a PEM file

Let's take this example

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAxoINivhkH/RG4L1rgotLyVj3YWBGh11M4Z8MdtNAGgzNwnH2
lBwOJ6K1O5R0mmo5vTSkx21TTTz4b7ealAEzpLHyXShUsdQey2CUv4X6rdHpn29O
E622mImYw4Ni7uJkhdWl3JxpLIu9ya/ZpsKy75W91eV8HyscLz3/zjekPv2y4hNU
du9unhGLGBXYrdEhA310tgHyHgHizmgPohPedReeTHIRQEr/epYFwORyioK2yOHC
6nKniWSioXjrmFziw/vHhl4rtB7ZxLs+M+csAVEr2EcyUb9HhMz4stPyCGDomXSh
gwIZsmJsdsA0tBDJkbx03+qyQfZ5V1XPdIMZUwIDAQABAoIBADE3QFVStTxI66O/
wt54W79da+P8IzBffPa5YLU7NfrfbspFSA27kPTZvdY6Pdik+pDC8xGwtMUDF1NX
cZ89Vwj/x2e6XplCTqo81VRQyvB6iVIqnfB6Ernp73KV6hrxDVwzNq6mJttRACp3
i15xijylYw4bmfT+UrwRwpDlsbad7sItqPF6m8cQ1qPmUw1IWKJpZtnnM0CeW1Zd
ZvPsMgGcXS0RCBkiCESUDftEB7JpTNmtCoMRai4CFKLlluU+C2NQ/F9mAOodHG5c
3iSDnos/KS2CtyZ4MGseSZ+Pe9+tbHgA9x6EHaE6WzFb95L+vQHlYcPD4cWn6g+M
rd/c3PECgYEA0f6CnyHKir81U9D68aZtbRBlQy2CMfO6czWQcijVkekfwdacFYn1
4yRyHCe4htvo5rzUvgQ0A5ggeuGMIRXUowXH1BPfjKjwYGCi78YYvmeQM++r9hVb
SJaMePJUZ7prLDqAhmgBzvXVnEmh1lJOI40RvLfBNClLNUWTR/6EKYMCgYEA8f9X
2HT0B2VgjKDbEA/V/kFhzFD8Ap+1nw5voAle2b+x5S6USfdaP5xhVWYfzCobTytJ
lmHSkiXmjP5eFzeb3L8qfqu4Rdnf+wapS1/PwCsZ4HI/Enc3Ln+7r5hmNvLVnch7
y+Vf0GQltDEB9oH9ouWFOB5+b8o6clfgBDHp1/ECgYA7Phspu8XBWOo5iDaToAk1
ALAgwKD/aKxMPmtO2ZhO/r7X16zXIsG84BZVPRuA6F+PYOx/4v2tmehn4mO/HcKM
b2ANw8GIlEykU/8DuBZY+SykilQwK5xCIT7mDC+lx+DebG6//G2uLoqh+d/vb+7d
drnvTMPz7EZsdAF2CSbN8wKBgBRom2j23AmvpAfYHQFqxHpP20aW4dn6zB9g4UsW
3zfv8bnJRtpCEQtiTdcc6LuYJXt0xBz5nP+UULA9V4QPvYZNXPEX3E+Pw1LxSd/A
cc3cFK+YEvektDOotBRn/t7NdfR7ju0wJ/d0KrXamDbI2bIeNzD3aWRWIr236X2R
FA6RAoGBAII5fdIo7oq8D2PIOnw6w+njz90PMzEdcMmXYb42hkaZBCk4CmxYaf6+
M8CAZ+HPyXARsan09Pvc6bviH2DuBtBNUYkumqbYwkGG+SUDEX7tujiA/xHfrzAC
m2vtOcWmt5eEtm/FabxSSnr2kCNNkXuvz+jT3JUG7X0760Ugk4+f
-----END RSA PRIVATE KEY-----
```

The content is base64 encoded, then let's decode it and look at the hex data:

```
from base64 import b64decode
decoded_certificate = b64decode(certificate).hex()
```

The output is:

```
308204a30201000282010100c6820d8af8641ff446e0bd6b828b4bc958f7616046875d4ce19f0c76d3401a0ccdc271f6941c0e27a2b53b
94749a6a39bd34a4c76d534d3cf86fb79a940133a4b1f25d2854b1d41ecb6094bf85faadd1e99f6f4e13adb6988998c38362eee26485d5
a5dc9c692c8bbdc9afd9a6c2b2ef95bdd5e57c1f2b1c2f3dffce37a43efdb2e2135476ef6e9e118b1815d8add121037d74b601f21e01e2
ce680fa213de75179e4c7211404aff7a9605c0e4728a82b6c8e1c2ea72a78964a2a178eb985ce2c3fbc7865e2bb41ed9c4bb3e33e72c01
512bd8473251bf4784ccf8b2d3f20860e89974a1830219b2626c76c034b410c991bc74dfeab241f6795755cf74831953020301000102820
01003137405552b53c48eba3bfc2de785bbf5d6be3fc23305f7cf6b960b53b35fadf6eca45480dbb90f4d9bdd63a3dd8a4fa90c2f311b0
b4c503175357719f3d5708ffc767ba5e99424eaa3cd55450caf07a89522a9df07a12b9e9ef7295ea1af10d5c3336aea626db51002a778b
5e718a3ca5630e1b99f4fe52bc11c290e5b1b69deec22da8f17a9bc710d6a3e6530d4858a26966d9e733409e5b565d66f3ec32019c5d2d
110819220844940dfb4407b2694cd9ad0a83116a2e0214a2e596e53e0b6350fc5f6600ea1d1c6e5cde24839e8b3f292d82b72678306b1e
499f8f7bdfad6c7800f71e841da13a5b315bf792febd01e561c3c3e1c5a7ea0f8caddfdcdcf102818100d1fe829f21ca8abf3553d0faf1
a66d6d1065432d8231f3ba7335907228d591e91fc1d69c1589f5e324721c27b886dbe8e6bcd4be04340398207ae18c2115d4a305c7d413
df8ca8f06060a2efc618be679033efabf6155b48968c78f25467ba6b2c3a80866801cef5d59c49a1d6524e238d11bcb7c134294b354593
47fe84298302818100f1ff57d874f40765608ca0db100fd5fe4161cc50fc029fb59f0e6fa0095ed9bfb1e52e9449f75a3f9c6155661fcc
2a1b4f2b499661d29225e68cfe5e17379bdcbf2a7eabb845d9dffb06a94b5fcfc02b19e0723f1277372e7fbbaf986636f2d59dc87bcbe5
5fd06425b43101f681fda2e585381e7e6fca3a7257e00431e9d7f10281803b3e1b29bbc5c158ea39883693a0093500b020c0a0ff68ac4c
3e6b4ed9984efebed7d7acd722c1bce016553d1b80e85f8f60ec7fe2fdad99e867e263bf1dc28c6f600dc3c188944ca453ff03b81658f9
2ca48a54302b9c42213ee60c2fa5c7e0de6c6ebffc6dae2e8aa1f9dfef6feedd76b9ef4cc3f3ec466c7401760926cdf302818014689b68
f6dc09afa407d81d016ac47a4fdb4696e1d9facc1f60e14b16df37eff1b9c946da42110b624dd71ce8bb98257b74c41cf99cff9450b03d
57840fbd864d5cf117dc4f8fc352f149dfc071cddc14af9812f7a4b433a8b41467fedecd75f47b8eed3027f7742ab5da9836c8d9b21e37
```

```
30f769645622bdb7e97d91140e910281810082397dd228ee8abc0f63c83a7c3ac3e9e3cfdd0f33311d70c99761be368646990429380a6c
5869febe33c08067e1cfc97011b1a9f4f4fbdce9bbe21f60ee06d04d51892e9aa6d8c24186f92503117eedba3880ff11dfaf30029b6bed
39c5a6b79784b66fc569bc524a7af690234d917bafcfe8d3dc9506ed7d3beb4520938f9f
```

Basically, what DER is doing is serialize-deserialize ASN.1 structures. It uses a simple **tag-length-value** format to serialize an ASN.1 structure, where **tag** denotes the field type (integer, bit string, etc..).
The table below summarizes the tags we are interested in:

Tag (HEX)	Type
02	Integer
03	Bit String
04	Octet String
05	NULL
06	Object Identifier
0C	UTF8 String
10 (or 30)	SEQUENCE and SEQUENCE OF
11 (or 31)	SET and SET OF
13	Printable String
16	IA5 String
17	UTCTime
18	Generalized Time

Exercise: do you recognize some fields in the hex data above? (Try on your own)

Recognizing data from the HEX data

The ordered fields contained in our private key are:

- Version
- Modulus (What we called `n`)
- Public Exponent (what we called `e`)
- Private Exponent (what we called `d`)
- Prime number `p`
- Prime number `q`
- CRT reconstruction exponent `d mod (p-1)`
- CRT reconstruction exponent `d mod (q-1)`
- CRT coefficient `q^(-1) mod p`

CRT stands for Chinese Remainder Theorem, and it is used for fast decryption (directly computing the d-power is too heavy). If you want to read more on this last part, please use these links:

- https://www.youtube.com/watch?v=NcPdiPrY_g8
- https://www.di-mgt.com.au/crt_rsa.html (with examples)