asn.1标准描述的公私钥结构,通过der编码形成.der格式文件,通过base64编码后形成pem格式文件

```
-- ====================
--  Main structures
-- ====================

RSAPublicKey ::= SEQUENCE {
    modulus           INTEGER,  -- n
    publicExponent    INTEGER   -- e
}

--
-- Representation of RSA private key with information for the CRT
-- algorithm.
--
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   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
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

The output will typically contain lines like this:

```
0:d=0  hl=4 l= 681 cons: SEQUENCE
```

.....

```
229:d=3  hl=3 l= 141 prim: BIT STRING
373:d=2  hl=3 l= 162 cons: cont [ 3 ]
376:d=3  hl=3 l= 159 cons: SEQUENCE
379:d=4  hl=2 l=  29 cons: SEQUENCE
381:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Identifier
386:d=5  hl=2 l=  22 prim: OCTET STRING
410:d=4  hl=2 l= 112 cons: SEQUENCE
412:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Identifier
417:d=5  hl=2 l= 105 prim: OCTET STRING
524:d=4  hl=2 l=  12 cons: SEQUENCE
```

.....

This example is part of a self signed certificate. Each line starts with the offset in decimal. d=XX specifies the current depth. The depth is increased within the scope of any SET or SEQUENCE. hl=XX gives the header length (tag and length octets) of the current type. l=XX gives the length of the contents octets.

BER encoding[edit]

The format for Basic Encoding Rules specifies a self-describing and self-delimiting format for encoding ASN.1 data structures. Each data element is encoded as a type identifier, a length description, the actual data elements, and, where necessary, an end-of-content marker. These types of encodings are commonly called type-length-value or TLV encodings. This format allows a receiver to decode the ASN.1 information from an incomplete stream, without requiring any pre-knowledge of the size, content, or semantic meaning of the data.[1]

Encoding structure[edit]

The encoding of data does generally consist of four components which appear in the following order:

Identifier octets *Type*	Length octets *Length*	Contents octets *Value*	End-of-contents octets

The End-of-contents octets are optional and only used if the indefinite length form is used. The Contents octet may also be omitted if there is no content to encode like in the NULL type.

Identifier octets[edit]

Types[edit]

Data (especially members of sequences and sets and choices) can be tagged with a unique tag number (shown in ASN.1 within square brackets []) to distinguish that data from other members. Such tags can be implicit (where they are encoded as the TLV tag of the value instead of using the base type as the TLV tag) or explicit (where the tag is used in a constructed TLV that wraps the base type TLV). The default tagging style is explicit, unless implicit is set at ASN.1 module-level. Such tags have a default class of context-specific, but that can be overridden by using a class name in front of the tag.

The encoding of a choice value is the same as the encoding of a value of the chosen type. The encoding may be primitive or constructed, depending on the chosen type. The tag used in the identifier octets is the tag of the chosen type, as specified in the ASN.1 definition of the chosen type..

The following tags are native to ASN.1:

Name	Value encodings	Tag number	
		Decimal	Hexadecimal
End-of-Content (EOC)	Primitive	0	0
BOOLEAN	Primitive	1	1
INTEGER	Primitive	2	2
BIT STRING	Both	3	3
OCTET STRING	Both	4	4
NULL	Primitive	5	5
OBJECT IDENTIFIER	Primitive	6	6
Object Descriptor	Both	7	7
EXTERNAL	Constructed	8	8
REAL (float)	Primitive	9	9
ENUMERATED	Primitive	10	A
EMBEDDED PDV	Constructed	11	B

UTF8String	Both	12	C
RELATIVE-OID	Primitive	13	D
Reserved		14	E
Reserved		15	F
SEQUENCE and SEQUENCE OF	Constructed	16	10
SET and SET OF	Constructed	17	11
NumericString	Both	18	12
PrintableString	Both	19	13
T61String	Both	20	14
VideotexString	Both	21	15
IA5String	Both	22	16
UTCTime	Both	23	17
GeneralizedTime	Both	24	18
GraphicString	Both	25	19
VisibleString	Both	26	1A
GeneralString	Both	27	1B
UniversalString	Both	28	1C
CHARACTER STRING	Both	29	1D
BMPString	Both	30	1E

Encoding[edit]

The identifier octets encode the element type as an ASN.1 tag, consisting of the class and number, and whether the contents octets represent a constructed or

primitive value. Note that some types can have values with either primitive or constructed encodings. It is encoded as 1 or more octets.

Octet 1									
8	7	6	5	4	3	2	1	8	
Tag class		P/C	Tag number (0–30)						
			31					More	

In the initial octet, bit 6 encodes whether the type is primitive or constructed, bit 7–8 encode the class of the type, and bits 1–5 encode the tag number. The following values are possible:

Class	Value	Description
Universal	0	The type is native to ASN.1
Application	1	The type is only valid for one specific application
Context-specific	2	Meaning of this type depends on the context (such as within a sequence, set or choice)
Private	3	Defined in private specifications

P/C	Value	Description
Primitive (P)	0	The contents octets directly encode

		the element value.
Constructed (C)	1	The contents octets contain 0, 1, or more element encodings.

Long form[edit]

Where the identifier is not universal, its tag number may be too large for the 5-bit tag field, so it is encoded in further octets.

The initial octet encodes the class and primitive/constructed as before, and bits 1–5 are 1. The tag number is encoded in the following octets, where bit 8 of each is 1 if there are more octets, and bits 1–7 encode the tag number. The tag number bits combined, big-endian, encode the tag number. The least number of following octets should be encoded; that is, bits 1–7 should not all be 0 in the first following octet.

Length octets[edit]

There are two forms of the length octets: The definite form and the indefinite form.

Form	Bits							
	8	7	6	5	4	3	2	1
Definite, short	0	Length (0–127)						
Indefinite	1	0						
Definite, long	1	Number of following octets (1–126)						
Reserved	1	127						

Definite form[edit]

This encodes the number of content octets and is always used if the type is primitive or constructed and data are immediately available. There is a short form and a long form, which can encode different ranges of lengths. Numeric data is encoded as unsigned integers with the least significant bit always first (to the right).

The **short form** consists of a single octet in which bit 8 is 0, and bits 1–7 encode the length (which may be 0) as a number of octets.
The **long form** consist of 1 initial octet followed by 1 or more subsequent octets, containing the length. In the initial octet, bit 8 is 1, and bits 1–7 (excluding the values 0 and 127) encode the number of octets that follow.[1] The following octets encode, as big-endian, the length (which may be 0) as a number of octets.

Octet 1									
1	0	0	0	0	0	1	0	0	
Long form	2 length octets								

Indefinite form[edit]
This does not encode the length at all, but that the content octets finish at marker octets. This applies to constructed types and is typically used if the content is not immediately available at encoding time.
It consists of single octet, in which bit 8 is 1, and bits 1–7 are 0. Then, 2 end-of-contents octets must terminate the content octets.

Contents octets[edit]

The contents octets encode the element data value.[1]
Note that there may be no contents octets (hence, the element has a length of 0) if only the existence of the ASN.1 object, or its emptiness, is to be noted. For example, this is the case for an ASN.1 NULL value.