1 DER Encoding

DER (Distinguished Encoding Rules) is a restricted variant of BER for producing unequivocal transfer syntax for data structures described by ASN.1. Like CER, DER encodings are valid BER encodings. DER is the same thing as BER with all but one sender's options removed.

DER is a subset of BER providing for exactly one way to encode an ASN.1 value. DER is intended for situations when a unique encoding is needed, such as in cryptography, and ensures that a data structure that needs to be digitally signed produces a unique serialized representation. DER can be considered a canonical form of BER. For example, in BER a Boolean value of true can be encoded as any of 255 non-zero byte values, while in DER there is one way to encode a boolean value of true.

Applying an encoding rule to the data structures described by an abstract syntax provides a transfer syntax that governs how bytes in a stream are organized when sent between computers. The transfer syntax used by DER always follows a *Tag*, *Length*, *Value format* (TLV).

Tag byte	Length	Value

The following table lists some of the data types supported and the tag byte

Data type	Tag bye
BIT STRING	`0x03`
BOOLEAN	`0x01`
INTEGER	`0x02`
NULL	`0x05`
OBJECT IDENTIFIER	`0x06`
OCTET STRING	`0x04`
SEQUENCE	`0x30`
SET	`0x31`
UTF8String	`0x0C`
PrintableString	`0x13`

Every byte contains 8 bits and can be represented with 8 digits (zeros or ones). For example, the byte `01011101` can be wrritten as `0x5D` in hexadecomal format and `93` in decimal or base 10. We will say that the first bit is the left most bit anf the last bit is the rigth most.

$$\begin{aligned} 0101 &= 1\cdot 1+0\cdot 2^1+1\cdot 2^2+0\cdot 2^3 = 5 \\ 1101 &= 1\cdot 1+0\cdot 2^1+1\cdot 2^2+1\cdot 2^3 = 13 = D \\ 0101 \quad 1101 &= 13\cdot 16+5 = 93 \end{aligned}$$

1.1 Encoding lengths and values

The Length field in a TLV triplet identifies the number of bytes encoded in the Value field. If the length to encode is less than 128, we need only one byte: the last bit is `0` and the the remaining seven bits stores the length.

If the length L satisfies $128 \le L \le 256^{126}$, then the first bit of the first byte is `0` and the remaining seven bits stores the number of bytes needed to represent L. Next the value L is stored starting by the most significant byte.

The value is stored as a list of bytes. For integers, we start with the most significant byte.

1.2 Examples

We want to encode the integer $n = 3656838764234$, we write it in base $b = 256$,

$$n = 202 + 86b + 147b^2 + 108b^3 + 83b^4 + 3b^5$$

and we need 5 bytes to encode the value, i. e., the length is 5. Then, the integer is encode with 7 bytes,

2	5	3	83	108	147	86	202

If the most significant byte of n (in our case 3) is bigger than 127, we have to add an extra byte with the value 0x00 before it.

In the second example, we want to encode binary data formed by 2935117 bytes. First we have to encode the length $L = 2935117$; since L is bigger than 127, we write

$$L = 77 + 201b + 44b^2$$

and we need 3 bytes to encode the length value. Then, the binary data can be encoded as follows:

4	3	44	201	77	2935117 bytes of data.......

1.3 The rsaEncription Obeject Identifier

In RFC3279, *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, the OID rsaEncrytion is defined

```
rsaEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                      rsadsi(113549) pkcs(1) pkcs-1(1) 1 }
```

or 1.2.840.113549.1.1.1.

The explanation can be found in https://crypto.stackexchange.com. The first numbers 1.2 are encoded in one byte `0x2A`, since $1 \cdot 40 + 2 = 42 = 00101010 =$ `0x2A`.

In base 2, $840 = 1101001000$, we split this number in 7 bit blocks starting from the right

$$0000110 \quad 1001000$$

and we add an eighth bit 0 to the last block and 1 to the first

$$10000110 \quad 01001000.$$

In decimal form this numbers are $134 =$ `0x86` and $72 =$ `0x48`.

Similarly, $113549 = 11011101110001101$, we split in

$$000110 \quad 1110111 \quad 0001101$$

and we add an eighth bit 0 to the last block and 1 to the first two

$$1000110 \quad 11110111 \quad 00001101.$$

In decimal form this numbers are $134 =$ `0x86`, $247 =$ `0xF7` and $13 =$ `0x0D`.

Then 1.2.840.113549.1.1.1 is encode as an Object Identifier of length 9,

```
0x06 0x09 0x2A 0x86 0x48 0x86 0xF7 0x0D 0x01 0x01 0x01
```

Finally, all is encoded as a sequence of length $13 =$ `0x0D` formed by the previous object and a NULL,

```
0x30 0x0D 0x06 0x09 0x2A 0x86 0x48 0x86 0xF7 0x0D 0x01 0x01 0x01 0x00 0x00
```