Network Security

Introduction to TLS using a web browser

- ***HTTPS Connections***
- ***X.509 Certificates***
- ***Base64 encodings***
- ***ASN.1 and their BER / DER encodings***

TLS 1.2 Cipher Suite

- **TLS: Transport Layer Security (Sec. for TCP / UDP)**
- **HTTPS: Implemented with a browser**
- **Example Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_ SHA256**
  - **ECDHE**: Elliptic Curve Diffie Hellman Ephemeral key agrement algorithm
  - **RSA**: Rivest, Shamir and Adleman asymmetric encryption and signature algorithms
  - **AES_128_GCM**: Advanced Encryption Standard (with a key length of 128 bits) used in Galois Counter Mode
  - **SHA256**: Secure Hash Algorithm (with hash values of length 256 bits) used in TLS specific PRNG and HKDF
    - PRNG: Pseudo Random Number Generator
    - HKDF: Hash based Key Derivation Function

Base64 encoding

- **Idea**: Each 3 consecutive bytes are encoded with 4 bytes of ASCII characters, which can be transmitted over different channels without any changes.
- **RFC 2045** - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
- **RFC 4648** - The Base16, Base32, and Base64 Data Encodings
- **Java implementation: java.util.Base64**

Base64 encoding

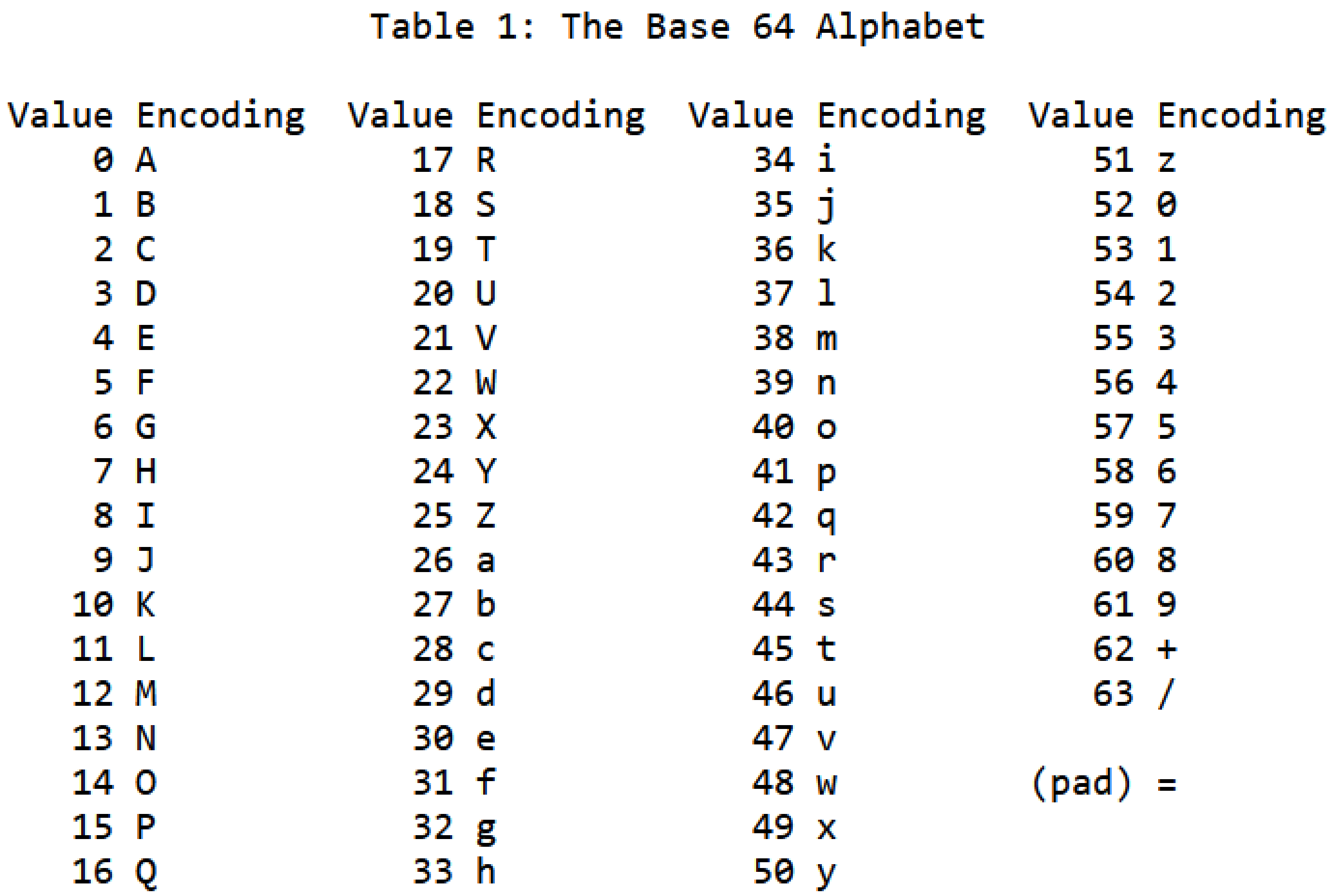
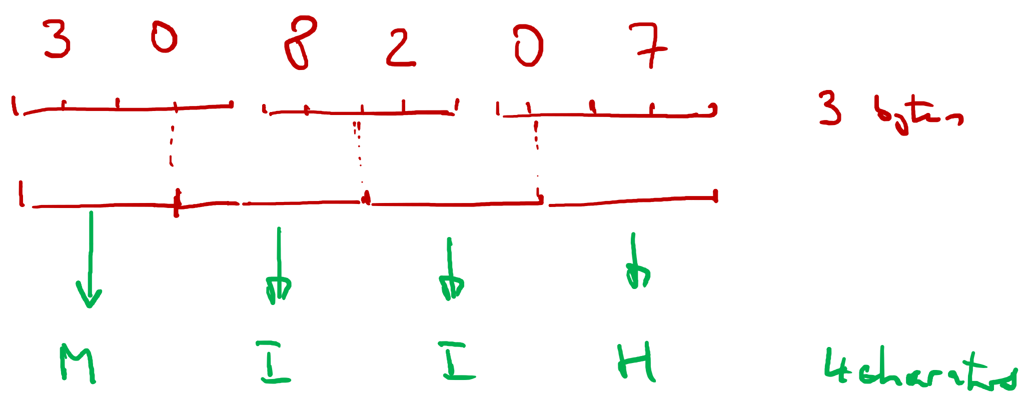


Table 1: The Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		



ASN.1 and BER/DER encodings

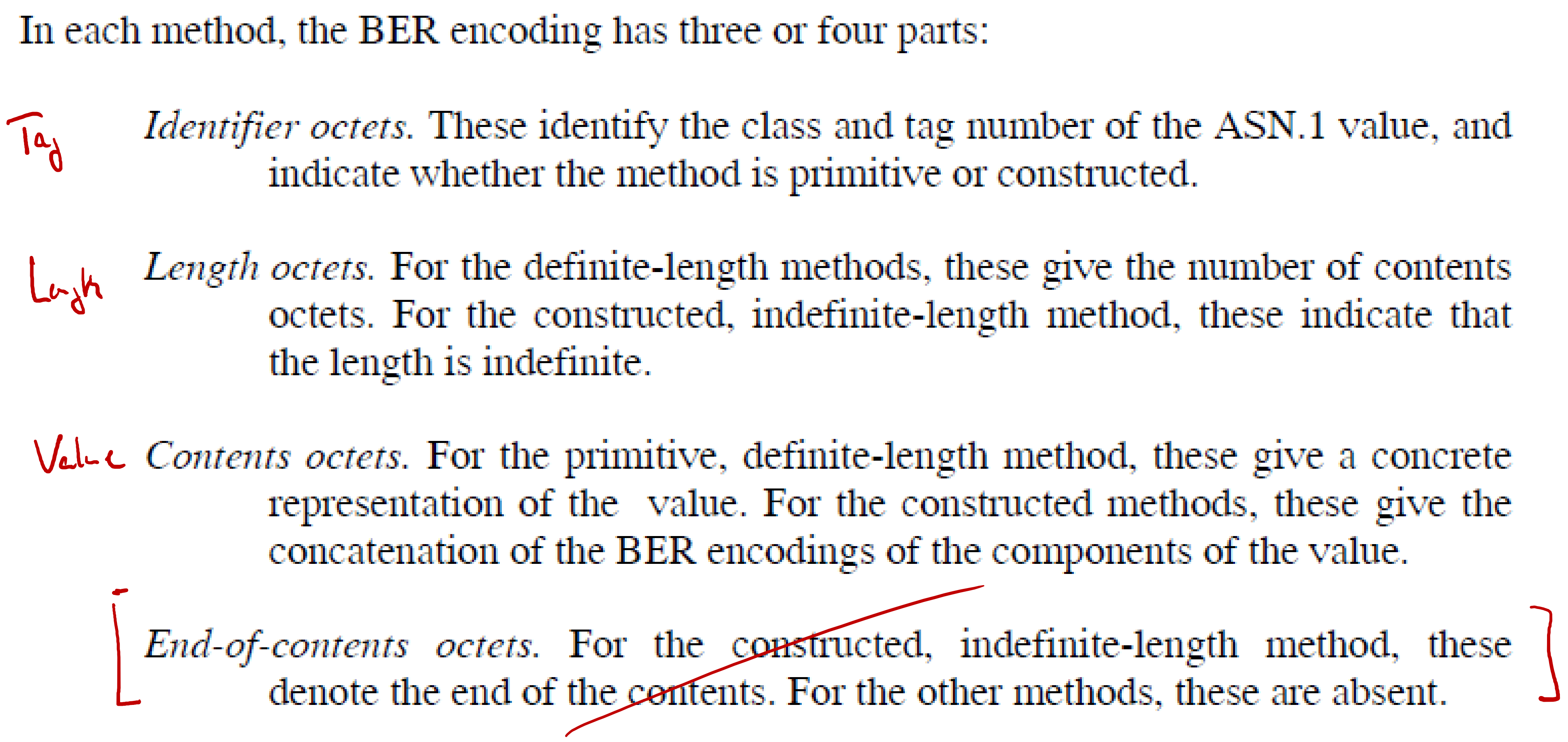
- **ASN.1**: OSI's Abstract Syntax Notation One
- **BER**: Basic Encoding Rules
- **DER**: Distinguished Encoding Rules

- **A Layman's Guide to a Subset of ASN.1, BER, and DER** - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
- **Object Identifier (OID) Repository**

Typical and important example of an ASN.1 specification:

- RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

ASN.1 and BER/DER encodings



In each method, the BER encoding has three or four parts:

Tag

Identifier octets. These identify the class and tag number of the ASN.1 value, and indicate whether the method is primitive or constructed.

Lngth

Length octets. For the definite-length methods, these give the number of contents octets. For the constructed, indefinite-length method, these indicate that the length is indefinite.

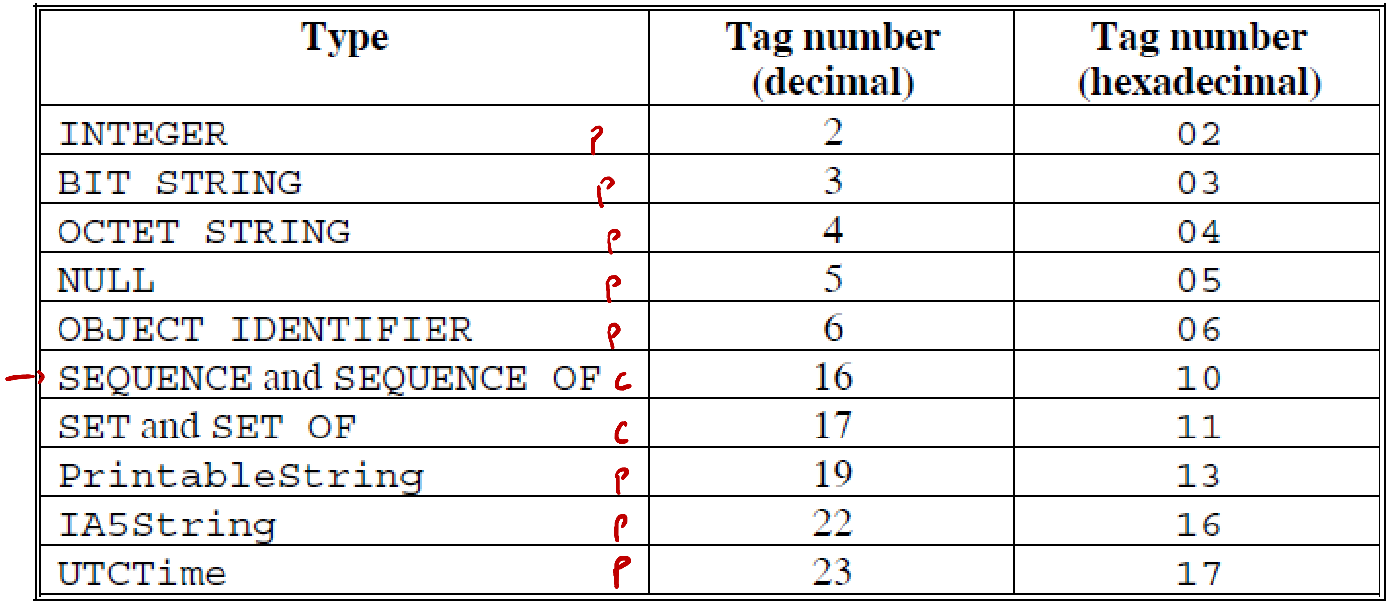
Value

Contents octets. For the primitive, definite-length method, these give a concrete representation of the value. For the constructed methods, these give the concatenation of the BER encodings of the components of the value.

End-of-contents octets. For the constructed, indefinite-length method, these denote the end of the contents. For the other methods, these are absent.

ASN.1 and BER/DER encodings

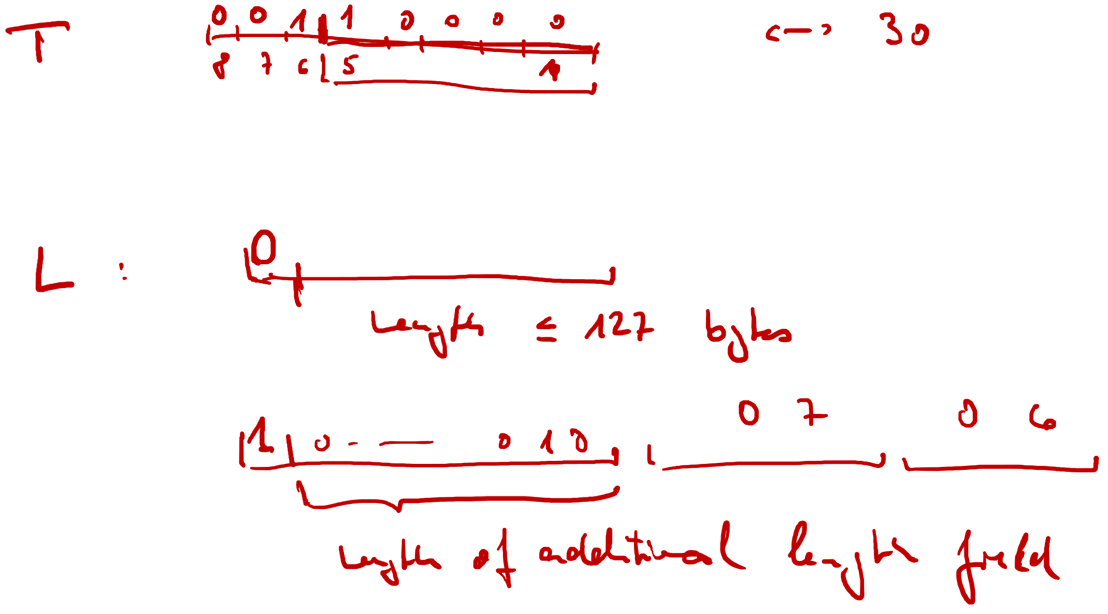
Tag



Type	Tag number (decimal)	Tag number (hexadecimal)
INTEGER p	2	02
BIT STRING p	3	03
OCTET STRING p	4	04
NULL p	5	05
OBJECT IDENTIFIER p	6	06
→ SEQUENCE and SEQUENCE OF c	16	10
SET and SET OF c	17	11
PrintableString p	19	13
IA5String p	22	16
UTCTime p	23	17

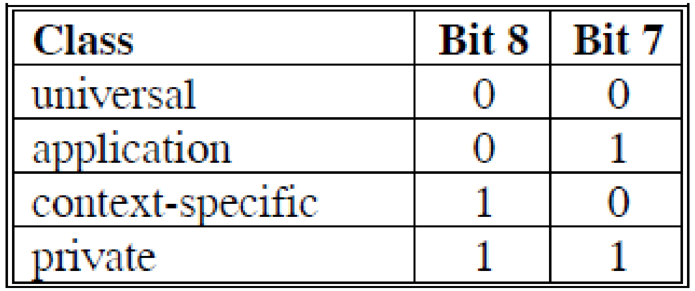
Table 1. Some types and their universal-class tags.

Ex. Sequence:



7 · 256 + 6

✱ **Bit 6** has value 0/1 iff data type is primitive/constructed



Class	Bit 8	Bit 7
universal	0	0
application	0	1
context-specific	1	0
private	1	1

Table 2. Class encoding in identifier octets.

RFC 5280: X.509 certificates and crls

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }
```

To_Be_Signed part of cert.

30 0D 06 09 ——— 05 00

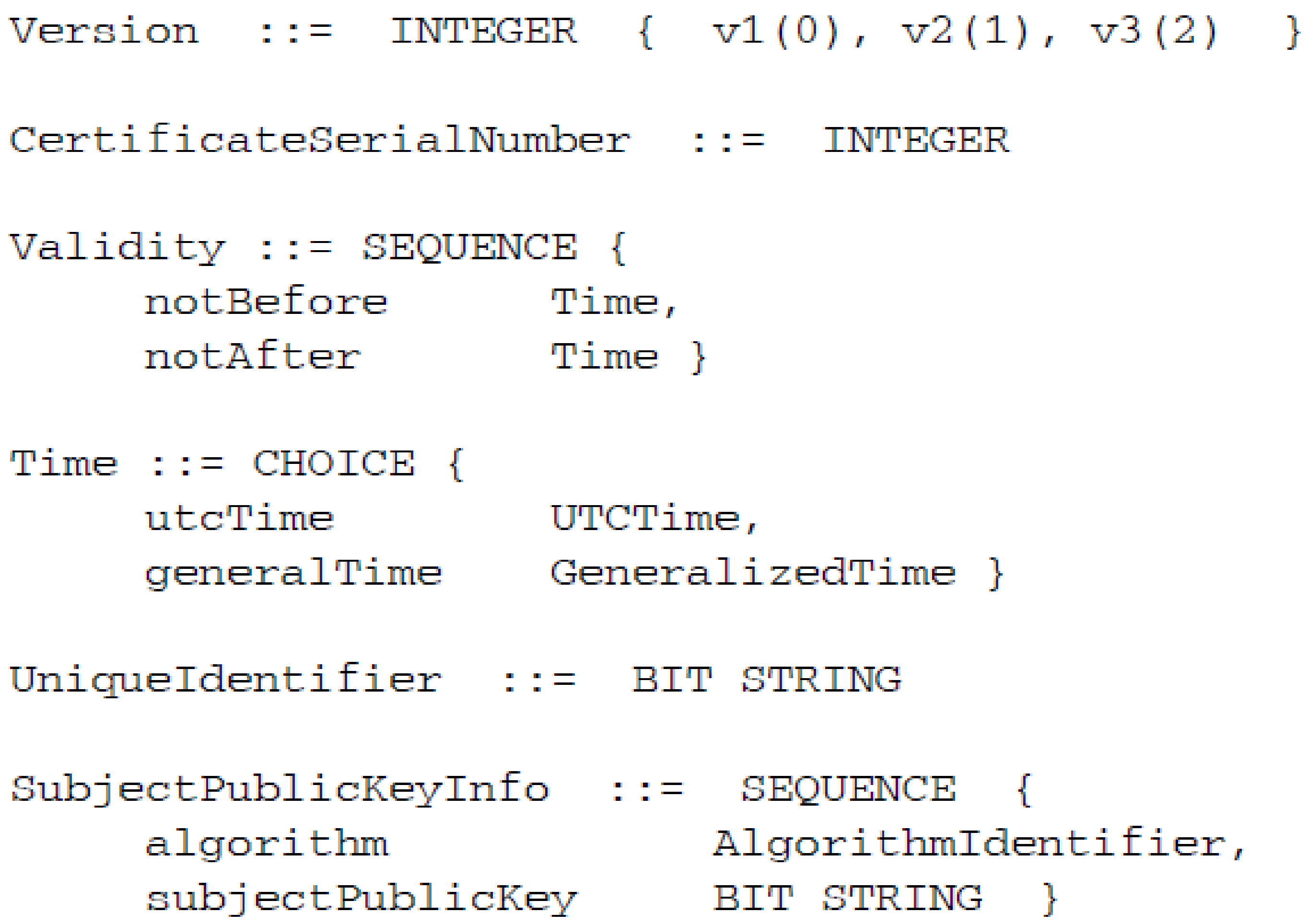
Object-Identifier

NULL

03 82 01 01 00 - - - —

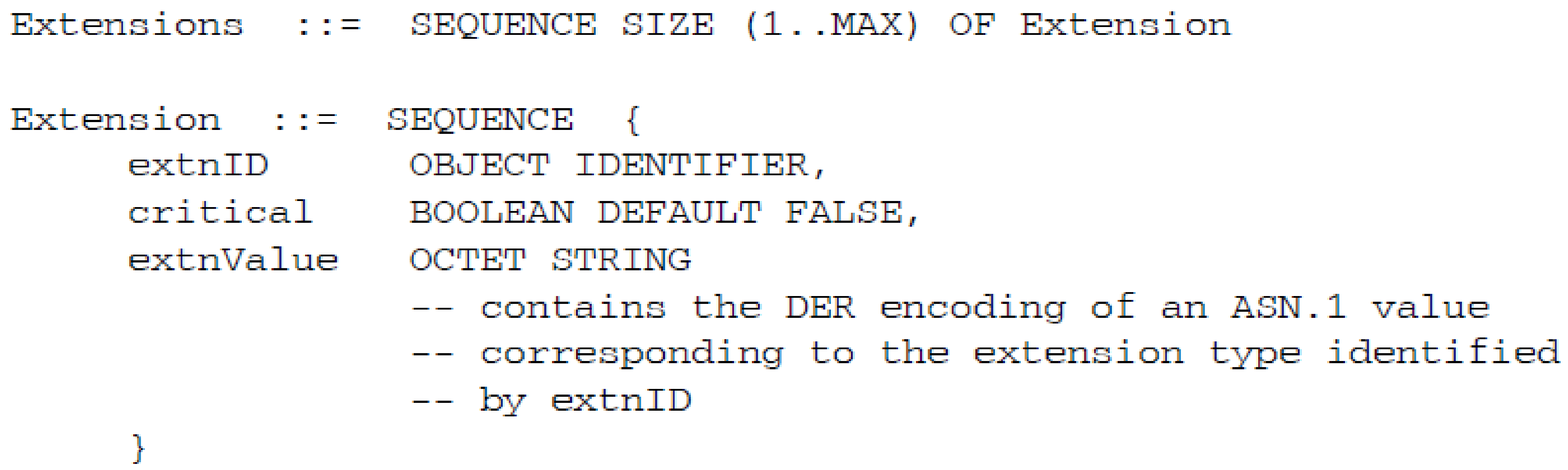
```
TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3

     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version MUST be v3
     }
```

RFC 5280: X.509 certificates and crls

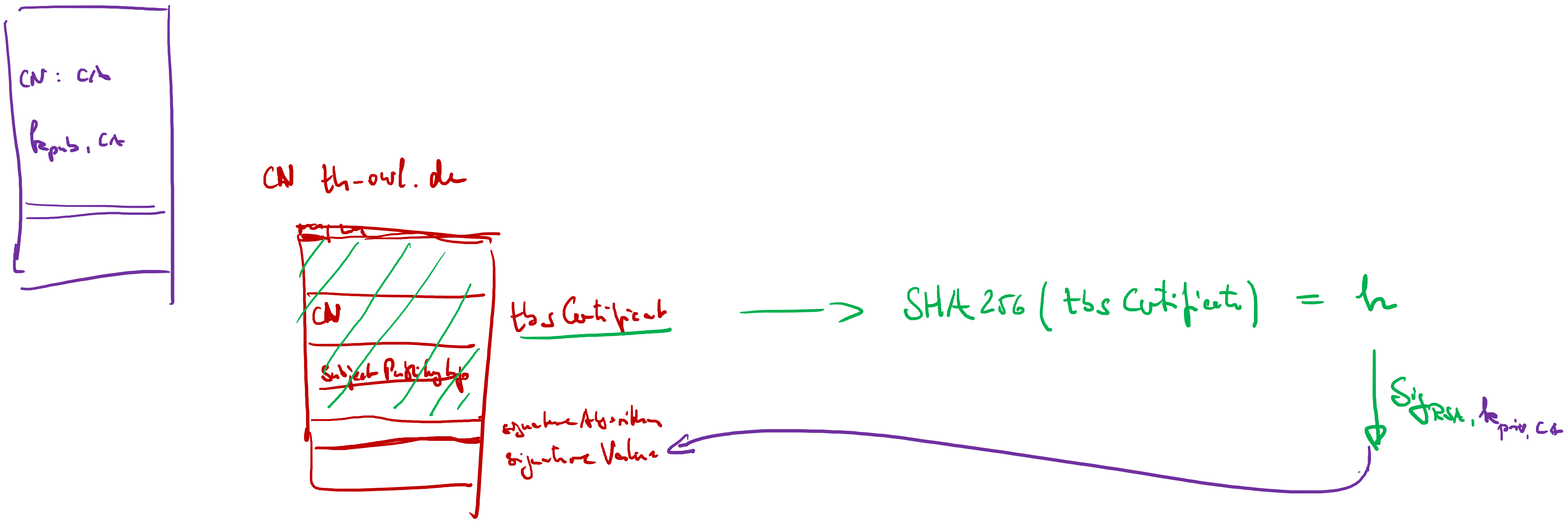


```
Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER

Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }
```

RFC 5280: X.509 certificates and crls



```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
     }
```


CN : CA
kpub, CA
CN th-owl.de
CN
Subject Public key Info
tbs Certificate
signature Algorithm
signature Value
SHA 256 (tbs Certificate) = h
Sig RSA, kpriv, CA