

RSA keys (PKCS#1)

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

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
}

OtherPrimeInfos ::= SEQUENCE SIZE(1..MAX) OF OtherPrimeInfo

OtherPrimeInfo ::= SEQUENCE {
    prime          INTEGER,  -- ri
    exponent       INTEGER,  -- di
    coefficient     INTEGER  -- ti
}
```

ECDSA keys/signatures (RFC 3279, RFC 5480, RFC 5915)

```
EcpkParameters ::= CHOICE {
    ecParameters  ECPParameters,
    namedCurve    OBJECT IDENTIFIER,
    implicitlyCA  NULL }

ECPParameters ::= SEQUENCE {
    version      ECPVer,          -- version is always 1
    fieldID      FieldID,        -- identifies the finite field over
                                -- which the curve is defined
    curve        Curve,          -- coefficients a and b of the
                                -- elliptic curve
    base         ECPPoint,       -- specifies the base point P
                                -- on the elliptic curve
    order        INTEGER,        -- the order n of the base point
    cofactor     INTEGER OPTIONAL -- The integer h = #E(Fq)/n
}

ECPVer ::= INTEGER {ecpVer1(1)}

Curve ::= SEQUENCE {
    a          FieldElement,
    b          FieldElement,
    seed       BIT STRING OPTIONAL }

FieldElement ::= OCTET STRING

ECPPoint ::= OCTET STRING
```

```

ECPrivateKey ::= SEQUENCE {
    version          INTEGER { ecPrivkeyVer1(1) } (ecPrivkeyVer1),
    privateKey       OCTET STRING,
    parameters [0] ECPParameters {{ NamedCurve }} OPTIONAL,
    publicKey [1] BIT STRING OPTIONAL
}

```

```

ECDSA-Sig-Value ::= SEQUENCE {
    r  INTEGER,
    s  INTEGER
}

```

Certificates (RFC 5280)

```

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

```

```

TBSCertificate ::= SEQUENCE {
    version          [0] 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] Extensions OPTIONAL
                    -- If present, version MUST be v3 -- }

```

```

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 }

```

```

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
}

```

```

AlgorithmIdentifier ::= SEQUENCE {
    algorithm          OBJECT IDENTIFIER,
    parameters        ANY DEFINED BY algorithm OPTIONAL }
    -- contains a value of the type
    -- registered for use with the
    -- algorithm object identifier value

Name ::= CHOICE { -- only one possibility for now --
    rdnSequence      RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::=
    SET SIZE (1..MAX) OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
    type      AttributeType,
    value     AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY -- DEFINED BY AttributeType

DirectoryString ::= CHOICE {
    teletexString      TeletexString (SIZE (1..MAX)),
    printableString    PrintableString (SIZE (1..MAX)),
    universalString     UniversalString (SIZE (1..MAX)),
    utf8String          UTF8String (SIZE (1..MAX)),
    bmpString           BMPString (SIZE (1..MAX))

id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }

AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier          [0] KeyIdentifier          OPTIONAL,
    authorityCertIssuer    [1] GeneralNames           OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }

KeyIdentifier ::= OCTET STRING

id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }

KeyUsage ::= BIT STRING {
    digitalSignature          (0),
    nonRepudiation           (1), -- recent editions of X.509 have
    -- renamed this bit to contentCommitment

    keyEncipherment          (2),
    dataEncipherment         (3),
    keyAgreement             (4),
    keyCertSign              (5),
    cRLSign                  (6),
    encipherOnly             (7),
    decipherOnly             (8) }

id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }

BasicConstraints ::= SEQUENCE {
    cA                      BOOLEAN DEFAULT FALSE,
    pathLenConstraint        INTEGER (0..MAX) OPTIONAL }

```

CMS

ContentInfo ::= SEQUENCE {
 contentType ContentType,
 content [0] EXPLICIT ANY DEFINED BY contentType }

ContentType ::= OBJECT IDENTIFIER

SignedData ::= SEQUENCE {
 version CMSVersion,
 digestAlgorithms DigestAlgorithmIdentifiers,
 encapContentInfo EncapsulatedContentInfo,
 certificates [0] IMPLICIT CertificateSet OPTIONAL,
 crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
 signerInfos SignerInfos }

DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

SignerInfos ::= SET OF SignerInfo

SignerInfo ::= SEQUENCE {
 version CMSVersion,
 sid SignerIdentifier,
 digestAlgorithm DigestAlgorithmIdentifier,
 signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
 signatureAlgorithm SignatureAlgorithmIdentifier,
 signatureValue,
 unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }

SignerIdentifier ::= CHOICE {
 issuerAndSerialNumber IssuerAndSerialNumber,
 subjectKeyIdentifier [0] SubjectKeyIdentifier }

SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

Attribute ::= SEQUENCE {
 attrType OBJECT IDENTIFIER,
 attrValues SET OF AttributeValue }

AttributeValue ::= ANY

SignatureValue ::= OCTET STRING