# Part II.
# The RPKI Repository

TMA PhD School 2019

– Tim Bruijnzeels & Martin Hoffmann –

NLNETLABS

# Agenda

- Data Structures
  - ASN.1 and DER.
  - X.509 and resource certificates.
  - CMS and signed objects.
  - ROAs.
- Repository structure and distribution
  - Manifests
  - rsync.

# The House of Cards

Resource Certificates

CMS

Distinguished Encoding Rules

Basic Encoding Rules

ASN.1

Resource Certificates

Internet X.509 profile

X.509

# ASN.1 and DER

# ASN.1

- ‘Abstract Syntax Notation One’
- Language to formally describe data structures and values.
- Independent of encoding
  - multiple encodings are available: ‘Encoding Rules’
- Standardized by ISO, IEC, and ITU-T
  - ITU-T X.680, ISO/IEC 8824-1

# General Structure

```
IPAddrBlocks            ::= SEQUENCE OF IPAddressFamily

IPAddressFamily         ::= SEQUENCE {    -- AFI & optional SAFI --
     addressFamily            OCTET STRING (SIZE (2..3)),
   ipAddressChoice          IPAddressChoice }

IPAddressChoice         ::= CHOICE {
   inherit                  NULL, -- inherit from issuer --
   addressesOrRanges        SEQUENCE OF IPAddressOrRange }

IPAddressOrRange        ::= CHOICE {
   addressPrefix            IPAddress,
   addressRange             IPAddressRange }

IPAddressRange          ::= SEQUENCE {
   min                      IPAddress,
   max                      IPAddress }

IPAddress               ::= BIT STRING
```

# Optional and Tagged Types

```
ASIdentifiers          ::= SEQUENCE {
   asnum                   [0] EXPLICIT ASIdentifierChoice OPTIONAL,
   rdi                     [1] EXPLICIT ASIdentifierChoice OPTIONAL }

ASIdentifierChoice     ::= CHOICE {
   inherit                 NULL, -- inherit from issuer --
   asIdsOrRanges           SEQUENCE OF ASIdOrRange }

ASIdOrRange            ::= CHOICE {
    id                     ASId,
    range                  ASRange }

ASRange                ::= SEQUENCE {
    min                    ASId,
    max                    ASId }

ASId                   ::= INTEGER
```

# Explicit v. Implicit Tagging

- Explicit
  - a value of the given tag wraps around the actual value.
- Implicit
  - the value simply carries a different tag.
- If not specifically mentioned, the default is defined in the module

```
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
-- ...
```

# Object Identifiers

```
id-pe-ipAddrBlocks  OBJECT IDENTIFIER ::= { id-pe 7 }

id-pkix             OBJECT IDENTIFIER ::= {
                      iso(1) identified-organization(3) dod(6) internet(1)
                      security(5) mechanisms(5) pkix(7) }

id-pe               OBJECT IDENTIFIER ::= { id-pkix 1 }
```

```
id-pe-ipAddrBlocks OBJECT INDENTIFIER ::= { 1 3 6 1 5 5 7 1 7 }
```

# Basic Encoding Rules (BER)

- Defined in ITU-T X.690 aka ISO/IEC 8825-1.
- Simple type – length – content encoding.

| Type | Length | Content |
|---|---|---|

- Primitive value: content contains the encoded data of the value.
- Complex value: content contains more values.

# Type: ‘Identifier Octets’

```
8    7    6    5    4    3    2    1
class    P/C   tag number < 32

8    7    6    5  4  3  2  1    8  7  6  5  4  3  2  1    8  7  6  5  4  3  2  1
class    P/C   1  1  1  1  1    1  .  .  .  .  .  .  .    0  .  .  .  .  .  .  .
```

- Tag: one of four classes + a positive number
- Classes:
  - ‘universal’ (00) for types defined in X.680
  - ‘context specific’ (10) for choosing options and alternatives
  - ‘application’ (01) and ‘private’ (11) also exist.

# Length Octets

- definite form
  - content has this many octets
- indefinite form
  - only allowed for constructed values
  - content of values until end-of-value marker

# Definite Form: Length < 128

- one octet

```
8 7 6 5 4 3 2 1

0 0 0 1 1 0 0 1     0x19   25
```

one octet definite form

- highest bit: 0
- lower bits: length

# Definite Form: Longer Length

- first octet: length of length
  - highest bit: 1
  - rest: number of octets to follow
- following octets
  - encode length as big-endian variable length integer

```
8 7 6 5 4 3 2 1   8 7 6 5 4 3 2 1   8 7 6 5 4 3 2 1
1 0 0 0 0 0 1 0   0 0 1 0 0 1 1 0   1 0 0 1 1 1 0 1
              2               0x26              0x9d
                                                9885
```

# Indefinite Form

- encoded as 0x80
- end-of-value
  - empty, primitive value with tag ‘universal 0’.
  - thus: 0x00 0x00

# Distinguished Encoding Rules (DER)

- Stricter sibling of BER
- Same basic rules but some restrictions
  - no indefinite length
  - every value has exactly one correct encoding
- Used in X.509 and pretty much RSA-developed crypto formats

# X.509 and Resource Certificates

# X.509 Public Key Certificates

- Public key certificates provide a cryptographic confirmation for the use of a public key by an entity.
- A key and a set of attributes formally describing the authorised user of the key are signed by a superiour entity,
  - key user: subject,
  - superiour entity: issuer.
- Additional information describing the subject can be included.

# Chains of Certificates

- A subject can be allowed to itself issue certificates.
- All entities that are allowed to issue certificates are called Certificate Authorities (CA).
- Certificates can be verified by following a chain from issuer to issuer until a well-known certificate is reached
  - Called trust anchor certifcate

# Internet Public Key Infrastructure

- X.509 assumes a strict hierarchy of CAs.
  - Developed for a telco-world directory service (X.500).
  - Unsuitable for the more chaotic Internet.
- IETF adopted and adjusted X.509 for Internet usage.
  - Internet Public Key Infrastructure (PKI), currently RFC 5280.
  - Profile on X.509,
    - mandatory and allowed attributes and algorithms,
    - rules for validation of certificates in Internet-land.

# Resource Certificates

- RPKI uses Internet PKI certificates to certify resource ownership.
- Profile on Internet PKI certificates
  - serverly limites the set of attributes and their values (RFC 6487),
  - exactly one algorithm for signatures and digests (RFC 7935).
- Includes attributes for address prefixes and autonomous system numbers (RFC 3779)

# Resource Certificates

- CA certificates
  - confirm transfer of resources from one organisation to another.
- EE certificates
  - issued by a resource holder for a statement about their resources
- TA certificates
  - self-signed CA certificates published and well known

# Resource Certificates

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

# TBSCertificate

```
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,
   subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
   extensions      [3]  EXPLICIT Extensions OPTIONAL
   }
```

must be v3

# Extension

```
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
     }
```

# Basic Constraints

```
id-ce-basicConstraints OBJECT IDENTIFIER ::=  { 2 5 29 19 }

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

- Is this certificate allowed to be used to issue certificates?

- Mandatory in CA certificates, forbidden elsewhere.

# Subject Key Identifier

```
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::=  { 2 5 29 14 }

SubjectKeyIdentifier   ::= KeyIdentifier
KeyIdentifier          ::= OCTET STRING
```

- SHA-1 hash of the subject's public key
- Mandatory.

# Authority Key Identifier

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { 2 5 29 35 }

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

- SHA-1 hash of the issuer's public key
- Mandatory, except optional TA certificates.

# Key Usage

```
id-ce-keyUsage OBJECT IDENTIFIER ::=  { 2 5 29 15 }

KeyUsage ::= BIT STRING {
        digitalSignature        (0),
        nonRepudiation          (1),
        keyEncipherment         (2),
        dataEncipherment        (3),
        keyAgreement            (4),
        keyCertSign             (5),
        cRLSign                 (6),
        encipherOnly            (7),
        decipherOnly            (8) }
```

- How is the key of the certificate allowed to be used?
  - CA certificates: mandatory with keyCertSign and cRLSign
  - EE certificates: mandatory with digitalSignatures.

# Extended Key Usage

```
id-ce-extKeyUsage OBJECT IDENTIFIER ::=  { 2 5 29 37 }

ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPurposeId ::= OBJECT IDENTIFIER
```

- Still: How is the key allowed to be used?
- Not allowed in CA or EE certificates.
- Allowed in certificates for other uses published via RPKI
  - Router keys for BGPSEC.

# Authority Information Access

```
id-pe-extKeyUsage OBJECT IDENTIFIER ::=  { 1 3 6 1 5 5 7 1 1 }

AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription
AccessDescription  ::=  SEQUENCE {
        accessMethod          OBJECT IDENTIFIER,
        accessLocation        GeneralName  }

id-ad-caIssuers OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 48 2 }
```

- “How do I get the issuer certificate?”
- Mandatory except in TA certificates.

# Subject Information Access (SIA)

```
id-pe-extKeyUsage OBJECT IDENTIFIER ::=  { 1 3 6 1 5 5 7 1 11 }

SubjectInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription
AccessDescription  ::=  SEQUENCE {
        accessMethod          OBJECT IDENTIFIER,
        accessLocation        GeneralName  }
```

- “How do I access services provided via this certificate?”
- Mandatory with a certain set of access methods.

# SIA in CA Certificates

- Where is all the material issued by this CA?

```
id-ad-caRepository OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 48 5 }
```

- Where is the manifest (the “index”) of all this material?

```
id-ad-rpkiManifest OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 48 10 }
```

# SIA in EE Certificates

- Each EE certificate signs exactly one object.
- Where is it?

```
id-ad-signedObject OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 48 11 }
```

# Certificate Policies

```
id-ce-certificatePolicies OBJECT IDENTIFIER ::=  { 2 5 29 32 }

certificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation
PolicyInformation ::= SEQUENCE {
     policyIdentifier   CertPolicyId,
     policyQualifiers   SEQUENCE SIZE (1..MAX) OF
                              PolicyQualifierInfo OPTIONAL }
CertPolicyId ::= OBJECT IDENTIFIER

id-cp-ipAddr-asNumber OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 14 2 }
```

- Mandatory with exactly one policy.
- RFC 6484.

# IP Resources

```
id-pe-ipAddrBlocks OBJECT IDENTIFIER ::=  { 1 3 6 1 5 5 7 1 7 }

IPAddrBlocks        ::= SEQUENCE OF IPAddressFamily
IPAddressFamily     ::= SEQUENCE {    -- AFI & optional SAFI --
   addressFamily        OCTET STRING (SIZE (2..3)),
   ipAddressChoice      IPAddressChoice }
IPAddressChoice     ::= CHOICE {
   inherit              NULL, -- inherit from issuer --
   addressesOrRanges    SEQUENCE OF IPAddressOrRange }
IPAddressOrRange    ::= CHOICE {
   addressPrefix        IPAddress,
   addressRange         IPAddressRange }
IPAddressRange      ::= SEQUENCE {
   min                  IPAddress,
   max                  IPAddress }
IPAddress           ::= BIT STRING
```

00 01: IPv4
00 02: IPv6

only include the bits of the network prefix

# AS Resources

```
id-pe-autonomousSysIds OBJECT IDENTIFIER ::=  { 1 3 6 1 5 5 7 1 8 }

ASIdentifiers       ::= SEQUENCE {
    asnum                [0] EXPLICIT ASIdentifierChoice OPTIONAL,
    rdi                  [1] EXPLICIT ASIdentifierChoice OPTIONAL}

ASIdentifierChoice  ::= CHOICE {
   inherit               NULL, -- inherit from issuer --
   asIdsOrRanges         SEQUENCE OF ASIdOrRange }

ASIdOrRange         ::= CHOICE {
    id                   ASId,
    range                ASRange }

ASRange             ::= SEQUENCE {
    min                  ASId,
    max                  ASId }

ASId                ::= INTEGER
```

# CRL Distribution Points

```
id-ce-extKeyUsage OBJECT IDENTIFIER ::=  { 2 5 29 31 }

CRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
DistributionPoint ::= SEQUENCE {
     distributionPoint       [0]     DistributionPointName OPTIONAL,
     reasons                 [1]     ReasonFlags OPTIONAL,
     cRLIssuer               [2]     GeneralNames OPTIONAL }
DistributionPointName ::= CHOICE {
     fullName                [0]     GeneralNames,
     nameRelativeToCRLIssuer [1]     RelativeDistinguishedName }
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
GeneralName ::= CHOICE {
  -- ,,,
  uniformResourceIdentifier [6]  IA5String, }
```

- Mandatory in everything except TA certificates.

# CRLs?

- Answer to “How do I revoke a certificate before it expires?”
  - Once published, users have it and can use it.
- Certificate Revocation List
- A signed list of the serial numbers of all the certificates that have been revoked.

# Certificate Revocation List

```
CertificateList  ::=  SEQUENCE  {
     tbsCertList          TBSCertList,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertList  ::=  SEQUENCE  {
     version                 Version OPTIONAL,
     signature               AlgorithmIdentifier,
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time ~~OPTIONAL~~,
     revokedCertificates     SEQUENCE OF SEQUENCE  {
          userCertificate         CertificateSerialNumber,
          revocationDate          Time,
          ~~crlEntryExtensions      Extensions OPTIONAL~~
                                  }  ~~OPTIONAL~~,
     crlExtensions           [0]  EXPLICIT Extensions OPTIONAL }
```

# CRL Extensions

- Authority Key Identifier (same as with certificates)
- CRL number

```
id-ce-cRLNumber OBJECT IDENTIFIER ::= { 2 5 29 20 }

CRLNumber ::= INTEGER (0..MAX)
```

# CMS and Signed Objects

# Cryptographic Message Syntax

- A framework for exchanging digitally signed, encrypted, digested, authenticated messages.
- Used for instance in S/MIME for secure email.
- RFC 5652.

# Cryptographic Message Syntax

- Provides different containers for content that has been
  - signed
  - enveloped
  - digested
  - encrypted
  - authenticated.
- Stack containers to achieve multiple purposes.

# Signed Object

- Profile on CMS' signed data container.
  - severly limits allowed attributes, values, and algorithms.
  - RFC 6488
- All statements published in RPKI are placed in these signed objects.

# Signed Object

```
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 }

EncapsulatedContentInfo ::= SEQUENCE {
   eContentType ContentType,
   eContent [0] EXPLICIT OCTET STRING OPTIONAL }

DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
```

# Certificate Set

```
CertificateSet ::= SET OF CertificateChoices

CertificateChoices ::= CHOICE {
   certificate Certificate,
   extendedCertificate [0] IMPLICIT ExtendedCertificate,   -- Obsolete
   v1AttrCert [1] IMPLICIT AttributeCertificateV1,         -- Obsolete
   v2AttrCert [2] IMPLICIT AttributeCertificateV2,
   other [3] IMPLICIT OtherCertificateFormat }
```

- Contains the EE certificate used for signing the object.
  - I.e., EE certificates are embedded in signed objects and won't appear stand-alone.

# SignerInfo

```
SignerInfos ::= SET OF SignerInfo

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

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

# SignedAttributes

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

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

AttributeValue ::= ANY
```

# Signed Attributes

- Content-Type,
  - 1.2.840.113549.1.9.3,
  - object identifier of the content type.
- Message-Digest
  - 1.2.840.113549.1.9.4
  - values of the message digest over the content.
- Signing-Time, Binary-Signing-Time: optional.

# Signature

```
SignatureValue ::= OCTET STRING
```

- Signature is not directly calculated over the content.
- Rather, it is calculated over the signed attributes.
  - Allows verification of the values of these attributes.
  - Message-Digest attribute contributes the link to the content.

# ROAs

# Route Origin Authorization

- A signed object listing a set of address prefixes that a autonomous system is allowed to announce.
  - Content type: 1.2.840.113549.1.9.16.1.24

```
RouteOriginAttestation ::= SEQUENCE {
   version [0] INTEGER DEFAULT 0,
   asID  ASID,
   ipAddrBlocks SEQUENCE (SIZE(1..MAX)) OF ROAIPAddressFamily }
```

# ROAIPAddressFamily

```
ROAIPAddressFamily ::= SEQUENCE {
   addressFamily OCTET STRING (SIZE (2..3)),
   addresses SEQUENCE (SIZE (1..MAX)) OF ROAIPAddress }

ROAIPAddress ::= SEQUENCE {
   address IPAddress,
   maxLength INTEGER OPTIONAL }
```

# Repository Content

# Recap: CA Certificates and caRepository

- TA certificates and CA certificates:
  - caRepository SIA where they keep all their issued material.
- This repository contains:
  - the CRL for the CA,
  - ROAs and other signed objects for this CA,
  - CA certificates for all transferred resources.

# Rsync

- software to synchronize the content of two directory trees
  - works locally and across the network.
  - uses a clever algorithm to minimize the data exchanged
- rsync daemon allows to offer rsync as a network service.
- Currently primary means to distribute the RPKI repository.
  - rsync URIs identify resources available for rsync via the daemon

# Repository Distributions

- Distributed Repository,
  - no single central service,
  - potentially lots and lots of locations.
- Each CA chooses the location of its repository.
  - Can be its very own server,
  - Can be under or aside of its parent CA.

# CA Repository Content

- The CA repository contains of all files found in the directory.
  - Whatever rsync puts there.
- Adding illegitimite objects is fine, they are signed.
- Except if they are replayed stale objects.
  - Stale: certificate is still valid but the object has been withdrawn.
  - CRL doesn't help if it is replaced by a stale version, too.

# Manifests

- A signed object that lists all files in the CA repository
  - not just the name but also a hash over the content.
- Included in the CA certificate under the rpkiManifest SIA.
- Normally located in the CA repository itself.

# Manifest

```
Manifest            ::= SEQUENCE {
   version                  [0] INTEGER DEFAULT 0,
   manifestNumber           INTEGER (0..MAX),
   thisUpdate               GeneralizedTime,
   nextUpdate               GeneralizedTime,
   fileHashAlg              OBJECT IDENTIFIER,
   fileList                 SEQUENCE SIZE (0..MAX) OF FileAndHash }

FileAndHash         ::= SEQUENCE {
   file                     IA5String,
   hash                     BIT STRING }
```

# Repository Structure

# Trust Anchor Locators

- We need to start somewhere.

```
rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0URYSGqUz2myBsOzeW1j
Q6NsxNvlLMyhWknvnl8NiBCs/T/S2XuNKQNZ+wBZxIgPPV2pFBFeQAvoH/WK83Hw
A26V2siwm/MY2nKZ+Olw+wlpzlZ1p3Ipj2eNcKrmit8BwBC8xImzuCGaV0jkRB0G
Z0hoH6Ml03umLprRsn6v0xOP0+l6Qc1ZHMFVFb385IQ7FQQTcVIxrdeMsoyJq9eM
kE6DoclHhF/NlSllXubASQ9KUWqJ0+Ot3QCXr4LXECMfkpkVR2TZT+v5v658bHVs
6ZxRD1b6Uk1uQKAyHUbn/tXvP8lrjAibGzVsXDT2L0x4Edx+QdixPgOji3gBMyL2
VwIDAQAB
```

# rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer

caRepository: rsync://rpki.ripe.net/repository/

rpkiManifest: rsync://rpki.ripe.net/repository/ripe-ncc-ta.mft

```
$ ls -1 rpki.ripe.net/repository/
2a7dd1d787d793e4c8af56e197d4eed92af6ba13.cer
aca
DEFAULT
ripe-ncc-ta.crl
ripe-ncc-ta.mft
```

# 2a7dd1d787d793e4c8af56e197d4eed92af6ba13.cer

caRepository: rsync://rpki.ripe.net/repository/aca/

rpkiManifest: rsync://rpki.ripe.net/repository/aca/Kn3R14fXk-\
TIr1bhl9Tu2Sr2uhM.mft

```
$ ls -1 rpki.ripe.net/repository/aca
HGp1AESLbyiopScGy7yW4b6s_T4.cer
Kn3R14fXk-TIr1bhl9Tu2Sr2uhM.crl
Kn3R14fXk-TIr1bhl9Tu2Sr2uhM.mft
qM_jralcLee1A8ndIB6R9r9Jz8A.cer
```