DATA DEFINITION AND PROCEDURES TO VERIFY COMPLIANCE WITH CERTIFICATE TEMPLATES

INVENTOR:

MASSIMILIANO PALA

Description:

The idea comprises two main parts: the definition of two data structures (the first capable of representing certificate templates and the second capable of carrying the result of a validation process), and the definition of the procedures required to validate a certificate against the data structure.

The first part of the invention focuses on the definition of a data structure that can be used to define certificate templates. The data structure's definition shall carry the details about the allowed contents for certificate profiles, together with the profile's description, and the indication of their level in the hierarchy (if applicable). The data structure's definition shall also contain the indication (in the form of an OID) of the Certificate Policy that the data structure relates to. The definition shall also allow the data structure to be validated for integrity and authentication from a validated source. For example, the structure shall be signed with an X.509 certificate issued by the Root CA (or Intermediate CA) to the CPA that carry a specific value in the Extended Key Usage that identifies the signing entity as an authorized CPA. The definition can be translated into different formats (e.g., XML, JSON, CBOR, DER, etc.) and can also be included in Certificate Policy documents (usually in a human-readable text format) to define the approved profiles of certificates under one or more PKIs.

The second data definition is aimed at providing an authenticated description of the result of the certificate-profile (or template) validation process. This data structure shall allow for the description of the result for one or more of the certificates present in a certiicate-profile (or template) definition against one or more certificates for one or more certificate-profile (or template) definition. This data structure may be authenticated (i.e., signed) by the entity that performed the validation.

The second part comprises the process with which the specific data structure that represents the certificate template is processed for different scenarios (this could be a separate patent application). In particular it covers the following scenarios:

A Certification Authority wants to validate that the contents of the issued certificates are compliant with the profiles defined in a CP

A Certificate Policy Authority wants to validate that the contents of certificates issued by one or more CAs are compliant with the established templates for the PKI

A user (application, monitoring system, etc.) wants to make sure that the certificates presented in a transaction is compliant with a profile (it can check, for example, the template against the certificate's template and verify that the OID associated with it is one of the allowed ones)

An auditor that wants to verify that certificates issued by a CA is conformant to the certificate profiles (or templates) defined in a Certificate Policy

The provisioning of a Certificate Policy compliance service where a trusted third party can provide an authenticated attestation for certificate-profiles (or template) compliance

Background :

In many ecosystems, because the flexibility provided by digital certificates, the contents of digital certificates need to be regulated to provide interoperability within the deployment environment.

Today, no standard exist that provide the required data definitions and procedures to (a) describe a profile (or template) for a certificate, and (b) to validate the compliance of a certificate to a particular profile.

This invention addresses this problem by providing a solution to both aspects of certificate-profile validation.

```
PKIXProfiles ::= SEQUENCE {
  certTemplates tbsCertificateTemplates
  algorithm     Algorithm,
  signature     BIT STRING,
  certificates  SEQUENCE OF (1..MAX) Certificate }

tbsCertificateTemplates ::= SEQUENCE {
  policyId      OBJECT IDENTIFIER OPTIONAL,
  templates     CertificateProfilesList }

CertificateProfilesList ::= SEQUENCE OF (1..MAX) CertificateProfile

CertificateProfile ::= SEQUENCE {
  id          OBJECT IDENTIFIER OPTIONAL,
  type        ProfileType,
  constraints   ConstraintsList }

ProfileType ::= ENUMERATION {
  notSpecified  (0),
  certificate   (1),
  p10request    (2),
  crl           (3),
  ocspResponse  (4) }

ConstraintsList ::= SEQUENCE OF (1..MAX) ProfileConstraint

ProfileConstraint ::= SEQUENCE {
  isOptional     [0] BOOLEAN OPTIONAL,
  mustCritical   [1] BOOLEAN OPTIONAL,
```

```
  mayCritical    [2] BOOLEAN OPTIONAL,
  target         ConstraintTarget }

ConstraintTarget ::= CHOICE {
  VersionConstraint  (0),
  SubjectConstraint  (1),
  IssuerConstraint   (2),
  SerialConstraint   (3),
  ValidityConstraint (4),
  PubKeyConstraint   (5),
  SigConstraint      (6),
  ExtConstraint      (7) }

VersionConstraint ::= SEQUENCE {
  minValue           [0] INTEGER OPTIONAL,
  maxValue           [1] INTEGER OPTIONAL }

SubjectConstraint ::= SEQUENCE {
  rdn                OBJECT IDENTIFIER,
  relativePosition   [0] INTEGER OPTIONAL,
  minSize            [1] INTEGER OPTIONAL,
  maxSize            [2] INTEGER OPTIONAL,
  regExpression      UTF8 STRING OPTIONAL }

IssuerConstraint ::= SEQUENCE {
  rdn                OBJECT IDENTIFIER,
  relativePosition   [0] INTEGER OPTIONAL,
  minSize            [1] INTEGER OPTIONAL,
  maxSize            [2] INTEGER OPTIONAL,
```

```
  regExpression      UTF8STRING }

SerialConstraint ::= SEQUENCE {
  minValue          [0] INTEGER OPTIONAL,
  maxValue          [1] INTEGER OPTIONAL
  minSize           [2] INTEGER OPTIONAL,
  maxSize           [3] INTEGER OPTIONAL }

ValidityConstraint ::= SEQUENCE {
  type              [0] TimeEncodingType OPTIONAL,
  minNotBefore          [1] GeneralizedTime OPTIONAL,
  maxNotBefore          [2] GeneralizedTime OPTIONAL,
  maxValidityYears      [3] INTEGER OPTIONAL,
  maxValidityDays       [4] INTEGER OPTIONAL,
  maxValidityMinutes    [5] INTEGER OPTIONAL,
  maxValiditySecs       [6] INTEGER OPTIONAL }

PubKeyConstraint ::= SEQUENCE {
  algorithm             Algorithm OPTIONAL,
  bitSizes              BitSizes OPTIONAL,
  minBitSize            [0] INTEGER OPTIONAL,
  maxBitSize            [1] INTEGER OPTIONAL }

BitSizes ::= SEQUENCE OF (1..MAX) INTEGER

ExtConstraint ::= SEQUENCE {
  target              OBJECT IDENTIFIER,
  tagValue            [0] INTEGER OPTIONAL,
  value               [1] ANY DEFINED by target OPTIONAL }
```