Infraestruturas de chave pública

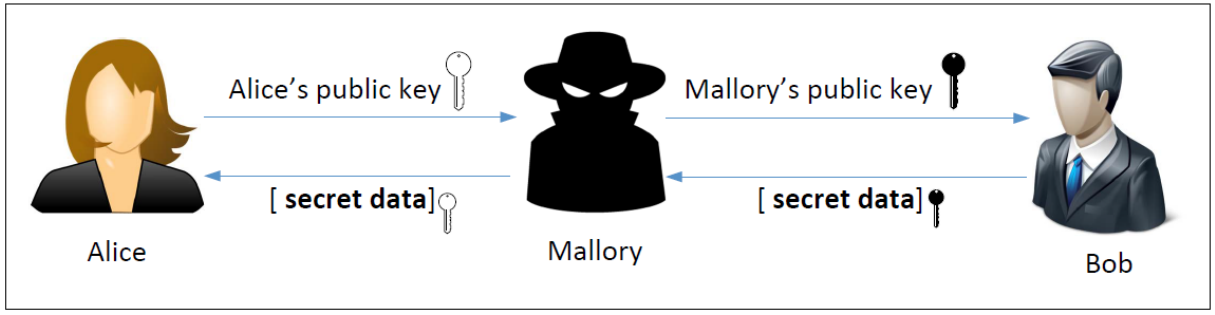
- Cerificados
- Caminhos de certificação
- Perfil PKIX
- Certificados e PKIX na JCA

ISEL – Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1 | 1959-007 Lisboa

Autenticação de chaves públicas

- Autenticidade de chaves públicas
  - "A chave *Key* pertence a *Name*?"

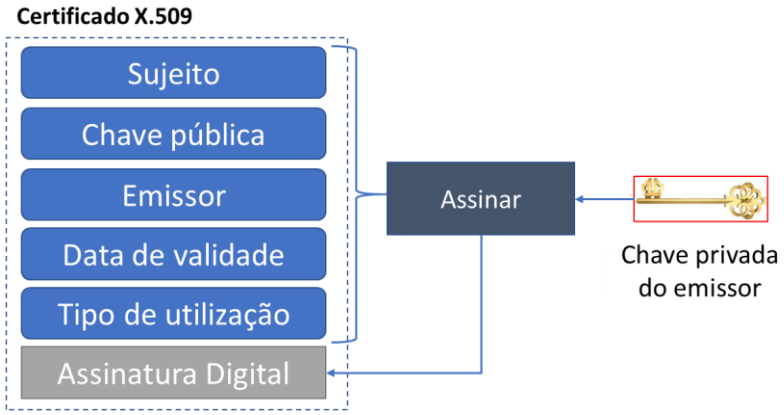
Exemplo de problema conhecido como *man-in-the-middle*



Adaptado de *seed labs*

- Se *Bob* usar as chaves públicas de *Mallory* como sendo de *Alice*
  - *Mallory* pode decifrar as mensagens enviadas para *Alice*, ou
  - *Mallory* pode modificar mensagens de *Alice* e assinar com a chave privada de *Mallory*
- As chaves públicas tem de ter garantia de autenticidade
  - Certificados: associação autenticada entre *identidade* e *chave pública*

Certificados: introdução

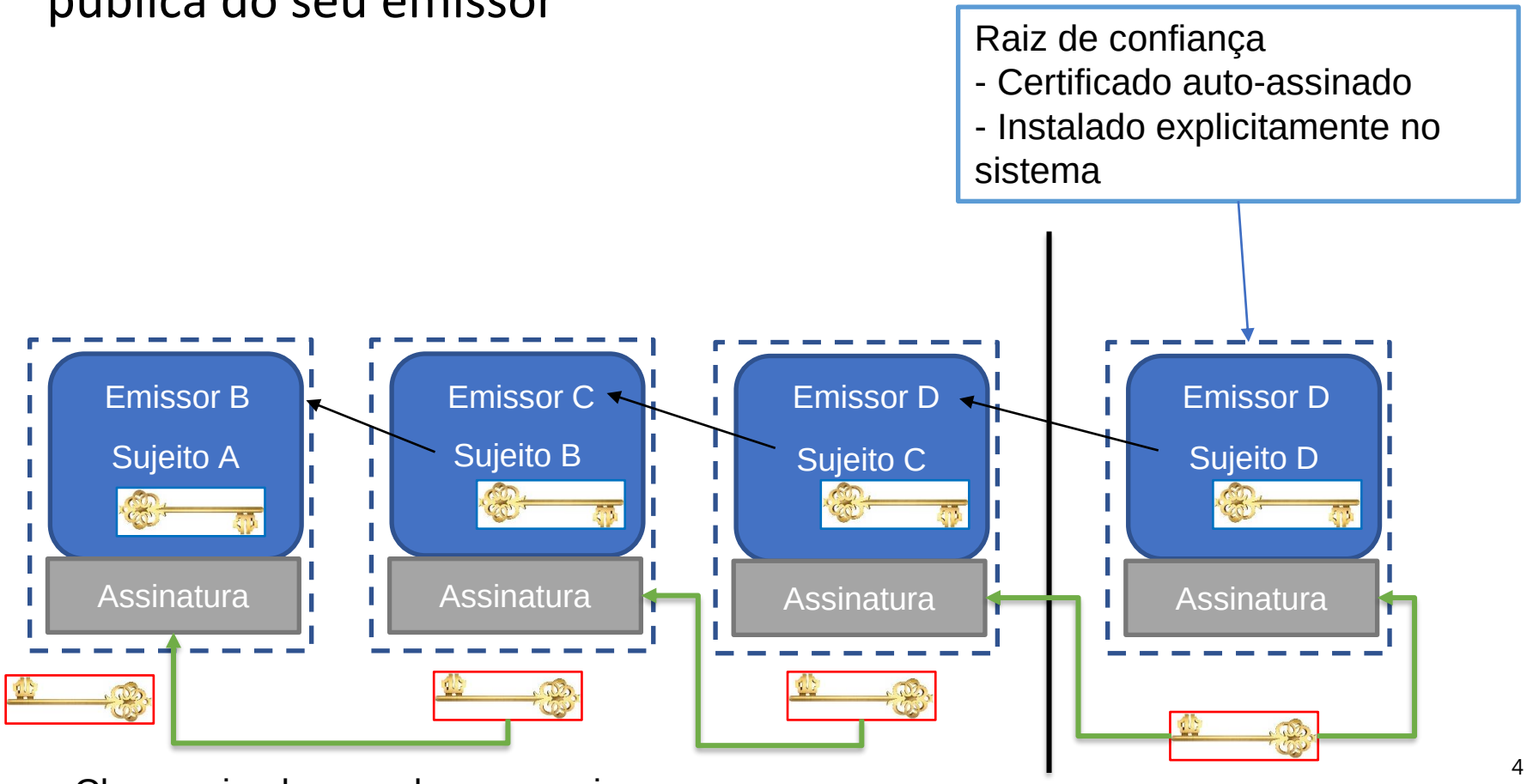


- Constituição dum certificado
  - Quem certifica – emissor
  - O que certifica
  - Outros atributos – validade, condições de aplicabilidade
  - Assinatura do emissor

- Certificados X.509
  - Quem certifica (emissor): Autoridade de Certificação (AC)
  - O que certifica: associação entre uma *chave pública* e um *nome* (identidade)
  - Outros atributos – validade, usos da chaves, extensões
  - Assinatura do emissor – assinatura digital realizada com a chave de assinatura (privada) do emissor

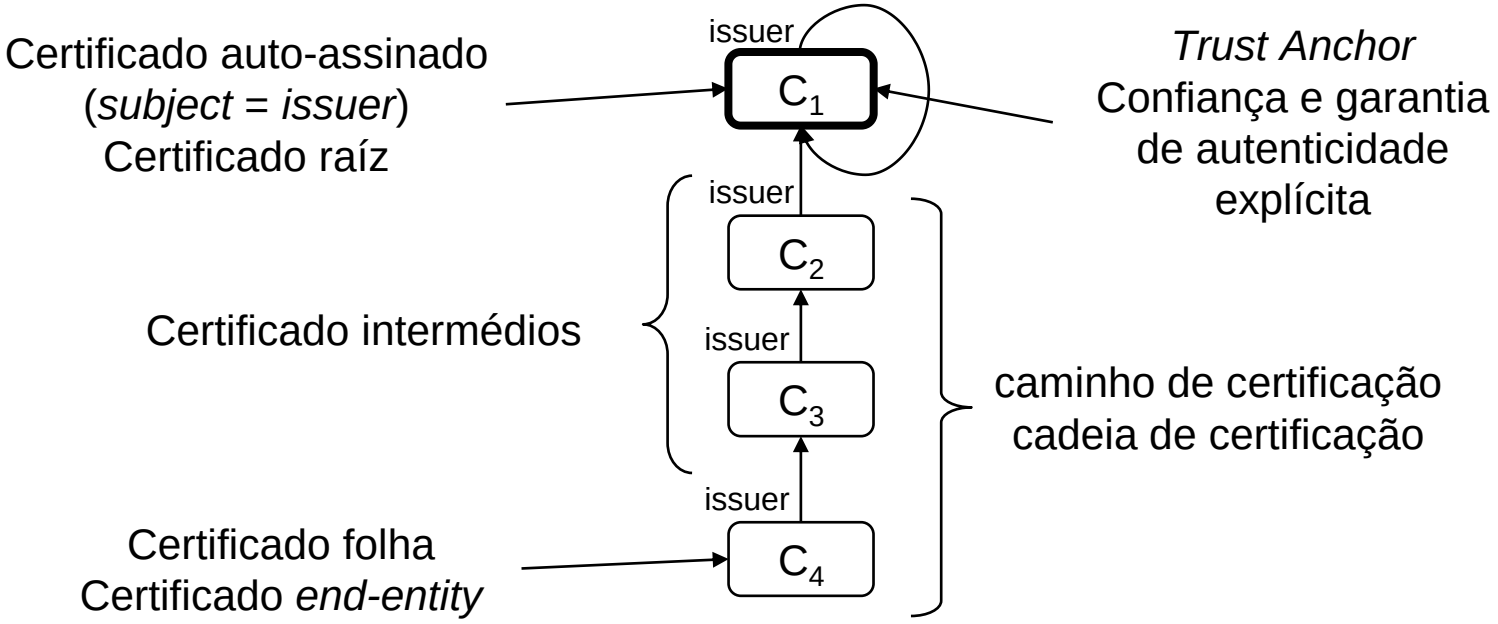
Verificação da cadeia de certificação

- Para verificar a assinatura de um certificado é preciso a chave pública do seu emissor

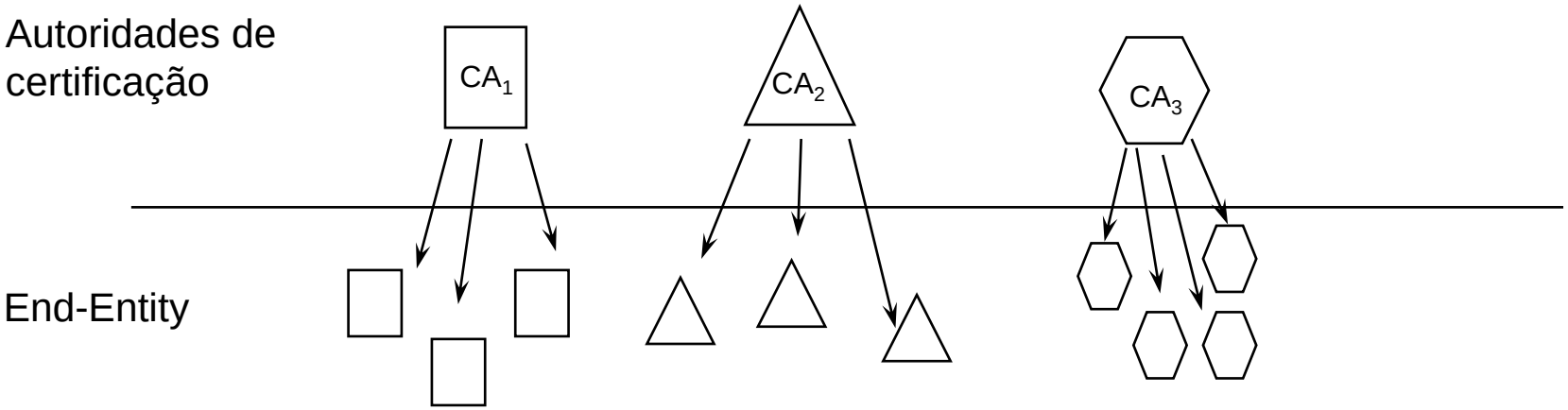


Caminho de certificação

- Recursão
  - Obter chave pública ⇒ validar certificado ⇒ obter chave pública (do *issuer*)
- Condição de paragem
  - *Trust anchor* - Certificado auto-assinado (*issuer* = *subject*)

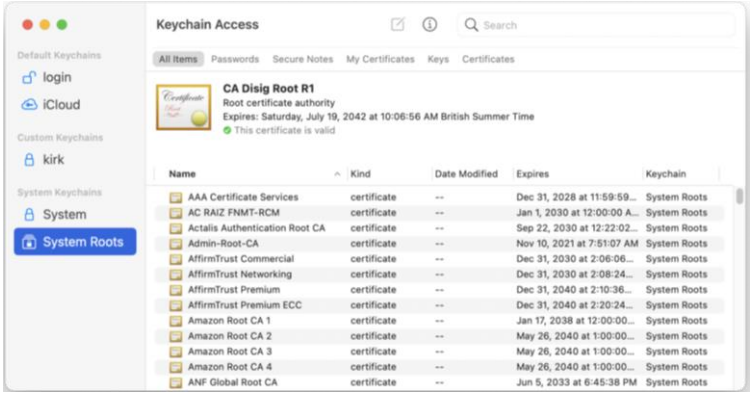


Modelo de domínios separados

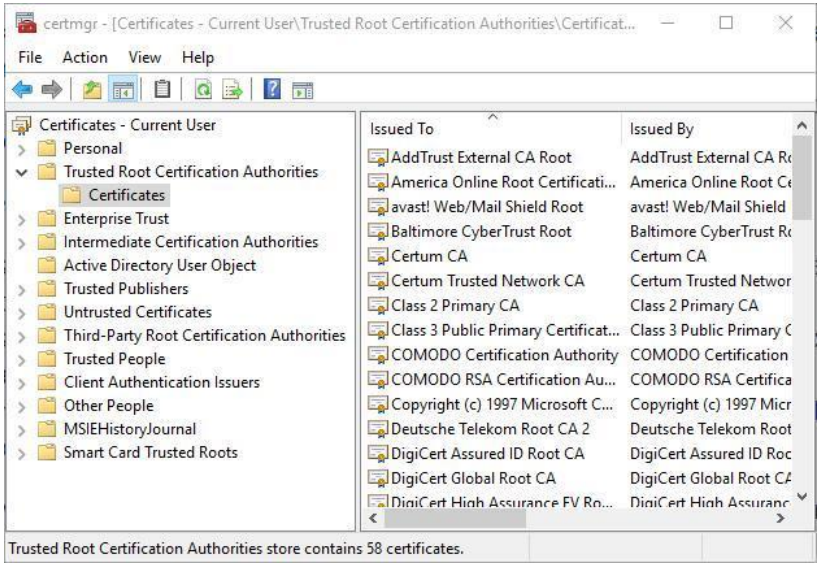


- Para existir interoperabilidade, todos os utilizadores devem confiar em todas as autoridades de certificação
- Modelo usado na *Internet* – não existe CA central

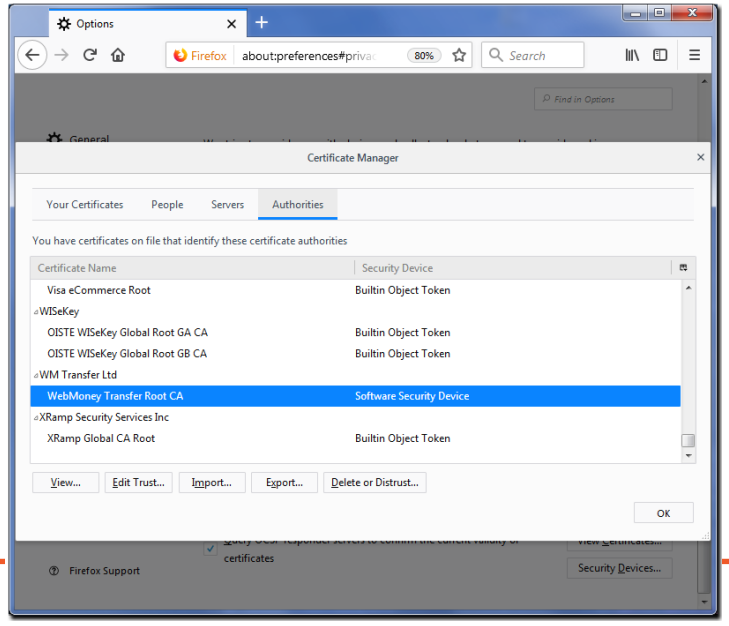
Exemplos de repositórios de confiança



MacOS



Windows



Firefox

Certificados e chaves privadas

- Os certificados guardam apenas a chave pública
- A chave privada fica "associada" ao certificado em armazenamento próprio
- Exemplos
  - Cartão de cidadão
    - certificado pode ser exportado mas a chave privada não saí da memória do cartão
    - operações "privadas", decifra e assinaturas, são feitas pelo hardware do cartão
  - Formatos para guardar chaves privadas
    - PEM format, https://tools.ietf.org/html/rfc7468
    - PKCS#12 / PFX, https://tools.ietf.org/html/rfc7292
  - Certificados de servidores aplicacionais
    - browser recebe apenas a chave pública
  - Certificados da série de exercícios

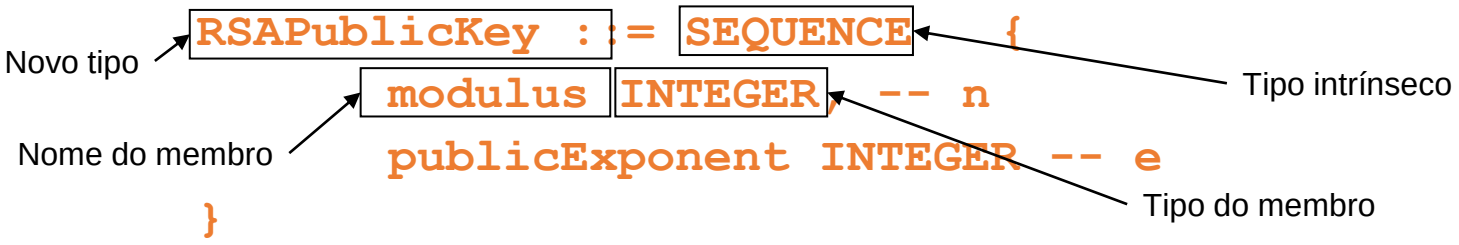
Demos

- Visualização de certificados (folhas e raízes)

ISEL – Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1 | 1959-007 Lisboa

Síntaxe - ASN.1

- *Abstract Syntax Notation 1*
  - Sintaxe e regras para a especificação de objectos abstractos
- Regras de codificação
  - Forma de representar os objectos abstractos como sequências de *bits*
  - DER – *Distinguished Encoding Rules*
  - BER – *Basic Encoding Rules*
- *Object Identifier* (OID)
  - Identificador único constituído por uma sequência de inteiros que representa uma hierarquia
  - Ex.: RSA "1.2.840.113549.1.1.1"
- Exemplo
  - chave pública RSA (norma PKCS #1)



Certificado X.509: constituição (1)

- Certificado

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

- Informação assinada

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

Certificado X.509: constituição (2)

- Validade

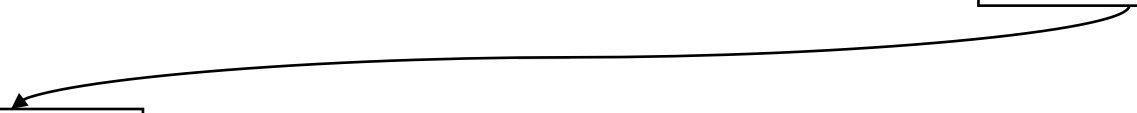
```
Validity ::= SEQUENCE {
  notBefore      Time,
  notAfter       Time
}
```

- Chave pública

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

- Extensões

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



```
Extension  ::=  SEQUENCE  {
  extnID       OBJECT IDENTIFIER,
  critical     BOOLEAN DEFAULT FALSE,
  extnValue    OCTET STRING
}
```

Extensões

- A versão 3 da norma X.509 acrescenta *extensões* à informação assinada (`tbsCertificate`)
- As extensões são a forma normalizada de acrescentar informação não considerada na norma base
- Constituição duma extensão:
  - Identificador da extensão
  - Valor da extensão
  - *flag critical* (se verdadeira, a extensão não pode ser ignorada)
- Perfil
  - Conjunto de extensões e respectiva semântica, usados num domínio de aplicação
  - ex.:
    - PKIX - *Public Key Infrastructure for the Internet*

Perfil PKIX

- Algumas extensões:
  - *Authority Key Identifier* – identificador da chave do emissor
  - *Subject Key Identifier* – identificador da chave do *subject*
  - ***Key Usage* – usos permitidos para o par de chaves**
  - ***Alternative Name* – nome alternativo (email, IP, URI)**
  - *Policy Identifiers* – identificador de política
  - ***Basic Constraints* – restrições ao uso do certificado**
  - *Name Constraints* – restrições ao espaço de nomes do certificado
  - *Policy Constraints* – restrições de política
  - *Extended Key Usage* – usos permitidos para o par de chaves
  - ***CRL Distribution Points* – pontos de distribuição das listas de revogação**

https://tools.ietf.org/html/rfc5280

KeyUsage

- Usos permitidos para o par de chaves

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


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

Subject Alternative Name

- Nome alternativo para o *subject*

```
id-ce-subjectAltName OBJECT IDENTIFIER ::=  { id-ce 17 }

SubjectAltName ::= GeneralNames

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
        otherName                   [0]     OtherName,
        rfc822Name                  [1]     IA5String,
        dNSName                     [2]     IA5String,
        x400Address                 [3]     ORAddress,
        directoryName               [4]     Name,
        ediPartyName                [5]     EDIPartyName,
        uniformResourceIdentifier   [6]     IA5String,
        iPAddress                   [7]     OCTET STRING,
        registeredID                [8]     OBJECT IDENTIFIER }
```

Basic Constraints

- A extensão *basic constrains* indica se o *subject* é uma autoridade de certificação e qual a maior dimensão do caminho de certificação

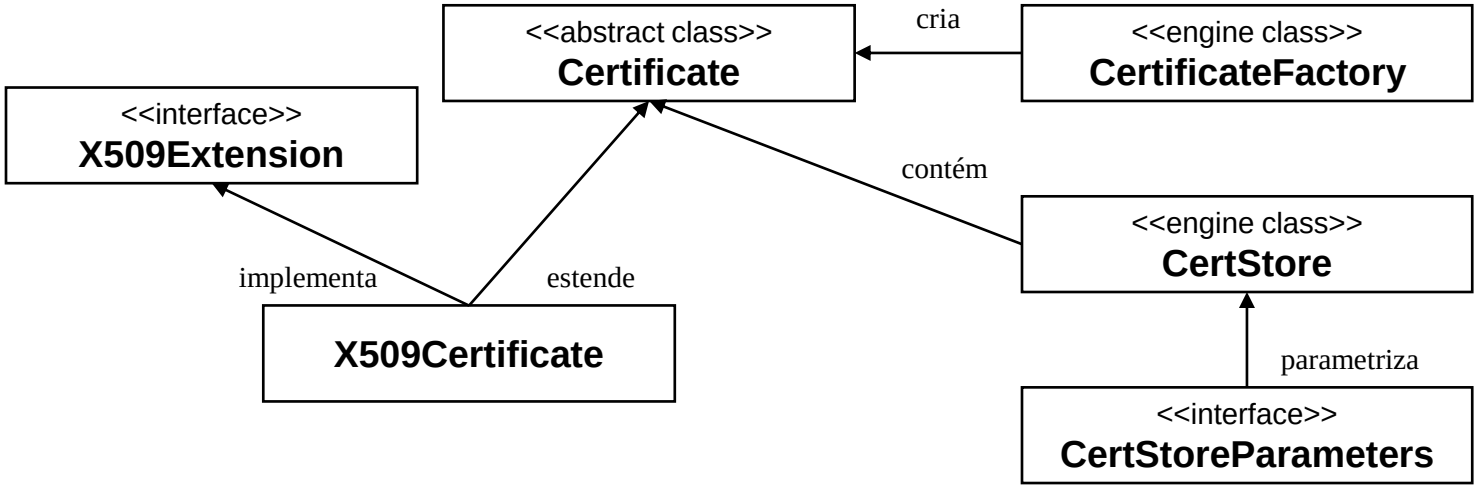
```
id-ce-basicConstraints OBJECT IDENTIFIER ::=  { id-ce 19 }

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

Certificados e PKIX na JCA

ISEL – Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1 | 1959-007 Lisboa

Classes *Certificate* e *CertificateFactory*

- **Certificate** – classe para a representação abstracta de certificados
- **X509Certificate** – estende a classe **Certificate** para o caso concreto de certificados X.509
- **CertificateFactory** – *engine class* para a criação de certificados ou cadeias de certificados com base na sua representação codificada, tipicamente em *stream*.
- **X509Extension** – Interface com métodos de acesso a todas as extensões presentes num certificado X.509



Key Stores

- Armazenamento de chaves e certificados
- Representação através da *engine class* **KeyStore**
- Três tipos de entrada
  - Chaves privadas e certificados associados (incluindo a cadeia)
  - Chaves simétricas
  - Certificados representando *trust anchors*
- Protecção baseada em *passwords*
  - Integridade de todo o repositório - uma password por repositório
  - Confidencialidade das entradas contendo chaves privadas ou secretas - uma password por entrada do repositório
- Formatos de ficheiro (tipos de *provider*)
  - JKS – Formato proprietário da *Sun*
  - JCEKS – Evolução do formato JCE com melhor protecção
  - PKCS12 – Norma PKCS#12 (usada nos ficheiros .pfx criados pelo *Windows*)

Entradas

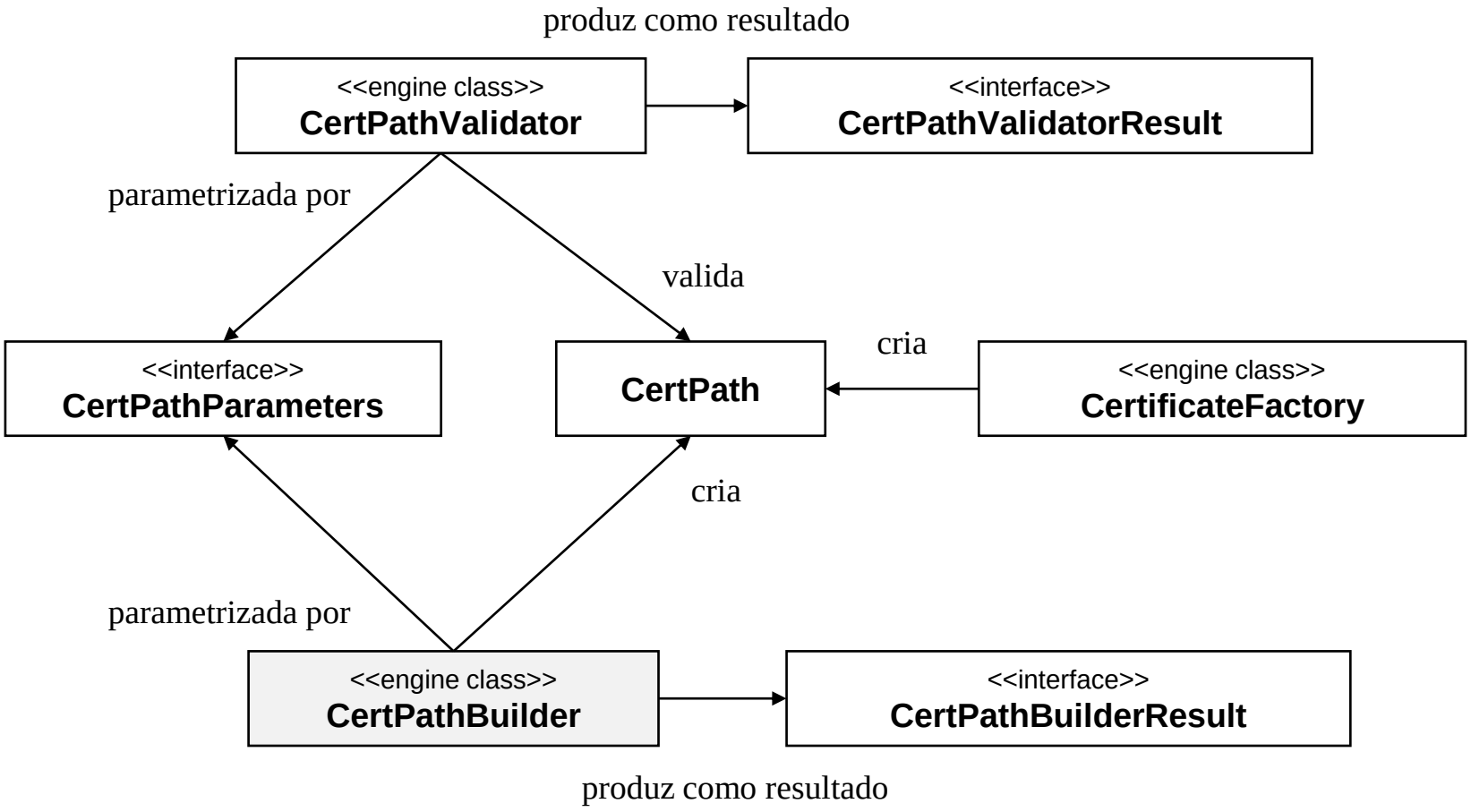
- Cada entrada tem associado um *alias*, do tipo **String**
- Interface base **Entry**

- **PrivateKeyEntry -** Chaves privadas e certificados associados
  - **ctor(PrivateKey, Certificate[])**
  - **Certificate getCertificate()**
  - **Certificate[] getCertificateChain()**
  - **PrivateKey getPrivateKey()**
- **SecretKeyEntry -** Chaves simétricas
  - **ctor(SecretKey)**
  - **SecretKey getSecretKey()**
- **TrustedCertificateEntry -** *Trust anchors*
  - **ctor(Certificate)**
  - **Certificate getTrustedCertificate()**

Classe KeyStore

- *Load* e *store*
  - **void load(InputStream, char[] password)**
  - **void load(LoadStoreParameter)**
  - **void store(OutputStream, char[] password)**
  - **void store(LoadStoreParameter)**
- Listagem
  - **Enumeration<String> aliases()**
- Acesso a entradas
  - **Entry getEntry(String alias, ProtectionParameter)**
  - **void setEntry(String alias, Entry, ProtectionParameters)**
  - **boolean is*Xxx*Entry(String alias)**
- Métodos especializados
  - **Key getKey(String alias, char[] password)**
  - …

Construção e validação de cadeias (1)

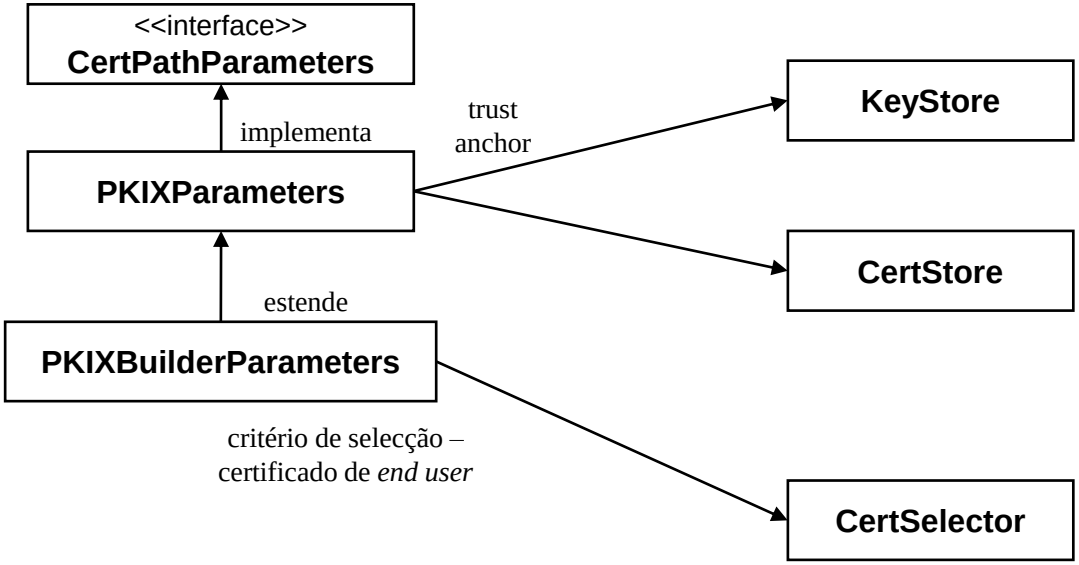


Construção e validação de cadeias (2)

- CertPathValidator
  - public final **CertPathValidatorResult**
    validate (**CertPath certPath**, **CertPathParameters params**)
      throws **CertPathValidatorException**,
                InvalidAlgorithmParameterException

- CertPathBuilder
  - public final **CertPathBuilderResult**
    build (**CertPathParameters params**)
      throws **CertPathBuilderException**,
                InvalidAlgorithmParameterException

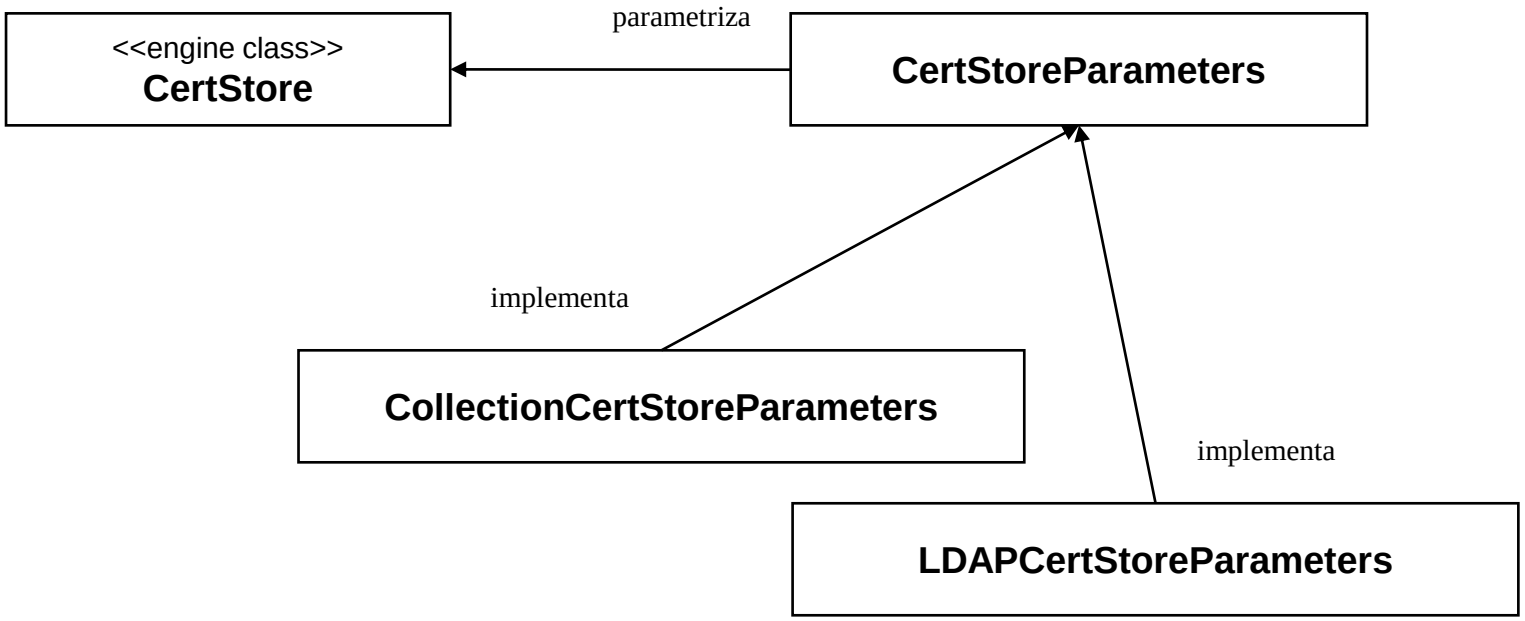
Parametrização (1)



Parametrização (2)

- Construção:
  - Conjunto de *trust anchors* – define os *issuers* dos certificados iniciais da cadeia
  - Conjunto de certificados (**`CertStore`**) – define os certificados que podem constituir a cadeia
  - Selector (X509CertSelector) – define os requisitos (ex. nome do *subject*) para o certificado final (*end user*)

Classe *CertStore*



Classe *CertSelector*