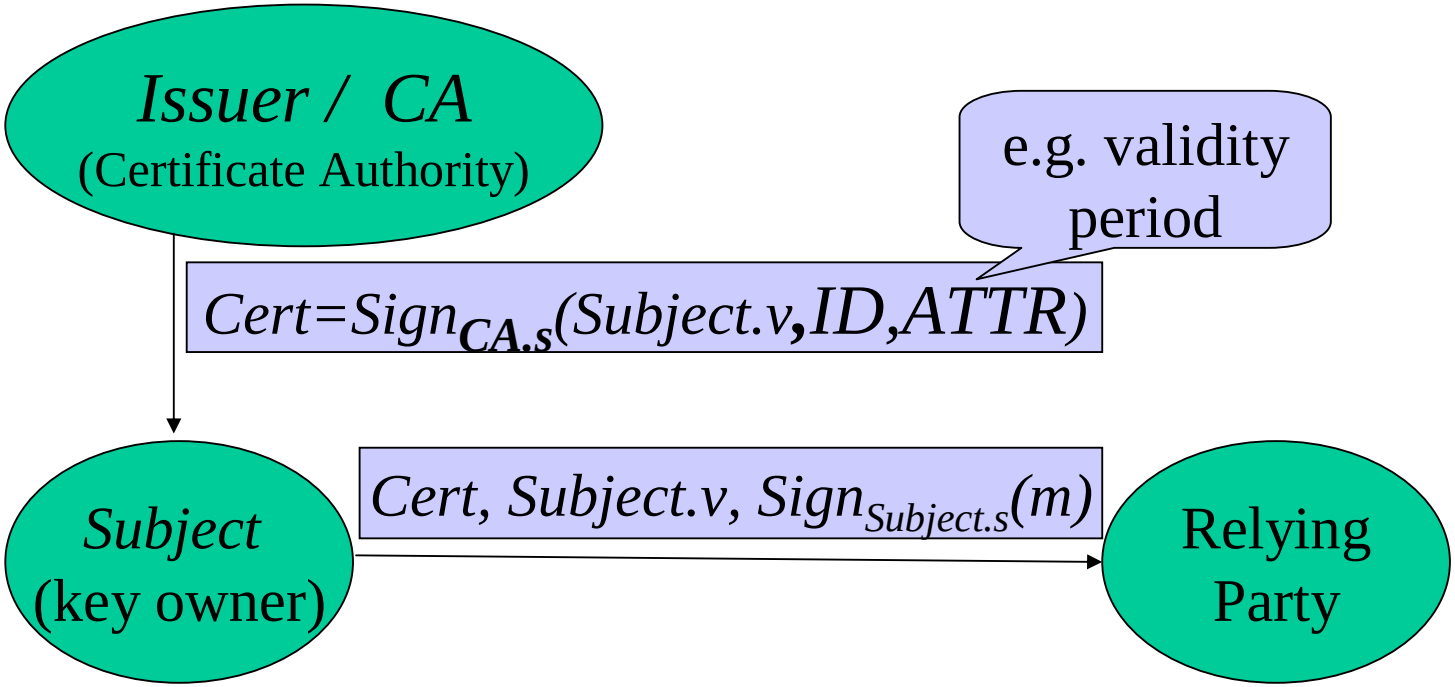
PKI

PPKE, ITK
Csapodi Márton

Public Key Certificates & Authorities

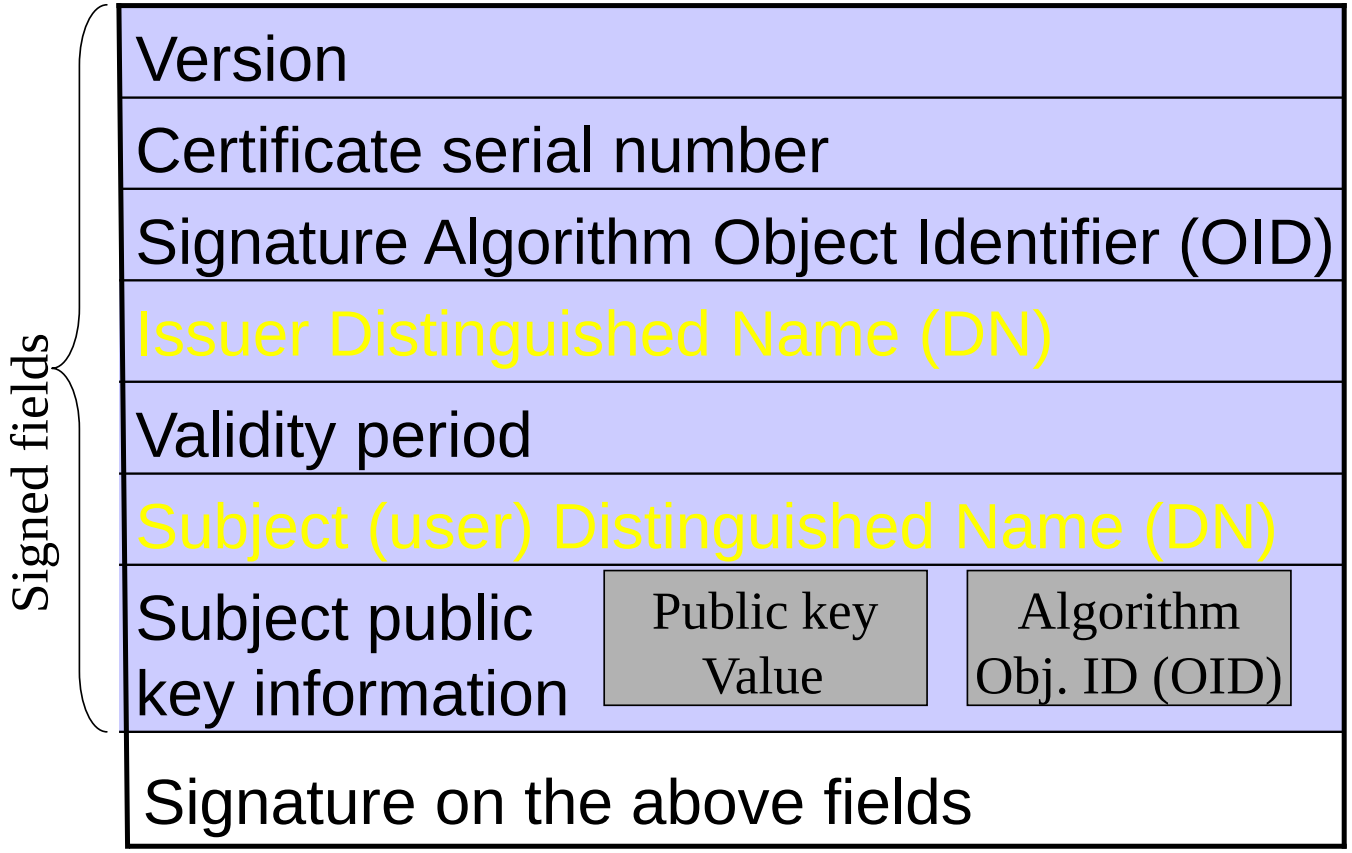
- *Certificate:* signature by Certificate Authority (CA) over subject's public key and attributes
- Attributes:
  - Validated by CA (liability?)
  - Used by **relying party** for decisions (e.g., use this website?)
  - **Questions:** Attributes? Identifiers? Format? …



X.509 public key certificates

- Public key **signed** by (trusted) **issuer (CA)**
  - Certificate: signed public key (and attributes)
  - CA: Certificate Authority (issuer)
- **X.509:** ITU's standard for certificates & usage
  - Widely adopted – in spite of complexity
- Main outcome of X.500 standard
  - ITU: International Telcos Union
  - Goal: trusted, centralized 'phone directory'
  - Global directory? No; but – X.509 widely used
    - Why global directory failed? Too complex, revealing
  - Identifiers: distinguished names
    - Goals: unique, meaningful, decentralized identifiers

Original (V1) X.509 Certs



Object Identifiers (OID):

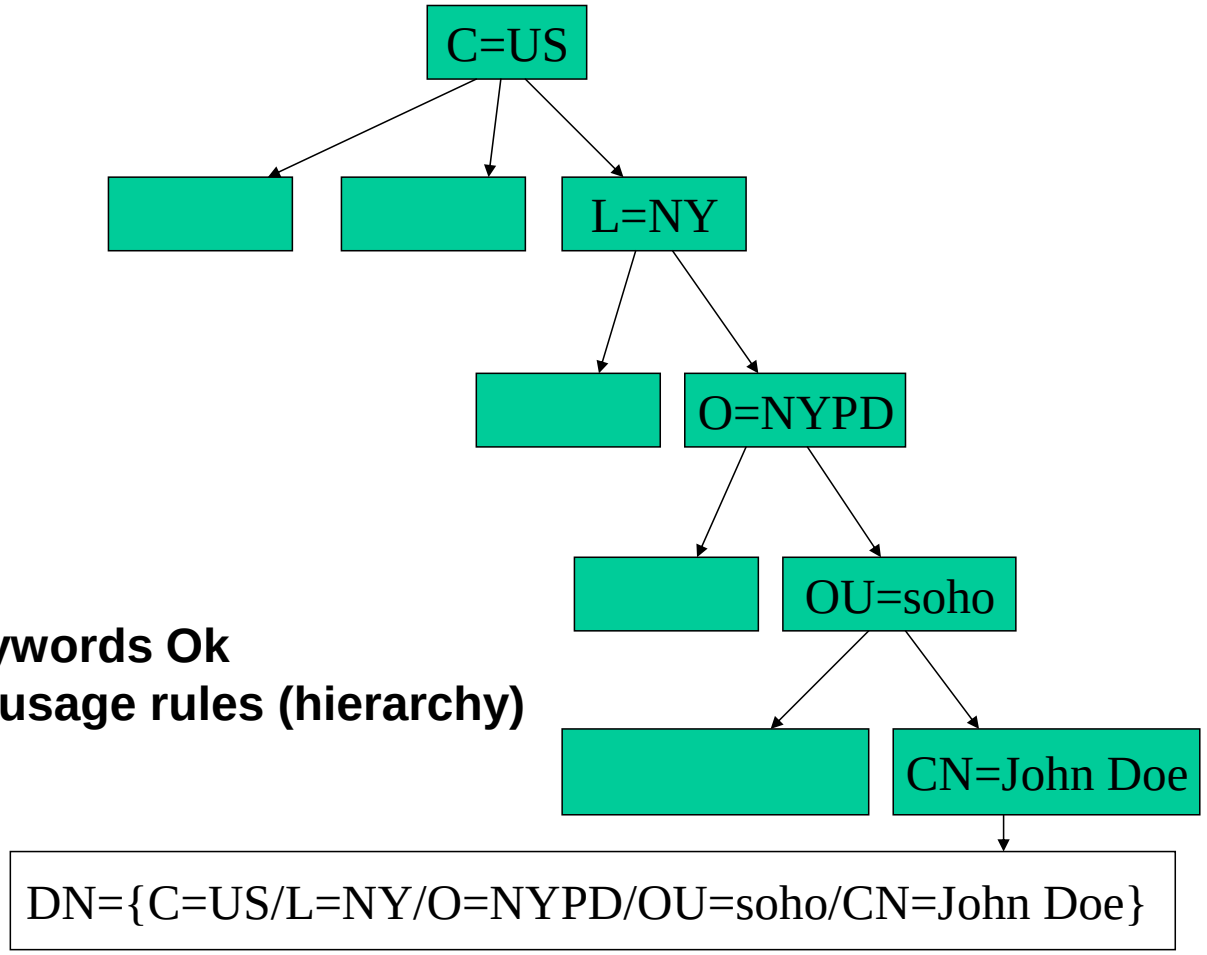
- Global, unique identifiers
- Sequence of numbers, e.g.: 1.16.840.1.45.33
  - Hierarchical

X.509 Distinguished Names (DN)

- Goal: meaningful, unique and decentralized identifiers
- Sequence of keywords, a string value for each of them
- Distributed directory, responsibility →*hierarchical DN*

Keyword	Meaning
C	Country
L	Locality name
O	Organization name
OU	Organization Unit name
CN	Common Name

Distinguished Name (DN) Hierarchy

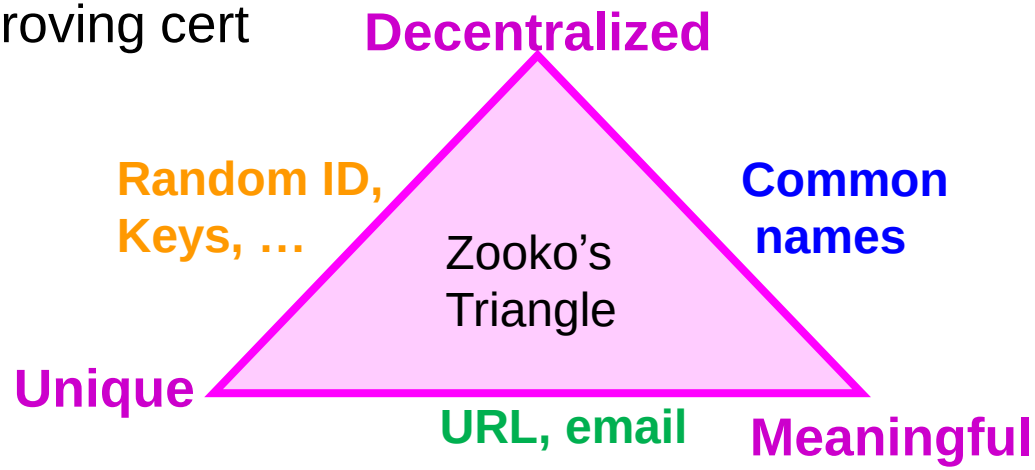


Comments:

1. **Other keywords Ok**
2. **No strict usage rules (hierarchy)**

Goals for Identifiers in Certificates

- Meaningful (to humans)
  - Memorable, reputation, off-net, legal
- Unique identification of entity (owner)
- Decentralized - with Accountability: assigned by any trusted certificate authority
  - Accountability: CA approving cert
- Pairs are easy:
  - Unique + Meaningful
  - Meaningful + Decentalized
  - Unique + Decentralized



- Zooko: can't have all three properties

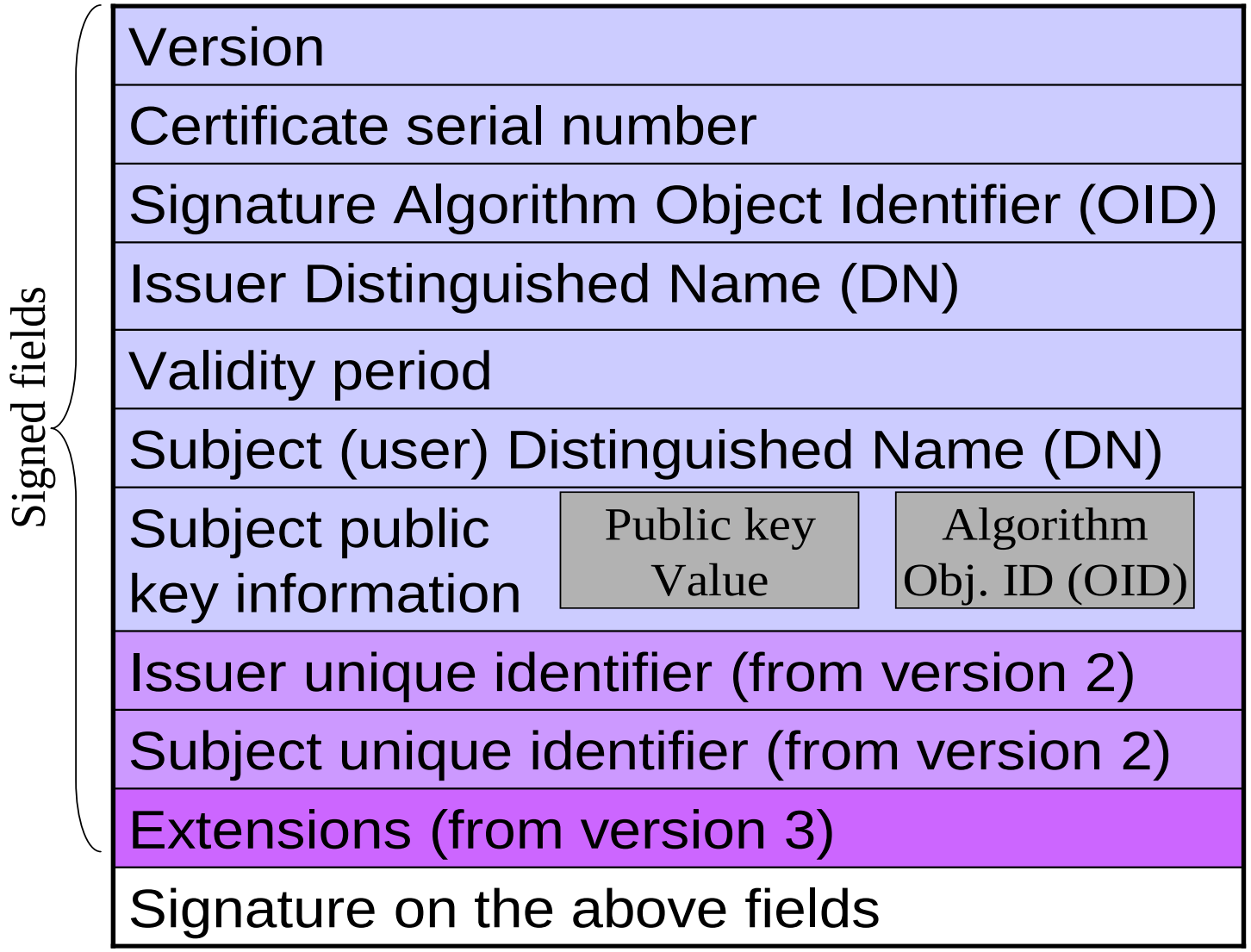
Distinguished Names - Evaluation

- **Decentralized?** 
  - Sure: any CA can select DN for its customers, sign cert
- **Unique ?** 
  - Could be, if each name space has one issuer
  - TLS reality: browsers trust 100s of CAs for **all** DN
- **Meaningful?** 
  - Usually: Julian Jones/UK/IBM
  - But not always: Julian Jones2/UK/IBM
    - Added ‘counter’ to distinguish → mistakes, loss of meaning
- X.509 response: v2: unique ID, v3: extensions

X.509 Certs & Subject Identifiers

- V1: Distinguished Name (for subject & issuer)
- V2: unique identifiers (for subject & issuer)
- V3: extensions
  - PKIX standard: SubjectAltName extension
    - Including DNSname
    - PKIX: Public Key Infrastructure working group of IETF
    - Widely adopted, including in SSL/TLS (& https)

X.509 Public Key Certificates

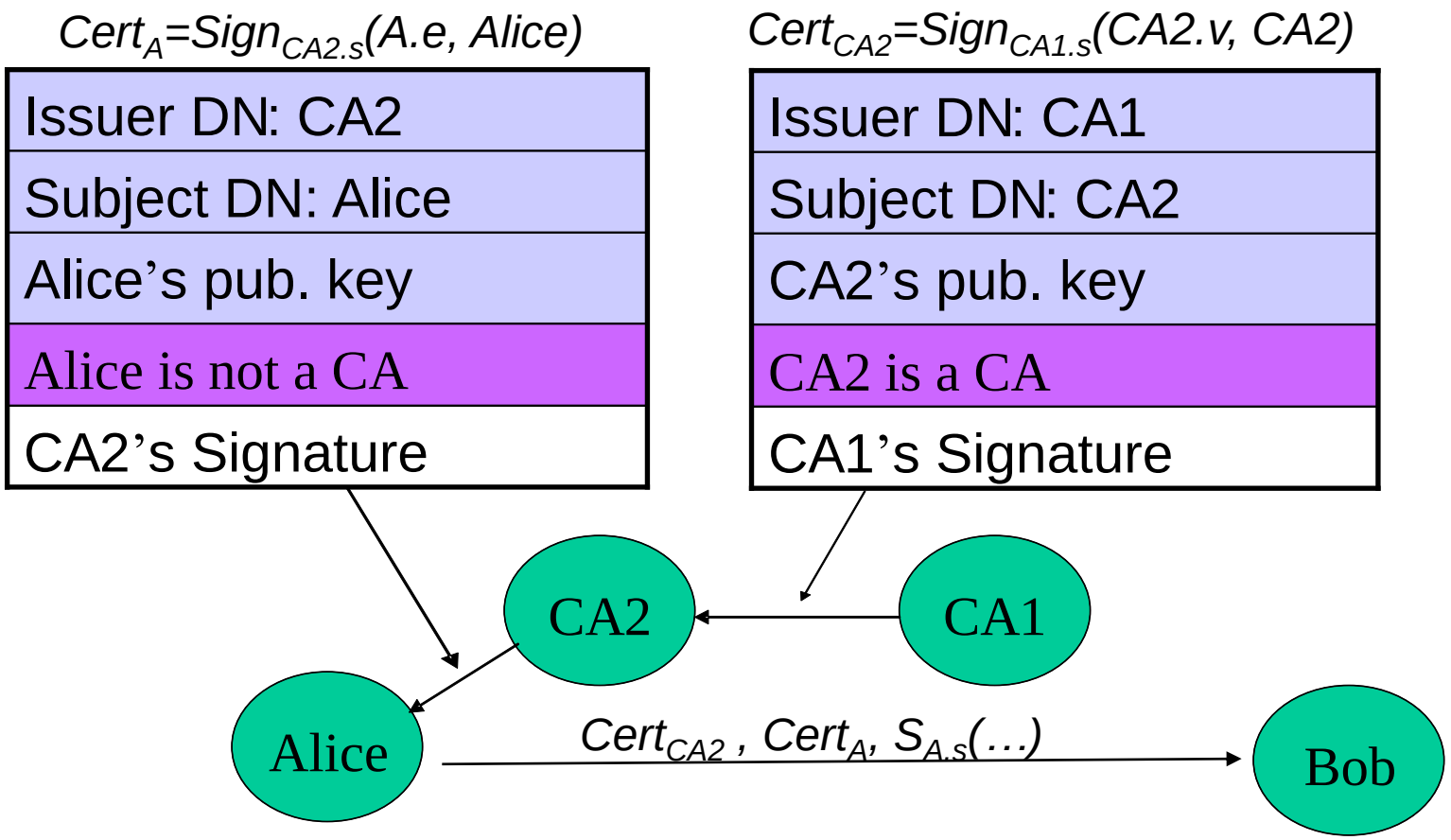


X.509 V3 Extensions Mechanism

- Each extension contains…
- Extension identifier
  - As an OID (Object Identifier)
  - E.g. `Naming constraints`
- Extension value
  - E.g. `Include C=IL`, `exclude dNSName=*.IBM.COM`
- Criticality indicator
  - If critical, relying parties MUST understand extension to use certificate
    - E.g. Naming constraints is `critical`
  - If non-critical, Ok to use certificate and ignore extension

Certificate Path

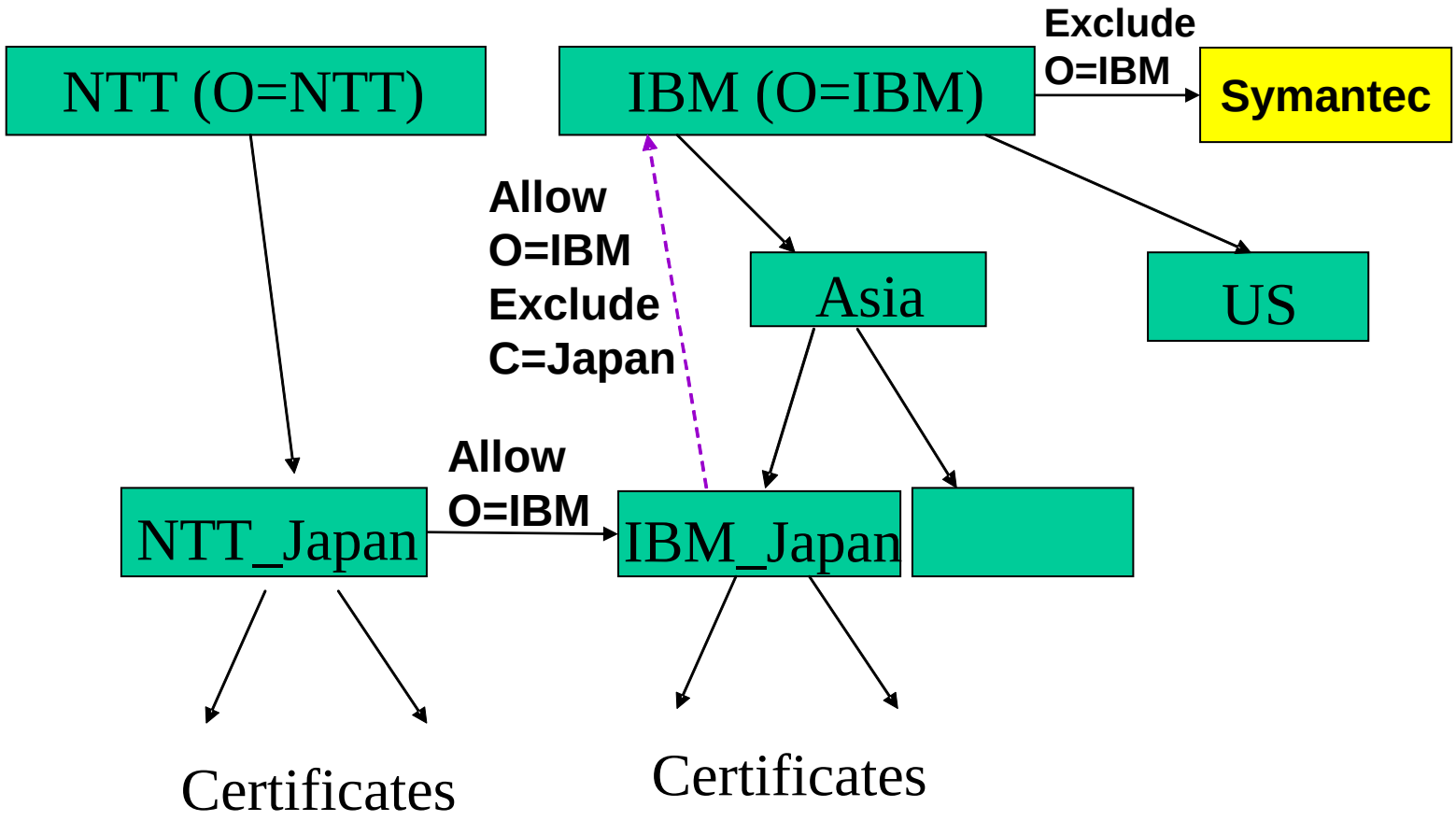
- Suppose relying party (browser) does not trust subject's CA…
- Solution: Certificate Path – a trusted CA certifies subject's CA



X.509v3/PKIX Standard Extensions

- Most important: Naming and Constraints extensions
- Certification path constraints extensions:
  - Basic constraints:
    - Goal: mark the (normal) case: subject isn't CA
    - `CA`: Subject is CA or end entity
    - `CertPathLength`
  - Naming constraints
    - Constraints on `DN` – in certs issued by subject
      - Only relevant when subject is a CA !
    - 'Allow' and 'Exclude'

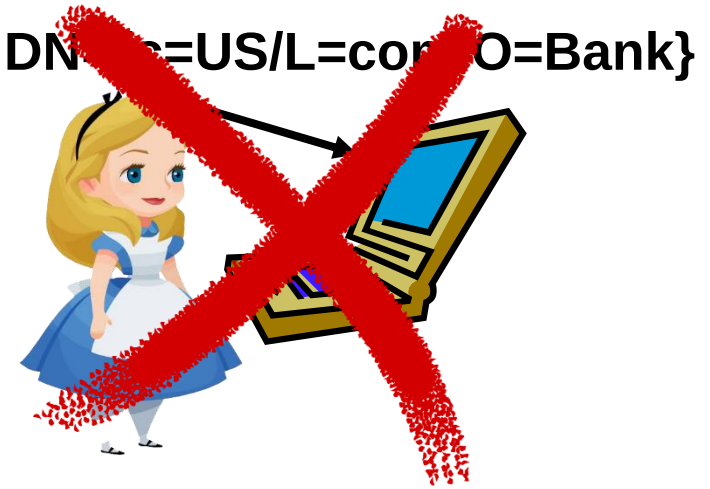
Applying naming constraints



- **NTT JP allows IBM JP to certify IBMers**
- **IBM JP allows IBM to certify all IBMers, except of IBM JP**
- **IBM trusts Symantec's certificates, except for O=IBM**

Reality: DNs aren’t usable identifiers

- Relying parties (users) don’t know the DN



- Hopefully, they know the domain (in URL)
- Naming extensions: alternative names
  - For TLS: `cert.SubjectAltName.dNSname`
  - Possible values: bank.com, *.bank.com (wildcard), …
  - May use also in naming constraints

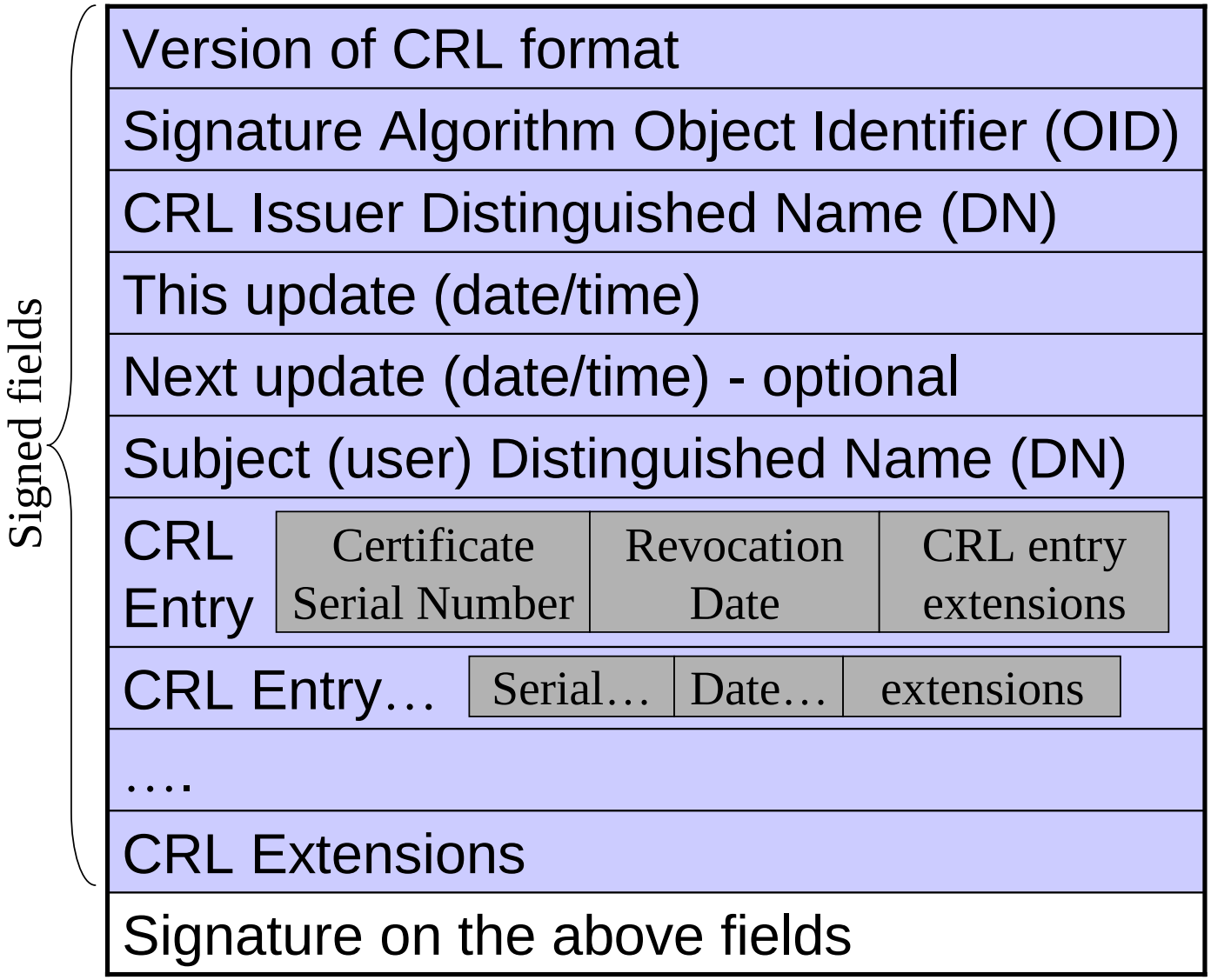
SSL / TLS PKI Challenges

- Many CAs `trusted' in browsers
- Every CA can certify any domain (name)
  - Since naming constraints NOT used
  - Two CAs can same name (equivocation)
  - To detect bad-CA: must find bad-certificate
    - No public, auditable log of certificates
- Several well-known failures
  - DigiNotar, Comodo, Stuxnet, …

Certificate Revocation

- Reasons for revoking certificate
  - Key compromise
  - CA compromise
  - Affiliation changed (changing DN or other attribute)
  - Superseded (replaced)
  - Cessation – not longer needed
- How to inform relying parties?
  - Do not inform – wait for end of (short?) validity period
  - Distribute *Certificate Revocation List (CRL)*
  - Ask - Online Certificate Status Protocol (OCSP)

X.509 CRL Format



Revocation is Difficult

- If CRLs contain all revoked certificates (which did not expire)… it may be huge!
- CRLs are (also) not immediate
  - Who is responsible until CRL is distributed?
  - What is the impact on non-repudiation?
- Solutions:
  - Online Certificate Status Protocol (OCSP)
  - More efficient CRL schemes (usually CRL extensions)
    - CRL distribution point – split certificates to several CRLs
    - Authorities Revocation List (ARL): list only revoked CAs
    - Delta CRL – only new revocations since last `base CRL`
    - Certificate Revocation Tree (more later)
  - Short validity for certificates

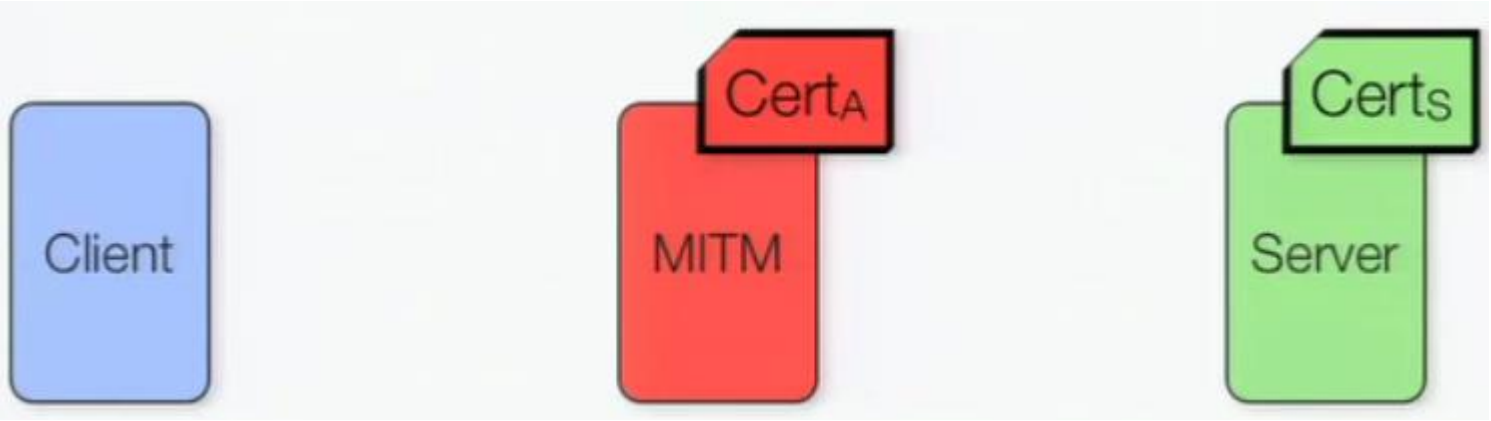
Short-Term Certificates

- Idea: short validity period of certificates, so no need to revoke them
- Concern: overhead of signing many certificates each (short) period
- Solutions:
  - Extend many certs with one signature: *hash tree*
    - $Sign_{CA.s}(date, valid{:}h(h(cert_A),h(cert_B),\ldots))$
    - Certificate revocation tree:
      $Sign_{CA.s}(date, all\ except{:}h(h(cert_A),h(cert_B),\ldots))$
  - Certificates includes a *hash chain*, e.g. for Jan 2005:
    $Cert_A=Sign_{CA.s}(A.s,"Alice",2005,h^{(11)}(x)))$
    - And for Feb 2005: $Cert_A$, $h^{(10)}(x)$
    - Validate incoming $Cert_A$, h_{10} by $h^{(11)}(x)=h(h_{10})$
    - Security based on random choice of x and h being one-way premutation
  - Often, requiring frequent CRL is more efficient

SSL / TLS PKI Challenges

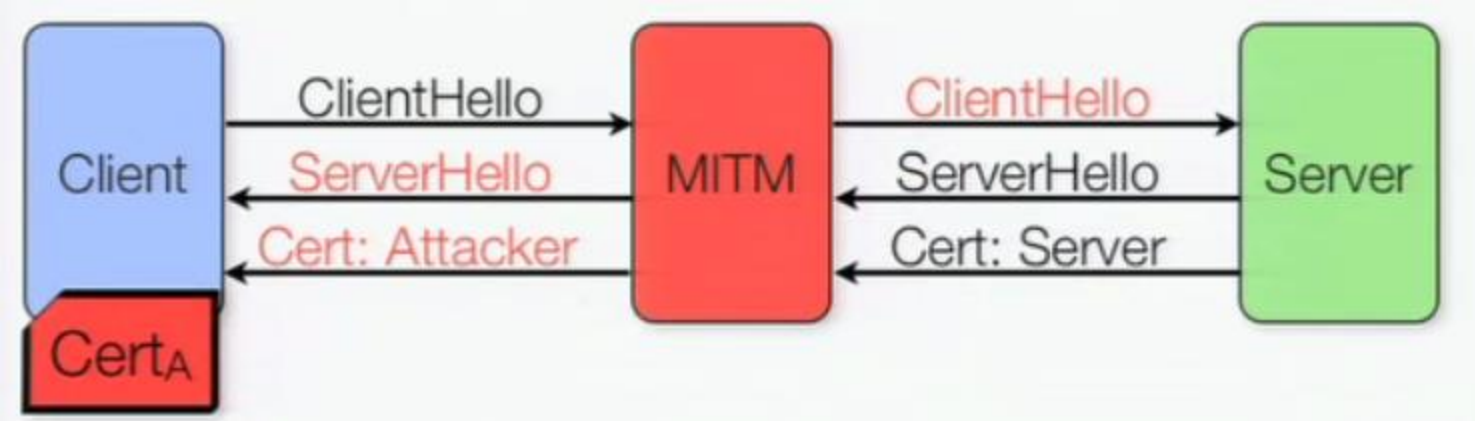
- Many CAs `trusted' in browsers
- Naming constraints NOT used
  - Every CA can certify any domain
- Several well-known failures
- DigiNotar, Comodo, Stuxnet, …

TLS Interception / MitM Attack



CertA is a fake-but-valid certificate for the identity of Server

TLS Interception / MitM Attack



Interception is used ethically, by 'locally' adding a CA, by many organizations, for filtering SSL/TLS traffic from malware, etc.

But also by attackers…

Defenses against Corrupt CAs

- **Use naming constraints to limit risk**
  - who can issue global TLDs (.com, etc.)?
- **‘Burned-in’ public keys (e.g., for Google)**
  - Detected MitM in Iran, using DigiNotar CA
- **Certificate / public-key pinning (HPKP)**
  - Server: I always use this PK / Cert / Chain
  - Client: remember and implement!
- Certificate Transparency (CT): Accountability
- Origin-bound certificates

Defenses against Corrupt CAs

- Use naming constraints to limit risk
- ‘Burned-in’ public keys (e.g., for Google)
- Certificate / public-key pinning (HPKP)
- **Certificate Transparency (CT): Accountability**
- Threshold schemes

Key Establishment & PKI : Conclusions

- Key Establishment: use PKs, cert for ‘handshake’
  - SSL/TLS: mature standard, widely used
    - Many vulnerabilities in older versions
    - Slow adoption of newer versions
- PKI & Trust: still challenging, active areas
  - SSL/TLS certs: too many legit CAs, no naming constraints
  - How to deal with rogue CAs?
    - Certificate Transparency (accountability) ?
  - Client certificates (authentication) ???