

---

CSE 3400/CSE 5850 - Introduction to Computer & Network  
Security  
/ Introduction to Cybersecurity

Lecture 12  
Public Key Infrastructure

Ghada Almashaqbeh  
UConn

**\*Adapted from the textbook slides**

---

---

# Outline

- ❑ Motivation.
- ❑ Public key infrastructure (PKI) components.
- ❑ PKI goals.
- ❑ X.509 PKI concepts.
- ❑ Intermediate CAs and trust path verification.
- ❑ Certificate revocation

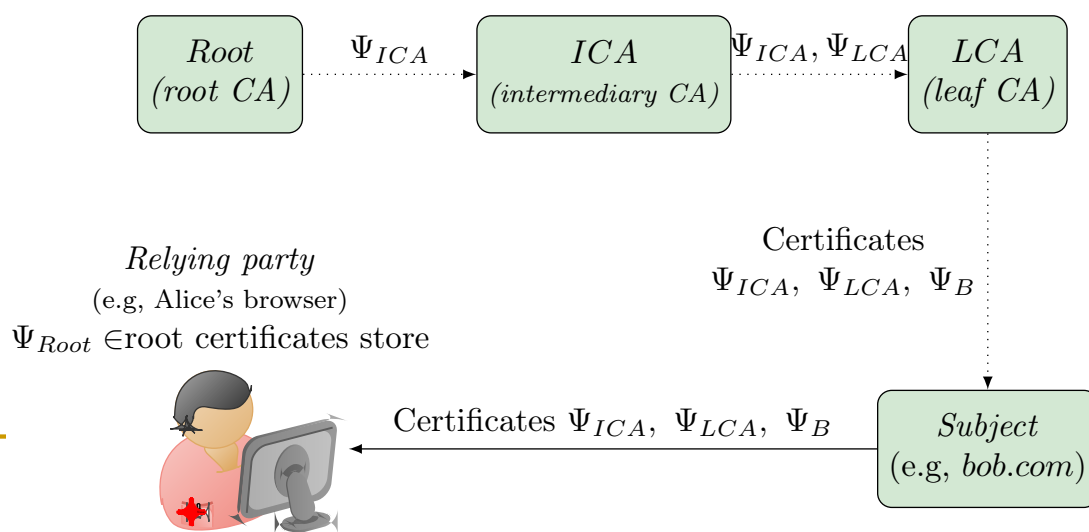
---

# Public keys are very useful...

- Secure web connections
- Software signing (against malware)
- Secure messaging, email.
- Cryptocurrency and blockchains.
- But ...How do we know the public key of an entity? And how can we trust that this entity is indeed who claims to be and owns a specific public key?
  - Mainly: signed by a **trusted Certificate Authority**
- **Public key infrastructure (PKI)** defines how to issue, manage and use such certificates.

# Public Key Certificates & Authorities

- **The big picture:** when receiving a party's (the **subject**) public key, it will be accompanied with a certificate.
  - A valid certificate means that the entity is who claims to be and she owns the corresponding the public key.
- **Certificate:** signature by a **Certificate Authority (CA)** over subject's public key and attributes
  - A chain of trust covering multiple CAs back to the root CA.
- **Attributes:** identity (ID) and others...
  - Validated by CA (liability?)
  - Used by **relying party** for decisions (e.g., use this website?)



---

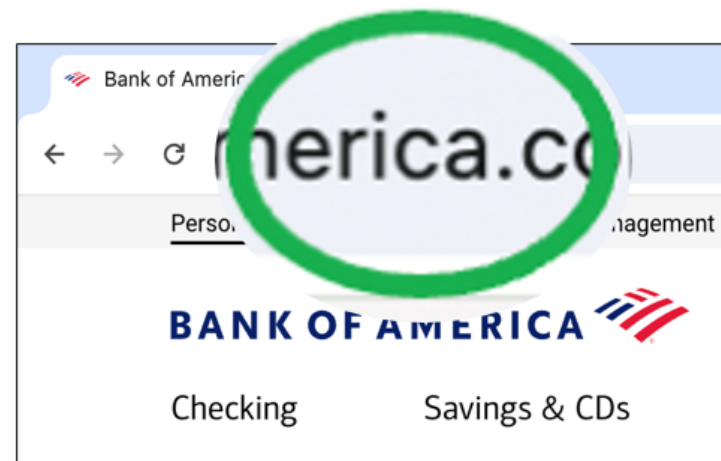
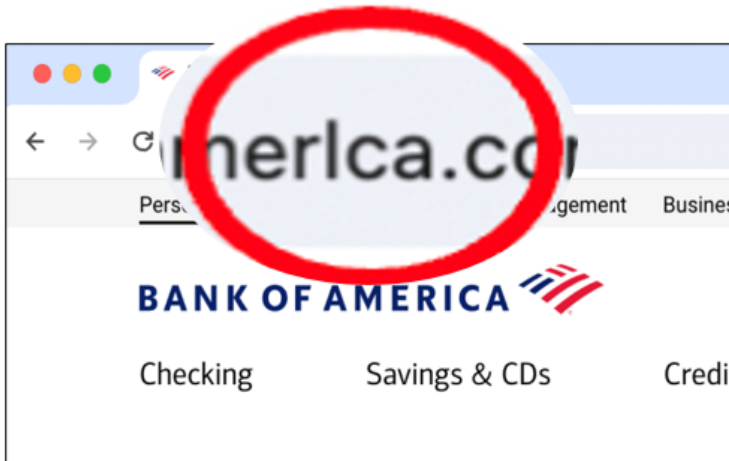
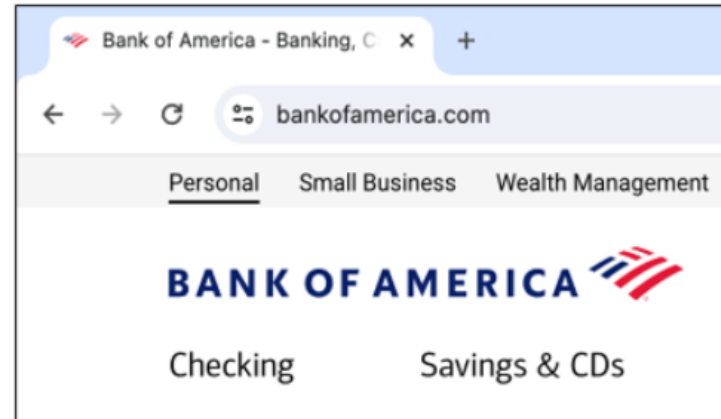
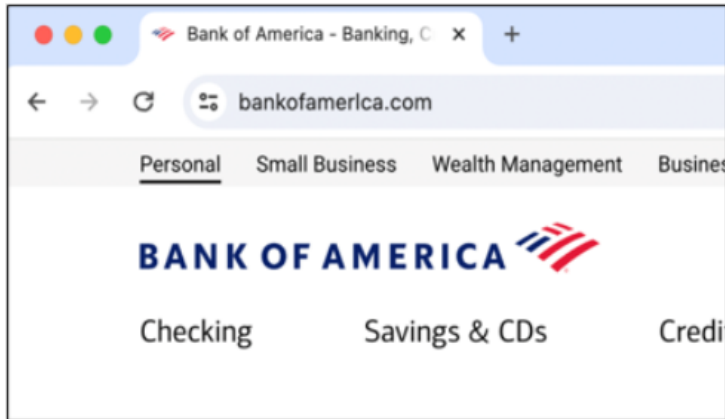
# Certificates are all about **Trust**

- Certificate:  $\psi_{Bob} = \text{Sign}_{CA.S}(\text{Bob.com}, \text{Bob.e}, \dots)$ 
  - CA attests that Bob's public key is *Bob.e*
- Do we **trust** this attestation to be true?
- Special case of **trust management**
  - Important problem far beyond PKI... still not resolved!

# Rogue Certificates

- Rogue certificates: equivocating or misleading (domain) name
- Attacker goals:
  - Impersonate: web-site, phishing email, signed malware..
  - Equivocating (same name): circumvent name-based security mechanisms, such as *Same-Origin-Policy (SOP)*, *blacklists*, *whitelists*, *access-control* ...
  - Name may be misleading even if not equivocating
- Types of misleading names:
  - Combo names: bank.com vs. **accts-bank.com**, **bank.accts.com**, ...
  - Domain-name hacking: accts.bank.com vs. **accts-bank.com**, ... or **accts-bank.co**
  - Homographic: paypal.com [l is L] vs. **paypal.com** [i is I]
  - Typo-squatting: bank.com vs. **banc.com**, **baank.com**, **banl.com**,...

# Example of Homographic Attacks



---

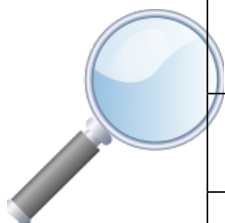
# PKI Failures

- Although the signature over the certificate verifies correctly, there is still a failure and the certificate must be revoked.
  - This is called a PKI failure.
- PKI failures include:
  - Subject key exposure.
  - CA failure.
  - Cryptanalysis certificate forgery.
    - Find collisions in the hash function used in the HtS paradigm,
    - or exploit some vulnerability in the digital signature scheme used for signing.



# Some Infamous PKI Failures

CA, year(s)	Description and Reference
Verisign, 2001	VeriSign issues Microsoft code-signing certificates to attacker [167].
Thawte, 2008	Validation failures of Thawte and StartSSL [455].
Comodo, 2008	CertStar, a reseller of Comodo, issued certificates without validation [299].
Comodo, 2011	Rogue certificates for major sites (e.g., Gmail) [261][291][343].
DigiNotar, 2011	DigiNotar CA compromised, 531 rogue certificates found, including for *.google.com, used for MitM against Iranian users [291][440].
TurkTrust, 2011-2012	TurkTrust issued intermediary CAs certificates to end entities; abused to issue certificate for *.google.com (detected on Dec. 2012) [282].
Trustwave, 2012	Trustwave issued intermediary CA certificate for eavesdropping [368].
ANSSI, 2013	ANSSI (French CA) issued intermediary CA certificate for MitM [445].
NICCA, 2014	Intermediary CA NICCA (India) issued rogue certs for Google domains [284].
CNNIC, 2015	Rogue certificates issued by MCS (Egypt), certified by CNNIC (China) [152][332].
WoSign, 2015	WoSign and StartCom (owned by WoSign) removed from revoked as CAs after validation and other failures
Symantec, 2015-17	Symantec issued unauthorized certs for over 176 domains [352].
DarkMatter, 2019	Mozilla, Google revoke intermediary CA of surveillance firm DarkMatter [68], refuse to make it a root CA.
Let's Encrypt, 2020	Let's Encrypt detected a bug in their CAA-validation code, affecting 3 million certificates [1].
TrustCor, 2022	Root CA TrustCor exposed as related to Spyware [318].



# PKI Goals/Requirements



**Trustworthy issuers:** Trust anchor/root CAs and Intermediary CAs; Limitations on Intermediary CAs (e.g., restricted domain names)



**Accountability:** identify issuer of a given certificate



**Timeliness:** limited validity period, timely **revocation**



**Transparency:** public log of all certificate; no 'hidden' certificates!



**Non-Equivocation:** one entity – one certificate



**Privacy:** why should CA know which site I use?

---

# X.509 Certificates

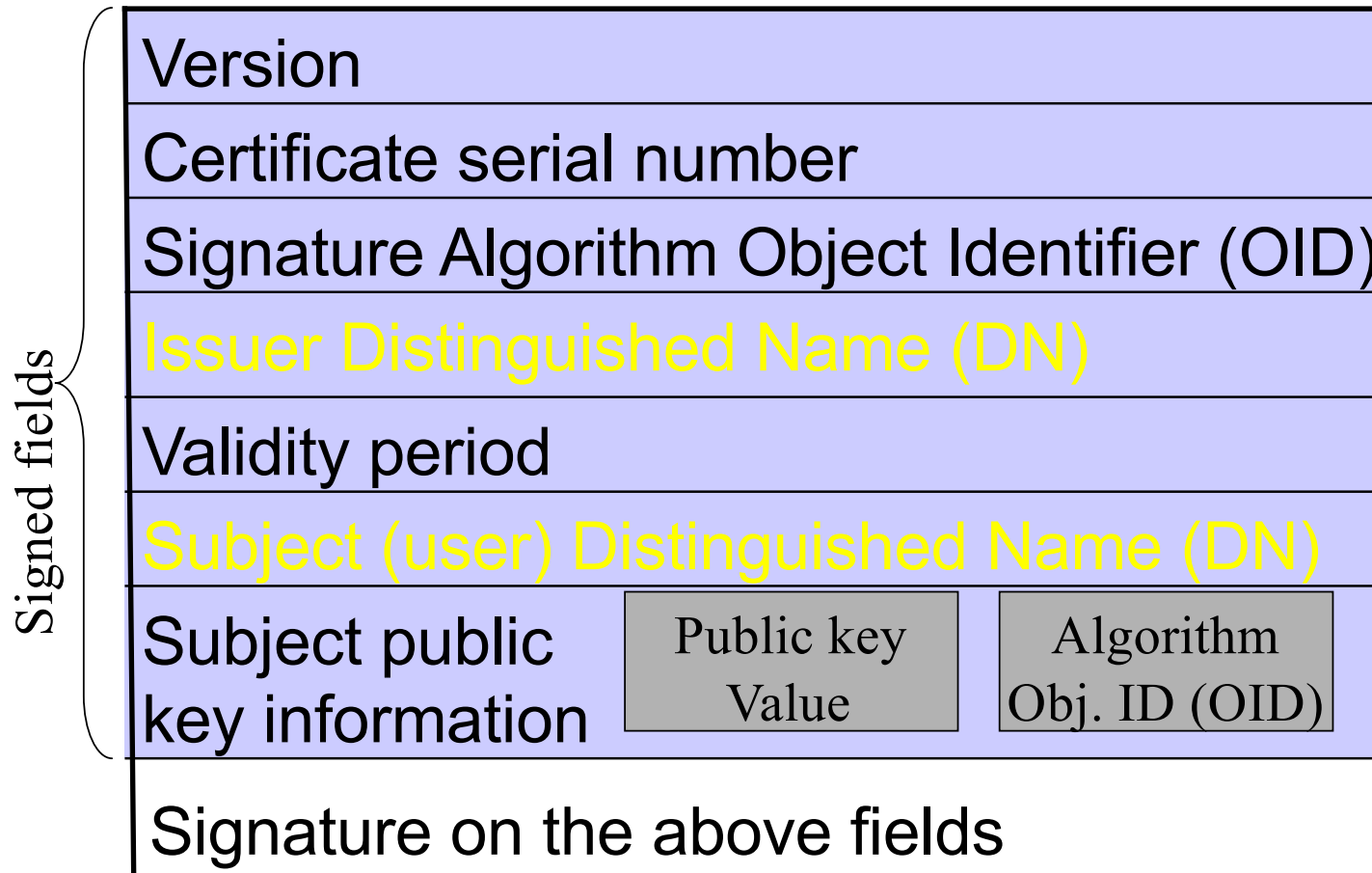
*Part of the X.500 Global Directory Standard*

---

# The X.509 Standard Certificate Format

- Published by ITU (International Telecommunication Union) in 1988 as part of the X.500 global directory standard.
- Idea: Signature binds **public key** to distinguished name (DN) and to other attributes
  - Some defined in X.509 standard, others in `extensions`
- Used widely in spite of complaints about its complexity.
  - SSL / TLS, code-signing, PGP, S/MIME, IP-Sec, ...

# X.509 V1 Certificate Format



# X.509 V1 Certificate Format

- **Version:** the version of X.509 (for V1 it is 1 and so on).
- **Certificate serial number:** a serial number of the certificate, unique among all of the certificates issued by this CA.
- **Signature algorithm OID:** an object identifier (OID) for the signature algorithm used to sign the certificate.
- **Issuer DN:** the Distinguished Name (DN) of the issuer of the certificate.
- **Validity period:** the period during which the certificate is supposed to be valid.
- **Subject DN:** the Distinguished Name (DN) of the subject of the certificate, i.e., the entity to whom the certificate was issued.
- **Subject public key information:** includes two parts, one containing the certified public key, and the other providing an OID to identify the public key algorithm with which this public key is to be used.
- **Signature (produced by CA):** a signature over the above fields.

---

# X.509 Certs & Subject Identifiers

- V1: Distinguished Name (for subject & issuer)
- V2: Unique identifiers (for subject & issuer)
- V3: Extensions (used in practice)
  - Some defined in X.509, others elsewhere
- V4: not covered in the textbook, not widely deployed

# X.509 Certificate Format – Later Versions

Signed fields	Version		
	Certificate serial number		
	Signature Algorithm Object Identifier (OID)		
	Issuer Distinguished Name (DN)		
	Validity period		
	Subject (user) Distinguished Name (DN)		
	Subject public key information	Public key Value	Algorithm Obj. ID (OID)
	Issuer unique identifier (from version 2)		
	Subject unique identifier (from version 2)		
	Extensions (from version 3)		
Signature on the above fields			

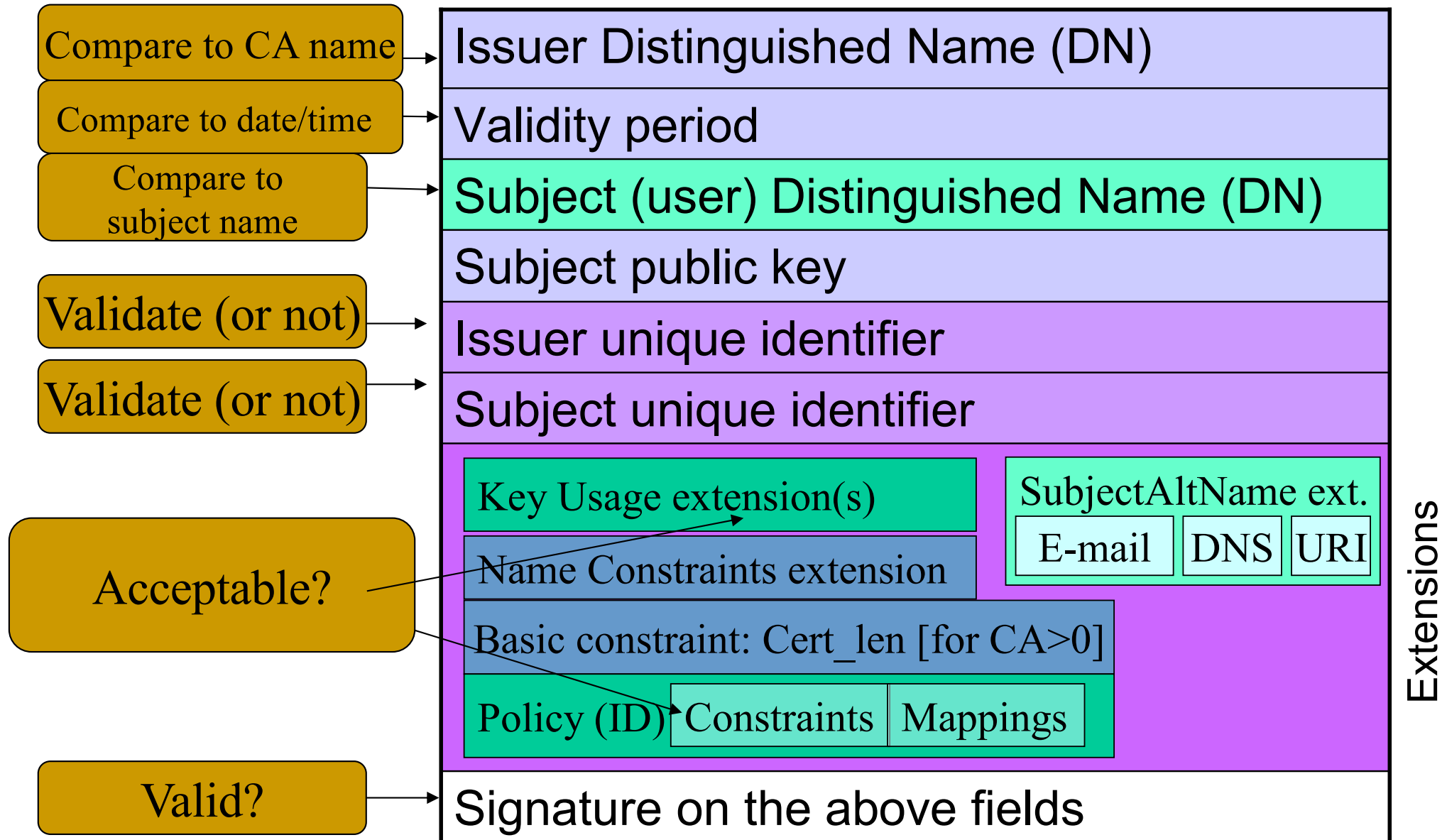


---

# X.509 Certificate Format – Later Versions

- **Issuer and subject unique identifiers (V2):**
  - ❑ Added to ensure uniqueness to handle situations where the DN may fail to ensure uniqueness.
  - ❑ Not widely used.
- **Extensions (V3):**
  - ❑ Additional fields to increase the expressiveness of X.509 certificates to facilitate more applications and end users.
  - ❑ Examples include limitations on which application the certificate or public key can be used for, certificate path constraints, policy constraints, etc.
  - ❑ We will not cover these in this course. More in a Network Security course.

# X.509 Certificate Validation (simplified)



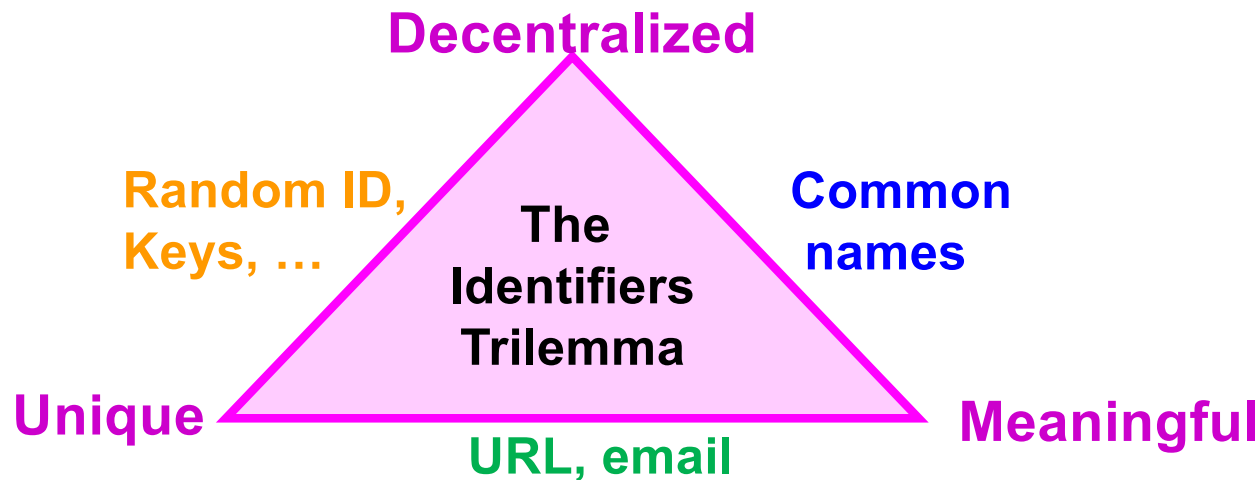
---

# Distinguished Names

- Most certificates contain identifiers
- Influenced by telecommunication providers.
  - Phone directory services are based on common names.
- Basic goals of identifiers:
  - **Meaningful** (to humans)
    - Memorable, reputation, off-net, legal
  - **Unique** identification of entity (owner)
  - **Decentralized** - with Accountability:  
assigned by trusted (certificate) authorities
    - Accountability: identification of the signing authority

# The Identifiers Trilemma

- Achieving the three goals: Meaningful, Unique, Decentralized, seems very challenging!
- Examples of achieving any two of the goals:
  - Unique + Meaningful: URL, email
  - Meaningful + Decentralized: common name
  - Unique + Decentralized: hash of key

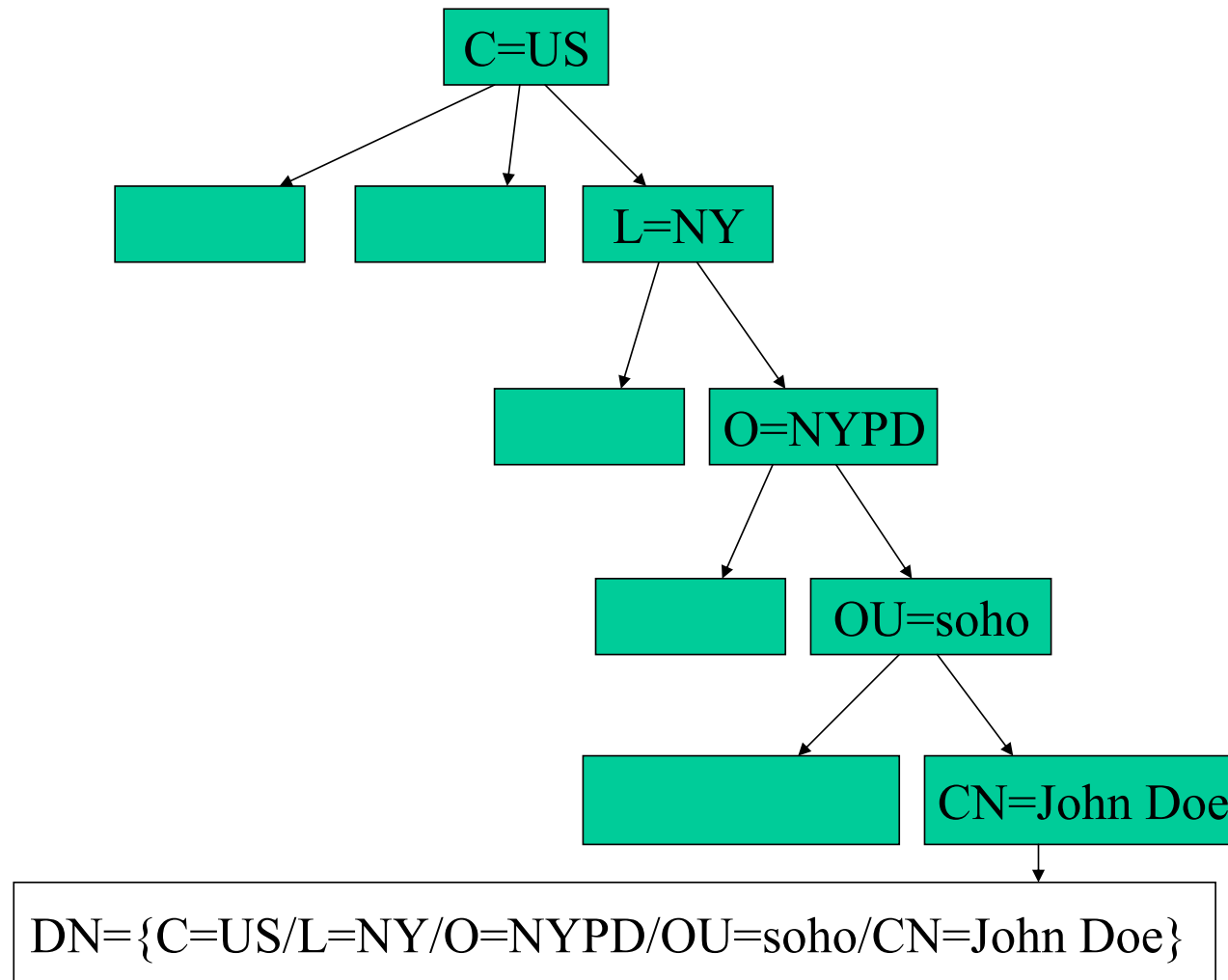


# X.500 Distinguished Names (DN)

- Sequence of keywords, a string value for each of them
- Distributed directory, responsibility → *hierarchical DN*

Keyword	Meaning
<b>C</b>	<b>Country</b>
<b>L</b>	<b>Locality or city name</b>
<b>O</b>	<b>Organization name</b>
<b>OU</b>	<b>Organization Unit name</b>
<b>CN</b>	<b>Common Name</b>

# Distinguished Name (DN) Hierarchy



---

# Intermediate CAs and Path Verification

---

# Why Intermediate CAs?

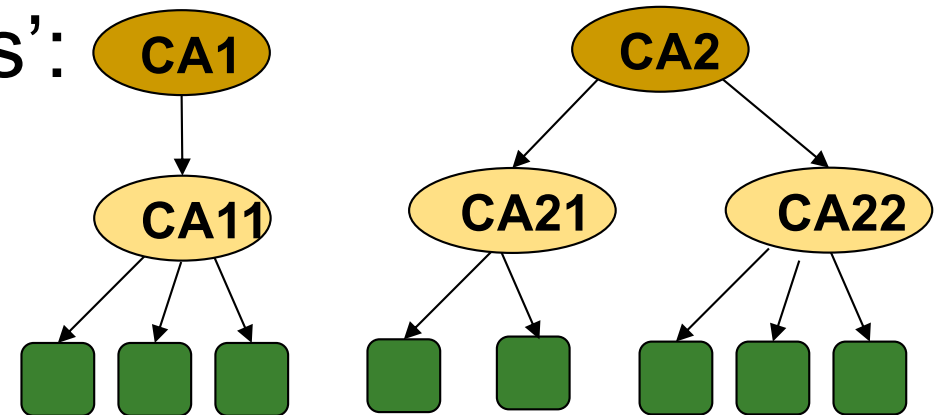
- Relying parties rely on trust anchor (or root) CA(s) to establish trust in a certificate.
- Large number of subjects to certify.
  - One (or a few) trust anchor CAs cannot handle all the load.
- An anchor or root CA certifies other CAs to become intermediate CAs.
  - So the root A certifies intermediate B, then B will sign certificates for subjects (B is an issuer).
- Certificate path validation allows validating such certificates that are issued by intermediate CAs.
  - Like tracing them back to the root CA.



# Certificate paths in different PKIs

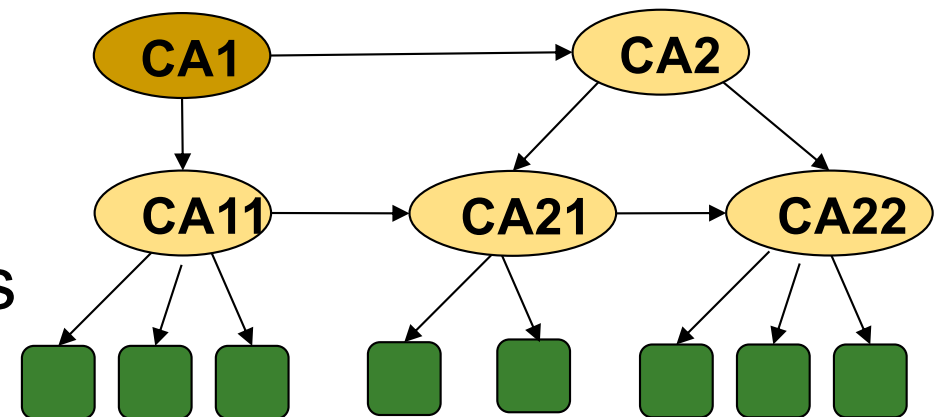
- **Web/TLS PKI (X.509):** 'root CAs'+'intermediate CAs':

- Root CA issues cert for intermediate CAs



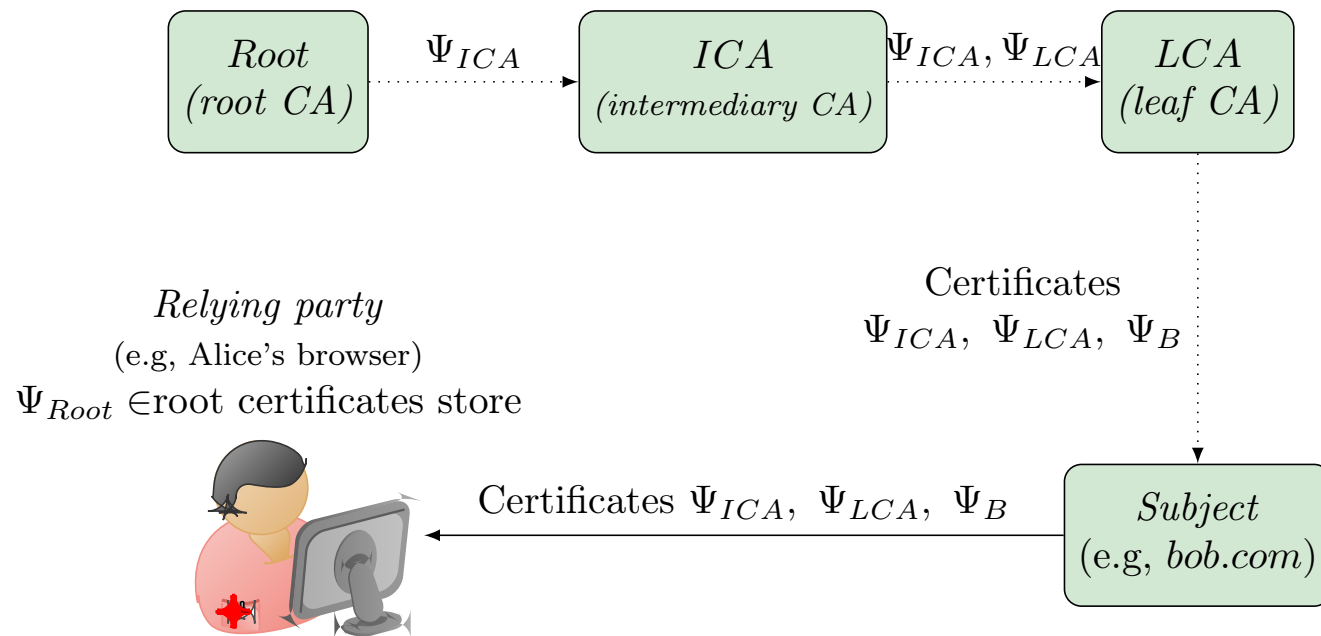
- **Web-of-Trust PKIs:**

- Directed graph, not tree
- Different variants/policies



# X.509 Validation of Certificate Paths

- Simply, validate all certificates in the chain all the way to the root CA.
- The root CA (self-signed) certificate is in the root store in Alice's browser.
- Let's trace the example below.



---

# Certificate Revocation

---

# Certificate Revocation

- Reasons for revoking certificates
  - Security issues:
    - Key compromise, CA compromise
  - Administrative issues:
    - Affiliation changed (changing DN or other attribute), public key has been replaced, subject has ceased operation (company dissolving).
- How to inform relying parties? Few options usually under three categories:
  - Prefetch: have revocation info in advance.
  - As-needed: ask for this info when a receiving a certificate and want to validate.
  - Neither: does not fall under any of the above, usually called network-assisted techniques.

# Certificate Revocation Techniques

- Prefetch:
  - ❑ Cons: higher storage and communication overhead,
  - ❑ Pros: lower response delay
- As needed:
  - ❑ Cons: higher response delays, reliability issues, privacy concerns.
  - ❑ Pros: lower storage and communication overhead
- We will study two techniques:
  - ❑ Distribute **Certificate Revocation List (CRL)** -- Prefetch
    - This is part of the X.509 standard.
  - ❑ Ask - **Online Certificate Status Protocol (OCSP)** – As needed

# CRLs

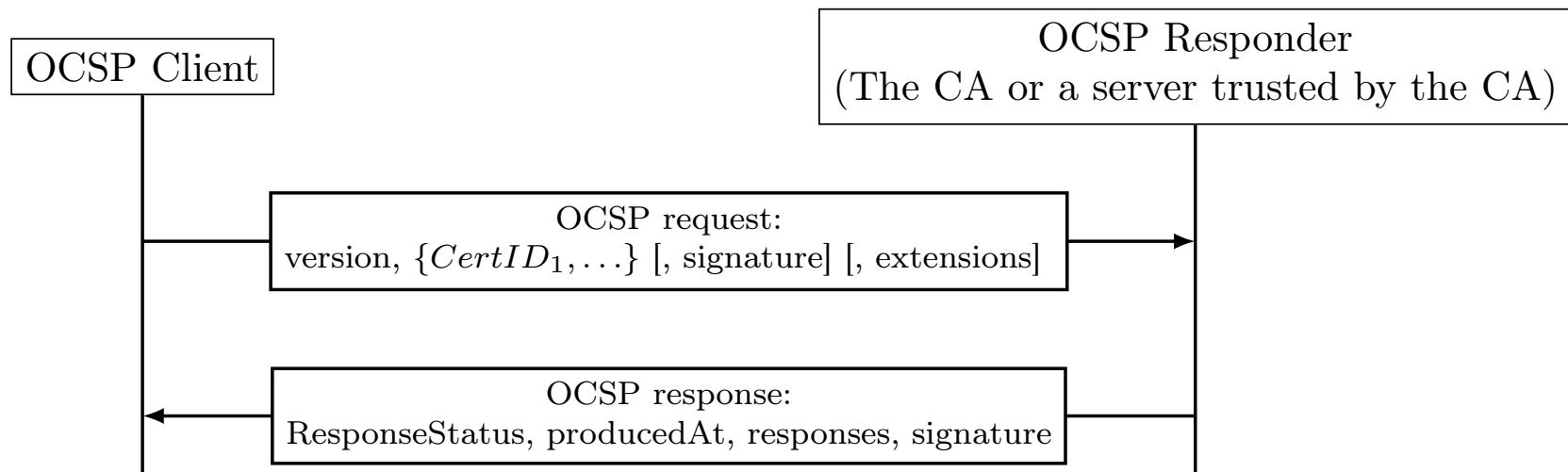
- A certificate revocation list (CRL) is simply a list of revoked certificates.
  - Distributed periodically by CAs.
- If CRLs contain all revoked certificates (which did not expire)... it may be huge!
  - Yes, large storage and communication overhead.
- CRLs are not immediate
  - Who is responsible until CRL is distributed?
  - Frequent CRLs → even more overhead!

# CRLs Optimization Solutions

- More efficient CRL schemes:
  - ❑ 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`
    - Need to keep CRLs for long period to check deltas → complicates implementation
- Browsers mostly do not check CRLs. Instead they usually use:
  - ❑ The Online Certificate Status Protocol (OCSP)

# Online Certificate Status Protocol (OCSP)

- Improve efficiency and freshness compared to CRLs
- Client asks CA about cert during handshake
- CA signs response (real-time)





---

# OCSP Challenges

- Privacy (expose domain and client to CA), load on CA, response delay, reliability (what if CA fails).
- Ambiguity:
  - When an OCSP server (or CA) cannot resolve the request, it replies with "certificate status is unknown".
- Reliability or failed requests.
  - Client failed to establish a connection with the OCSP server.
  - Or client's request is invalid (not signed, or not authorized).

---

# Ambiguous/Failed OCSP Responses

- What should the client do?
  - ❑ Wait forever – unrealistic!
  - ❑ Hard-fail: terminate the connection since certificate is unknown/not received.
    - Safe!
  - ❑ Ask user: application display a message asking the user how to proceed.
  - ❑ Soft-fail: pretend that a response has been received and continue as the certificate is not revoked.
    - Common choice for browsers!
    - But, a man in the middle attacker may block the OCSP response to make a revoked cert go through!

---

# Conclusion

- PKI is an essential component of the Internet.
- Yet, it is a complicated module with many issues related to security, privacy, and performance.
  - ❑ To many, this is a solved problem, but that is not the case.
  - ❑ Several questions related to how detect rogue certificates, how to handle CA failure, revocation, etc., how to audit these parties, how to reduce trust,...
  - ❑ How to handle all of these issues in an efficient way?
    - Remember, we all want a Web that is highly responsive!

---

# Covered Material From the Textbook

- ❑ Chapter 8:
  - ❑ Sections 8.1,
  - ❑ and Sections 8.3 and 8.4 (only the topics we covered from both sections)

---

# Thank You!

