CSE 3400/CSE 5850 - Introduction to Cryptography and Cybersecurity
/ Introduction to Cybersecurity

Lecture 12
Public Key Infrastructure

Ghada Almashaqbeh

UConn

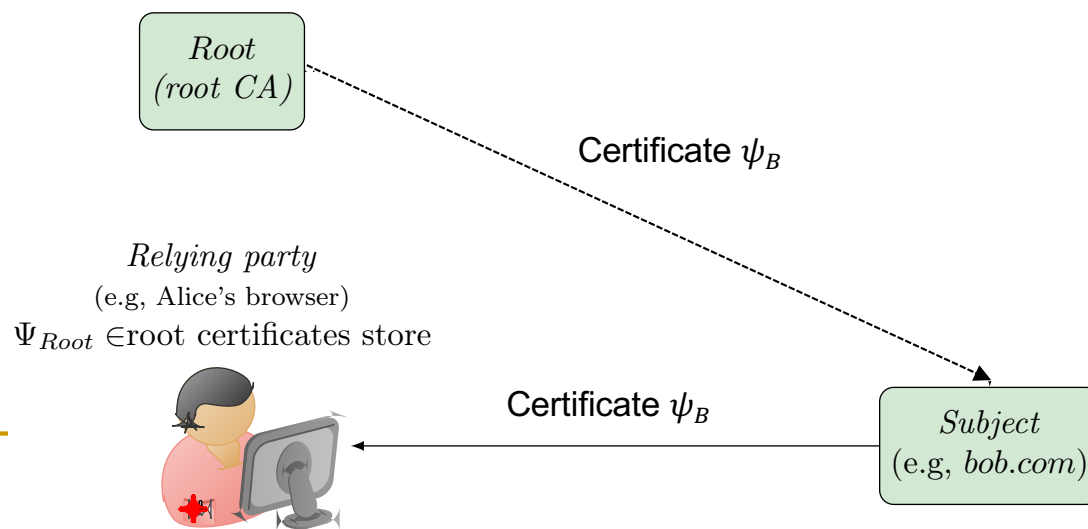*Adapted from the textbook slides

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

- 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 that she owns a specific public key?

  - Mainly: the key must be signed by a **trusted Certificate Authority (CA).**

- *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.
- **Attributes:** identity (ID) and others…
  - Validated by CA (liability?)
 - Used by **relying party** for decisions (e.g., use this website?)

Root
(root CA)

Certificate ψ_B

Relying party
(e.g, Alice's browser)
$\Psi_{Root} \in$ root certificates store
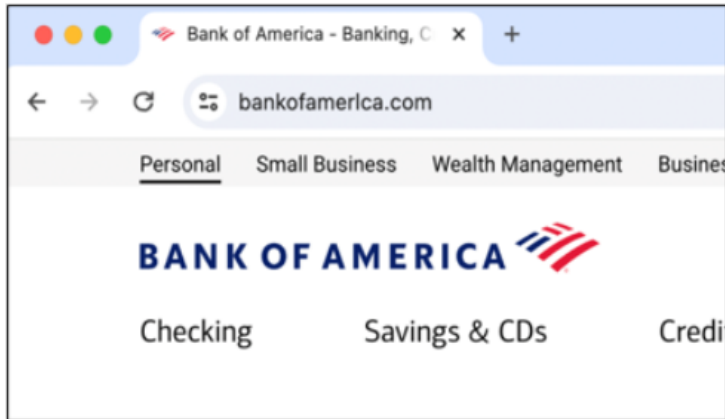
Certificate ψ_B

Subject
(e.g, bob.com)

4

Certificates are all about **Trust**

- **Certificate:** $\psi_{Bob} = Sign_{CA.s}(Bob.com, Bob.e, \ldots)$
  - 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: certificates that contain wrong or misleading information.
  - So they should fail PKI validation.
- Attacker goals:
  - Impersonate: website, phishing email, signed malware..
  - Equivocating (same name): circumvent name-based security mechanisms, such as *blacklists, access-control* …
- 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. paypaI.com [i is l]
  - Typo-squatting: bank.com vs. banc.com, baank.com, banl.com,…

Example of Homographic Attacks
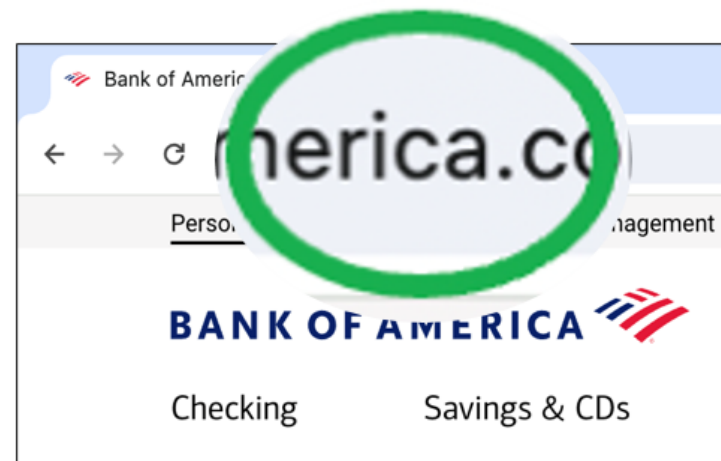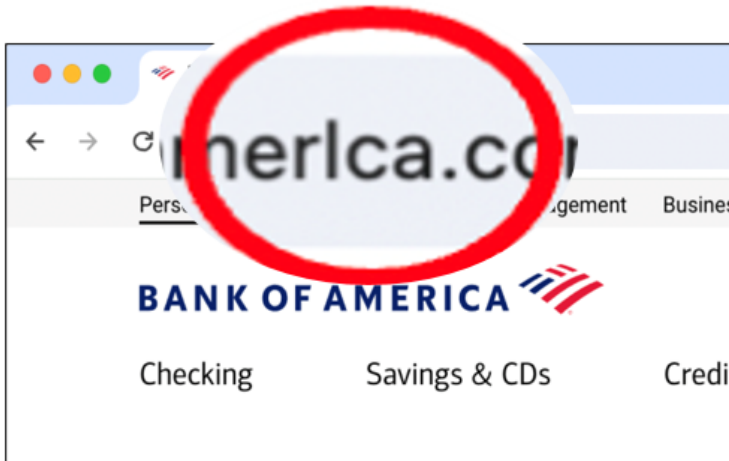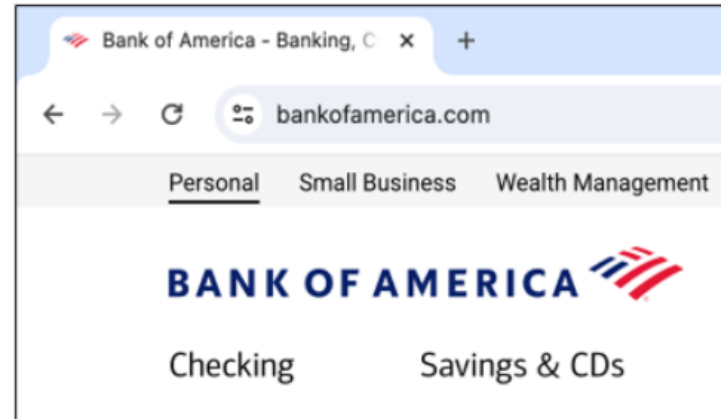
PKI Failures

- 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:

  - Corrupt CA.

  - Validation failure.

  - Exposed CA private key.

  - 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].
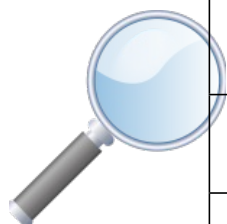
PKI Goals/Requirements

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 certificates; 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 despite complaints about its complexity.

  - SSL/TLS, code-signing, IP-Sec, …

X.509 V1 Certificate Format

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) |

| Signature on the above fields |

13
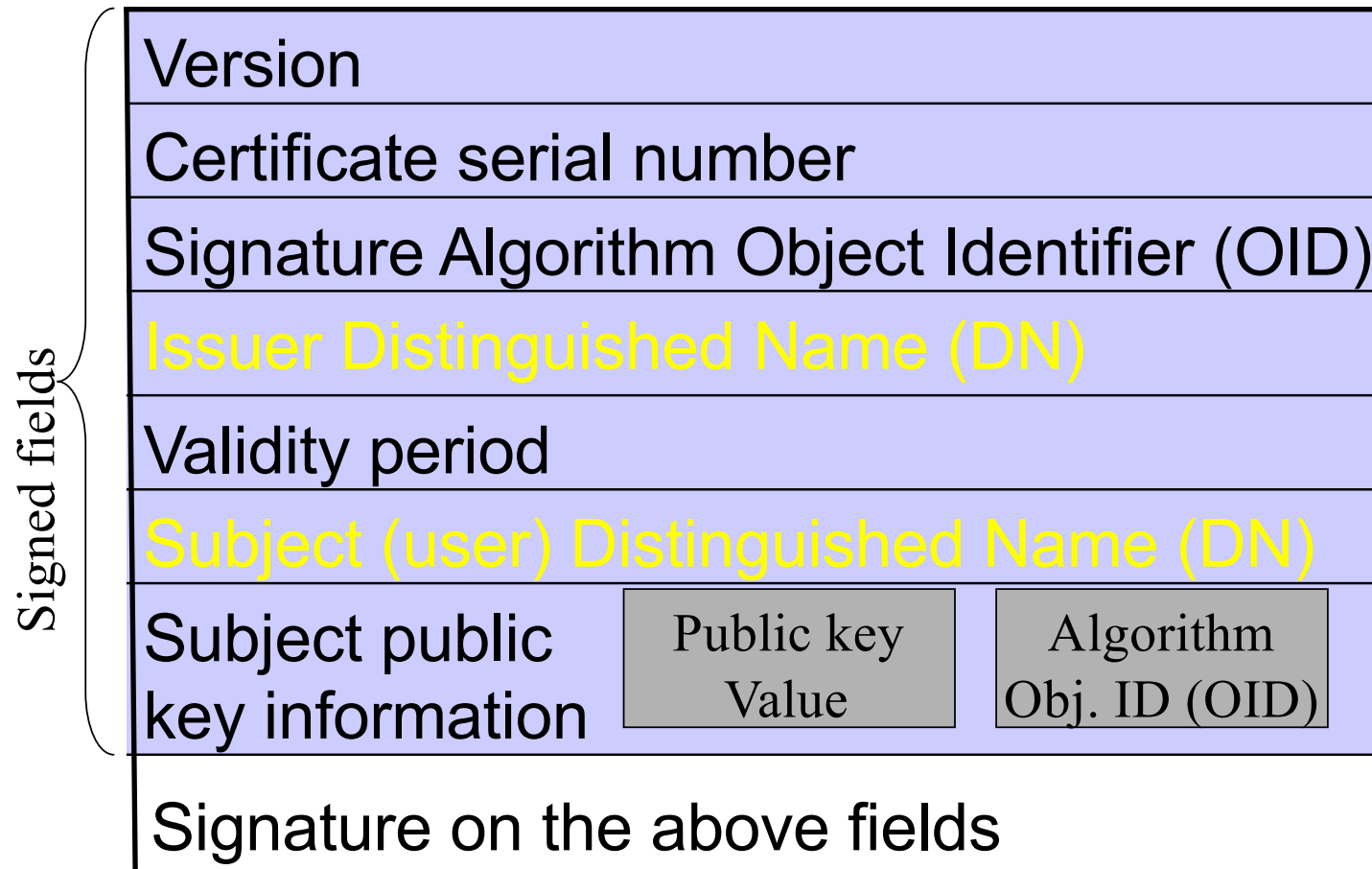
X.509 V1 Certificate Format

- **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 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

X.509 Certificate Format – Later Versions

Signed fields

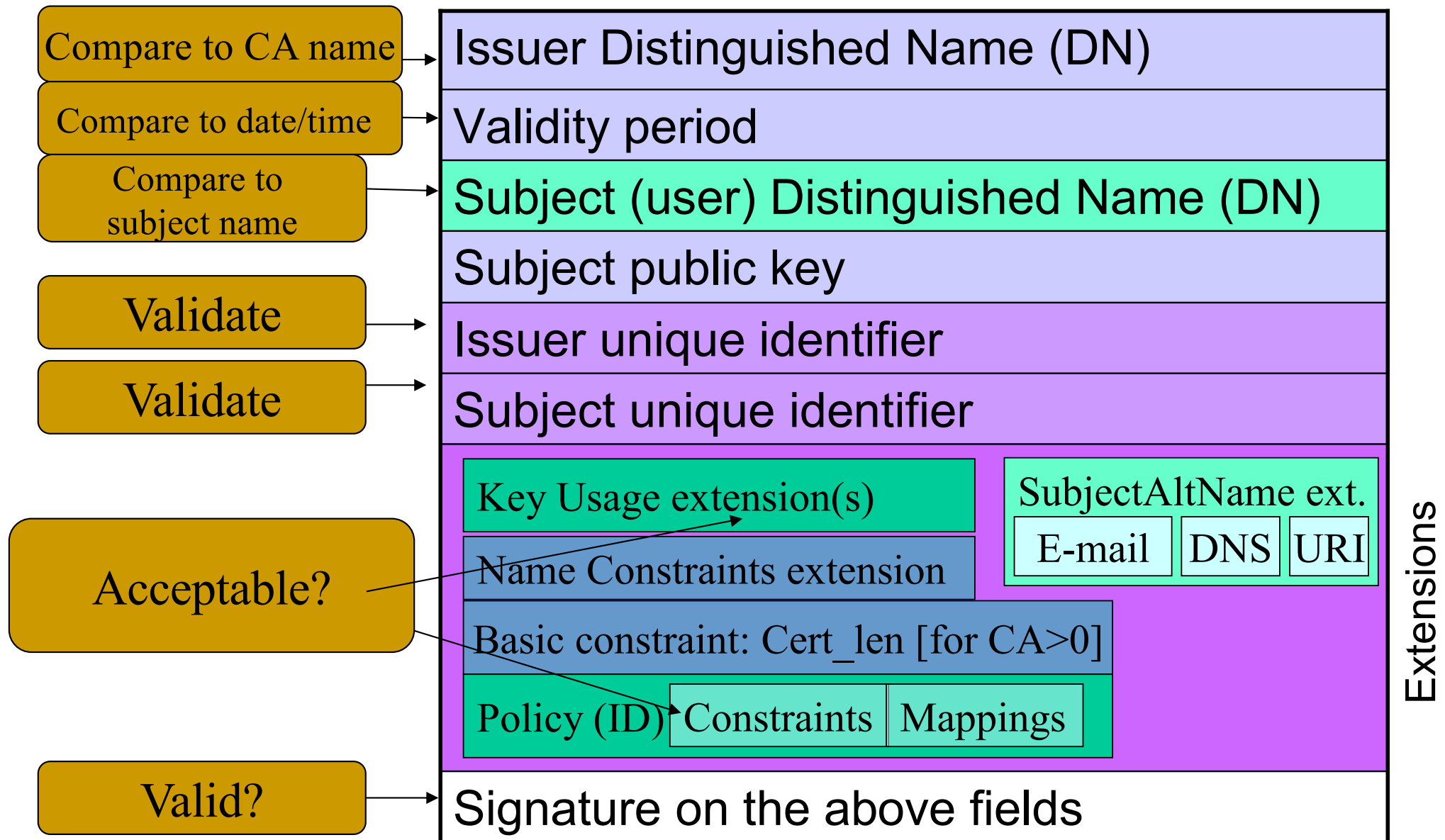| Version |
| Certificate serial number |
| Signature Algorithm Object Identifier (OID) |
| Issuer Distinguished Name (DN) |
| Validity period |
| Subject (user) Distinguished Name (DN) |
| Subject public key information — Public key Value — Algorithm Obj. ID (OID) |
| Issuer unique identifier (from 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 details are in a Network Security course.

X.509 Certificate Validation (simplified)

Compare to CA name → Issuer Distinguished Name (DN)

Compare to date/time → Validity period

Compare to subject name → Subject (user) Distinguished Name (DN)

Subject public key

Validate → Issuer unique identifier

Validate → Subject unique identifier

Extensions

Key Usage extension(s)

SubjectAltName ext.

E-mail | DNS | URI

Name Constraints extension

Acceptable?

Basic constraint: Cert_len [for CA>0]

Policy (ID) | Constraints | Mappings

Valid? → Signature on the above fields

18

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, etc.
  - 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 <u>two</u> of the goals:
  - Unique + Meaningful: URL, email
  - Meaningful + Decentralized: common name
 - Unique + Decentralized: hash of key

Decentralized

Random ID, Keys, …

The Identifiers Trilemma

Common names

Unique

URL, email

Meaningful
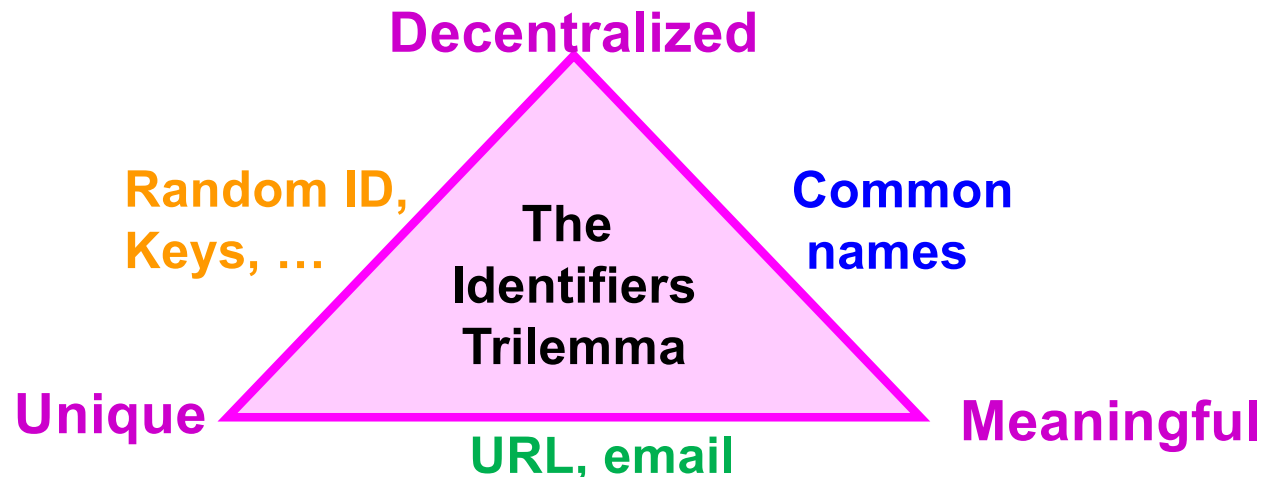
Intermediate CAs and Path Verification

Why Intermediate CAs?

- Relying parties rely on root CA(s) to establish trust in a certificate of a particular subject party.

- Large number of subjects to certify.
  - One (or a few) root CAs cannot handle all the load.

- A root CA certifies other CAs to become intermediate CAs.
  - So the root CA A certifies intermediate CA B, then B will sign certificates for subjects (B is an issuer).
  - Intermediate CAs can certify (beside subjects) other intermediate or leaf CAs.
  - Leaf CAs can certify only subjects.

- Certificate path validation allows validating such certificates that are issued by intermediate CAs.
  - Like tracing them back to the root CA.

- *Who certifies a root CA?*

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.
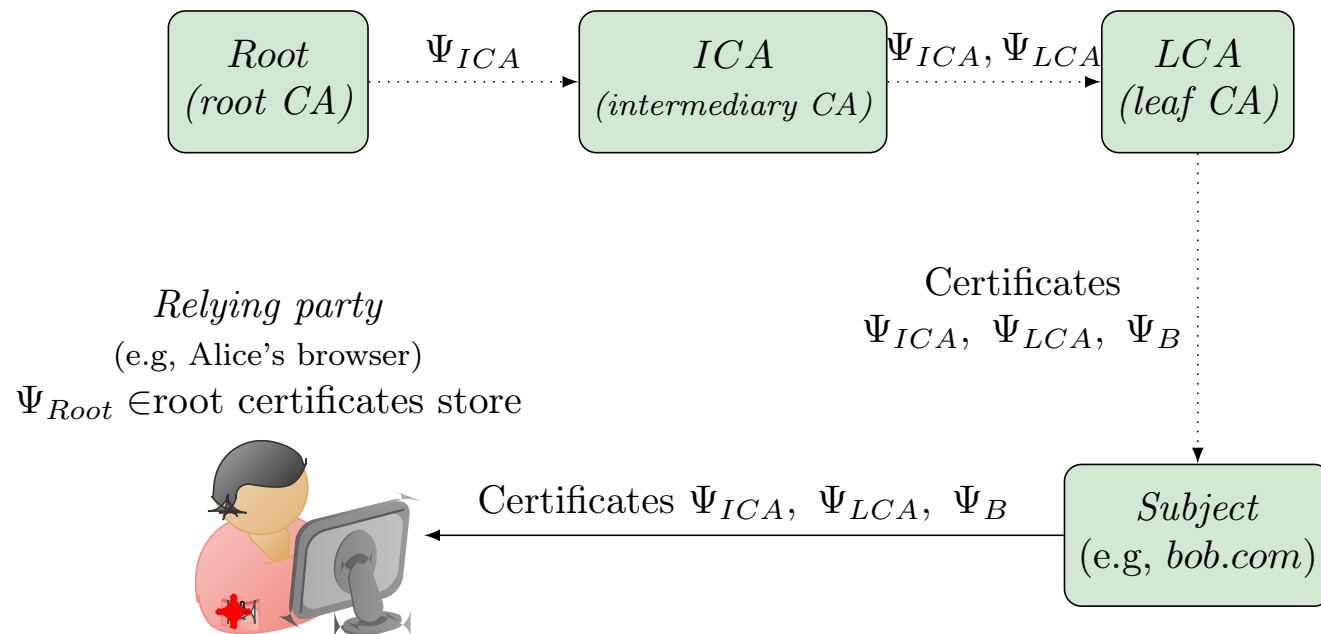
Certificate Revocation

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 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.
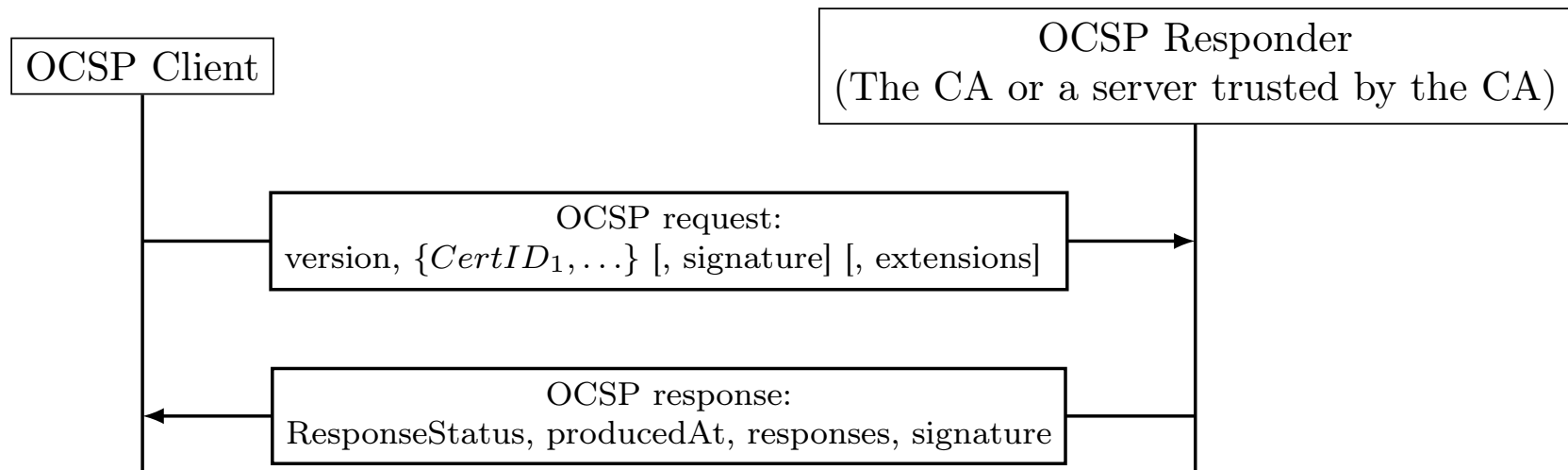   - Ask - **Online Certificate Status Protocol (OCSP) –** *As needed*

CRLs

- A certificate revocation list (CRL) is simply a list of revoked certificates.
    - Distributed periodically by CAs.
- If CRLs contain all revoked certificates, it could 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
- **Another more efficient approach is 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 Client		OCSP Responder (The CA or a server trusted by the CA)
	OCSP request: version, $\{CertID_1, \ldots\}$ [, signature] [, extensions] $\rightarrow$	
	$\leftarrow$ OCSP response: ResponseStatus, producedAt, responses, signature	

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), so no response will be received.

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 (MitM) 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 open questions related to how to detect rogue certificates, how to handle CA failure, revocation, etc., how to audit these parties, how to reduce trust,…
  - How to handle all 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!