Public Key Infrastructures (PKI)

Sushmita Ruj

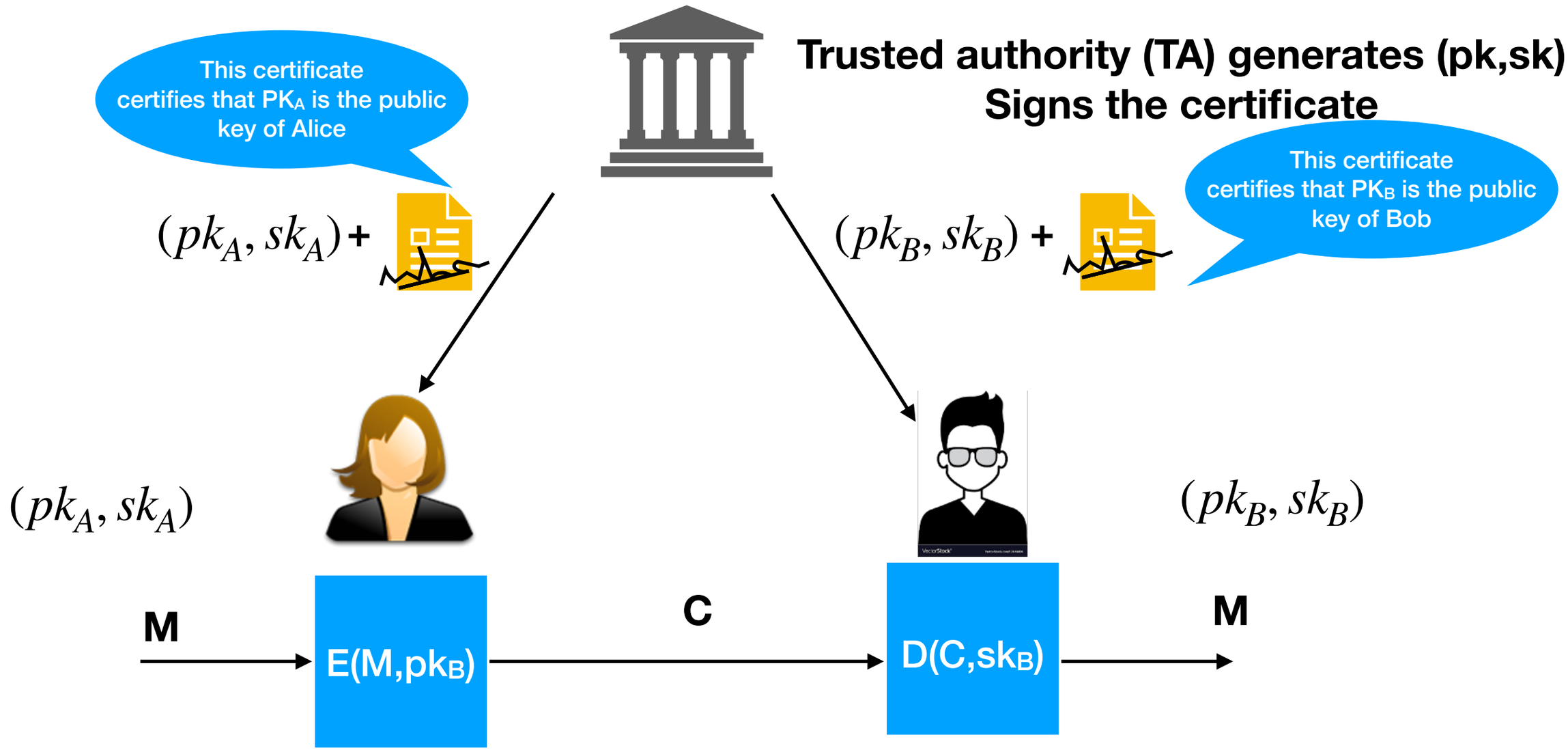
Recap

- Digital Signatures
- RSA Signatures
- Digital Signature Algorithm
- Construction of signature algorithms

This Lecture

- Public Key Infrastructure
- PGP
- Certificates
- Certificate Revocation
- Certificate Transparency

Public Key Infrastructure (PKI)



TA in real are Digicert, Lets Encrypt, Identrust, GoDaddy etc.

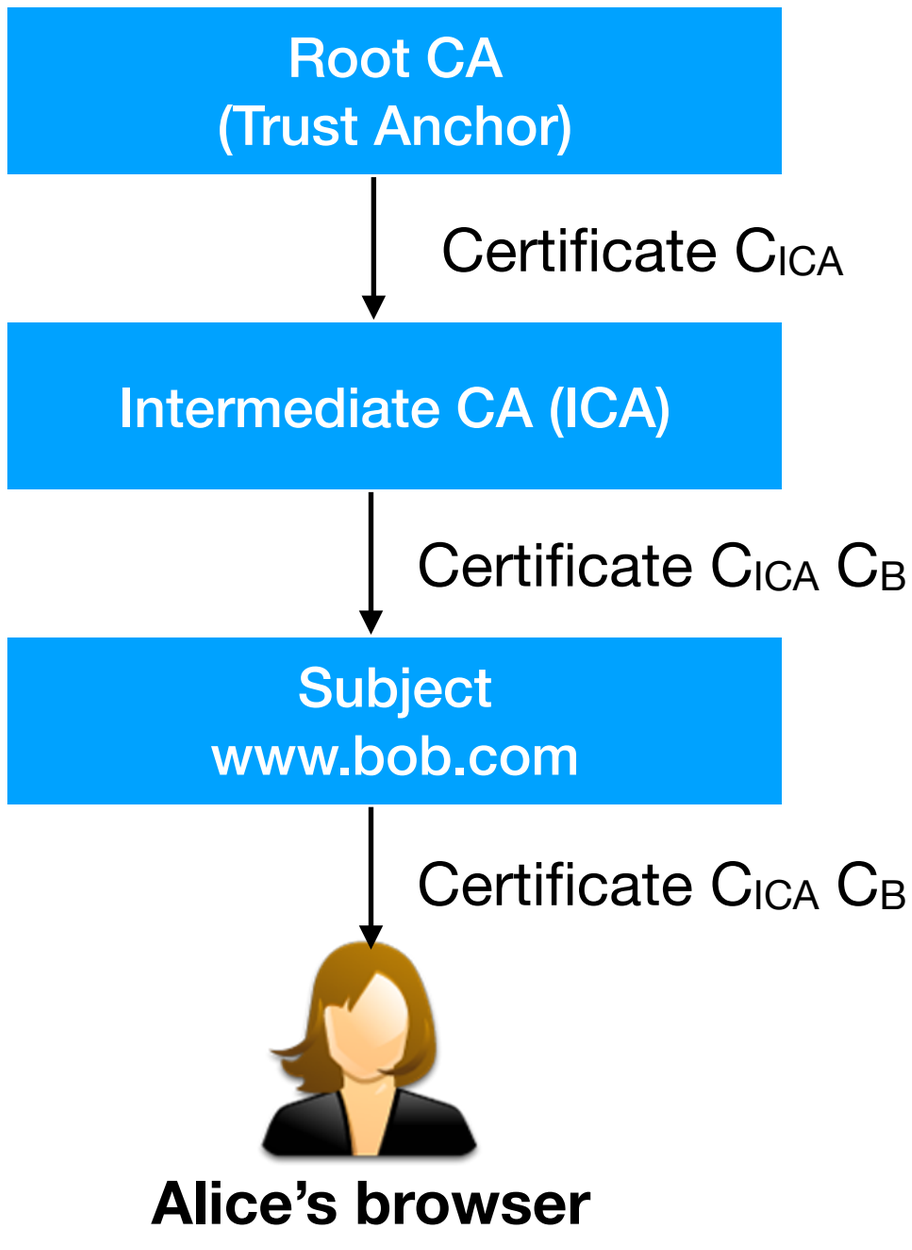
Web-PKI

- PKI deployed by **TLS/SSL, browsers, web-servers**
- Browsers contain keys of **Root CAs** (trust anchors)
- Root CAs defined by (four) **root programs** (of Google, MS, Mozilla, Apple)
- Root CA certifies **Intermediate CAs (ICA)**
- **Subject (website) certs** issued by **intermediate CA**

Web-PKI

Root CAs defined by **root programs**
(of Google, MS, Mozilla, Apple)

Subject (website) certs issued by intermediate CA



Rogue Certificates

- Rogue cert: equivocating or misleading (domain) name
- Attacker goals:
  - Impersonation: phishing email, Phishing websites, 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 (‘cybersquatting’):
  - Combo: 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 I]
  - Typo-squatting: bank.com vs. banc.com, baank.com, banl.com,…

PKI Failures

2001	VeriSign: attacker gets code-signing certs
2008	Thawte: email-validation (attackers' mailbox)
2008,11	Comodo not performing domain validation
2011	DigiNotar compromised, over 500 rogue certs discovered
2011	TurkTrust issued intermediate-CA certs to users
2012	Trustwave issued intermediate-CA certificate for eavesdropping
2013	ANSSI, the French Network and Information Security Agency, issued intermediate-CA certificate to MitM traffic management device
2014	India CCA / NIC compromised (and issued rogue certs)
2015	CNNIC (China) issued CA-cert to MCS (Egypt), who issued rogue certs. Google and Mozilla removed CNNIC from their root programs.
2013-17	Audio driver of Savitech install root CA in Windows
2015,17	Symantec issued unauthorized certs for over 176 domains
2019	Mozilla, Google *software* blocks *customer-installed* Kazathhstan root CA (Qaznet)
2019	Mozilla, Google revoke intermediate-CA of DarkMatter, and refuse to add them to root program

How to ensure trust?

Applied Cryptography, Amir Herzberg

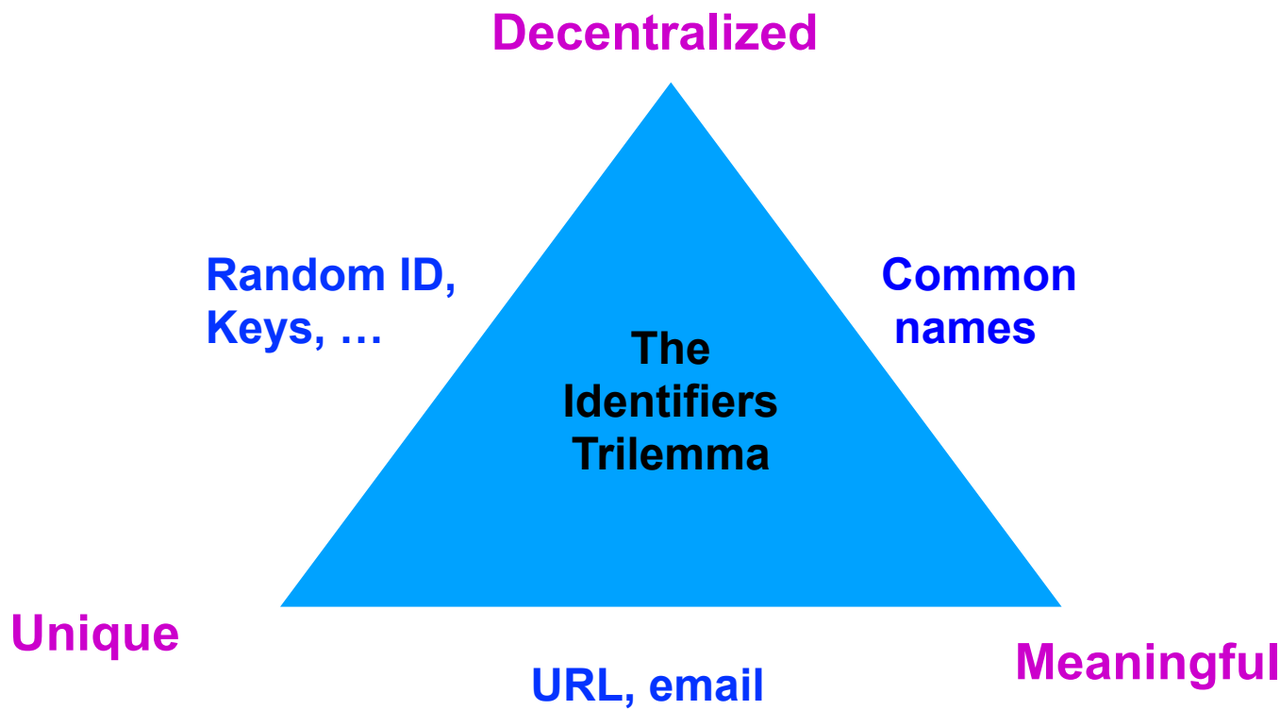
Desirable Properties of Certificates

- Trusted issuers: Root CA and ICA
- Validity period: Each certificate has a validity period, Timely revocation
- Transparency: Public log of certificates. Certificate Transparency
- Non Equivocation: All information is spelt out properly, one entity-one certificate
- Client privacy: Why should CA know which site I am accessing? Not present in tradition OCSP, but can be added

Identifiers in Certificates

- Most certificates contain identifiers
  - Aka identity-certificates
- Basic goals of identifiers:
- Meaningful (to humans)
  - Memorable, reputation, off-net, legal
- Unique identification of entity (owner)
- Decentralized - with Accountability: assigned by a trusted (certificate) authority
  - Accountability: identification of assigning authority

Identifier's Trilemma



Hard to achieve all!

X.509 Certificates

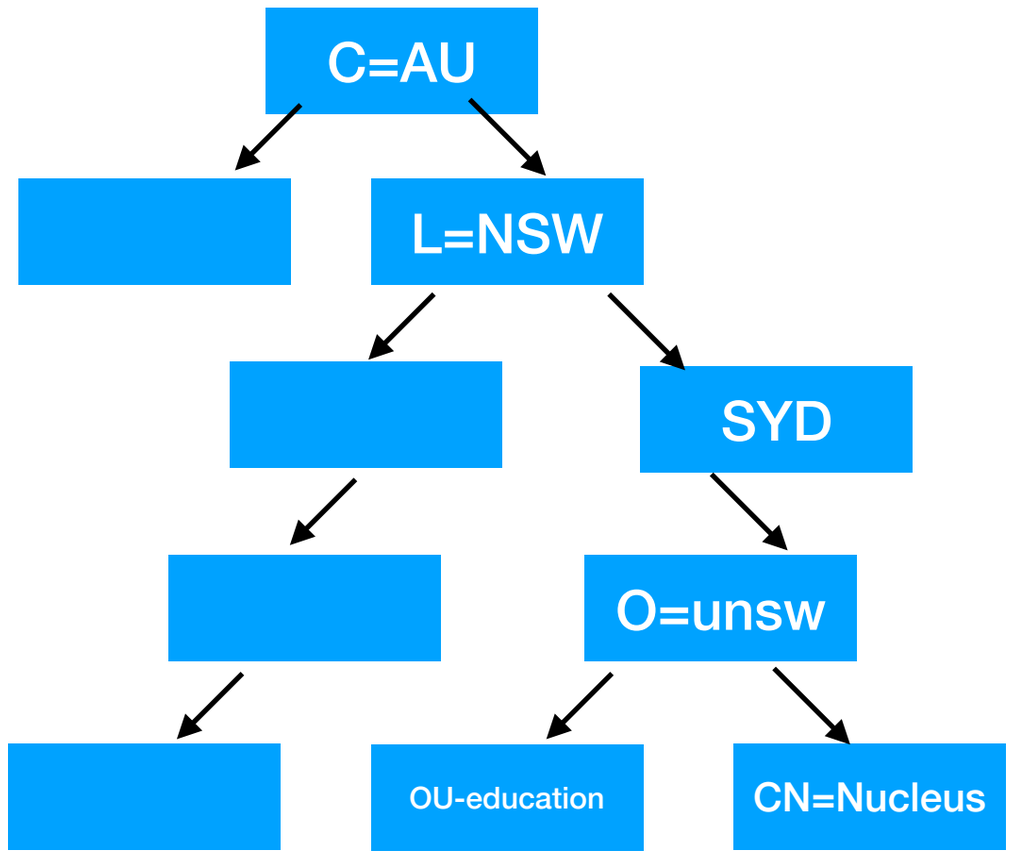
X.500 Certificate

- X.500: an International Telcos Union (ITU) standard, first issued 1988
- Idea: trusted global directory of certificates
  - Operated by hierarchy of trustworthy telcos
  - Never happened
    - Too complex, too revealing, high trust on telcos
    - Directory bind identifiers to attributes
- Standard attributes (incl. public key)
- Standard identifiers: Distinguished Names
  - Goal: unique, meaningful, decentralized identifiers

X.500 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



Features

- **Decentralized?**
  - Separate name spaces
- **Unique ?**
  - Could be, if each name space has one issuer
  - TLS reality: browsers trust 100s of CAs for **every name**
- **Meaningful?**
  - Usually: David Jones/UNSW/AUS
  - But not always: David Jones2/UNSW/AUS
    - Added ‘counter’ to distinguish → mistakes, loss of meaning

- **Distinguished Name fields may expose**
  - Organizational sensitive information (e.g. unit)
  - Privacy
- **Handling changes in position, organizations**
- **Multiple, related hierarchies:**
  - International organizations, divisions…
    - David Jones/UNSW/AUS or David Jones2/UNSW/AUS

X.509 Certificate

- X.509: authentication mechanisms of X.500
- Initially: Authenticate to Directory (PW, Pub key)
  - To maintain entity's record
- Later (and now): **X.509 certificate**
  - Signature binds **public key** to distinguished name (DN)
  - And to other attributes
    - Some defined in X.509 standard, others in `extensions`
- Used widely
  - SSL / TLS, code-signing, PGP, S/MIME, IP-Sec, …
  - In spite of complexity

Digital Certificate (X.509 v3)

- Certificate
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before
    - Not After
  - Subject name
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

X.509 V3 Extensions Mechanism

- Each extension contains…
- Extension identifier
  - As an OID (Object Identifier)
  - E.g. `Name constraints`
- Extension value
  - E.g. `Permit C=IL`, `Exclude dNSName=IBM.COM`
- **Criticality indicator**
  - **If critical**, relying parties MUST NOT use a certificate with any unknown critical extension
  - **If non-critical:** use certificate w/o unknown critical extensions; **ignore** unknown (non-critical) extensions
  - X.509/PKIX: extension MUST/MAY/CAN'T be critical

SubjectAltName (ESN) Extension

- Bound identities to the subject
  - In addition/instead of Subject Distinguished Name
  - Same extension may contain multiple ESNs
- Goal: unique and meaningful names
  - Common: DNS name (dNSName), e.g., a.com
    - TLS/SSL allows wildcard domains (*.a.com)
  - Or: email address, IP address, URI, other
- IssuerAltName (IAN) extensions
  - Similar – for issuer

Key Usage, Identifier Extensions

- Key-usage extension.
  - X.509: may be critical, PKIX: must be critical
  - Use of the public key being certified
    - Encrypt, verify-signature, verify-certificate, …
- Extended key usage extension
  - Additional optional use of the key: **Non-critical**
  - Details/restrictions related to `key usage' : **Critical**
- Subject/authority key identifier
  - Used when subject/CA has many keys; non-critical

Certificate Policy Extension

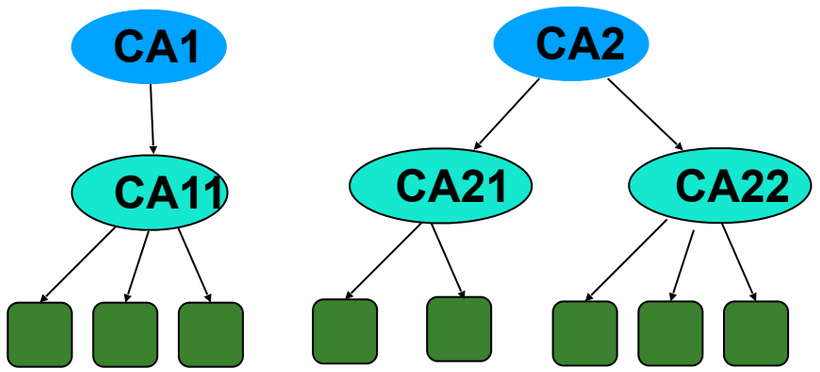
- Policies used/set by issuer
- Always critical
- Most important: method of subject validation
  - **Organization-Validated**
    - ‘Classical’ certificate; a person from CA checks subject
  - **Domain-Validated**
    - Automated check, e.g., send email to certified domain
  - **Extended validation**
    - Through checks, only for known organizations, companies
- Policy identified by Object Identifier (OID)
- **Do users know which was used? How ?**

X.509 V3

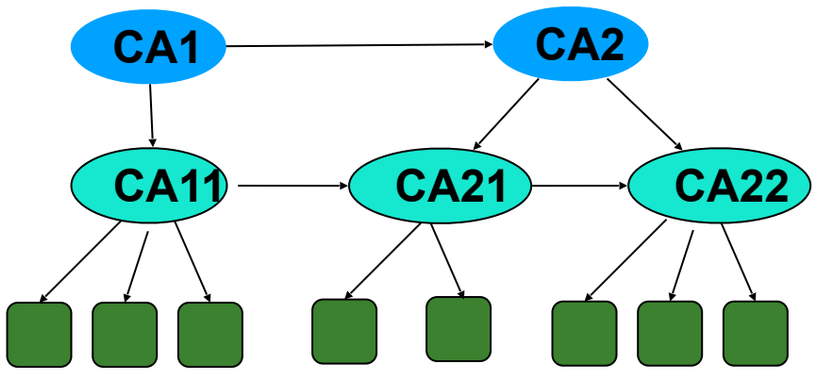
- Certificate
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before
    - Not After
  - Subject name
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

Certificate paths in different PKIs

- **Web/TLS PKI:** ‘root CAs’+‘intermediate CAs’:
  - Root CA issues cert for intermediate Cas



- **Web-of-Trust PKIs:**
  - Directed acyclic graph
  - Different variants/policies



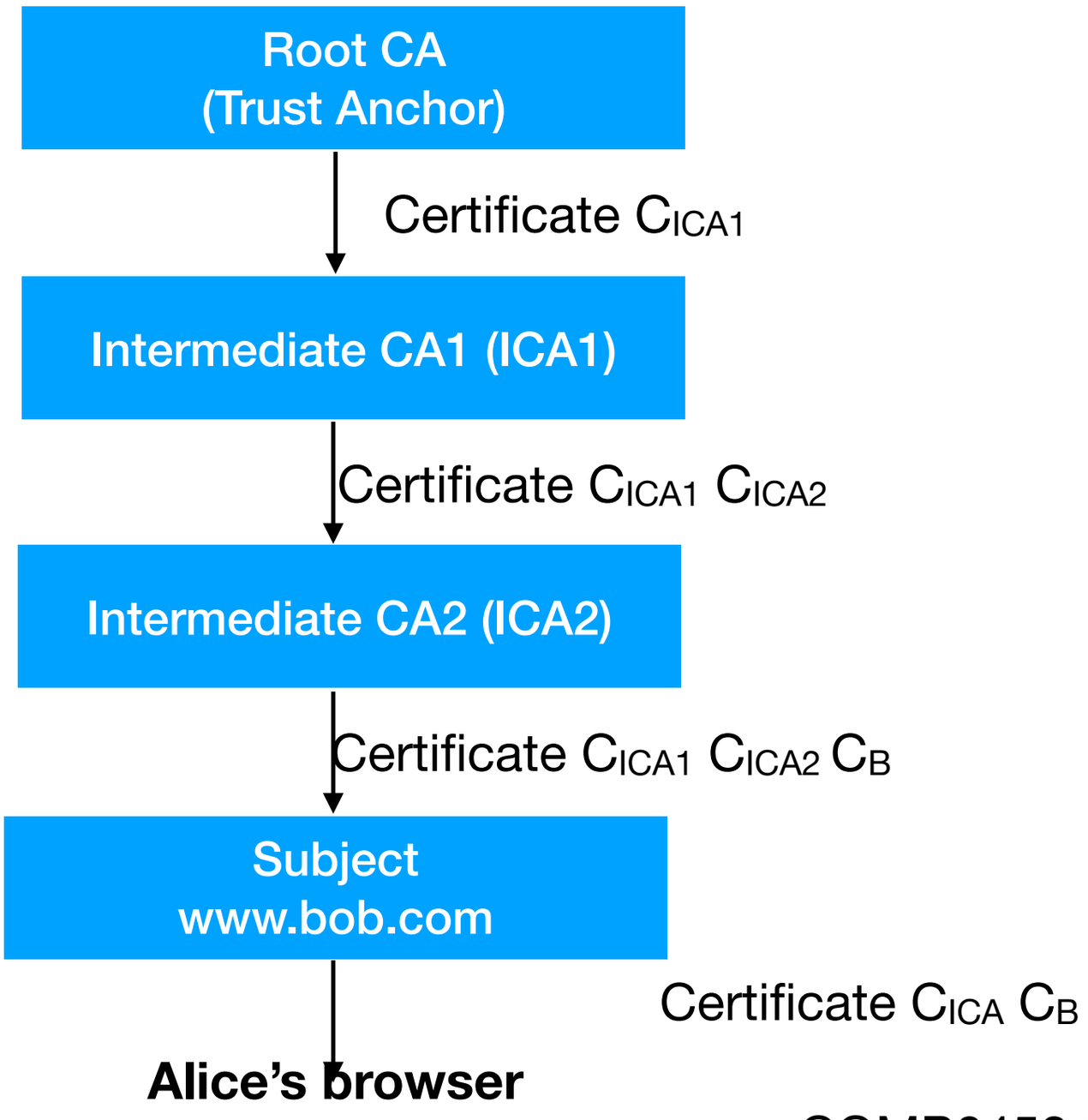
Pretty Good Privacy (PGP)

- PGP's friends-based Web-of-Trust:
  - Everyone is subject, CA and relying party
  - As a CA, certify (pk, name) for `friends'
  - As a subject, ask friends to sign for you
  - As a relying party, trust certificates from friends
    - Or also from friends-of-friends? Your policy….
    - Should you trust all your friends (equally)?

PKIX certificate path constraints

Root CAs defined by **root programs** (of Google, MS, Mozilla, Apple)

Subject (website) certs issued by intermediate CA



Certificate-Path Constraints Extensions

- Basic constraints:
  - Is the subject a CA? (default: FALSE)
  - Maximal length of additional CAs in path
    - pathLengthConstraint
- Policy constraints:
  - Require certificate-policies along path
  - Allow/forbid `policy mappings'
  - Details in textbook (or RFC)
- Name constraints
  - Constraints on DN and SubjectAltName - in certs issued by subject
    - Only relevant when subject is a CA !
  - 'Permit' and 'Exclude'

Certificate Revocation

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

CRL

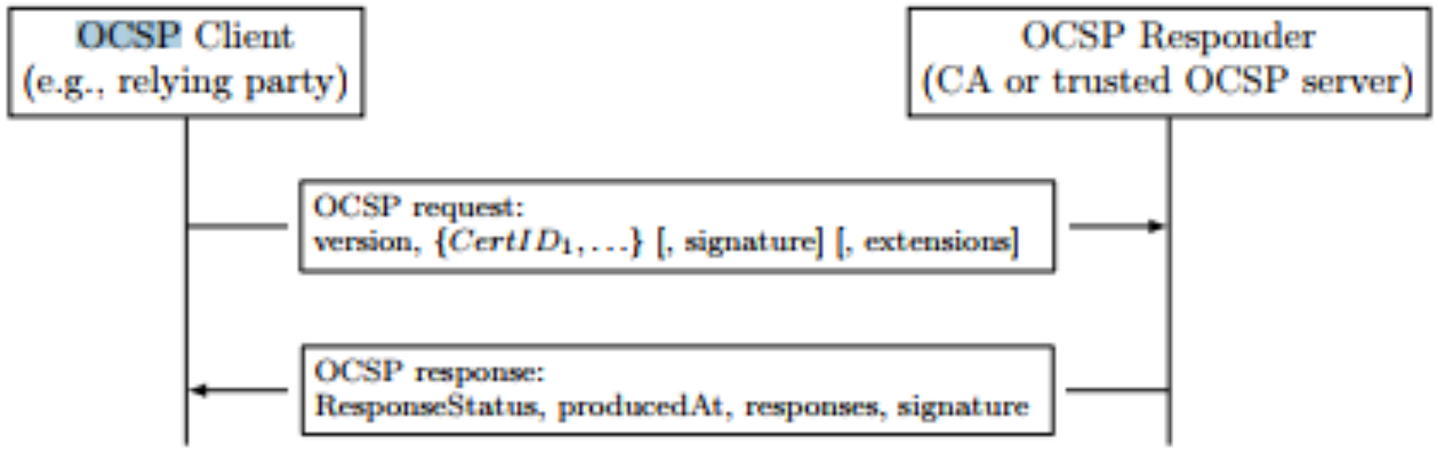
- If CRLs contain all revoked certificates (which did not expire)… it may be huge!
- CRLs are not immediate
  - Who is responsible until CRL is distributed?
  - Frequent CRLs ➔ even more overhead!
- Solutions:
  - More efficient CRL schemes
    - CRL distribution point – split certificates to several CRLs
    - Authorities Revocation List (ARL): list only revoked CAs
    - Delta CRL – only new revocations since last `base CRL`
- Browsers mostly do not check CRLs. Instead:
  - Online Certificate Status Protocol (OCSP)
  - 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
- Solution: Hash-chain short-term certificate renewal
- Yearly-signed certificate, monthly-preimage-renewal
- December: sign new yearly cert, with $h^{(12)}(x)$
  - Random x
  - Each month, expose a preimage: $h^{(11)}(x)$, $h^{(10)}(x)$, …
    - Validate extension, e.g.: $h^{(11)}(x)=h(h^{(10)}(x))$

Online Certificate Status Protocol (OCSP)

- Improve efficiency, freshness cf. to CRLs
- Client asks CA about cert during handshake
- CA signs response (real-time)



TLS handshake with OCSP

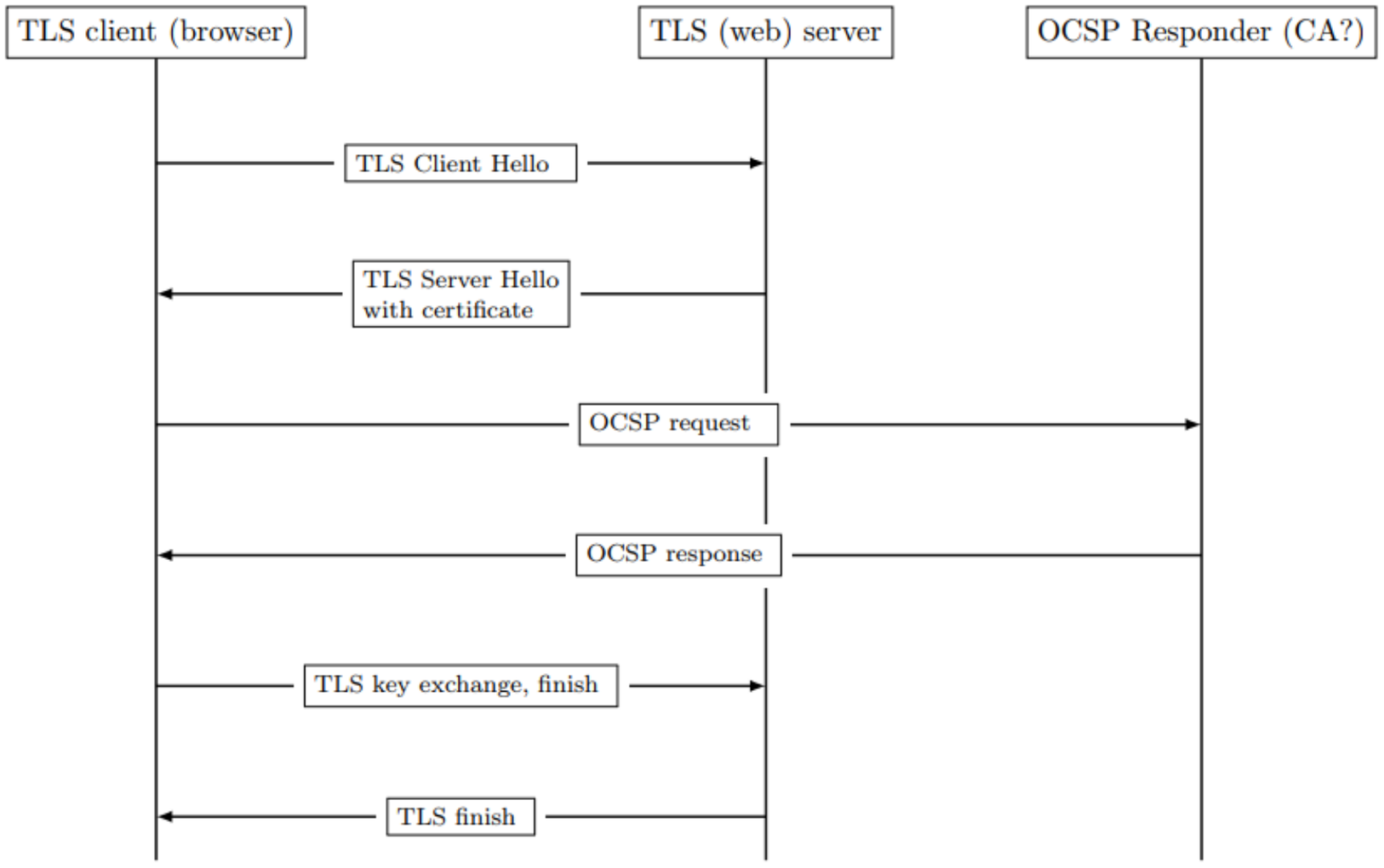


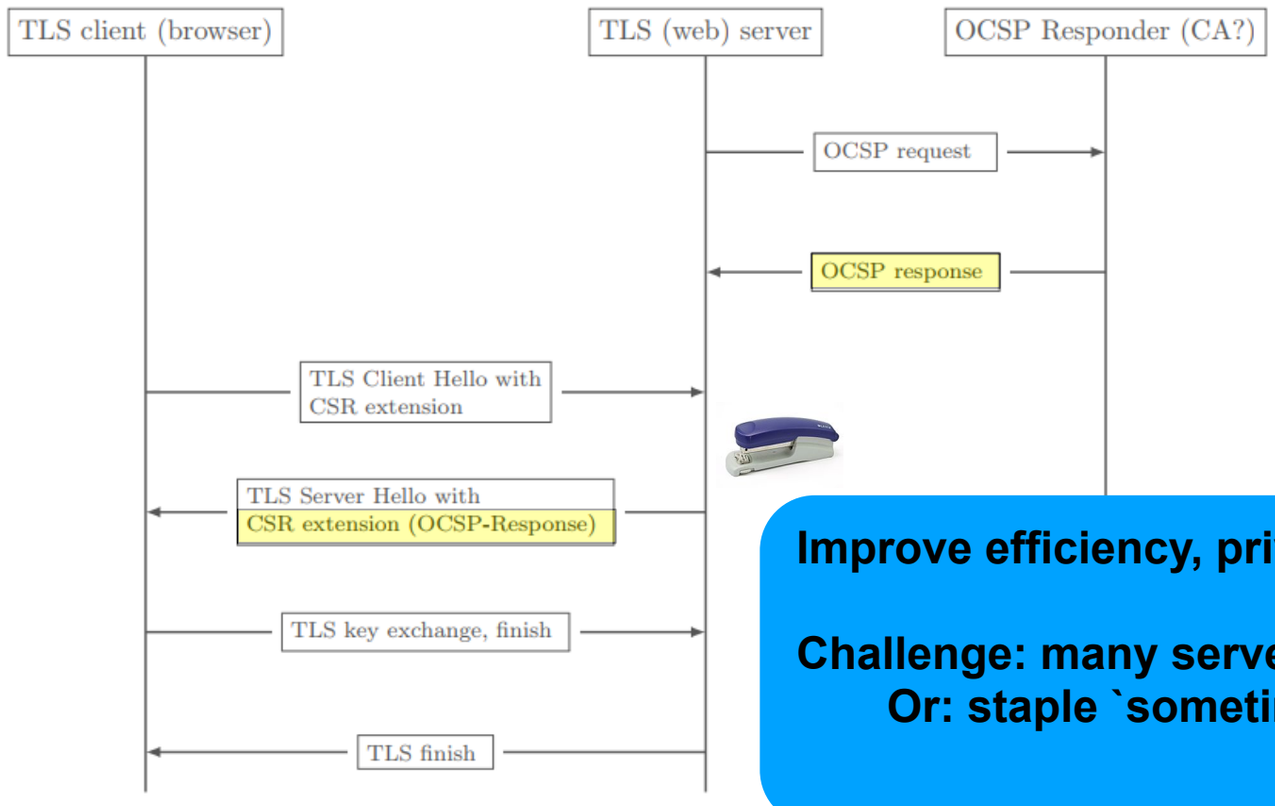
Figure 1: TLS handshake using OCSP (no stapling).

Online Certificate Status Protocol (OCSP)

- Client asks CA about cert during handshake
- CA signs response (real-time)
- Challenges:
  - Privacy: expose (domain, client) to CA
  - Load on CA
  - Delay (latency): on average, almost a second
  - Reliability: what if CA fails? No connectivity?
  - ➔ Most browsers skip OCSP or soft-fail: continue w/o OCSP response
- Better way to do OCSP?

OCSP-Stapling

Server runs OCSP, sends (`staples') the CA-signed response (CSR) during TLS handshake



Improve efficiency, privacy, reliability

Challenge: many servers don't staple!
Or: staple `sometimes/usually'

OCSP-Stapling

- Server runs OCSP, sends (`staples') the CA-signed response during TLS handshake
- Challenge: many servers don't staple!
  - Or, worse: staple `sometimes/usually'
  - So, try OCSP? Connect anyway? Disconnect?
- Solution: `Must-staple' cert. extension
  - RFC 7633
  - Mark as not critical
    - As it may not be supported by some browsers
- Most browsers don't use CRLs. Why?
  - Efficiency, freshness concerns
- OCSP: check cert-status `as needed'
- Signed responses (from trusted CA/server)

OCSP Optimisations

- Use Merkel Trees for membership proofs
- Use Merkel Trees for certificate revocation

Certificate Failure

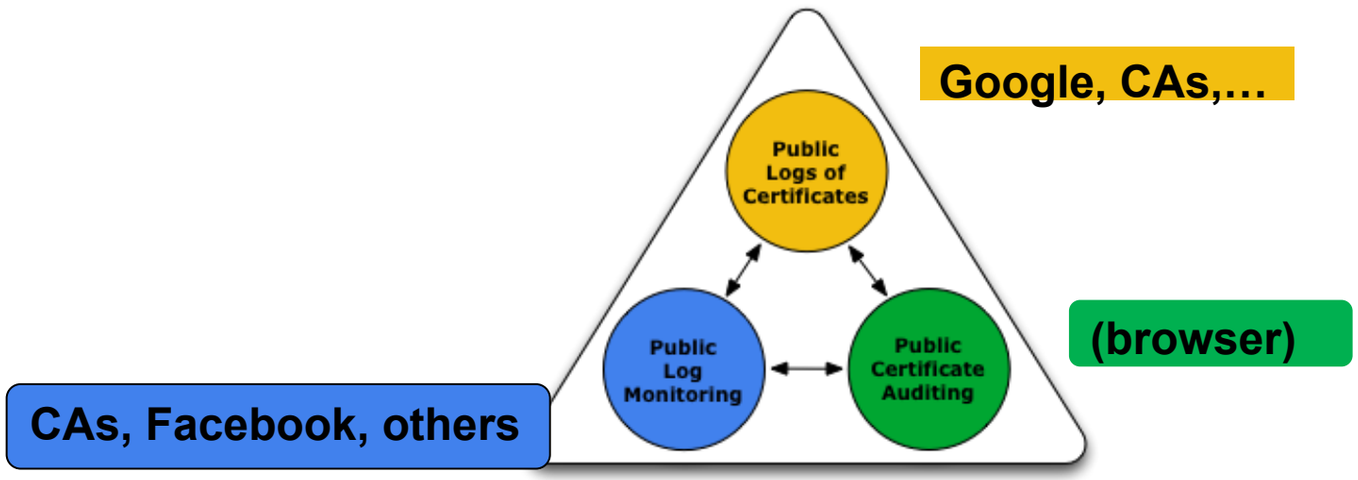
Why and How CAs fail?

- Many CAs `trusted' in browsers (as root)
- 'Domain-Validated' certificates
- Several well-known failures
  - DigiNotar, Comodo, Stuxnet, …
- Every CA can certify any domain (name)
  - Name constraints NOT applied (esp. to roots)
  - Some CAs may be negligible or even rogue
- Bad certificates:
  - Equivocation: rogue certificates for same name as a legit cert
  - Misleading certificates, e.g., similar name
- Can we improve defense against bad CA?

Defences against CA Failure

- **Use name constraints to limit risk**
  - who can issue global TLDs (.com, etc.)?
- **Static key pinning:** `burned-in' public keys
  - Detected MitM in Iran: rogue DigiNotar cert of Google
  - Limited: changing keys? Which keys to preload ?
- **Dynamic Pinning: HTTP Public-Key Pinning (HPKP)**
  - Server: I always use this PK / Cert / Chain
  - Client: remember, implement, detect & report attacks
  - Concerns: key loss/exposure, changing keys (recover security)
- Still, **Trust-On-First-Use (TOFU)** can be helpful
  - E.g. for security policies: OCSP-must-stapling or CAs-pinning
- Certificate Transparency (CT): **Accountability**
  - **Public, auditable certificates log**

Certificate Transparency

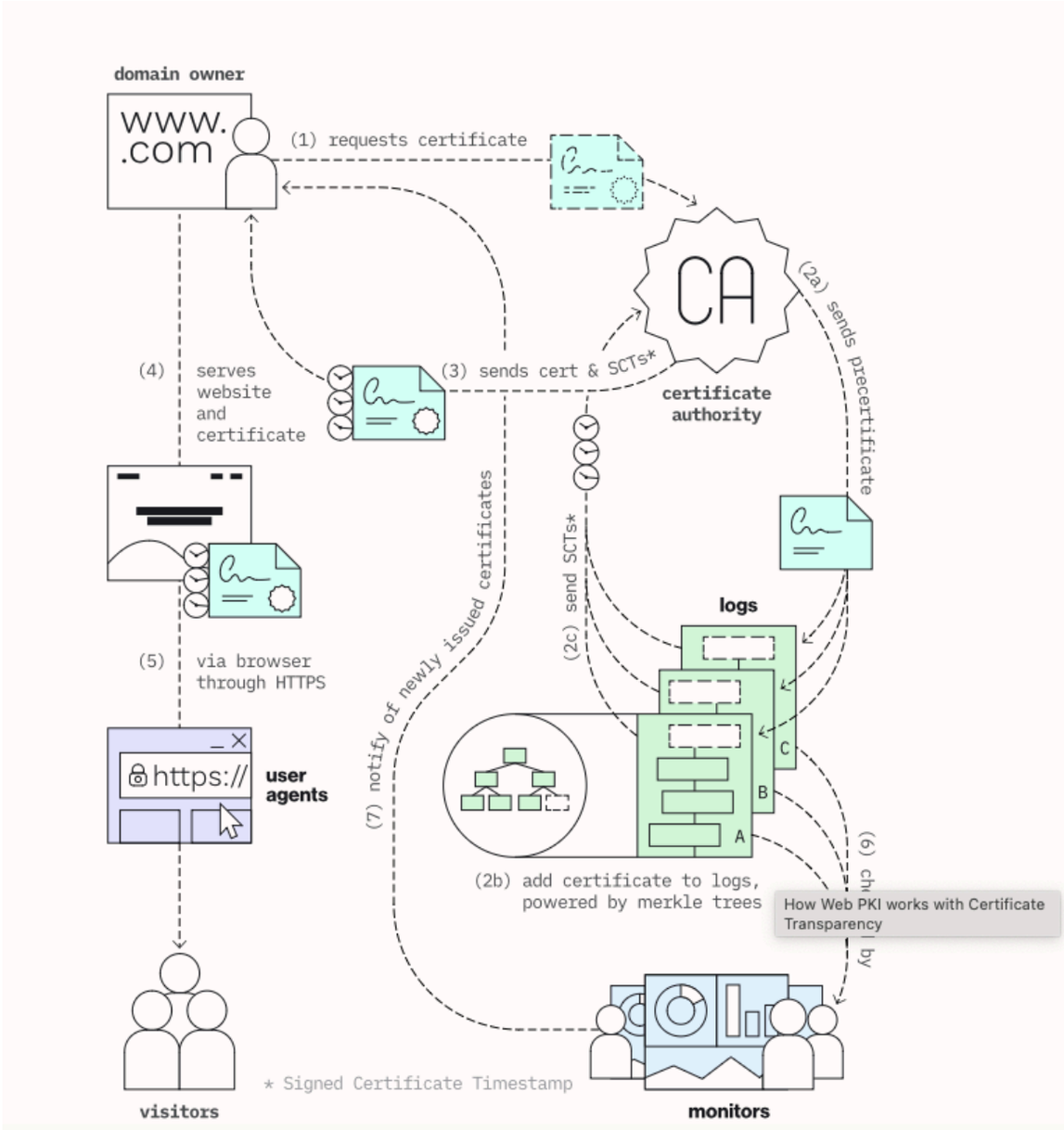


- **Loggers** provide public logs of certificates
- **Monitors** monitor certificates logged for detection of suspect certificates
- **Auditing** (auditors?): check for misbehaving loggers

CT Goals

- Easier to detect, revoke rogue certificates
- Easier to detect, dis-trust rogue CAs:
  No (real) accountability without transparency !
- What about rogue loggers ?
  - Option 1: Honest-Logger CT (HL-CT) [RFC6962]
    - Assume honest logger [out of two loggers – redundancy]
    - In Google we trust ?
  - Option 2: NTTP-Secure CT (NS-CT):
    - Monitors, relying-parties detect misbehaving loggers
    - Relying party decides which **monitor(s)** it relies on (trusts) !
    - Original CT goal: ‘no trusted third party’

Certificate Transparency



https://certificate.transparency.dev/

Thank you