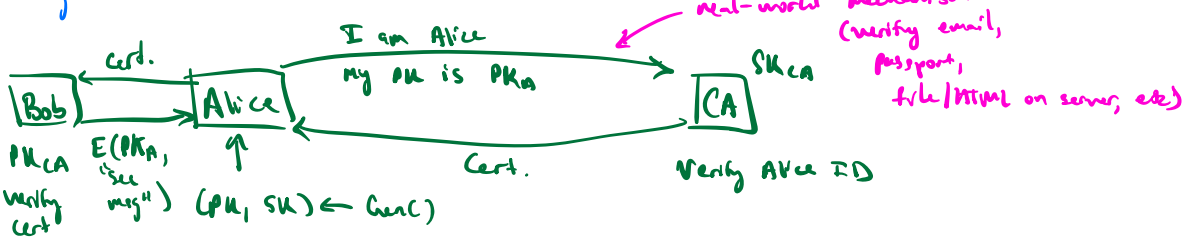
Public Key Mgmt: Certificates

How does Bob obtain Alice's PK_A?

Single domain cert. auth. (CA)



Cert: Binds PK_A to physical identity of Alice

$$\text{Cert} = \left[\text{issuer name} \mid \text{subject name} \mid PK_A \mid \text{validity period} \mid \ldots \right] + \left[\text{CA's sig on pre-cert} \right]$$

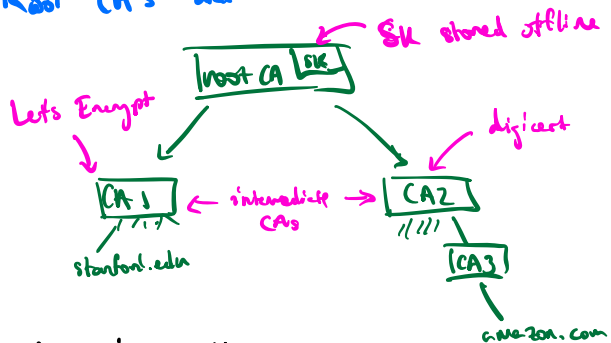
issuer name → CA; subject name → Alice; validity period → Exp. date

pre-cert

Notes.

1. Alice talks to CA during keygen
2. CA does not have Alice's SK!
3. Everyone needs PK_{CA} to verify certs.

Root CA's and intermediate CA's



Total of 60 root CAs, 1200 root CAs worldwide

Amazon's Cert:

$$\left[(\text{amazon.com} \mid PK_A \mid CA_3)_{\text{sig}},\ (CA_3 \mid PK_3 \mid CA_2)_{\text{sig}},\ (CA_2 \mid PK_2)_{\text{root sig}} \right]$$

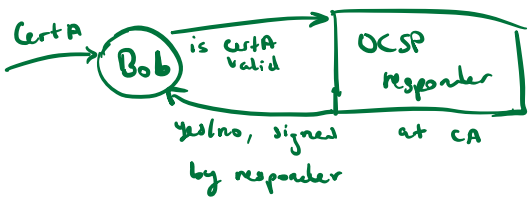
Cert. chain of depth 3

Certificate Revocation

Suppose Stanford's SK is stolen ⇒ need to revoke cert

Lines of defense:

1. expiry date: cert only valid until expiry (Let's Encrypt: 3 mo)
2. CRL set: cert. revocation list
   list of serial #'s of revoked certs
   shipped daily by Google to Chrome, Firefox, etc
3. OCSP: Online cert. status protocol



Problem: 1. Bob gets no response
   → fail open: cannot prevent traffic if OCSP offline
2. Privacy violation: CA knows every site you've been to
3. Performance: what if OCSP is slow?

⇒ on its way out for these reasons.

4. Short-lived certs (4 days)
   → Site maintains cache of 4-day certs
   → Every day, request a new 4-day cert
   → To revoke: CA stops issuing new cert

⇒ not in use yet.

Negligent CA's

2011: Diginotar hacked ⇒ *.google.com, gmail.com, etc, hacked

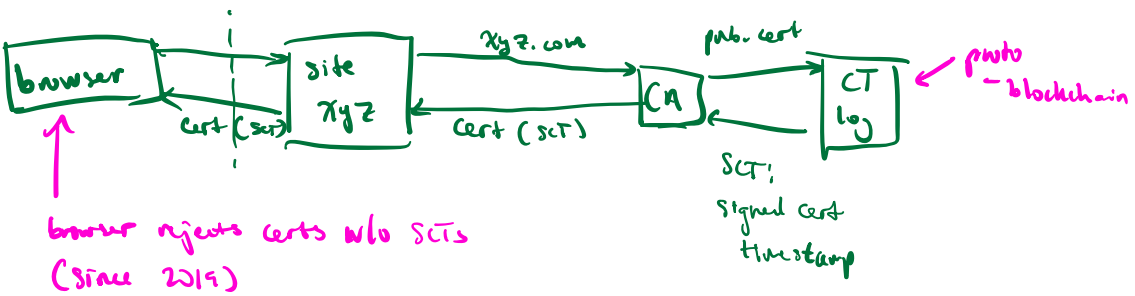
⇒ CA untrusted by browsers

Solution:

1. Pinning: browser ships w/ list of allowable CAs for some domains ← pins
   ex. gmail.com is pinned to GTS CA.

2. Cert transparency (CT)

   Require CA to publish log of all issued certs



XYZ admin periodically scans all logs to see if any bad certs were issued for xyz.com.