Alice gave me your PK → Bob

PKb → M → Enc(PKb, m) → m

Attacker MitM → PKadv → PKb

Enc(PKadv, m) → Enc(PKb, M) → M

Trusted directory

random

drawn

by Alice

Alice

PKtd

packet signed with PKb

PKb: sign(Kid) = PKtd

Bob's name

PK MitM

Assume update happens securely

Updating a key

Replay attack:

Attacker replays old information
(old sig with old PK)
Alice embeds nonce in her request
checks sig from TD to contain
nonce & to verify with PK_TD & contains Bob’s name
⇒ knows PK of Bob is latest

Drawbacks of TD

- Scalability (store & serve all PKs)
- TD is a central point of attack/trust
- difficult to recover from TD compromise
- updating key requires trust
- TD has to be always available
  - central point of failure
Approach 2: Digital Certificates

Association between name & PK by a CA (Certificate Authority)
   e.g. VenSign

Certificate: $\text{Sign}$(SK_{CA}, \text{Bob's PK is } \text{0x5f} \ldots) \Rightarrow \text{cert}_{Bob}

Assume browsers have PK_{CA} hardcoded

[Anyone can serve PK_{Bob}, cert_{Bob}]

Alice checks:
   - cert_{Bob} verifies with PK_{CA}, is not expired, is for Bob

Alice no longer contacts TN to fetch PK_{Bob}, but can contact local server, e.g. Bob's Server
Alice → give me your PK

PKb, cert → SKb, PKb

CA

Better than TD:
1. can contact Bank (or anyone) to obtain PK
2. + better scalability compared to TD

Certificate hierarchies & chains

Versign CA → only certifies level underneath, presidents

UC President

Stanford president

David, Raluca

C1 = sign(SKca, "UC Pre. has PKu", expiry)

C2 = sign(SKca, "David has PKd", expiry)

When I ask for David’s PK; I will receive PKd, C1, C2,
- check PKu using C1 using PKca
- check PKd using C2 and knowledge of PKu

David@berkeley.edu

path of authority
Root servers serve cert. edu

Revocation

How can we revoke a certificate that has not yet expired?

- Wait till expiry, make expiry shorter
- Revocation lists: CA could push revocation
  Sign (SKCA, "Revoke cert") into browsers; not ideal solution because
  browsers might not be downloading lists frequently

Long-term problem with CAs:

- CAs can be compromised or could be deceived to sign incorrect certificates

- Transparency logs promise to address this problem