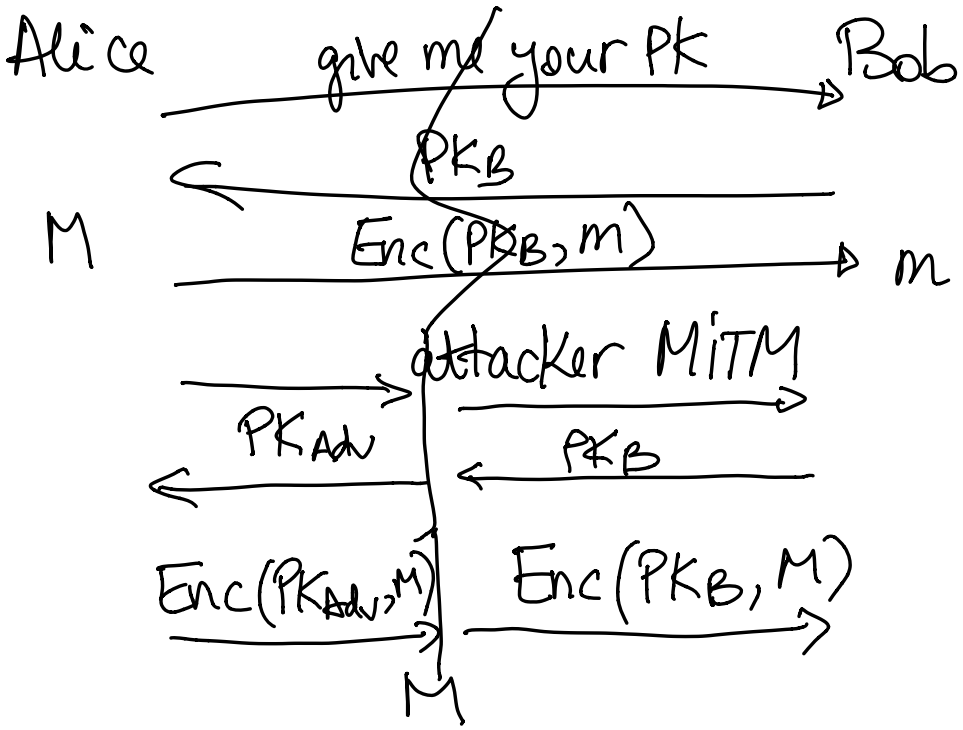
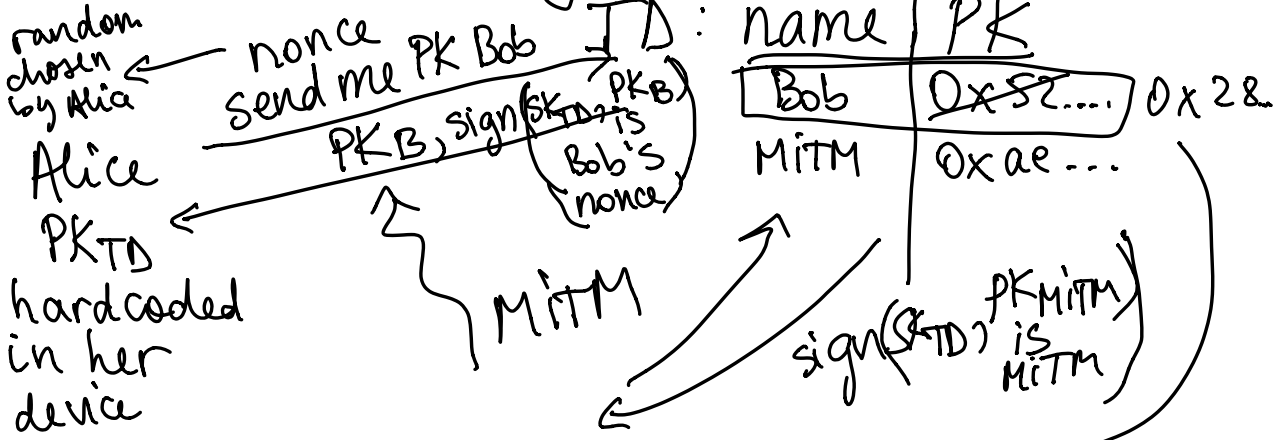


Trusted directory



Updating a Key

Assume update happens securely

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)
eg. Verisign

certificate: $\text{sign}(SK_{CA}, \text{Bob's PK is } 0x5f\ldots, \text{expiry date}) = cert_{Bob}$

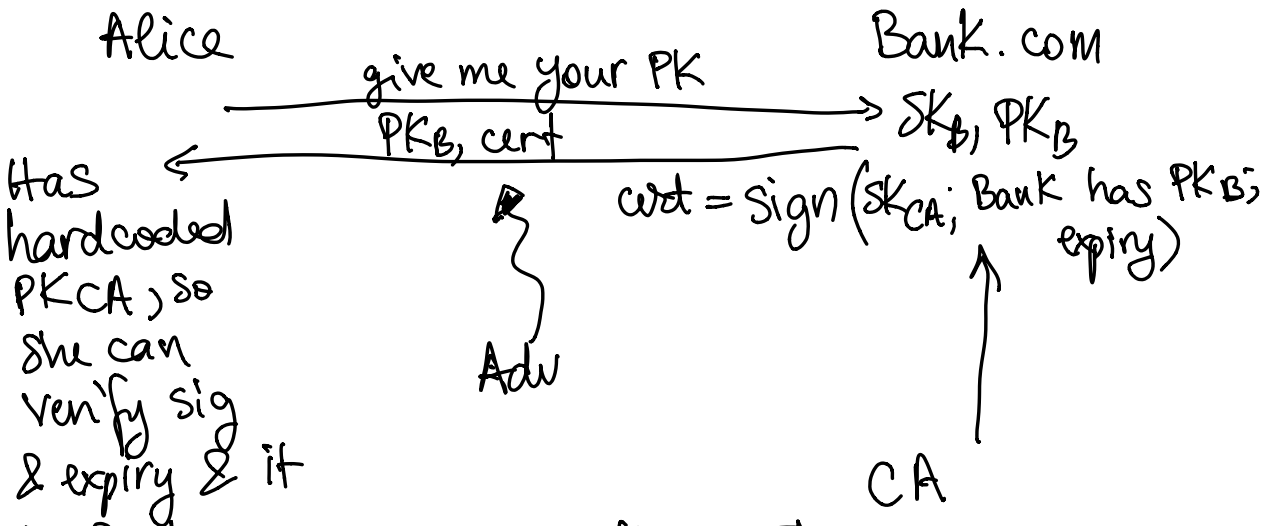
assume browsers have PK_{CA} hardcoded

+ 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 TD to fetch PK_{Bob}, but can contact local server, e.g. Bob's server

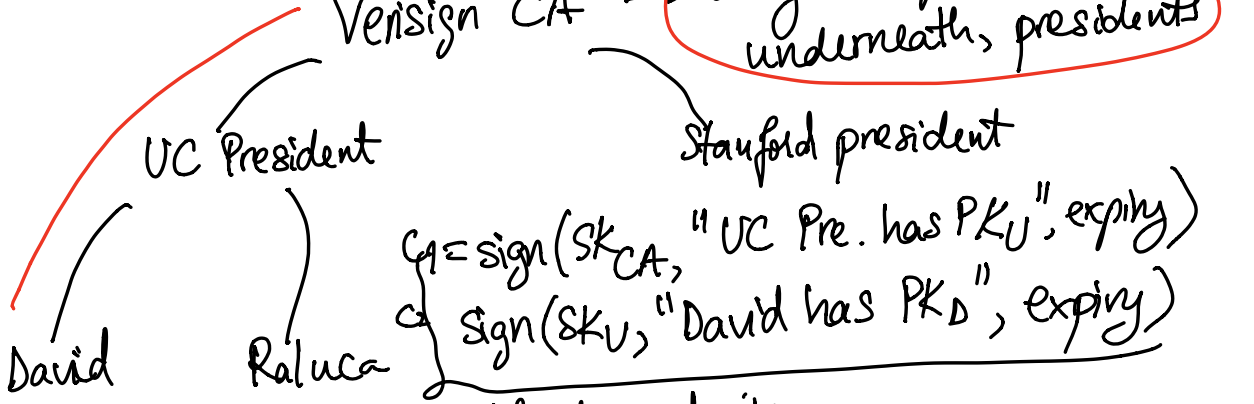


Better than TD:
+ can contact Bank (or anyone) to obtain PK

Certificate hierarchies & chains

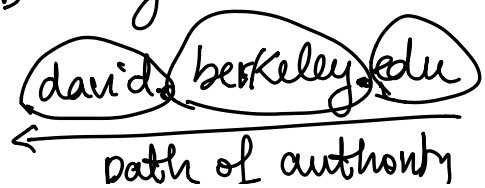
+ better scalability compared to TD

Verisign CA → only certifies level underneath, presidents



When I ask for David's PK; I will receive PK_D, C_1, C_2, PK_U

- check PK_U using C_1 using PK_{CA}
- check PK_D using C_2 and knowledge of PK_U



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(SK_{CA}, "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