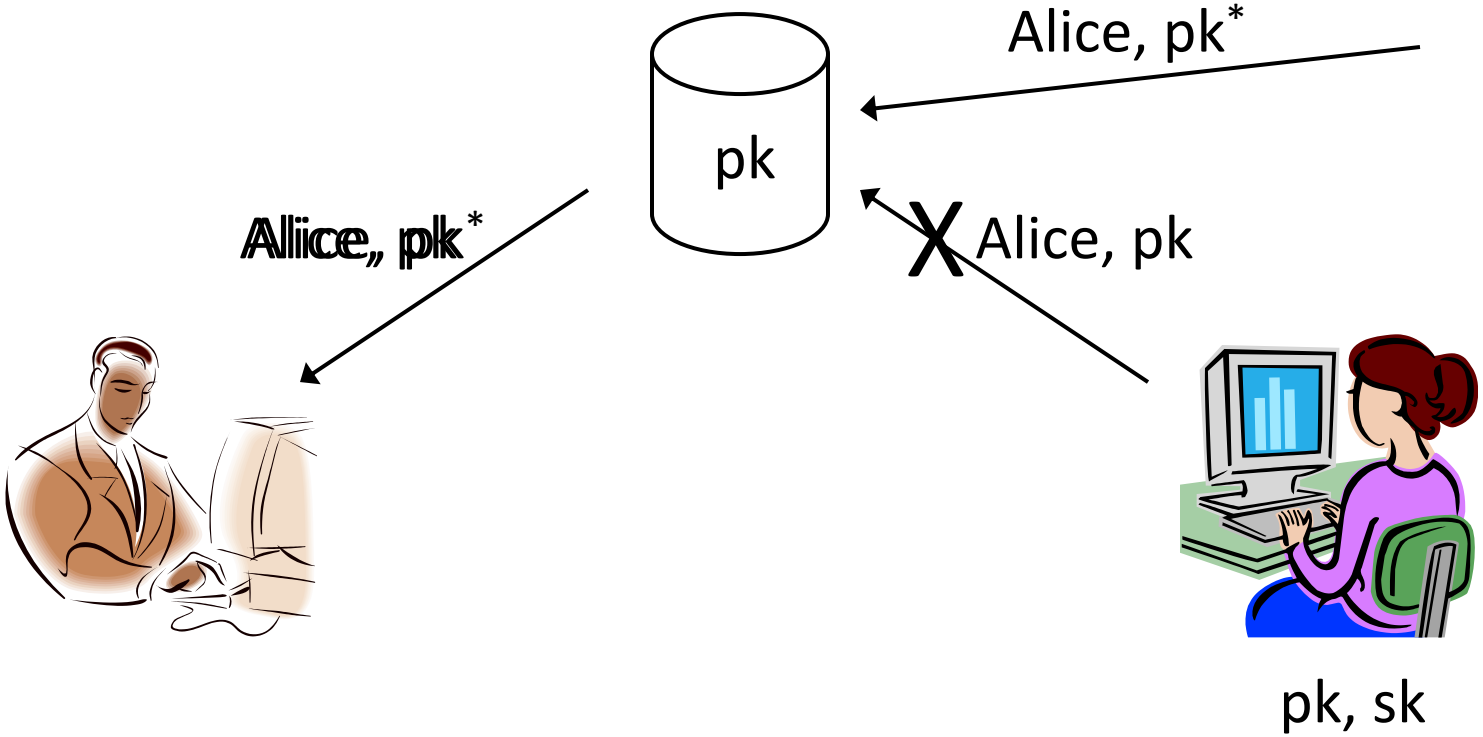
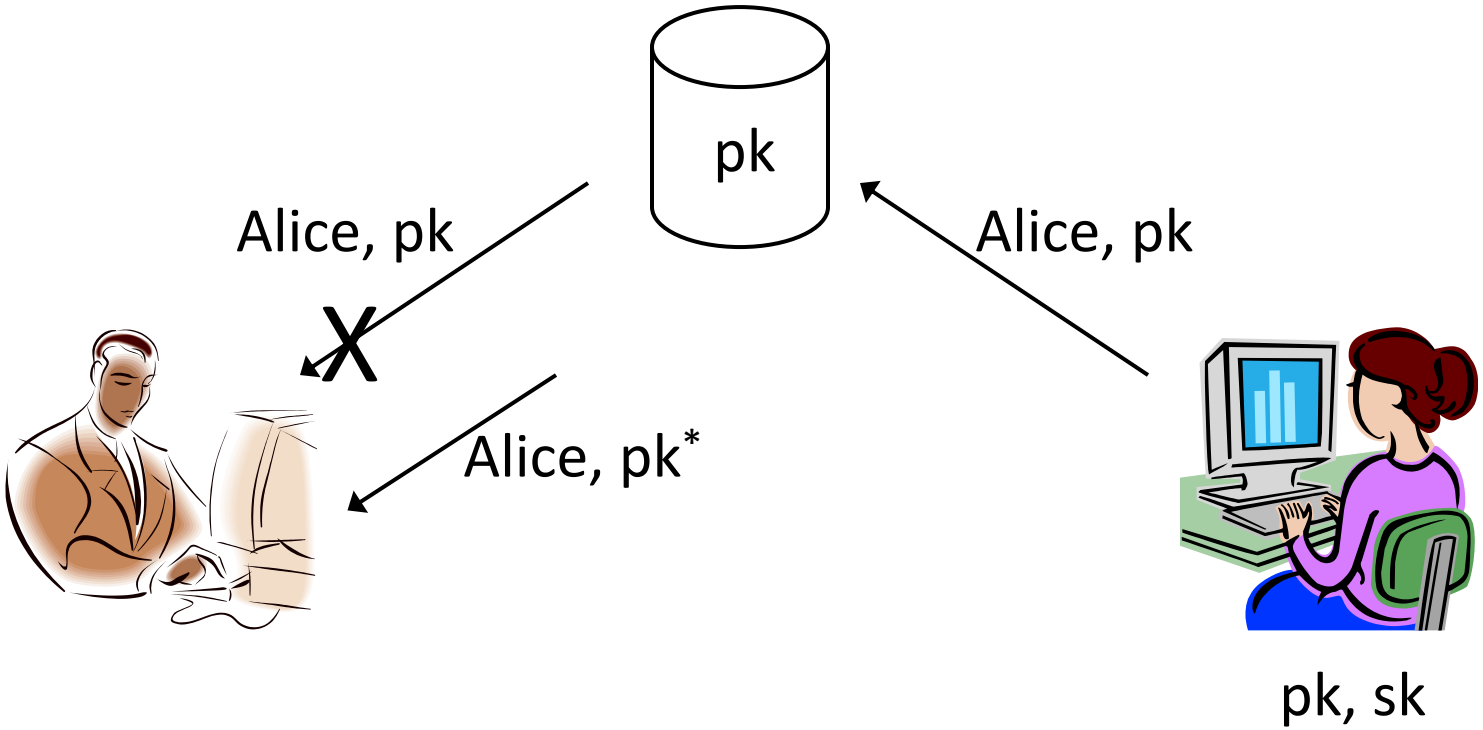
Public Key Infrastructure

Slides by Prof. Jonathan Katz.
Lightly edited by me.

Public-key distribution



Public-key distribution



Use signatures for secure key distribution!

- Assume a trusted party with a public key known to everyone
  - CA = certificate authority
  - Public key pk_{CA}
  - Private key sk_{CA}

Use signatures for secure key distribution!

- Alice asks the CA to sign the *binding* (Alice, pk)

$$\text{cert}_{\text{CA}\rightarrow\text{Alice}} = \text{Sign}_{\text{sk}_{\text{CA}}}(\text{Alice, pk})$$

- (CA must verify Alice's identity out of band)

Use signatures for secure key distribution!

- Bob obtains Alice, pk, and the certificate $\text{cert}_{\text{CA}\to\text{Alice}}$ …
  - … check that $\text{Vrfy}_{\text{pK}_{\text{CA}}}((\text{Alice}, \text{pk}), \text{cert}_{\text{CA}\to\text{Alice}}) = 1$
- Bob is then assured that pk is Alice's public key
  - As long as the CA is trustworthy…
    - Honest, and properly verifies Alice's identity
  - …and the CA's private key has not been compromised

Chicken-and-egg problem?

- How does Bob get pk_{CA} in the first place?

- Several possibilities…

“Roots

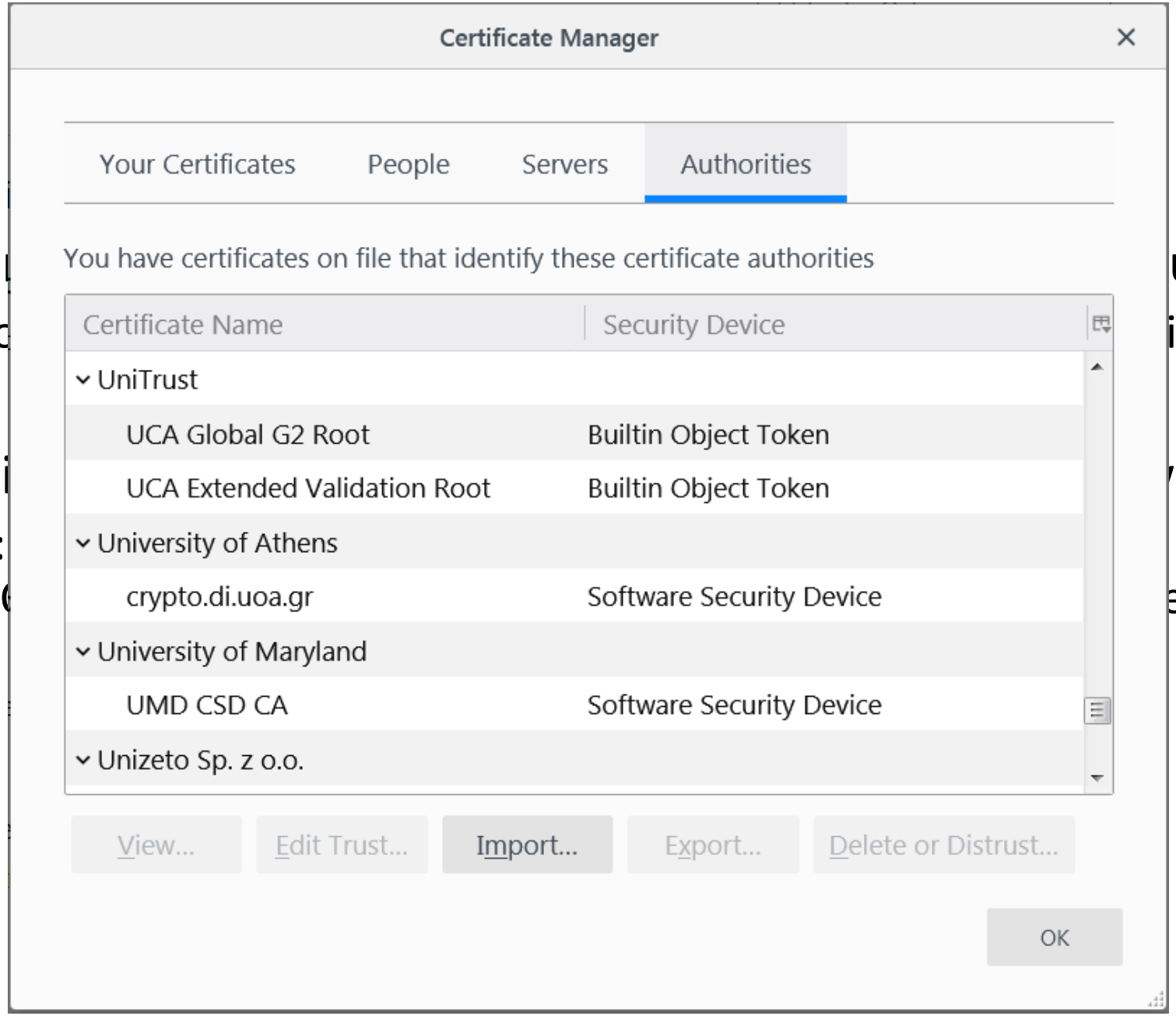
- Bob only
  - Need to
- E.g., distri
  - Firefox:
    Tools->

ublic keys

ic keys

yser

es



“Web of trust”

- Obtain public keys *in person*
  - “Key-signing parties”
- Obtain “certificates” on your public key from people who know you
- If A knows pk_B, and B issued a certificate for C, then C can send that certificate to A
  - What trust assumptions are being made here?

Public repository

- Store certificates in a central repository
  - E.g., MIT PGP keyserver
- To find Alice's public key
  - Get all public keys for "Alice," along with certificates on those keys
  - Look for a certificate signed by someone you trust whose public key you already have

PKI in practice…

- Does not work quite as well as in theory…
  - Proliferation of root CAs
    - Compromises of CAs
  - Revocation
  - Users/browsers may not verify certificates