PennState

CSE 443: Introduction to Computer Security
Module: Key Management

Prof. Syed Rafiul Hussain
Department of Computer Science and Engineering
The Pennsylvania State University

Acknowledgements: Some of the slides have been adopted from
Trent Jaeger (Penn State), Patrick McDaniel (Penn State), Ninghui Li (Purdue) and William Enck (NCSU)

Key Distribution/Agreement

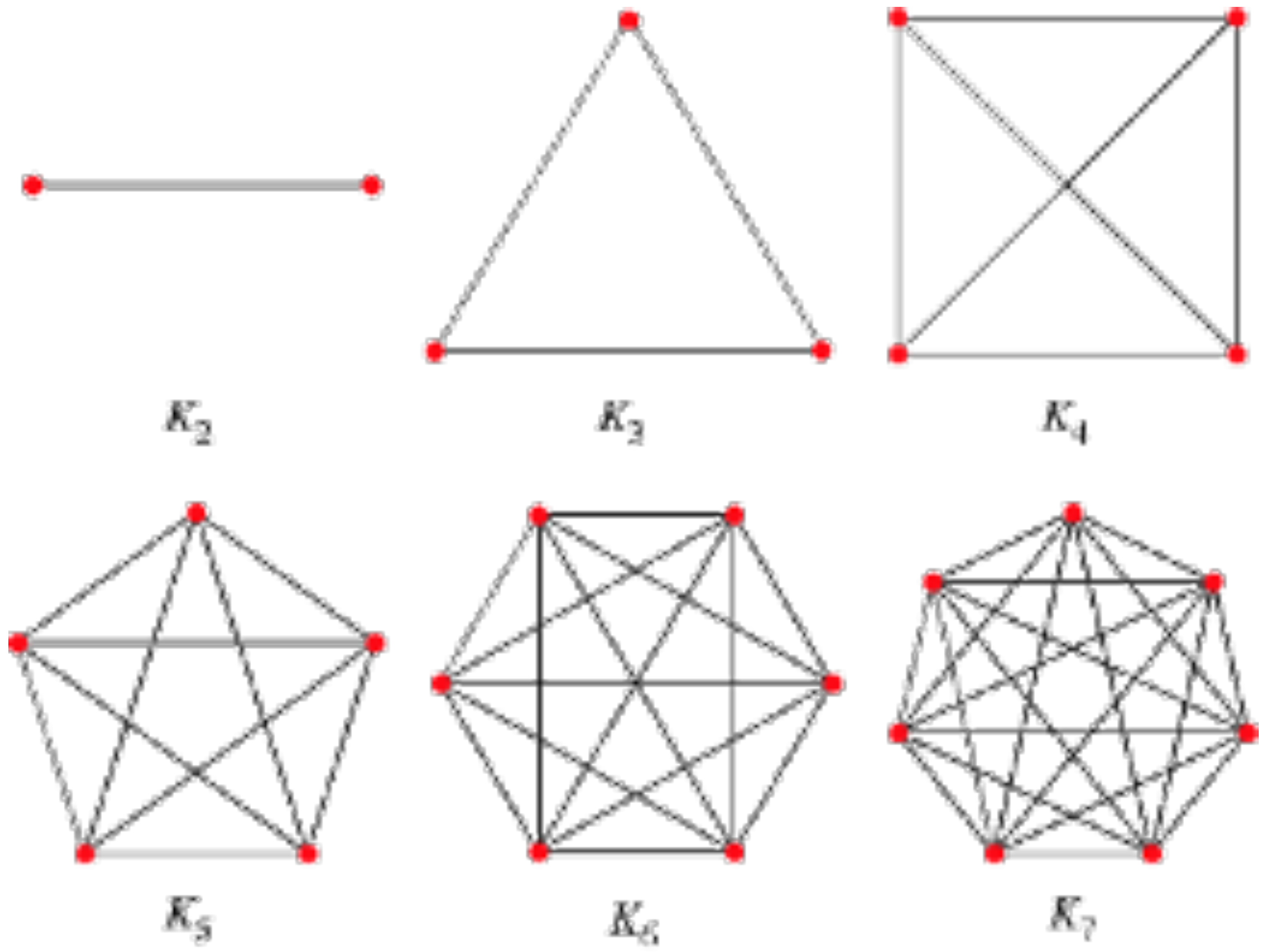
- Key Distribution is the process where we assign and transfer keys to a participant
  - Out of band (e.g., passwords, simple)
  - During authentication (e.g., Kerberos)
- Key Agreement is the process whereby two parties negotiate a key
  - 2 or more participants
- Typically, key distribution / agreement occurs in conjunction with or after authentication
  - However, many applications can pre-load keys

Key Distribution

- Secure key distribution without asymmetric cryptography is difficult
- Simple approach: send key though an out-of-band channel

Key Distribution

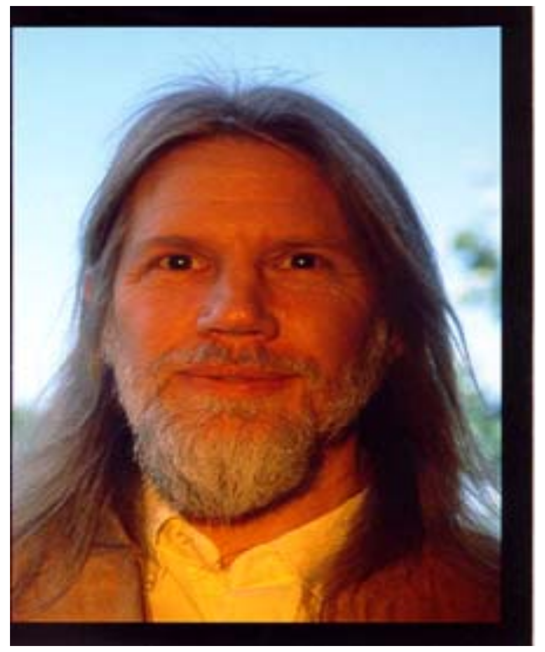
- Pairwise key distribution requires plastic cups



Key Agreement

- What happens if there is no out-of-band communication channel to share the key?
  - Diffie-Hellman Key Agreement protocol discussed in the last lecture.
  - Setup: We pick a prime number p and a base g ($<p$)
    - This information is public
    - E.g., $p=13$, $g=4$
  - Step 1: Each principal picks a private value x ($<p-1$)
  - Step 2: Each principal generates and communicates a new

$$A = g^x \bmod p$$



  - Step 3: Each principal generates the secret shared key z

$$Key = g^{xy} \bmod p$$

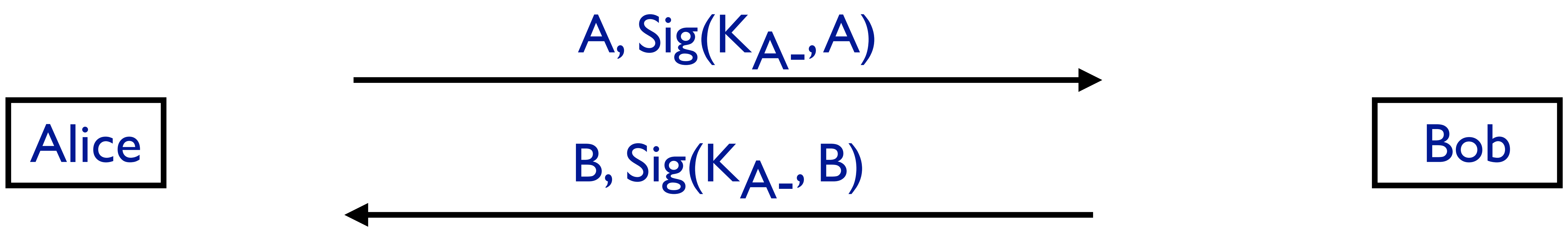
Meet-in-the-Middle Attack

- This is key agreement, not authentication
  - You really don't know anything about who you have exchanged keys with

•Alice and Bob think they are talking *directly* to each other, but Mallory is actually performing **two separate exchanges**

Authenticated DH

- Alice and Bob need a way to authenticate the received A and B values
  - Multiple ways to do this, here's one (vuln to replay)
-



Alice's public key is K_{A+} and private key is K_{A-}

Bob's public key is K_{B+} and private key is K_{B-}

$A = g^x \bmod p$

$B = g^y \bmod p$

Alice: $Key = g^{xy} \bmod p$

Bob: $Key = g^{xy} \bmod p$

Perfect Forward Secrecy

- Why use authenticated DH vs. Alice choosing a secret k, signing it, and encrypting it with Bob's public key?
- Answer: it provides perfect forward secrecy
  - K is valid just for the session (ephemeral)
  - K cannot be computed later if the adversary obtains
    - All network traffic
  - Either (or both) of Alice and Bob's private keys (e.g., via subpoena)

How do we verify it's correct public key?

- Every user has his/her own public and private key
- Public keys are all published in a database
- Alice gets Bob's public key from the database

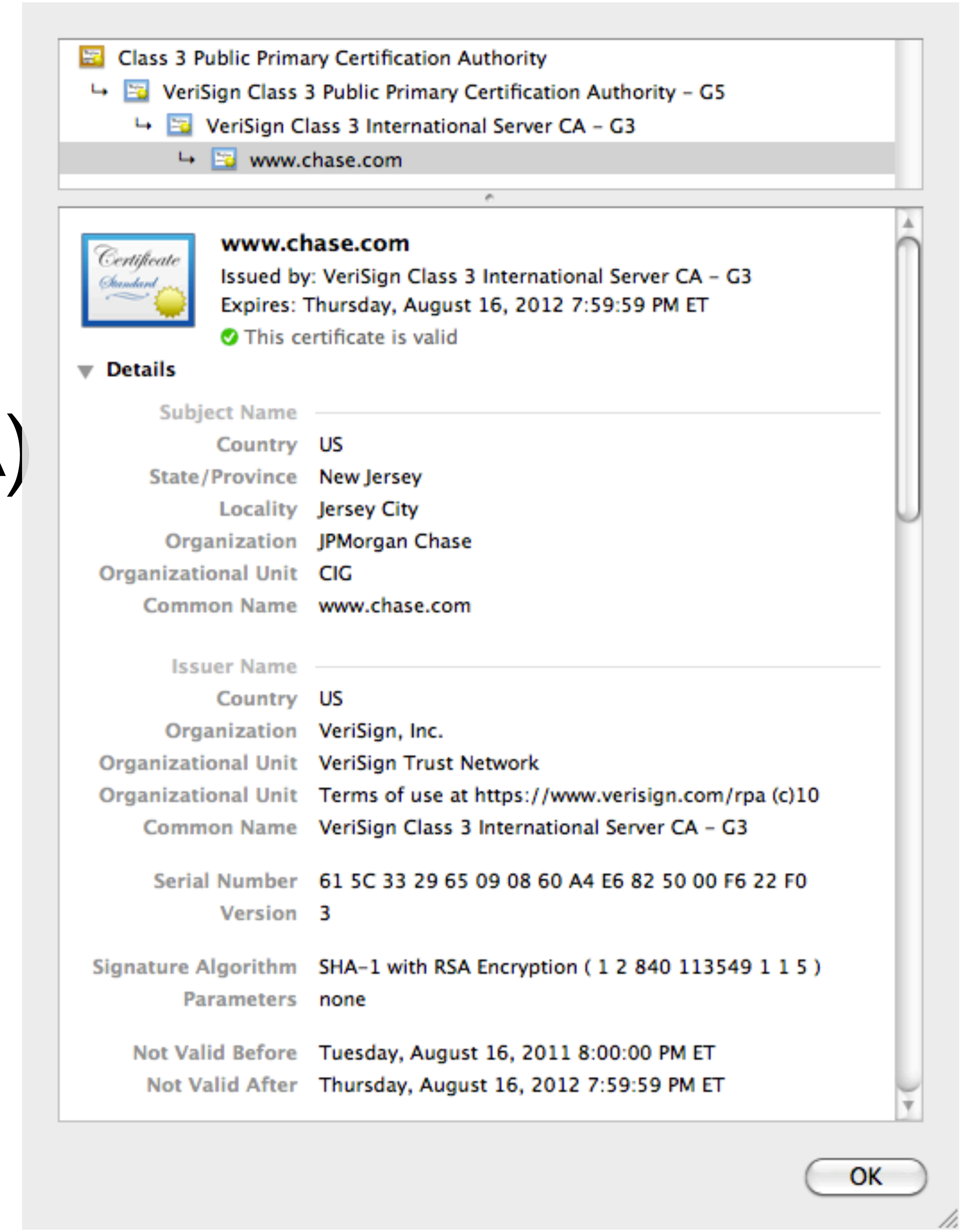
What's the problem with this approach?

Solving the Turtles Problem

- We need a trust anchor
  - There must be someone with authority
  - Requires a priori trust
- Solution: form a trust hierarchy
  - "I believe X because …"
  - "Y vouches for X and …"
  - "Z vouches for Y and …"
  - "I implicitly trust Z."

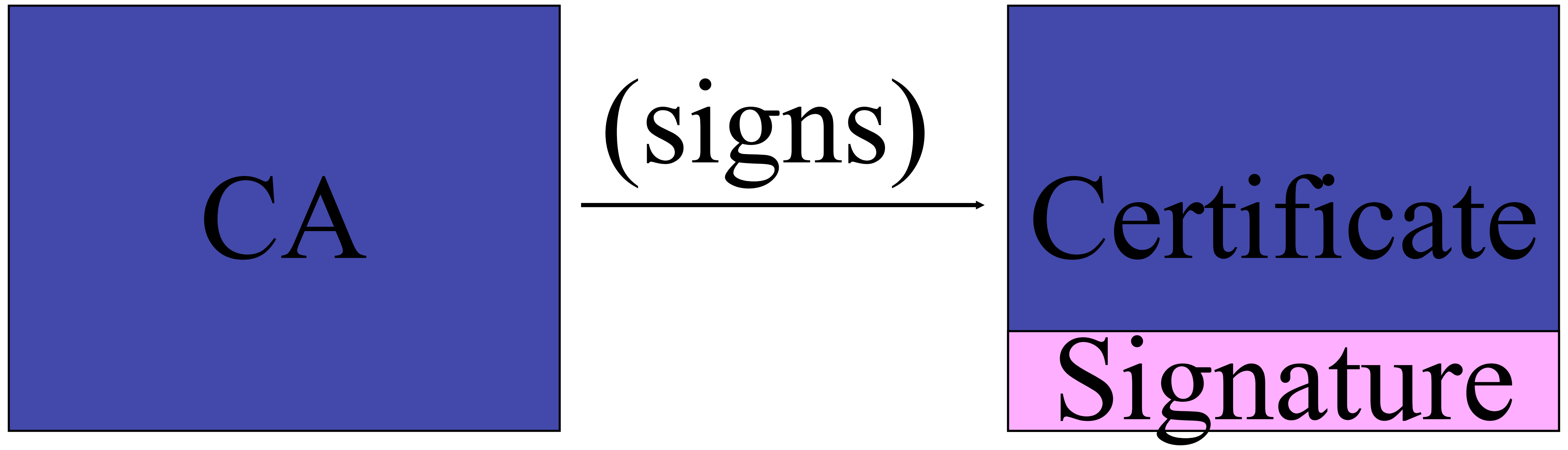
What is a certificate?

- A certificate …
  - … makes an association between a user identity/job/attribute and a private key
  - … contains public key information {e,n}
  - … has a validity period
  - … is signed by some *certificate authority* (CA)
  - ... identity may have been vetted by a *registration authority* (RA)
- Issued by CA for some purpose
  - Symantec is in the business of issuing certificates
  - People trust Symantec (formerly Verisign) to vet identity



Why do I trust the certificate?

- A collections of "root" CA certificates
  - … baked into your browser
  - … vetted by the browser manufacturer
  - … supposedly closely guarded (yeah, right)
- Root certificates used to validate certificate
  - Vouches for certificate's authenticity



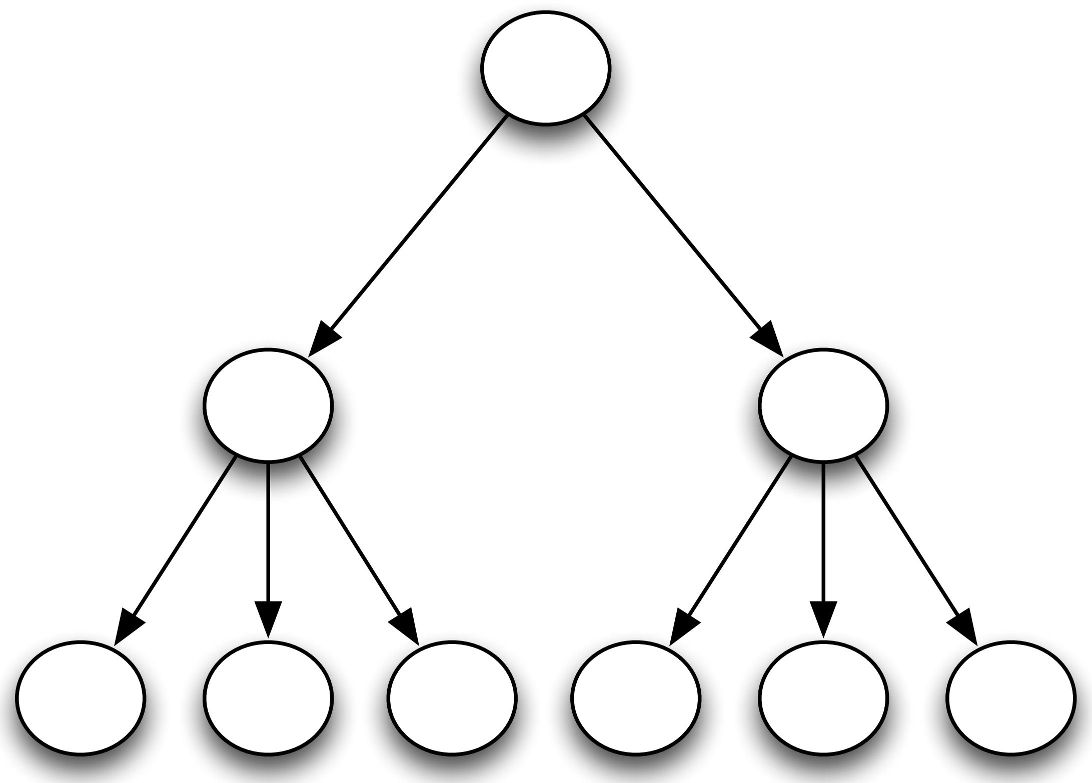
Public Key Infrastructure

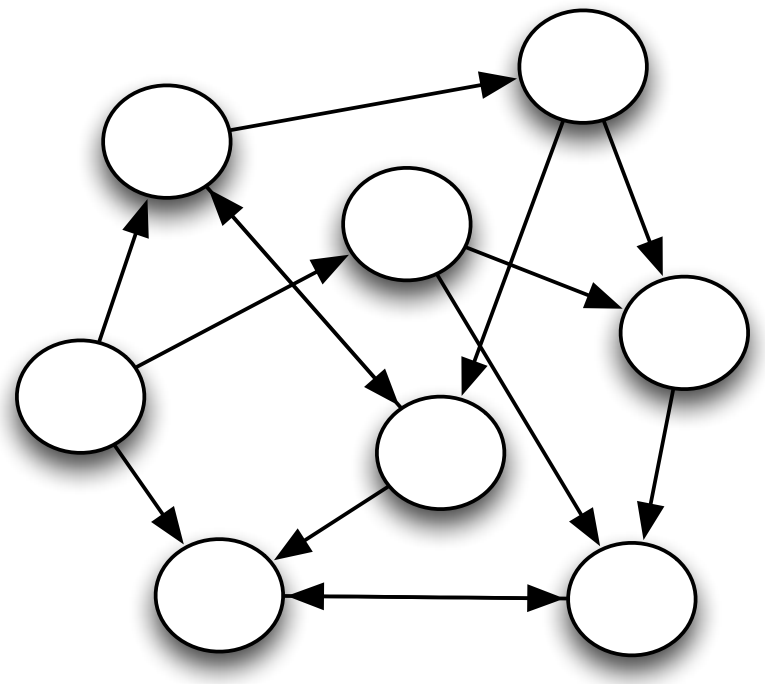
- System to "*securely distribute public keys (certificates)*"
  - Q: Why is that hard?

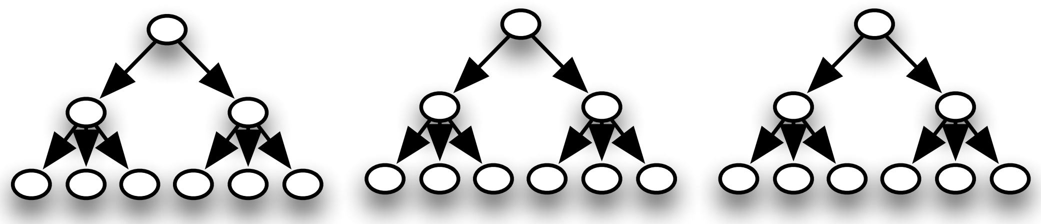
- Terminology:
  - Alice signs a certificate for Bob's name and key
    - Alice is issuer, and Bob is subject
  - Alice wants to find a path to Bob's key
    - Alice is verifier, and Bob is target
  - Anything that has a public key is a principal
  - Anything trusted to sign certificates is a trust anchor
    - Its certificate is a root certificate

Possible PKI Constructions

- Monarchy
  - Single globally trusted third party
- Anarchy
  - No globally trusted third party
    - e.g., Using MIT's PGP keyserver
- Oligarchy
  - Multiple globally trusted third parties
    - Model used in the Internet

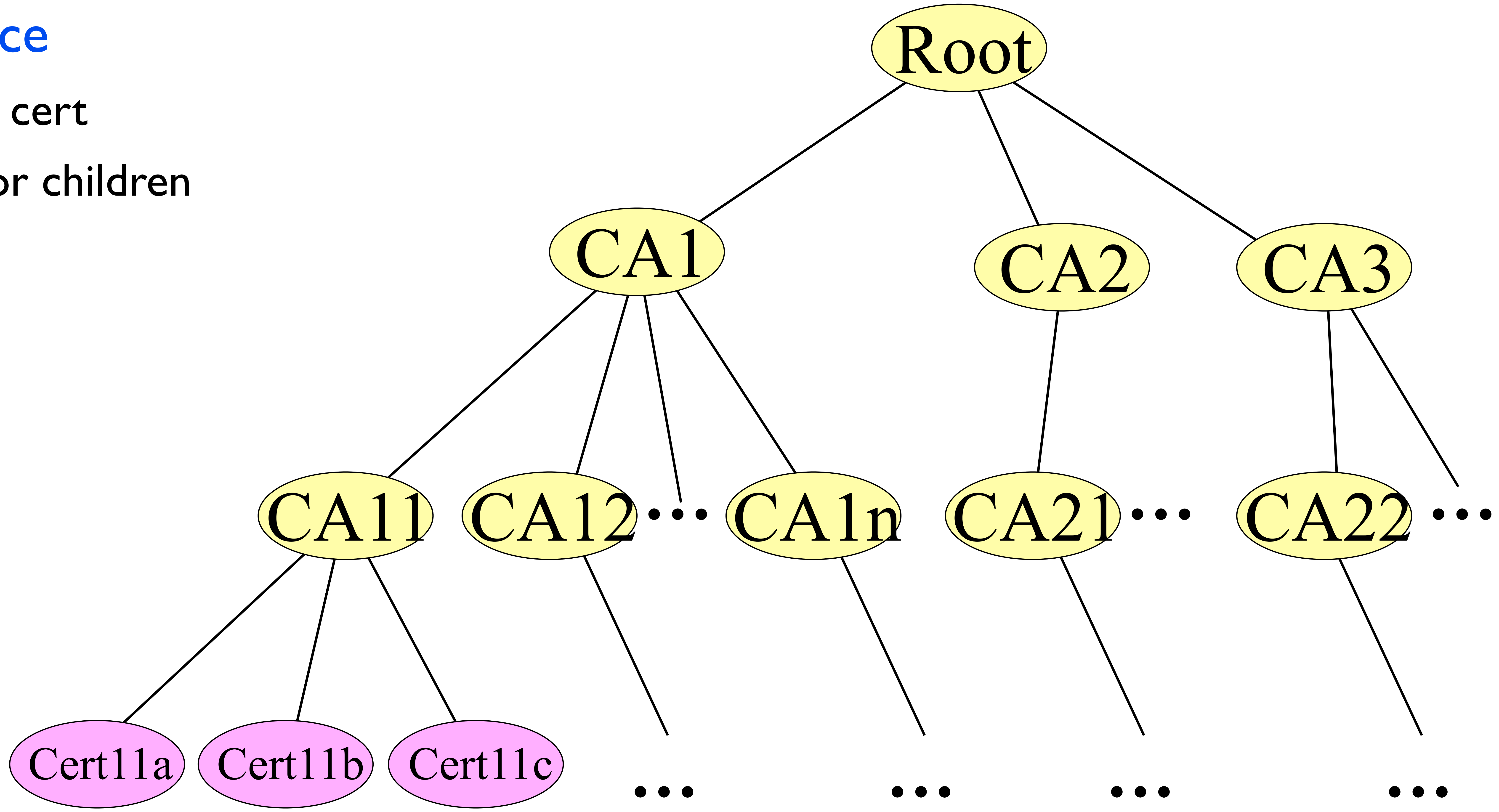






The Internet PKI?

- Rooted tree of CAs
- Cascading issuance
  - Any CA can issue cert
  - CAs issue certs for children

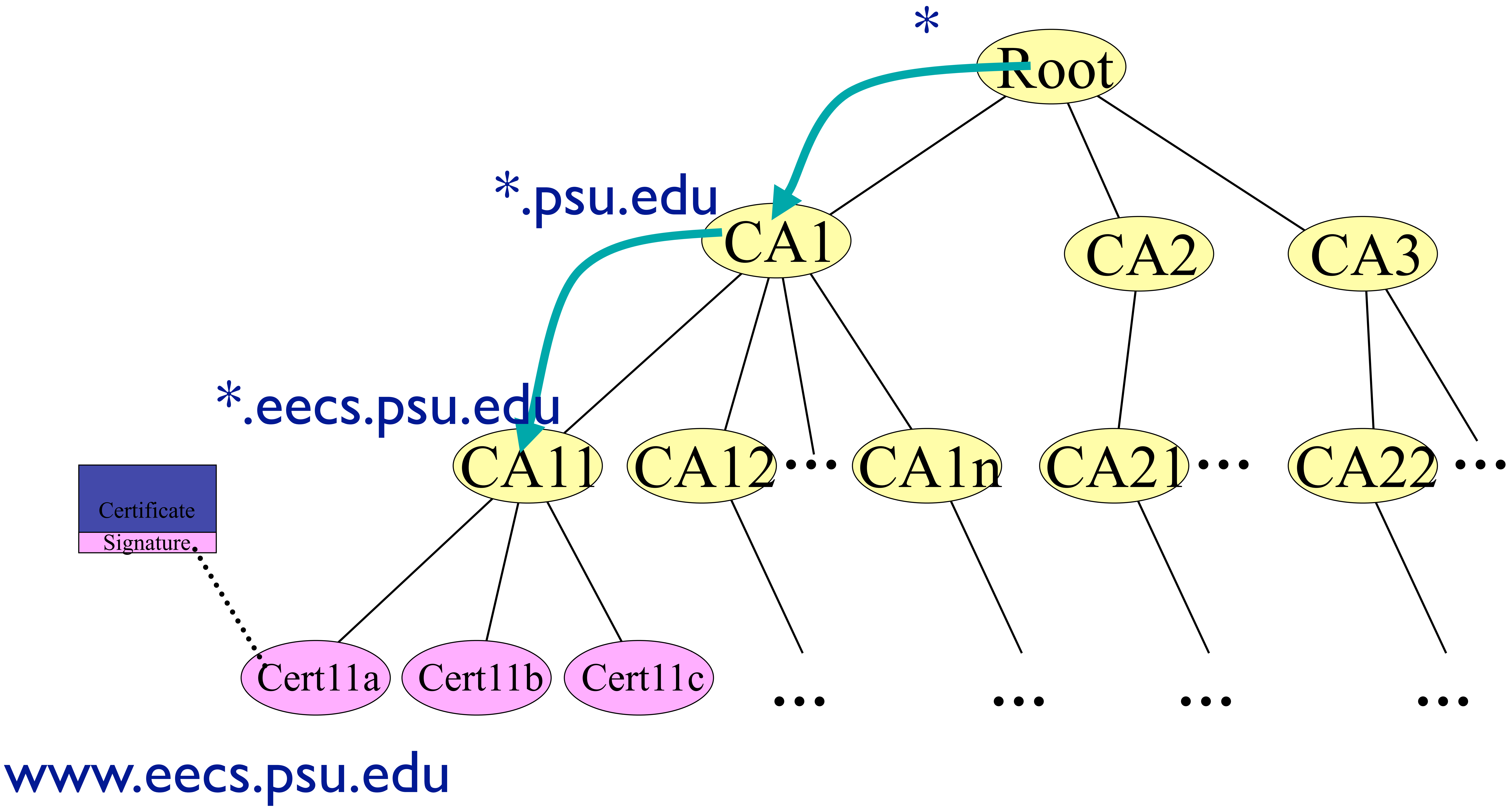


Obtaining a Certificate

- Alice has some identity document AID and generates a keypair (K_{A-}, K_{A+})
- A → CA : {K_{A+}, ID_A}, $Sig_{K_{A-}}$ ({K_{A+}, ID_A})
  - CA verifies signature -- proves Alice has K_{A-}
  - CA may (and should!) also verify ID_A offline
- CA signs {K_{A+}, ID_A} with its private key (CA-)
  - CA attests to binding between A+ and ID_A
- CA → A : {K_{A+}, ID_A}, Sig_{CA-} ({K_{A+}, ID_A})
  - this is the certificate; Alice can freely publish it
  - anyone who knows CA+ (and can therefore validate the CA's signature) knows that CA "attested to" {K_{A+}, ID_A}

Important: CA does not learn K_{A-}!

Certificate Validation



Certificate Authorities

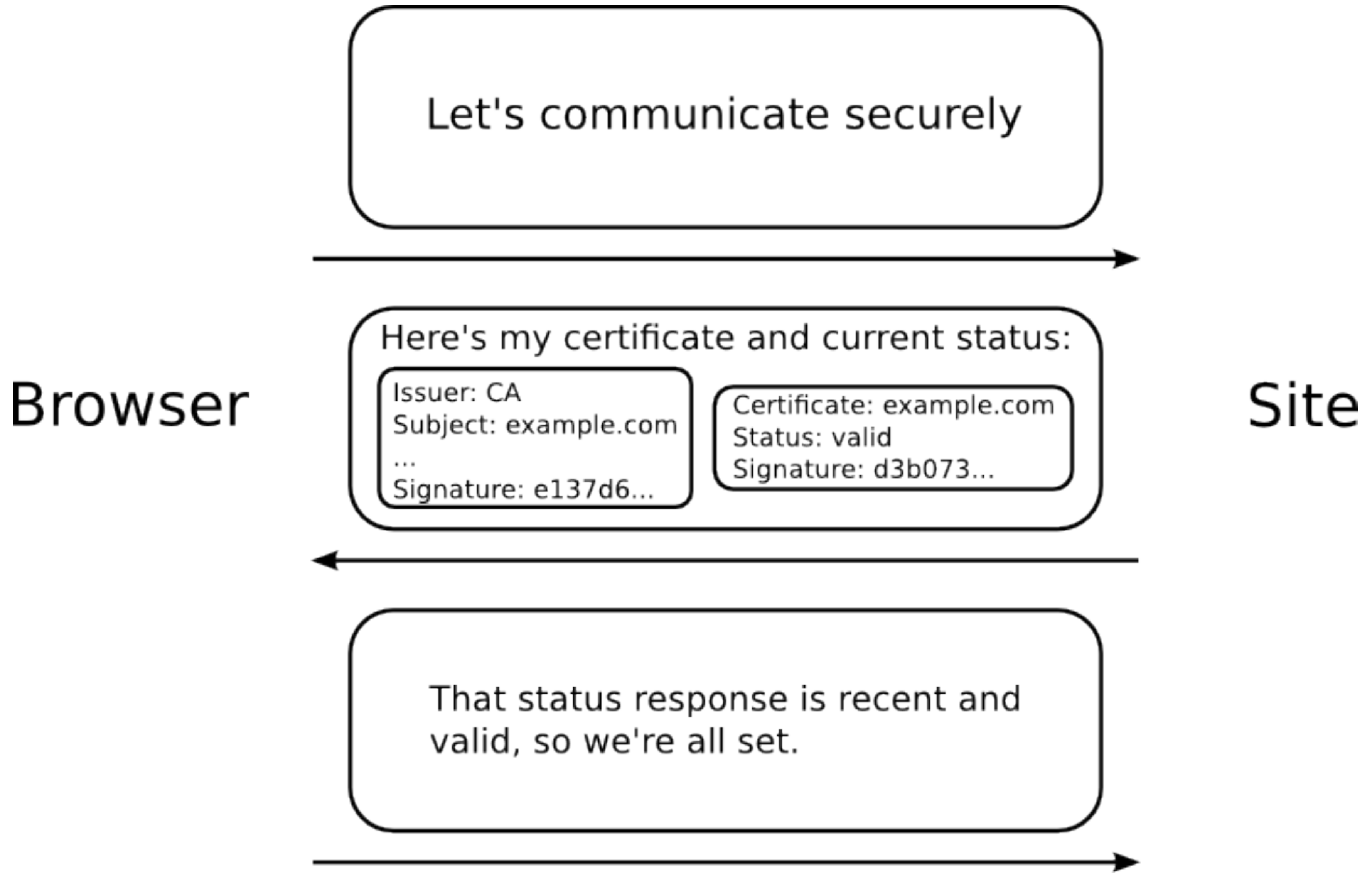
- Guarantee connection between public key and end entity
  - Man-in-the-Middle no longer works undetected
    - (If you verify the identity in the certificate against peer)
  - Guarantee authentication and non-repudiation
    - (If a CA doesn't make a mistake)
  - Privacy/confidentiality not an issue here
    - Only concerned with linking key to owner
- Distribute responsibility
  - Hierarchical structure
    - (Doesn’t exist in practice-- no good way to restrict delegation)
-

PKI and Revocation

- Certificate may be revoked before expiration
  - Lost private key
  - Compromised
  - Owner no longer authorized
- Revocation is hard …
  - The “anti-matter” problem
  - Verifiers need to check revocation state
    - Loses the advantage of off-line verification
  - Revocation state must be authenticated

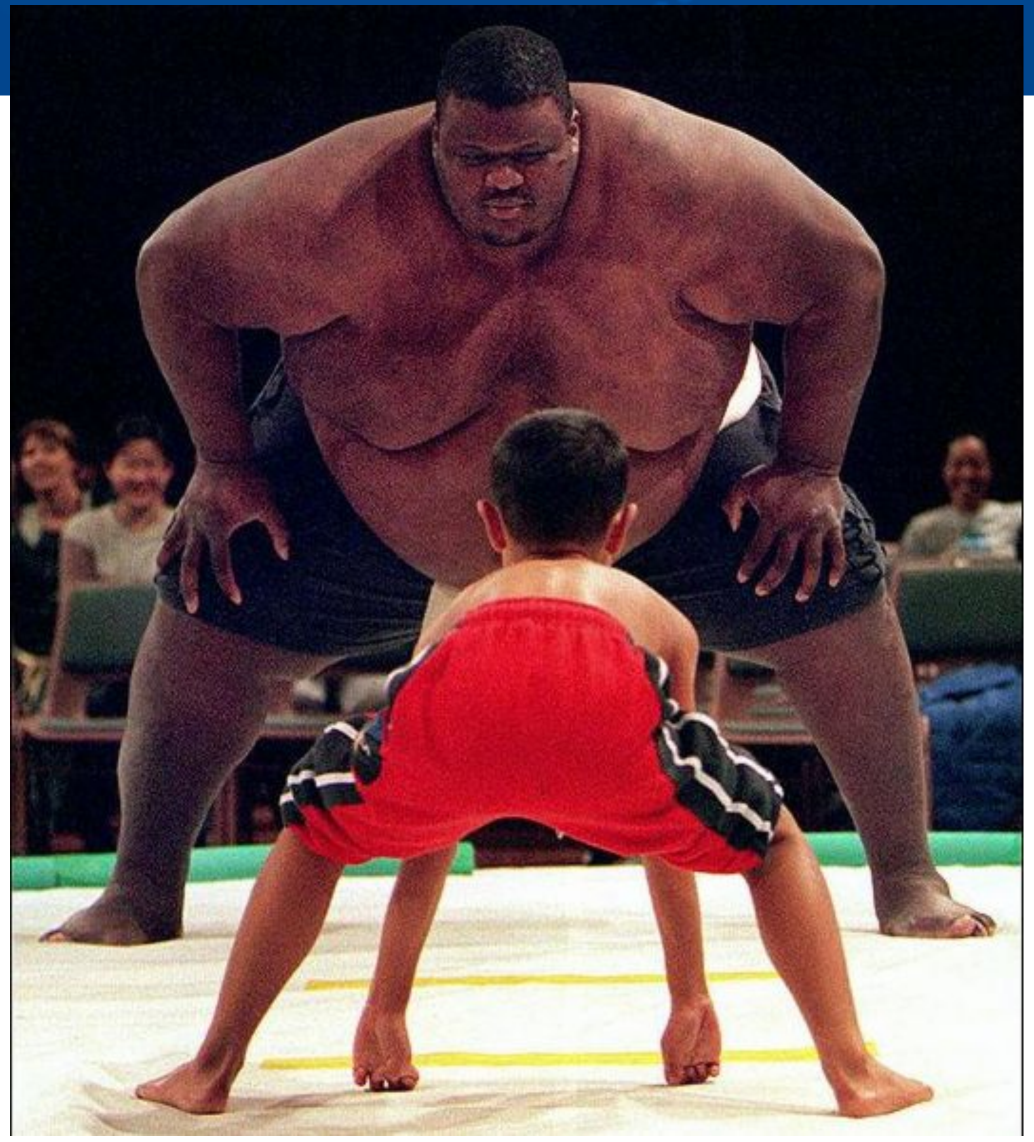
Revocation Mechanisms

- Certificate revocation lists (CRL)
  - Periodically issued
  - Delta CRLs when CRLs get too large
- Online certificate revocation server
  - Answers revoked = yes/no for a particular certificate
    - Implemented by OCSP protocol
  - Disadvantages?
  - OCSP-stapling



PKI Challenges

PennState



- Must trust a CA
  - Which one?
  - What is it trusted to do?
- Key storage
  - Who can access my key?
  - Similar problem for Kerberos, SSH, etc.
- Certificate bindings must be correct
  - Which John Smith is this?
  - Who authorizes attributes in a certificate?
  - How long are these value valid?
  - What process is used to verify the key holder?

Pretty Good Privacy

- Alternative infrastructure for public key
  - Peer-to-Peer approach
  - E.g., for email
- Key management is manual
  - Public key exchange between peers
  - Add public key to personal 'keyring'
  - Can authenticate messages from these parties
- Used mainly by computer security types
  - Johnny can't encrypt
  - GNU Privacy Guard

PKI (Circa 2009)