CS 114: Network Security

Lecture 6 - Key Agreement and PKI

Prof. Daniel Votipka
Spring 2023

(some slides courtesy of Prof. Micah Sherr)

Administrivia

- Exam 1 on Feb 16th in class.
- Homework 1, part 2 due Feb. 28th at 11:59pm
- There's a written component this time!
- pcap file has been uploaded
- HW1p1 reference solution posted soon

Public Key Cryptography

- Each key pair consists of a public and private component: k^+ (public key), k^- (private key)

$$D_{k-}(E_{k+}(m)) = m$$

- Public keys are distributed (typically) through public key certificates
  - Anyone can communicate secretly with you ***if they have your certificate***

RSA Key Generation

- Choose distinct primes p and q
- Compute n = pq
- Compute $\Phi(n) = \Phi(pq) = (p-1)(q-1)$
- Randomly choose $1<e<\Phi(pq)$ such that e and $\Phi(pq)$ are coprime. e is the **public key exponent**
- Compute $d=e^{-1} \bmod(\Phi(pq))$. d is the **private key exponent**

$$E_{k+}(M) : C = M^e \bmod n$$

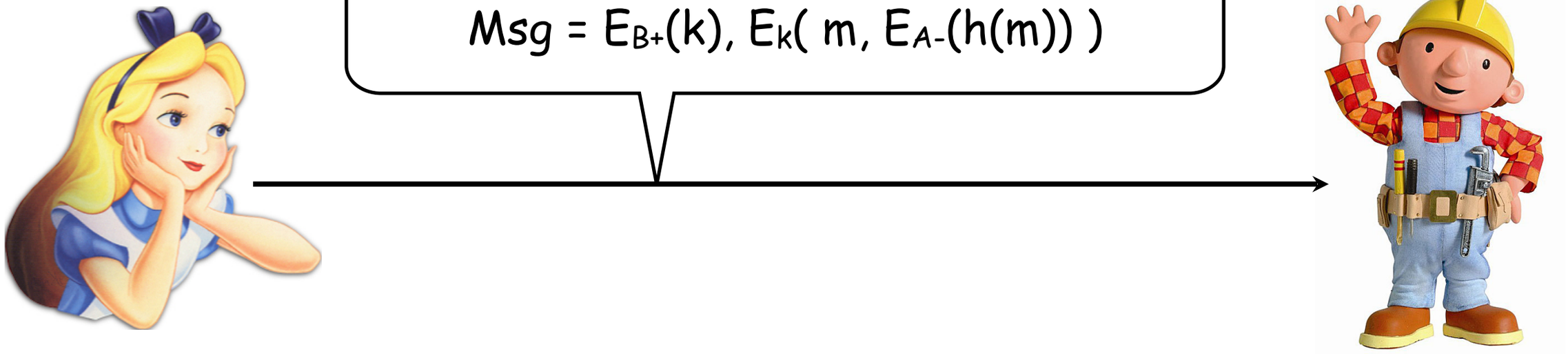
$$D_{k-}(C) : M = C^d \bmod n$$

Properties of a Digital Signature

- **No forgery possible**
- **No alteration/Integrity**
- **Non-repudiation**

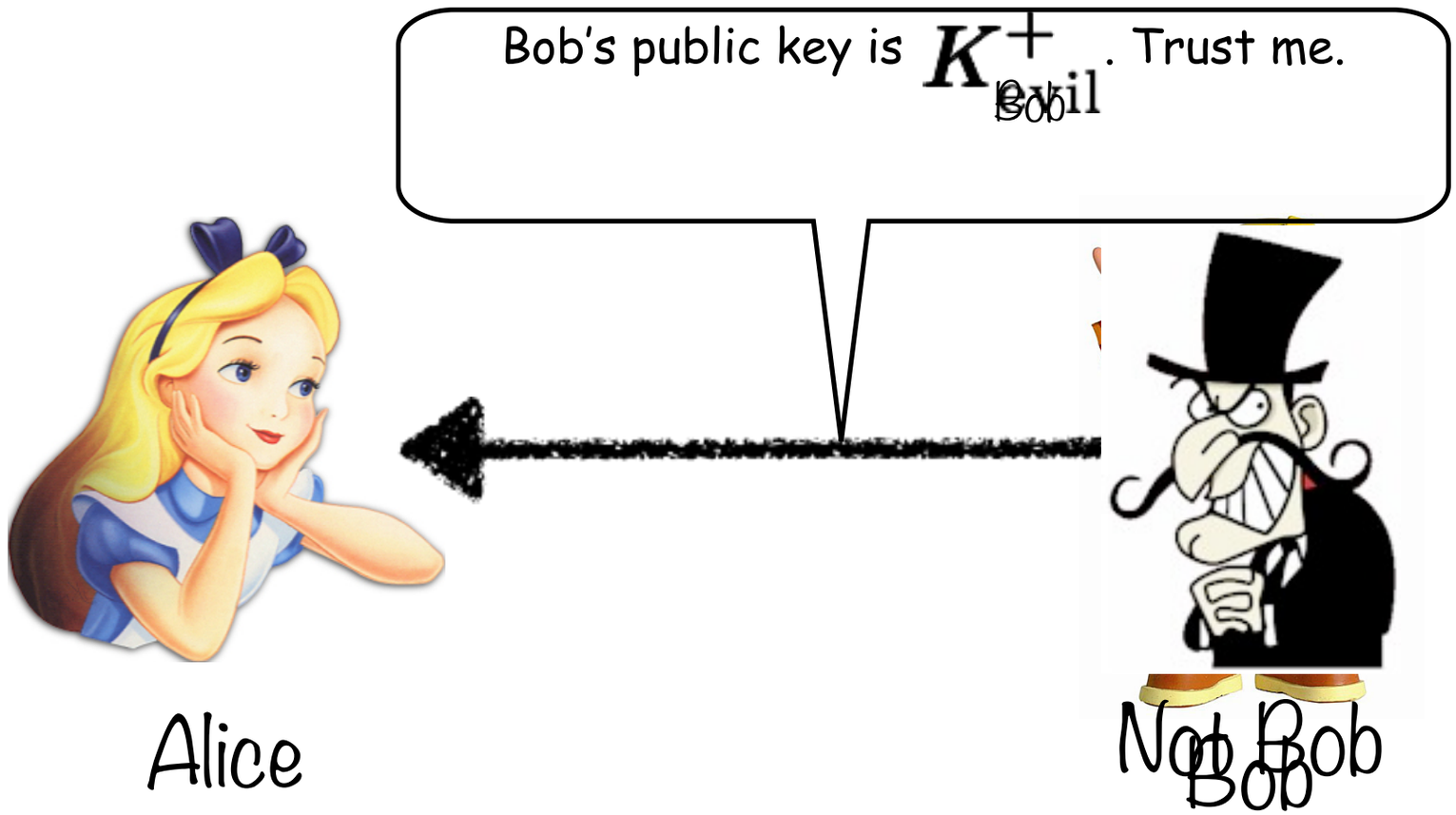
Hybrid Cryptosystems

Define m = "CS114 is awesome"



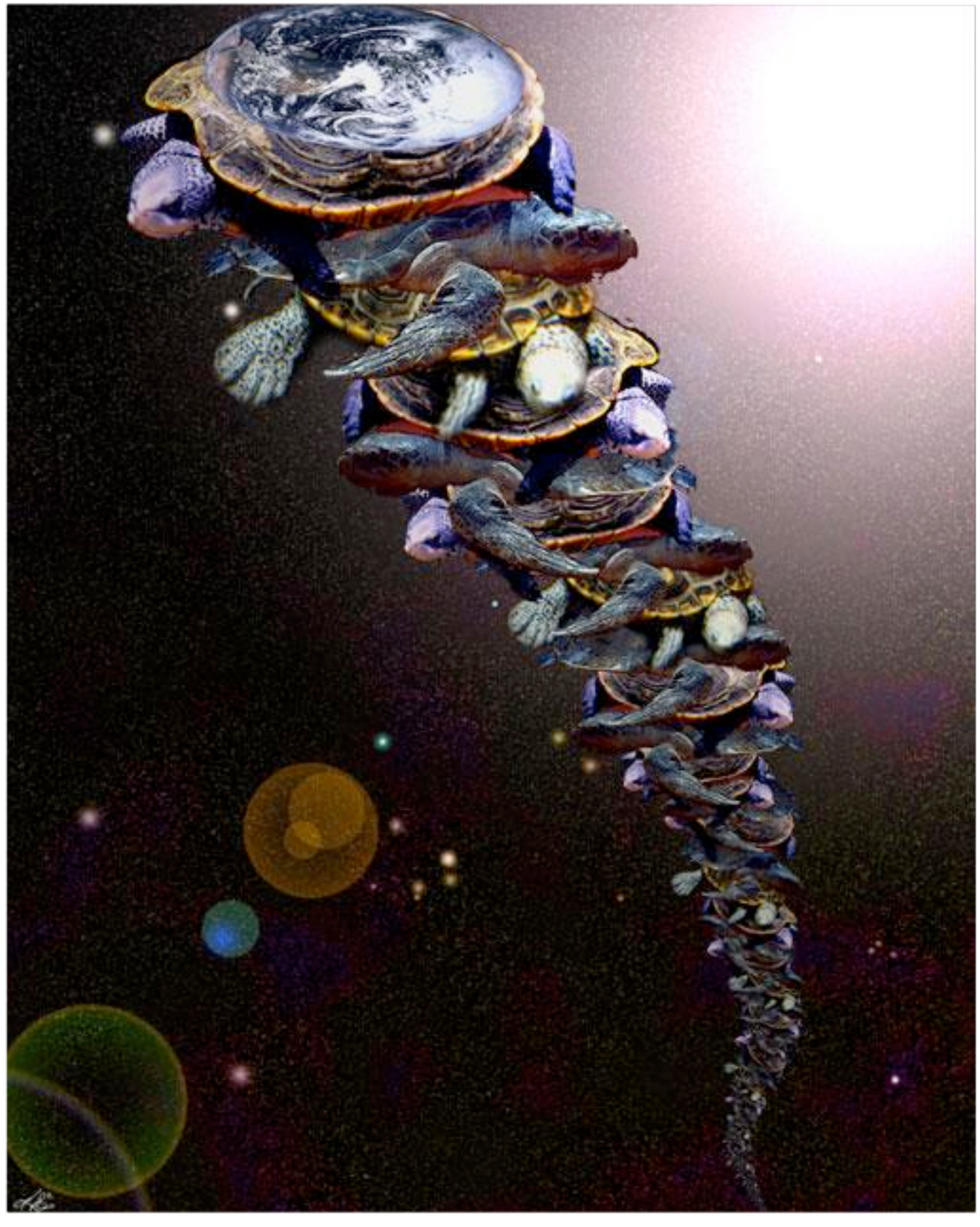
(A^+, A^-) is Alice's long-term public-private key pair.
(B^+, B^-) is Bob's long-term public-private key pair.
k is the session key; sometimes called the **ephemeral key**.

How do we *verify* we're using the correct public key?



Short answer:
We can't.

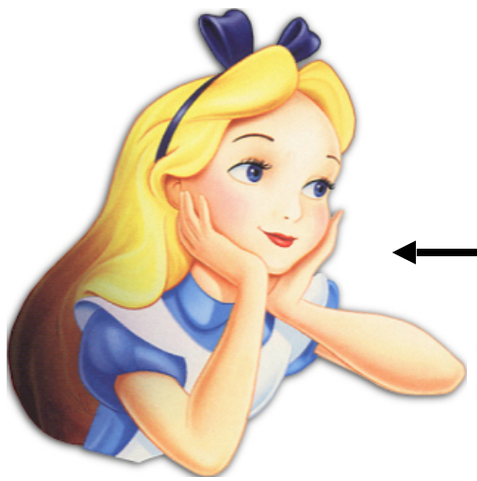
It's turtles all the way down.



Key Agreement, Part I: Sharing a Private Key

Encryption and Message Authenticity

Src = Alice, Dest = Bob
Msg = E_{k1}{{"network security is fun",
MAC_{k2}("network security is fun")}}



Alice

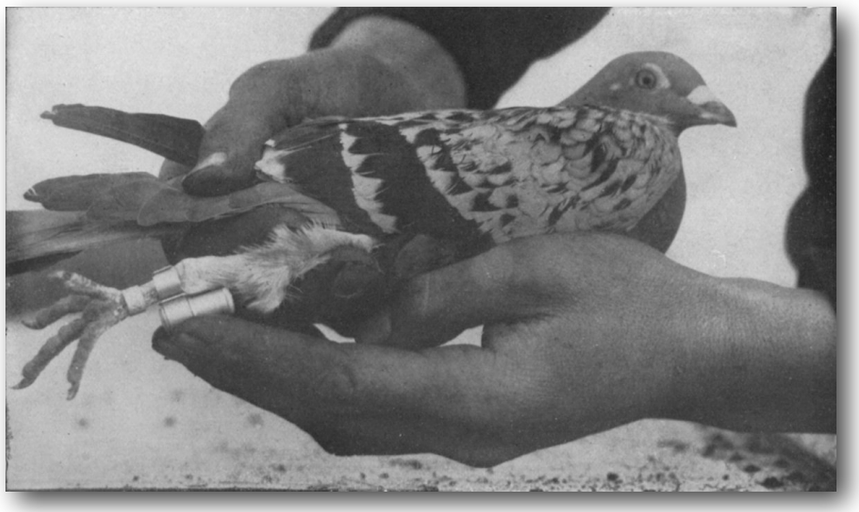
Bob

Key Distribution

- Suppose Alice has an channel for communicating with Bob.
- Alice and Bob wish to use this channel to established a shared secret.
- However, Eve is able to learn everything sent over the channel.
- If Alice and Bob have no other channel to use, can they establish a shared secret that Eve does not know?

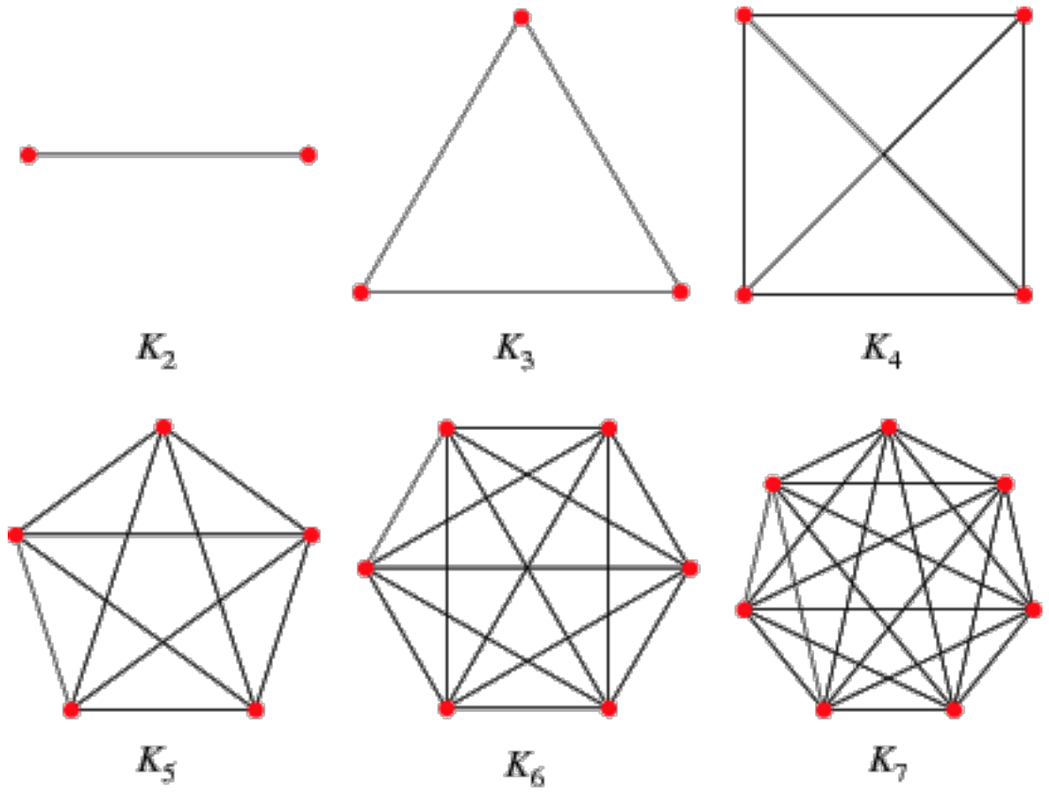
Key Distribution

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



Key Distribution

- Pairwise key distribution requires $\binom{N}{2}$ plastic cups



Key Distribution and Key Agreement

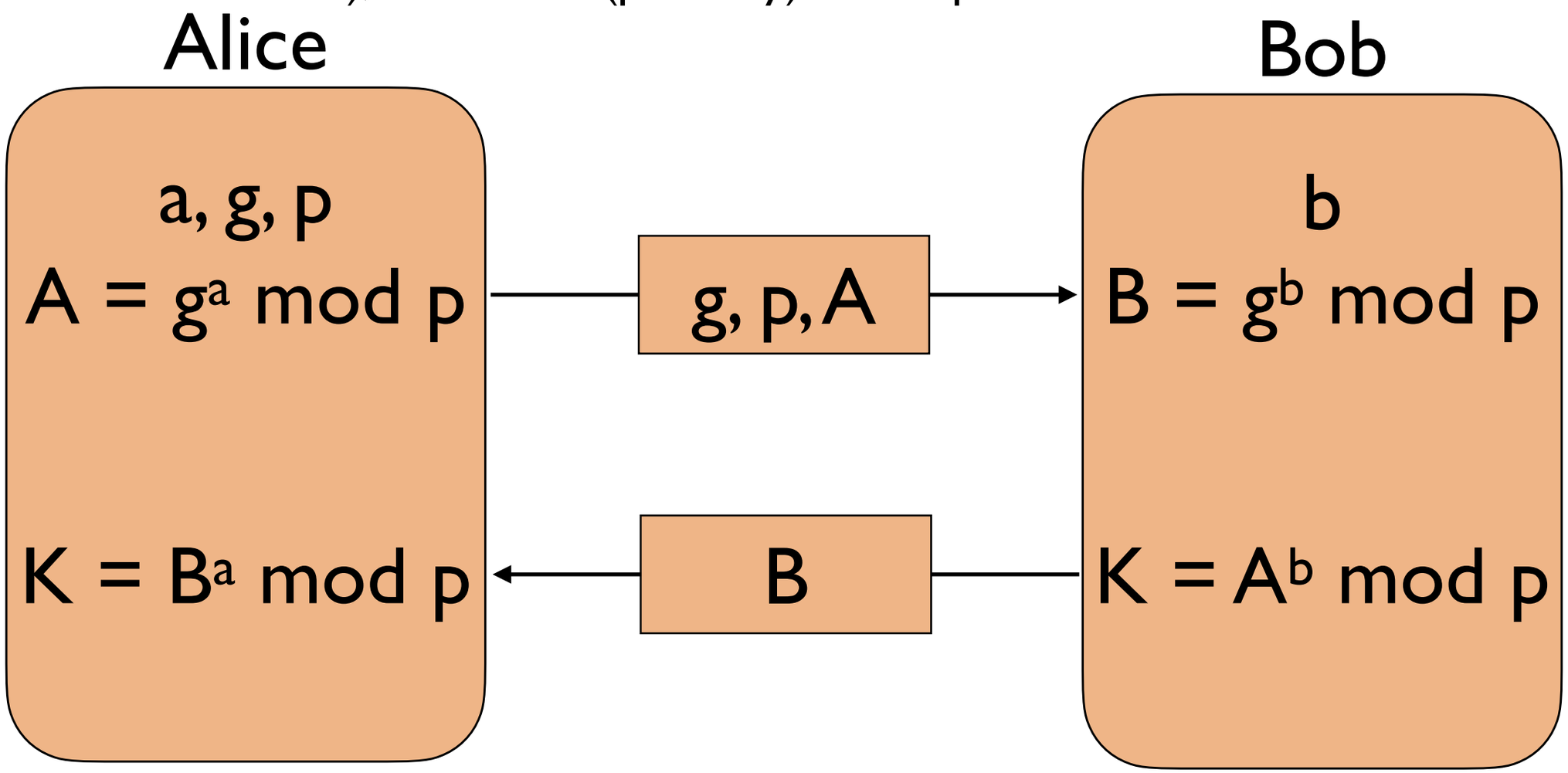
- **Key Distribution** is the process where we assign and transfer keys to a participant
- **Key Agreement** is the process whereby two or more parties negotiate a key

Diffie-Hellman (DH) Key Agreement

- The DH paper started the modern age of cryptography, and indirectly the security community
  - Negotiate a secret over an insecure media
  - E.g., "in the clear" (seems impossible)
    - Idea: participants exchange intractable puzzles that can be solved easily with additional information
- Mathematics are very deep
  - Use the hardness of computing discrete logarithms in finite field to make secure

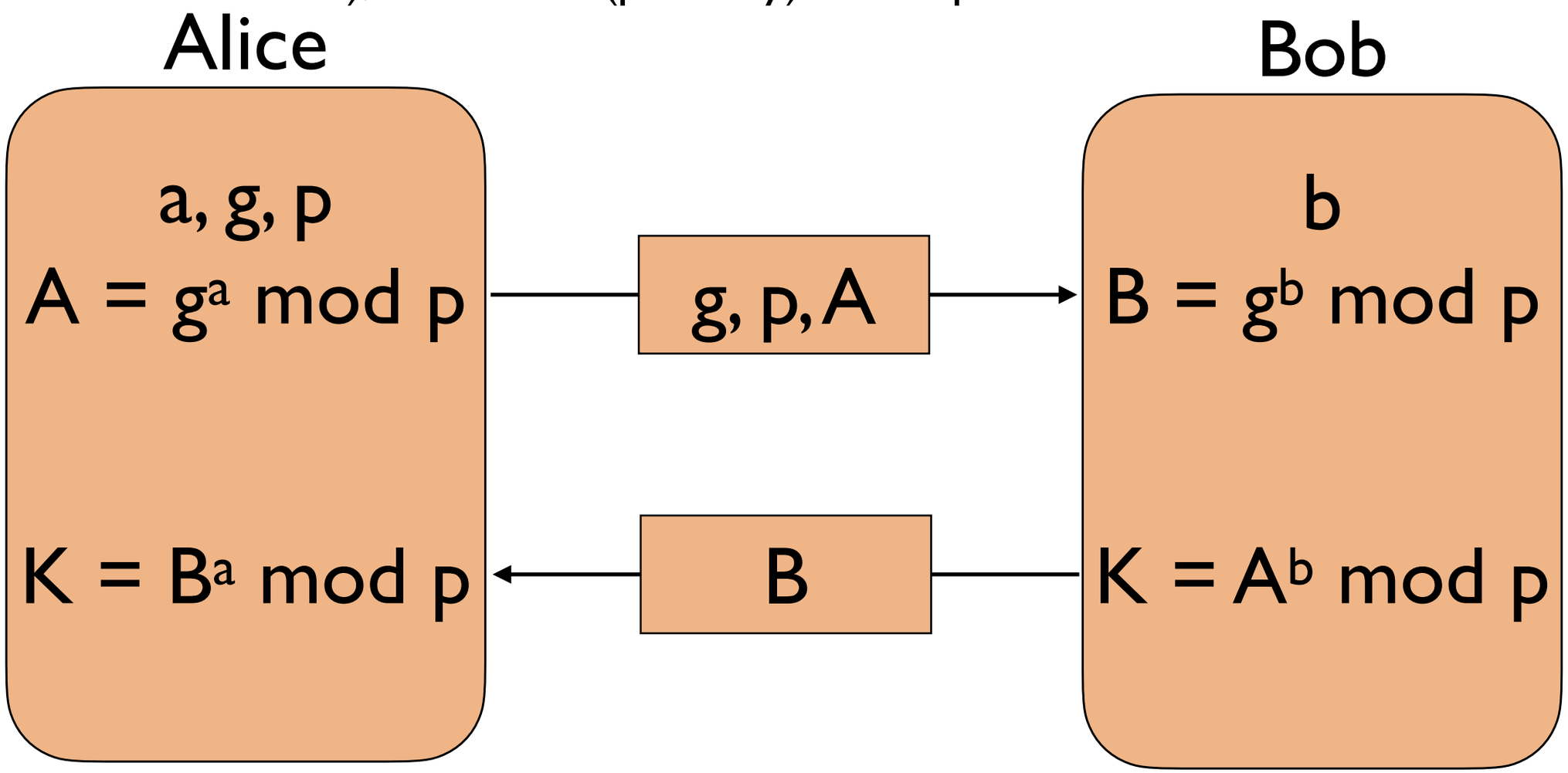
Diffie-Hellman (DH) Key Agreement

- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



Diffie-Hellman (DH) Key Agreement

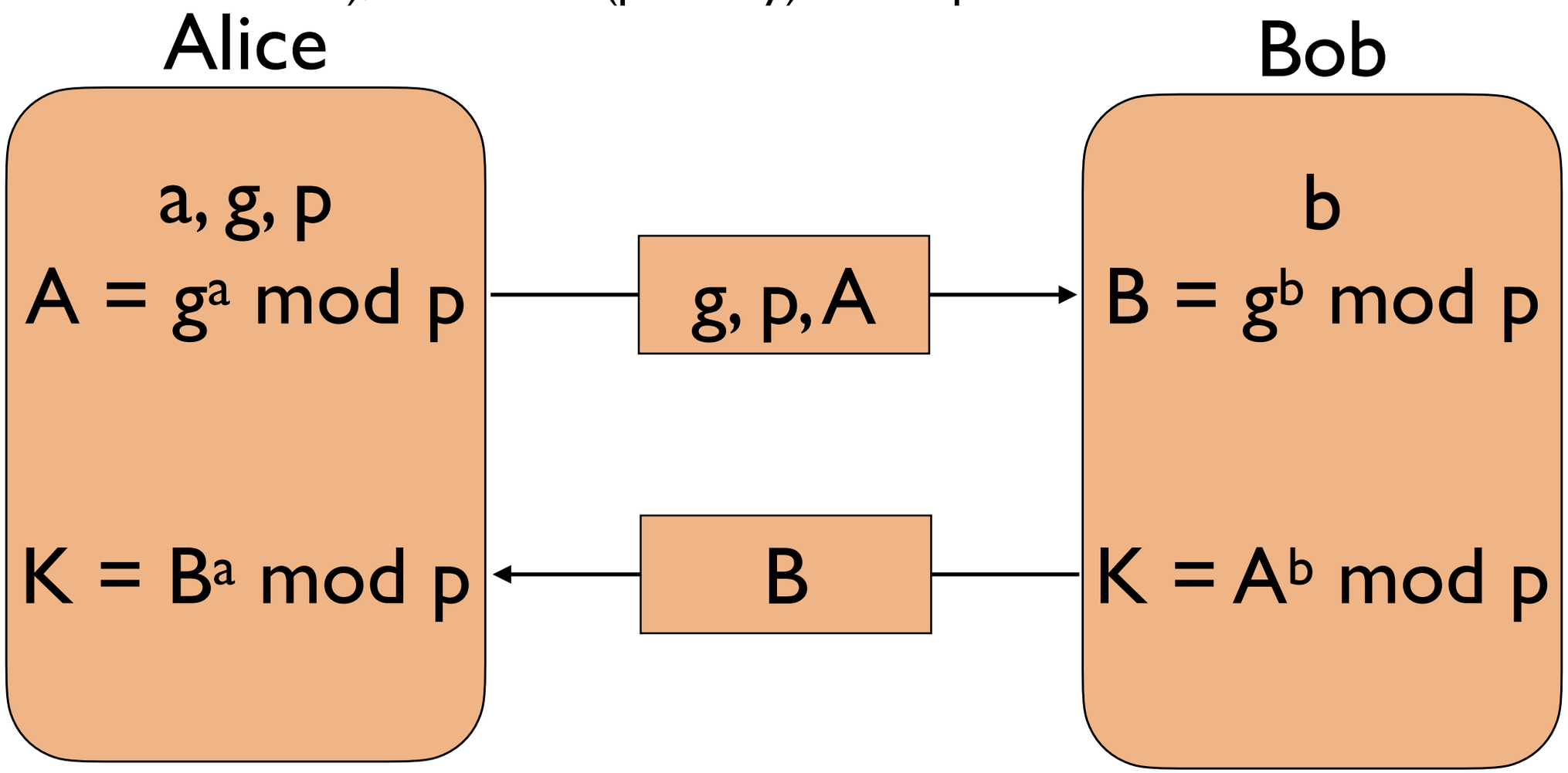
- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



$K = B^a \bmod p$

Diffie-Hellman (DH) Key Agreement

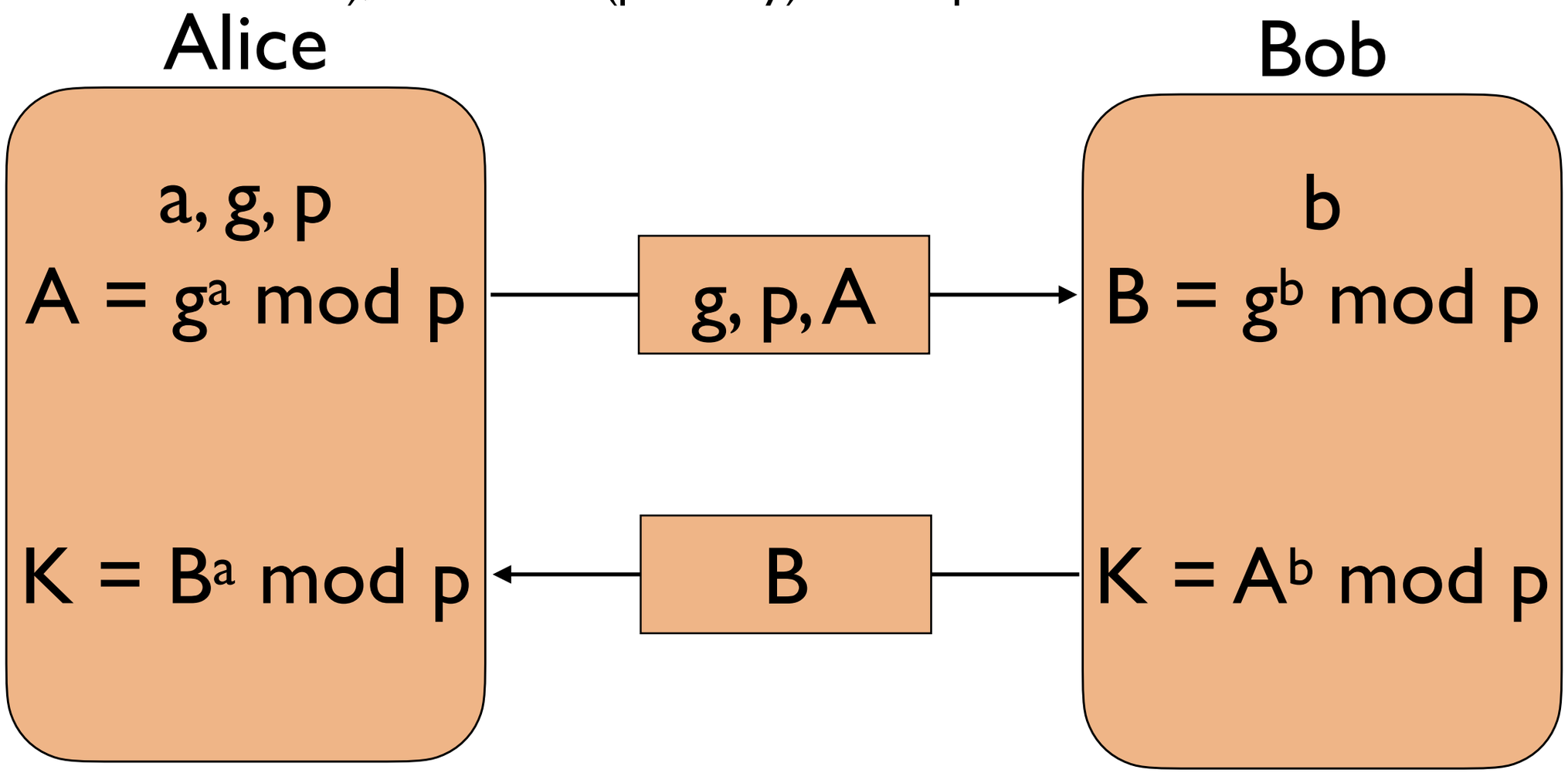
- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



$K = B^a \bmod p = (g^b \bmod p)^a \bmod p$

Diffie-Hellman (DH) Key Agreement

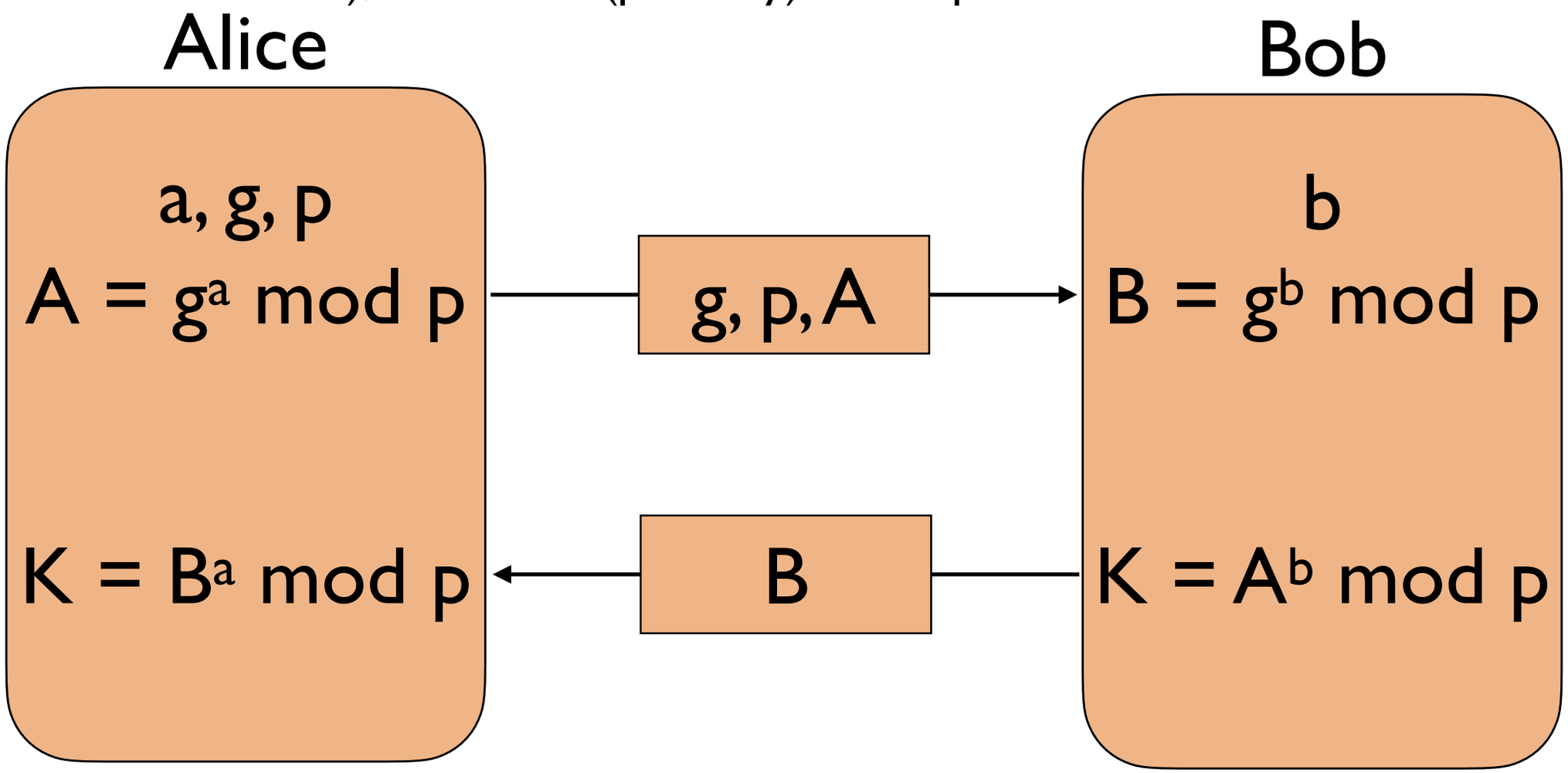
- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



$K = B^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ab} \bmod p$

Diffie-Hellman (DH) Key Agreement

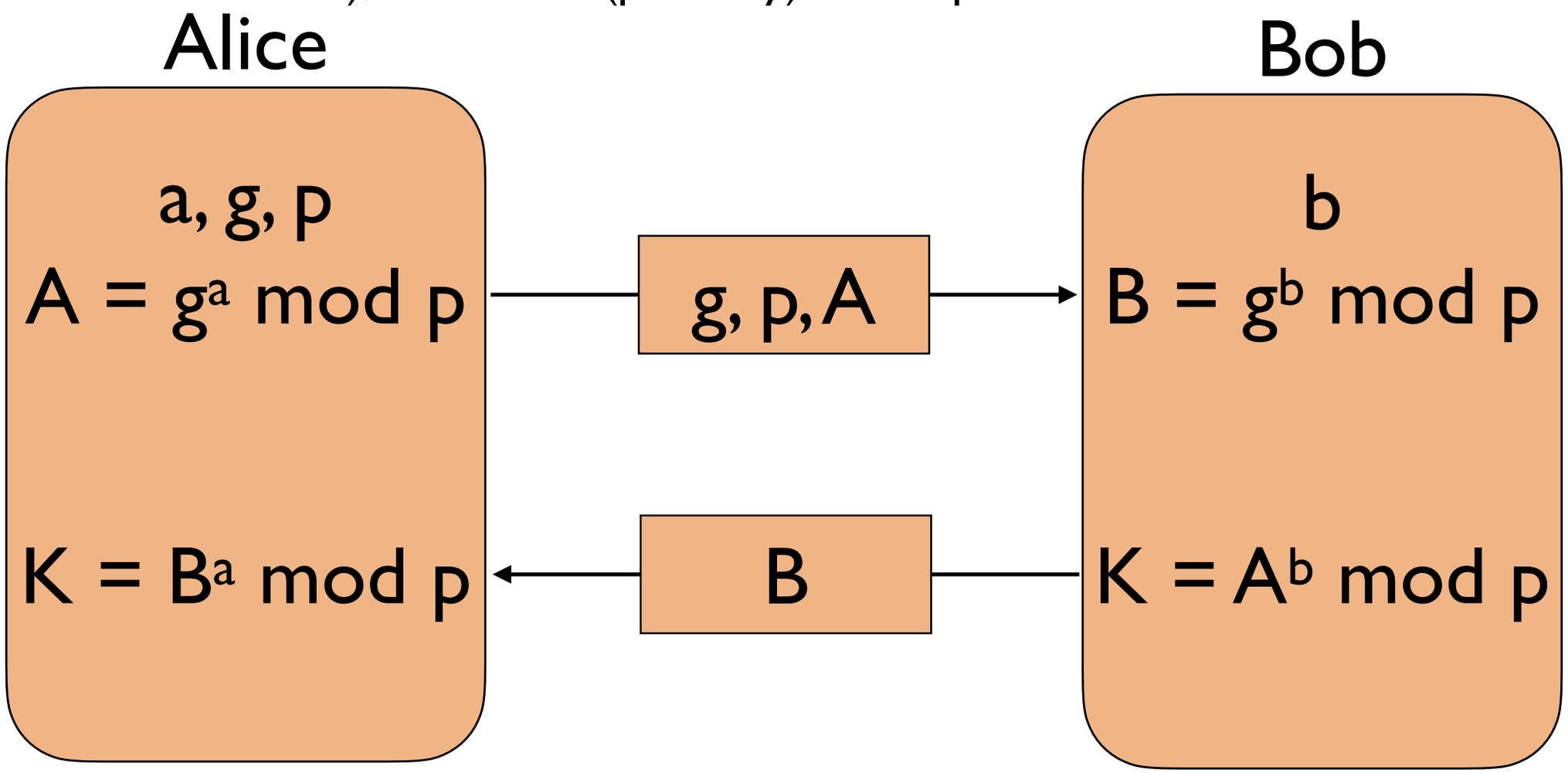
- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



$K = B^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ab} \bmod p = (g^a \bmod p)^b \bmod p$

Diffie-Hellman (DH) Key Agreement

- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



$K = B^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ab} \bmod p = (g^a \bmod p)^b \bmod p = A^b \bmod p$

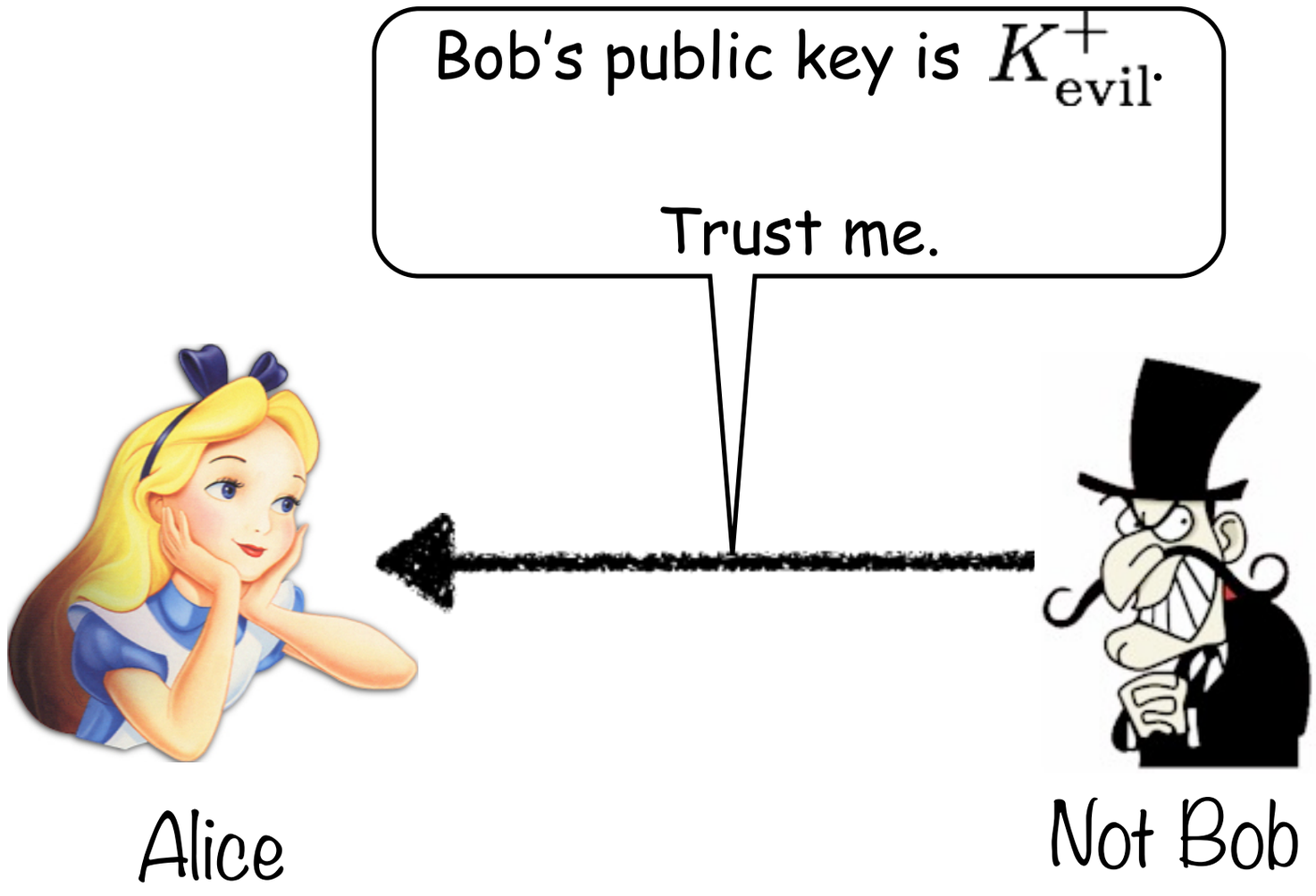
Attacks on Diffie-Hellman

- Subject to **Man-in-the-Middle** (MitM) attack
  - 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
- Fix: Authenticated DH exchange
  - The parties sign the exchanges (more or less)
  - Requires pre-shared knowledge or trusted third party

Key Agreement, Part II: Public Key Distribution

How do we *verify* we're using the correct public key?

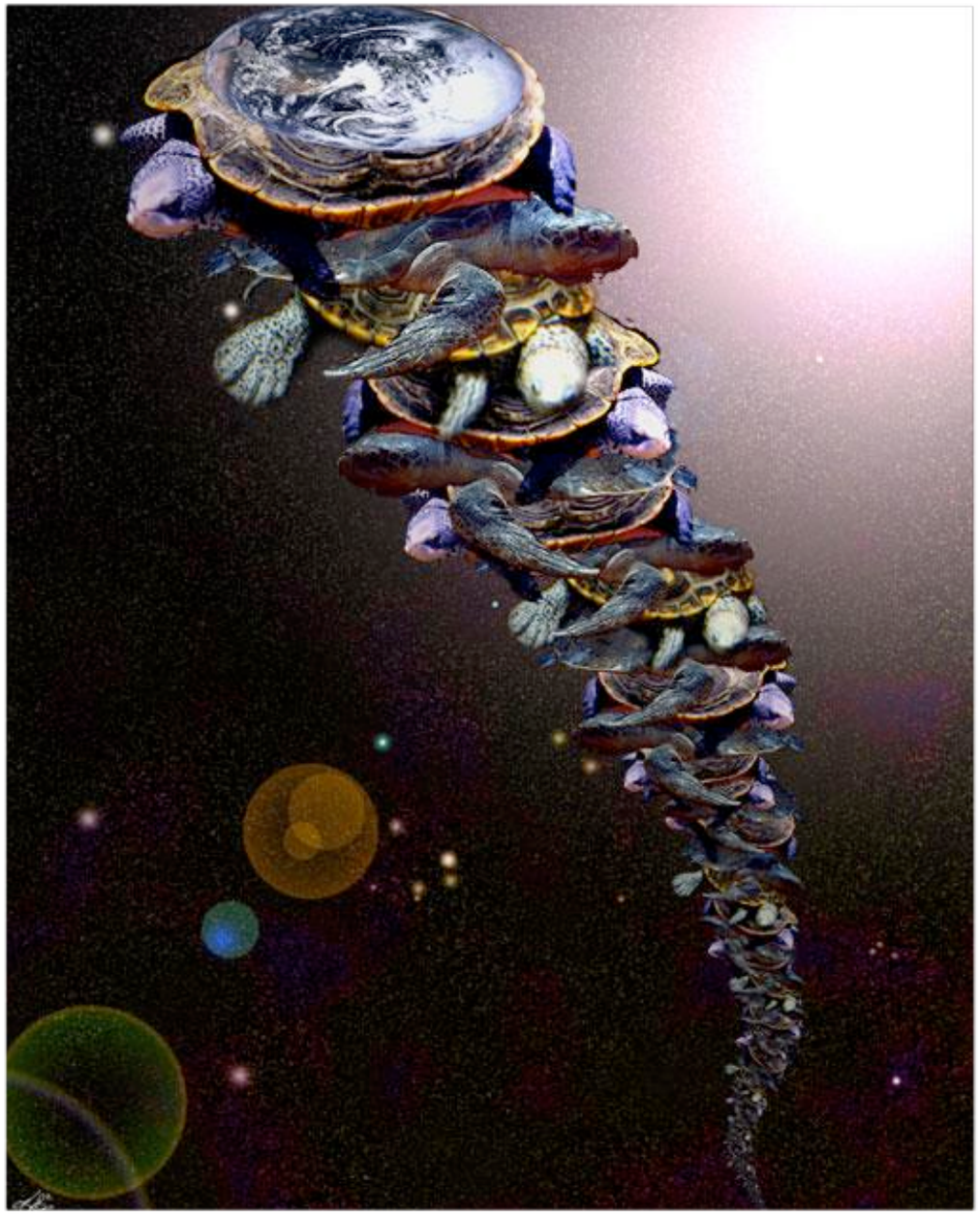


Why not just use a database?

- Every user has his/her own public key and private key.
- Public keys are all published in a database.
- Alice gets Bob's public key from the database
- Alice encrypts the message and sends it to Bob using Bob's public key.
- Bob decrypts it using his private key.
- **What's the problem with this approach?**

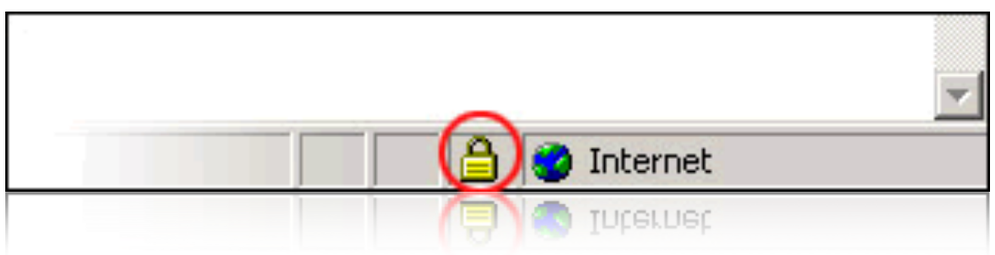
Solving the Turtles Problem

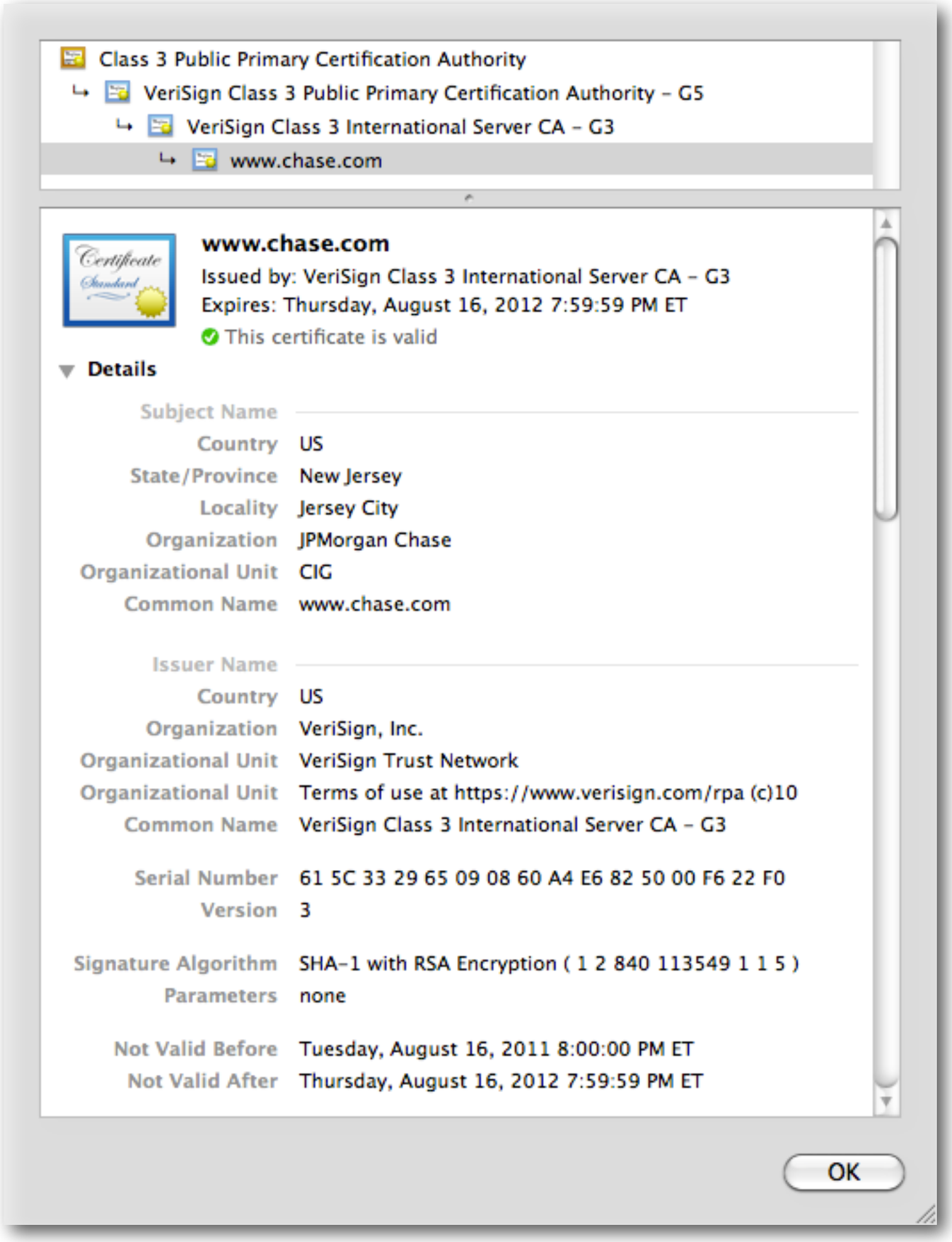
- We need a **trust anchor**
  - there must be someone with authority
  - requires *a priori* trust
- Solution: form a trust hierarchy
  - "I believe **X** because..."
  - "**Y** vouches for **X** and..."
  - "**Z** vouches for **Y** and..."
  - "I <u>implicitly</u> trust **Z**."



Browser

Certificate



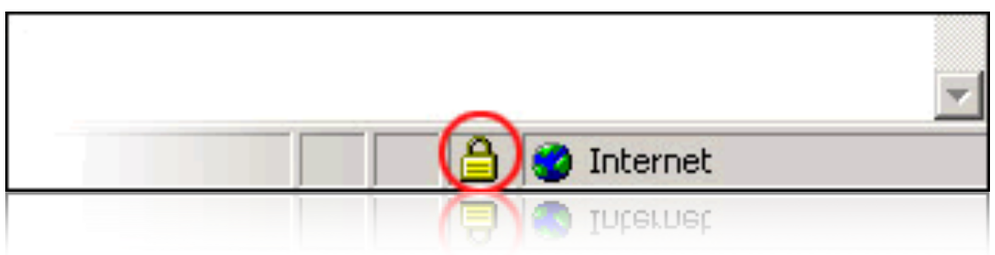


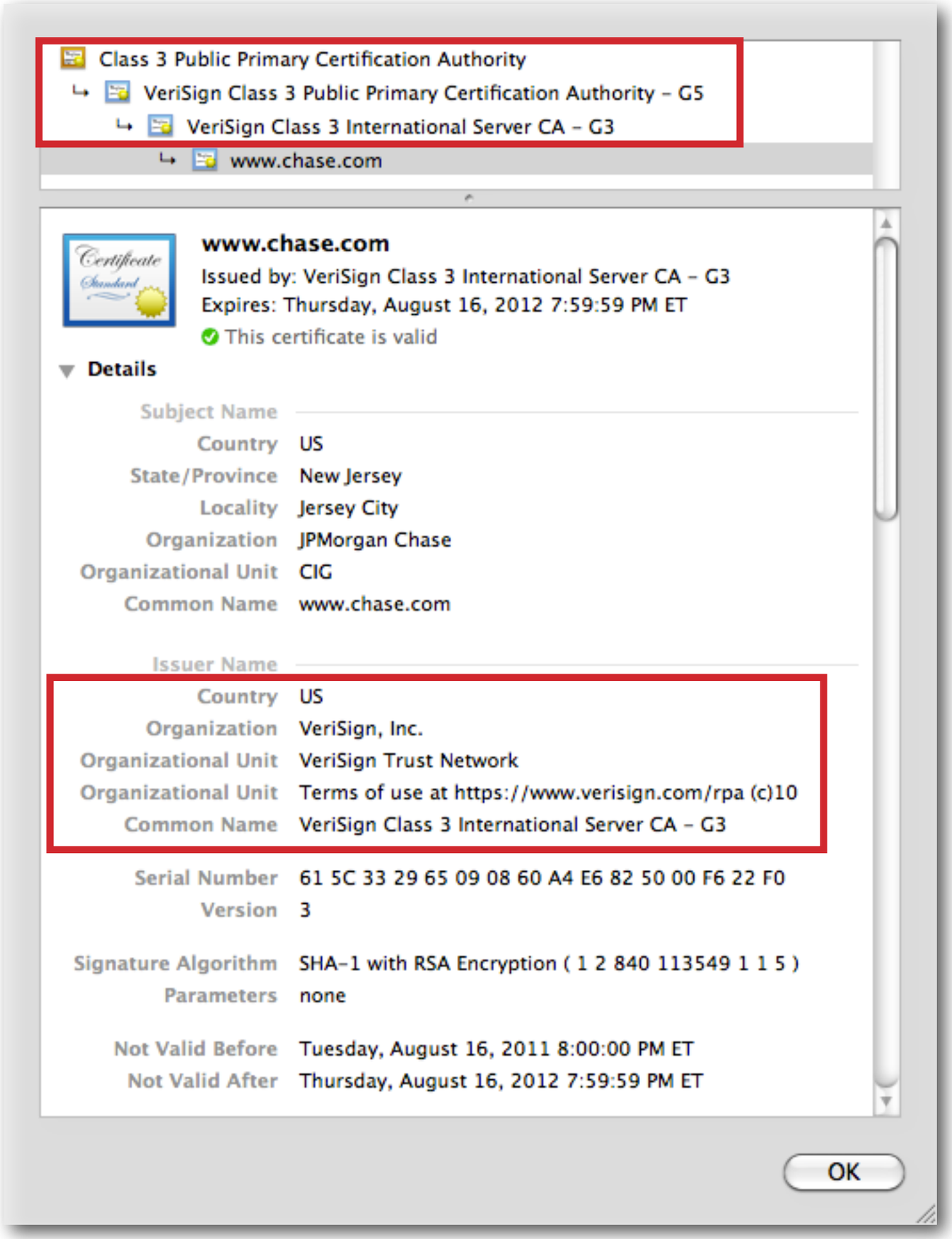
What's a certificate?

- A certificate …
  - … makes an association between an identity 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)
- People trust CA (e.g., Verisign) to vet identity

Browser

Certificate





Why do I trust the certificate?

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

Certificate Manager

Your Certificates | People | Servers | Authorities | Others

You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
▼ The Go Daddy Group, Inc.	
Go Daddy Secure Certification Authority	Software Security Device
Go Daddy Class 2 CA	Builtin Object Token
▼ The USERTRUST Network	
Network Solutions Certificate Authority	Software Security Device
Register.com CA SSL Services (OV)	Software Security Device
UTN-USERFirst-Hardware	Builtin Object Token
UTN - DATACorp SGC	Builtin Object Token
UTN-USERFirst-Network Applications	Builtin Object Token
UTN-USERFirst-Client Authentication and Email	Builtin Object Token
UTN-USERFirst-Object	Builtin Object Token
▼ Türkiye Bilimsel ve Teknolojik Araştırma Kurumu…	
TÜBİTAK UEKAE Kök Sertifika Hizmet Sağlayıcı…	Builtin Object Token
▼ TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hiz…	
TÜRKTRUST Elektronik Sertifika Hizmet Sağlay…	Builtin Object Token
▼ University of Pennsylvania	
DSL CA Authority	Software Security Device
▼ Unizeto Sp. z o.o.	
Certum CA	Builtin Object Token
▼ ValiCert, Inc.	
RSA Public Root CA v1	Software Security Device
http://www.valicert.com/	Builtin Object Token
http://www.valicert.com/	Builtin Object Token

View… | Edit… | Import… | Export… | Delete…

OK

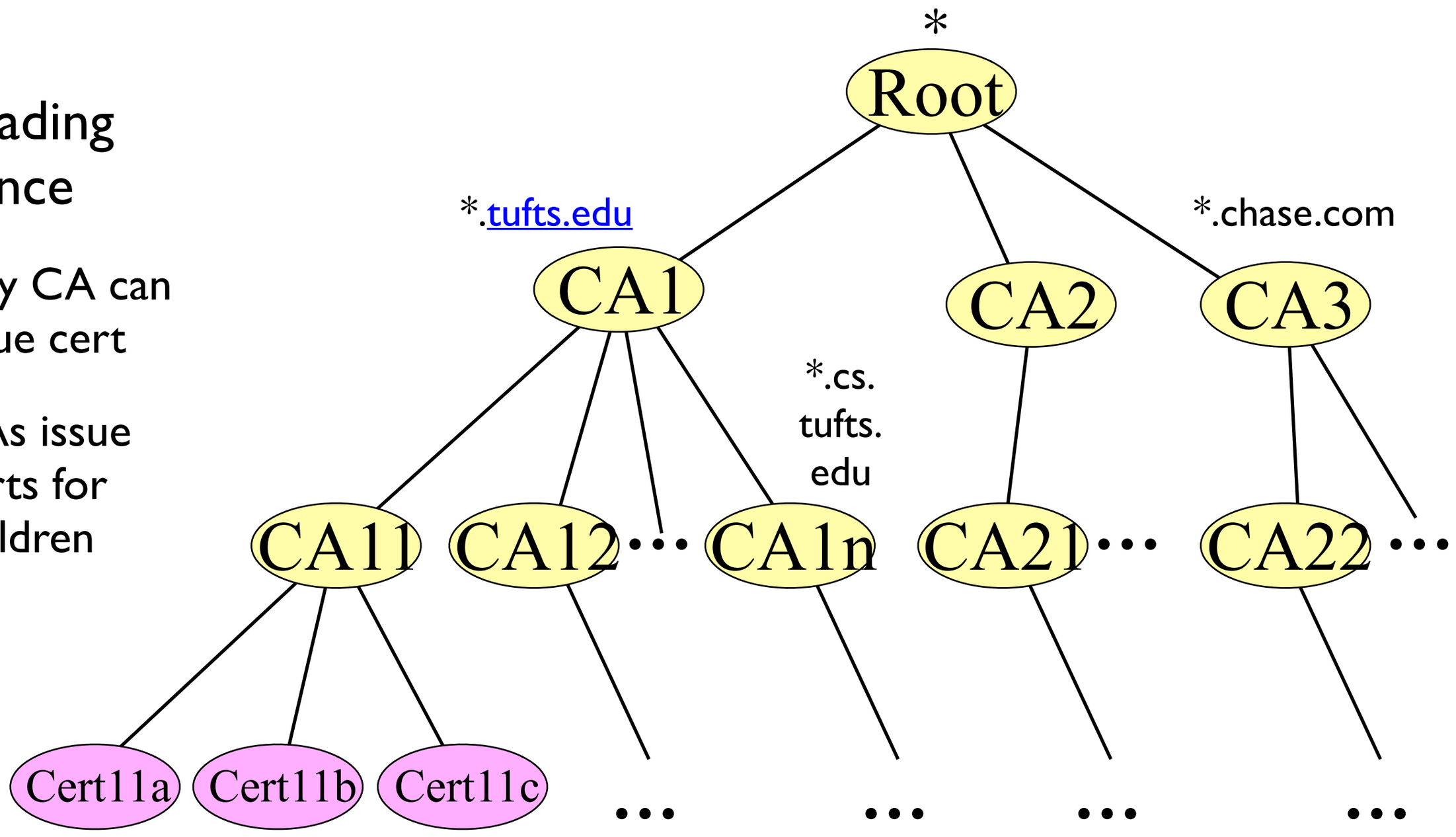
Public Key Infrastructure

Public Key Infrastructure

- Hierarchy of keys used to authenticate certificates
- Requires a **root of trust** (i.e., a **trust anchor**)

What is a PKI?

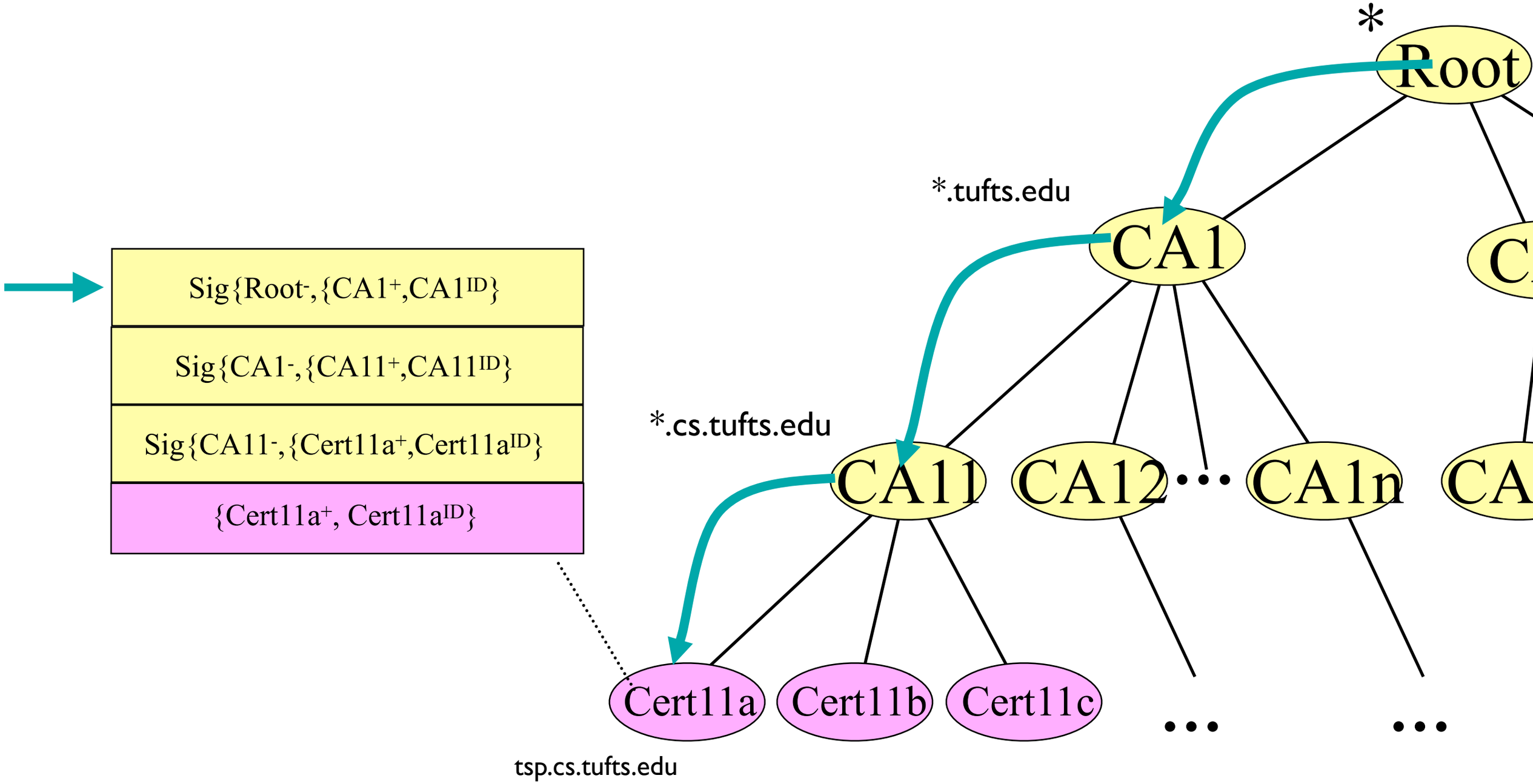
- Rooted tree of CAs
- Cascading issuance
- Any CA can issue cert
- CAs issue certs for children



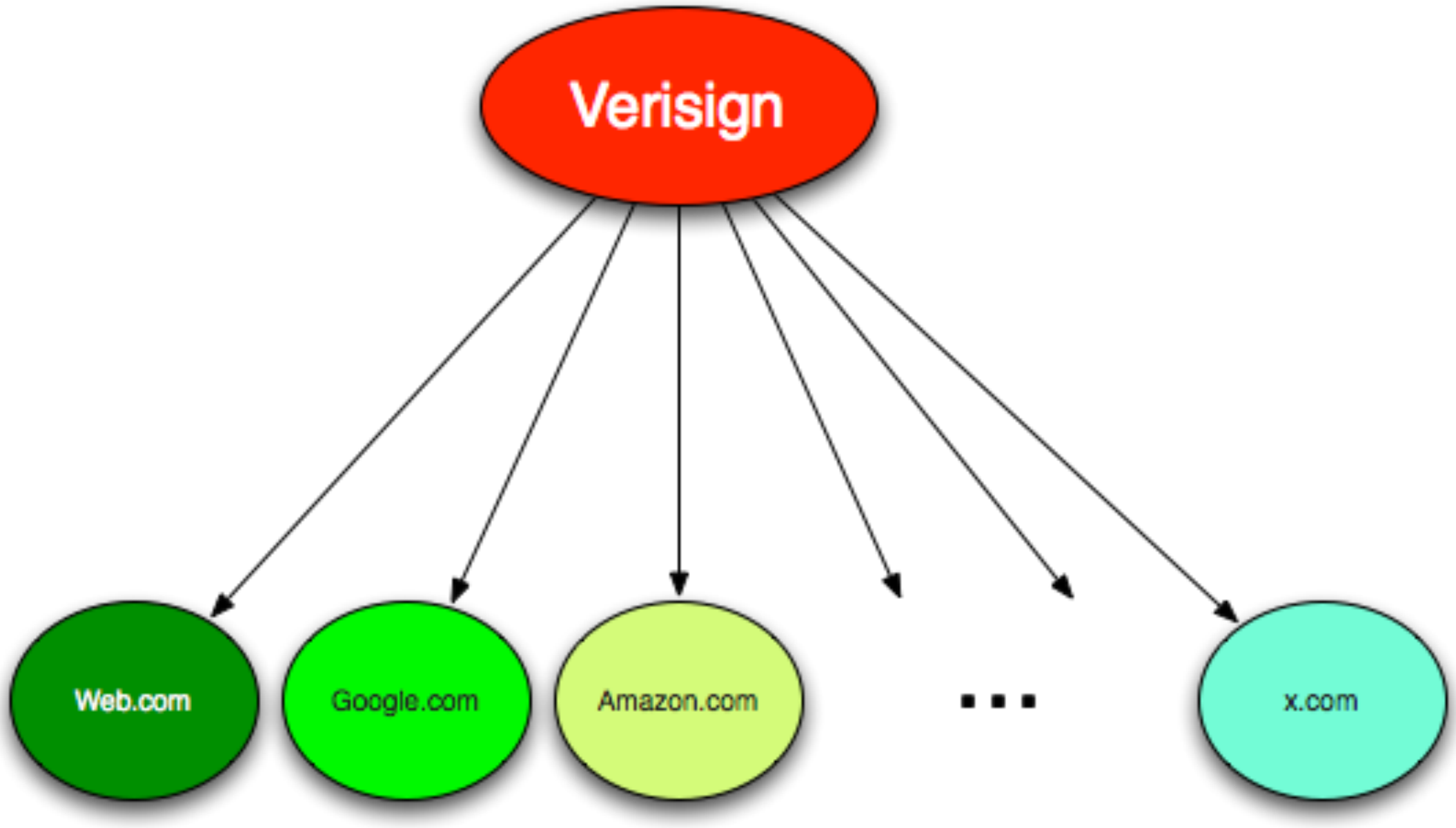
Obtaining a Certificate

1. Alice has some identity document A^{ID} and generates a keypair (A^-, A^+)
2. $A \rightarrow CA : \{A^+, A^{ID}\}, Sig(A^-, \{A^+, A^{ID}\})$
   - CA verifies signature -- proves Alice has A^-
   - CA may (and should!) also verify A^{ID} offline
3. CA signs $\{A^+, A^{ID}\}$ with its private key (CA^-)
   - CA attests to binding between A+ and A^{ID}
4. $CA \rightarrow A : \{A^+, A^{ID}\}, Sig(CA^-, \{A^+, A^{ID}\})$
   - 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" $\{A^+, A^{ID}\}$
   - note that CA never learns A^-

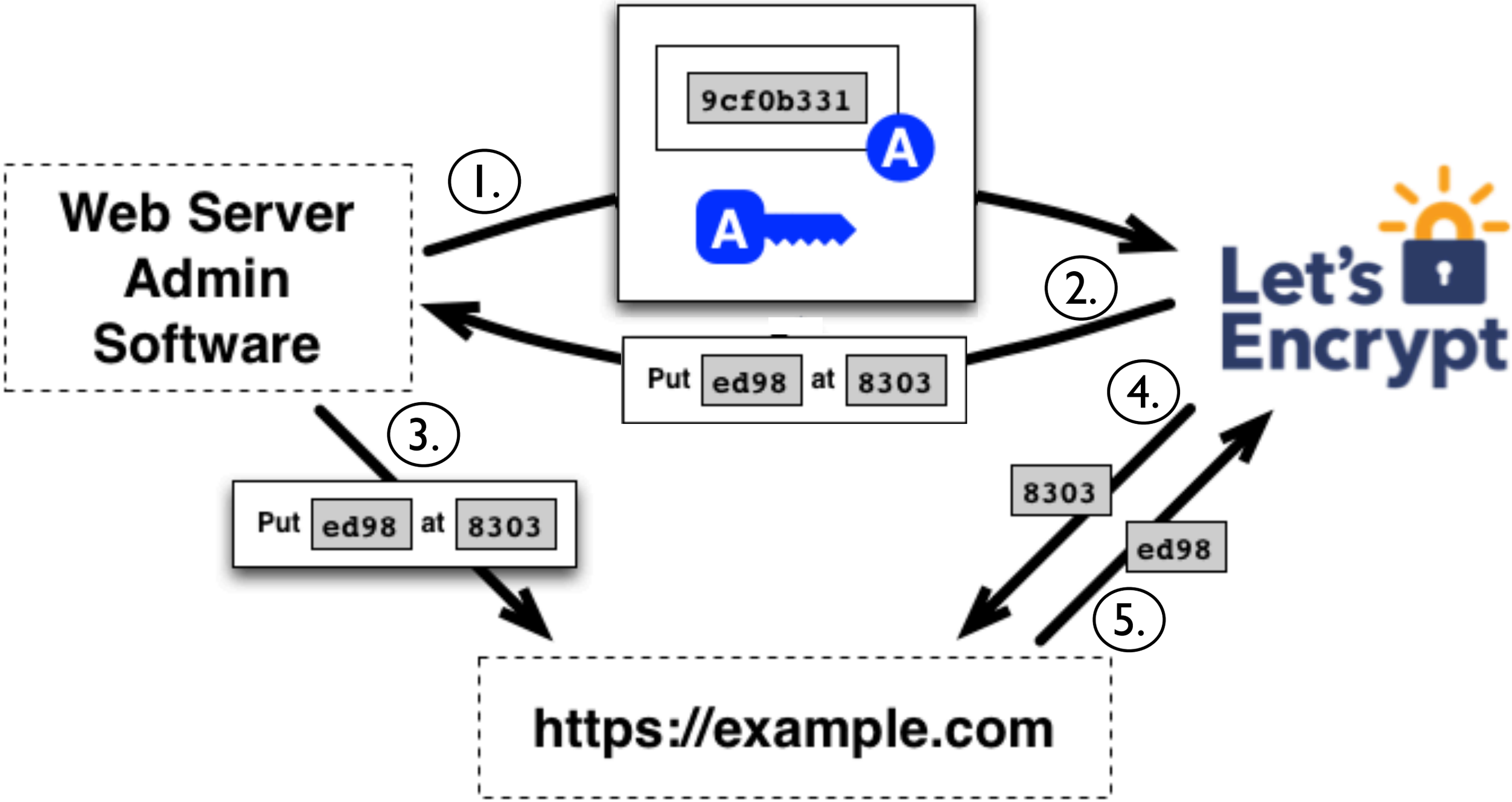
Certificate Validation



PKIs in Reality




Let's
Encrypt

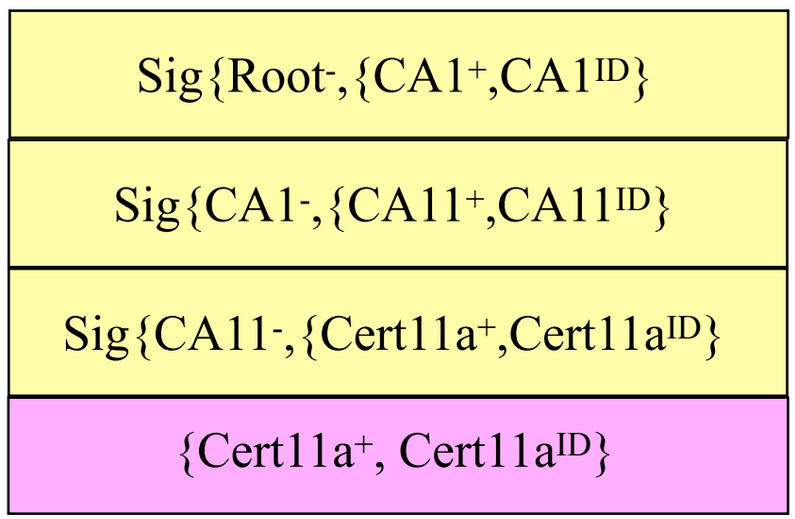

Web Server Admin Software
1.
9cf0b331
A
A
2.
Let's Encrypt
Put ed98 at 8303
3.
Put ed98 at 8303
4.
8303
ed98
5.
https://example.com

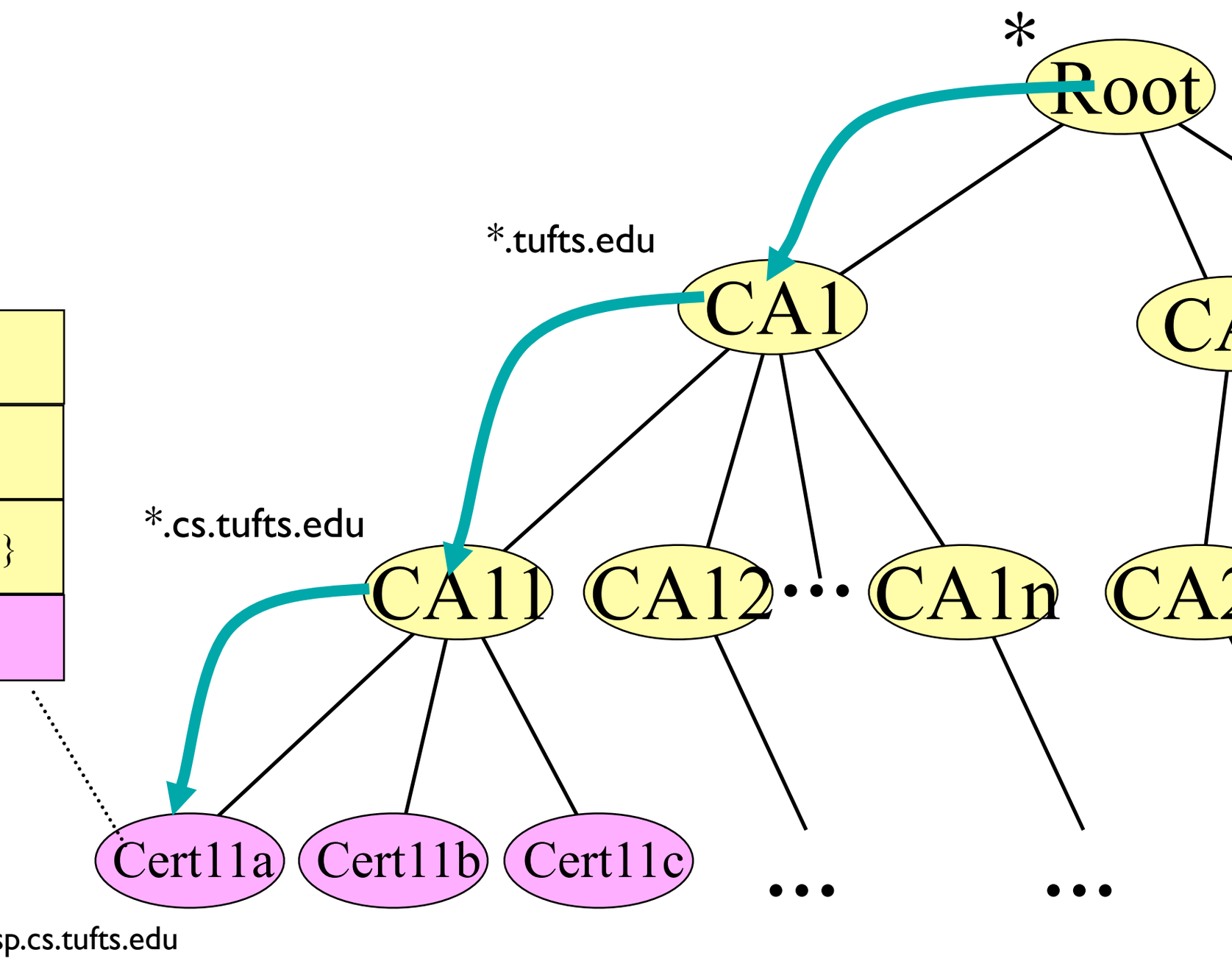
When PKI Goes Wrong!

PKI and Revocation

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

Certificate Validation





PKI and Revocation

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

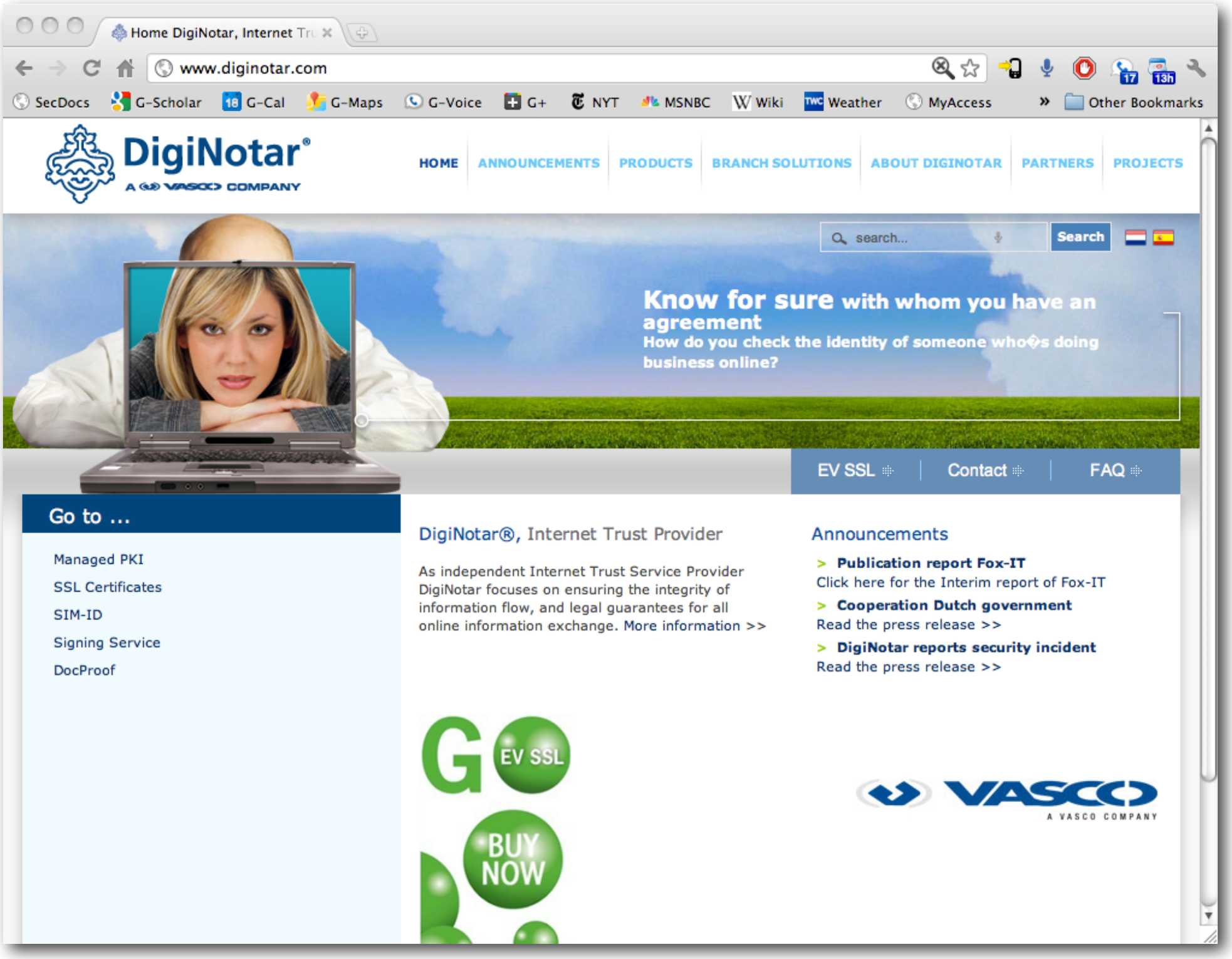
60% not revoked

20% 2yr+ TTL

"Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed", Zhang et. al., IMC '14

- Any CA may sign any certificate
- Browser weighs all root CAs equally
- *Q: Is this problematic?*

The DigiNotar Incident



DigiNotar Incident

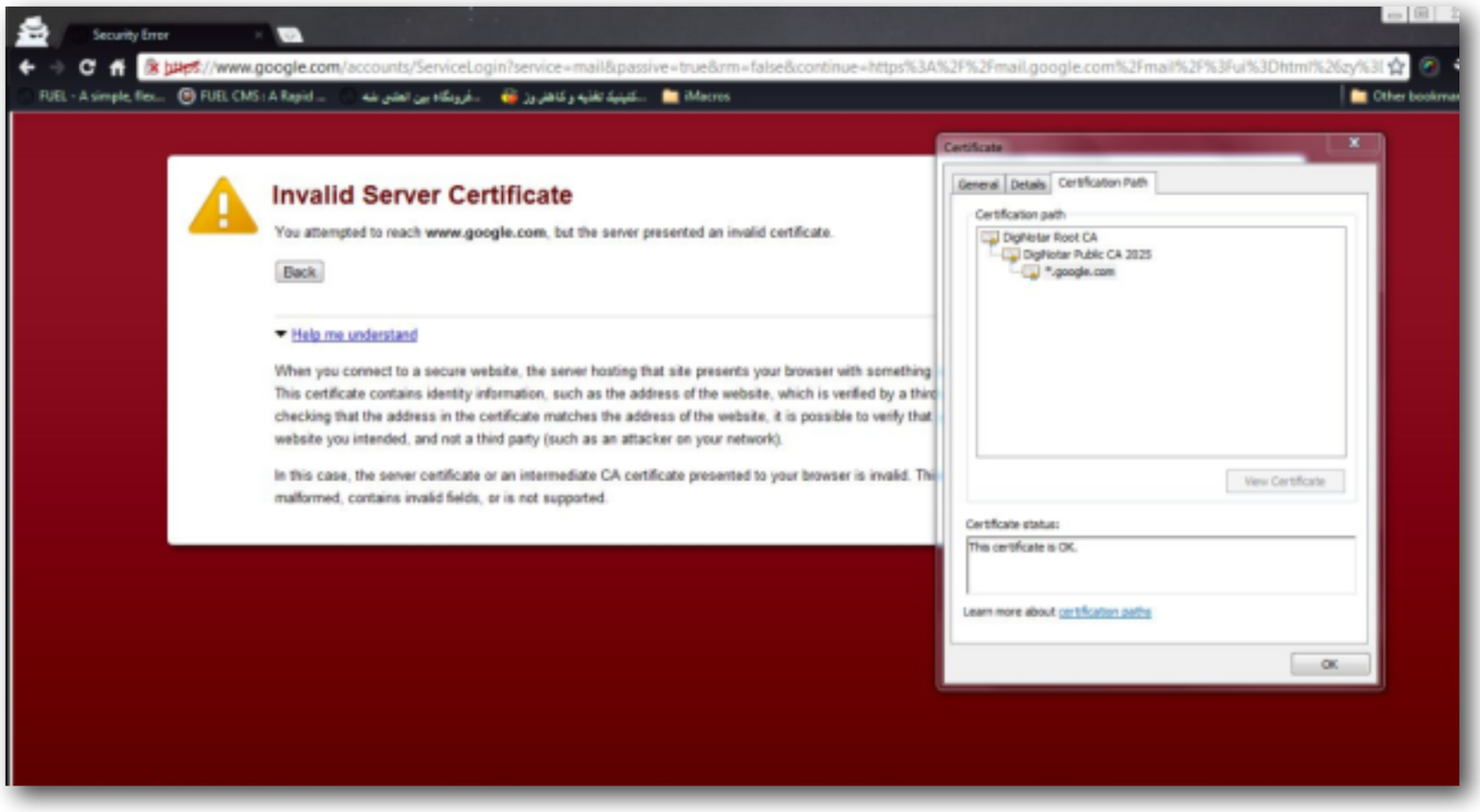
DigiNotar is a CA based in the Netherlands that is (well, was) trusted by most OSes and browsers

July 2011: Issued fake certificate for gmail.com to site in Iran that ran MitM attack...

... this fooled most browsers, but...

DigiNotar Incident

- As added security measure, Google Chrome hardcodes fingerprint of Google's certificate
- Since DigiNotar didn't issue Google's true certificate, this caused an error message in Chrome



Meta-Issue:
How much should we trust CAs?

(Because right now, we trust them a lot.)

Key Management Summary

- Key management is HARD
- PKI is not a panacea
- Devil is in the details