WILLIAM & MARY
CHARTERED 1693

CSCI 780:
IoT Security

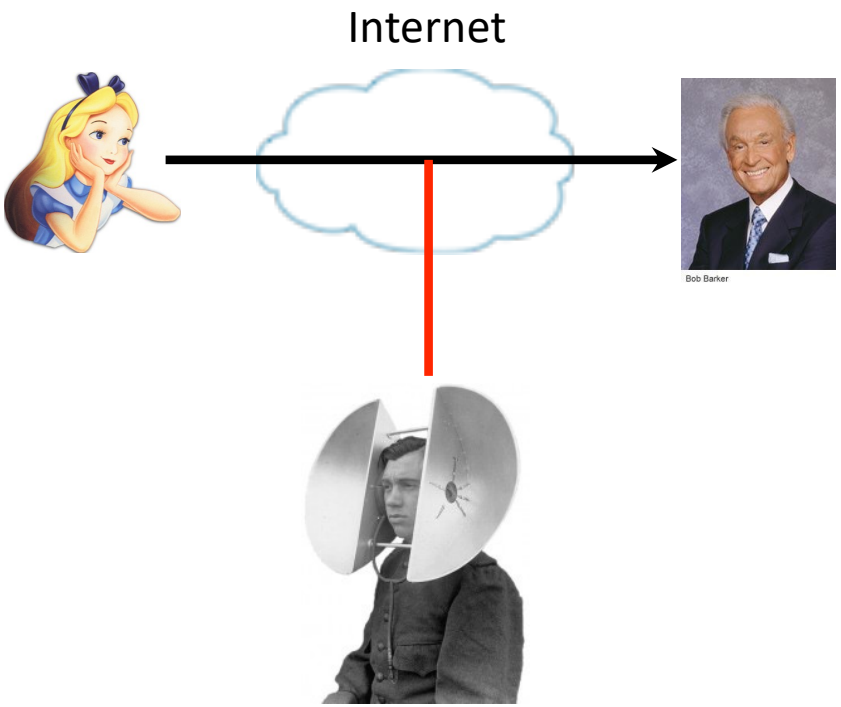
Lecture 5

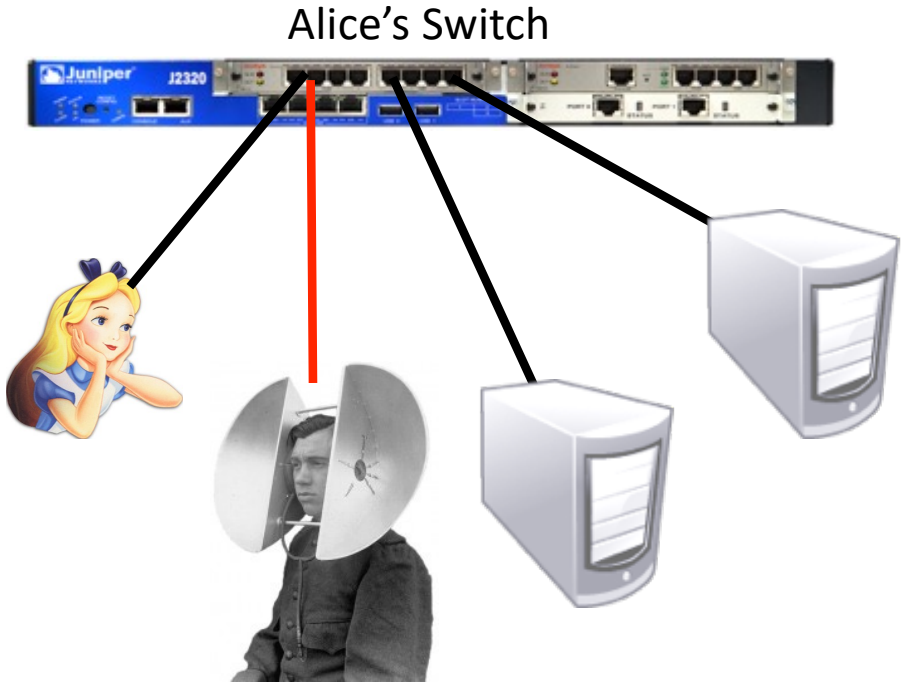
Prof. Adwait Nadkarni

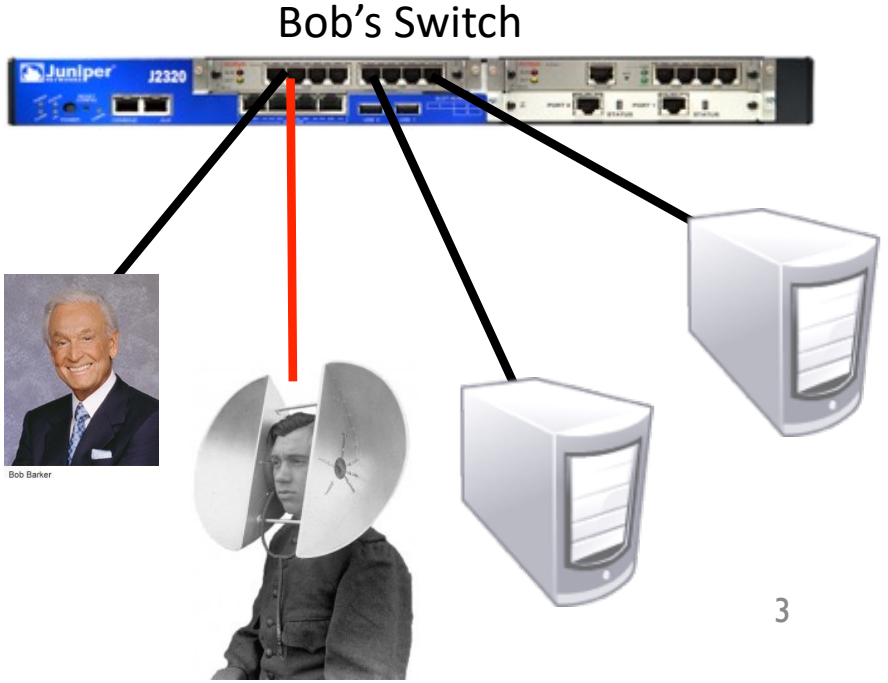
(Derived from slides by William Enck, Micah Sherr, Patrick McDaniel, and Peng Ning)

Very brief intro to cryptography

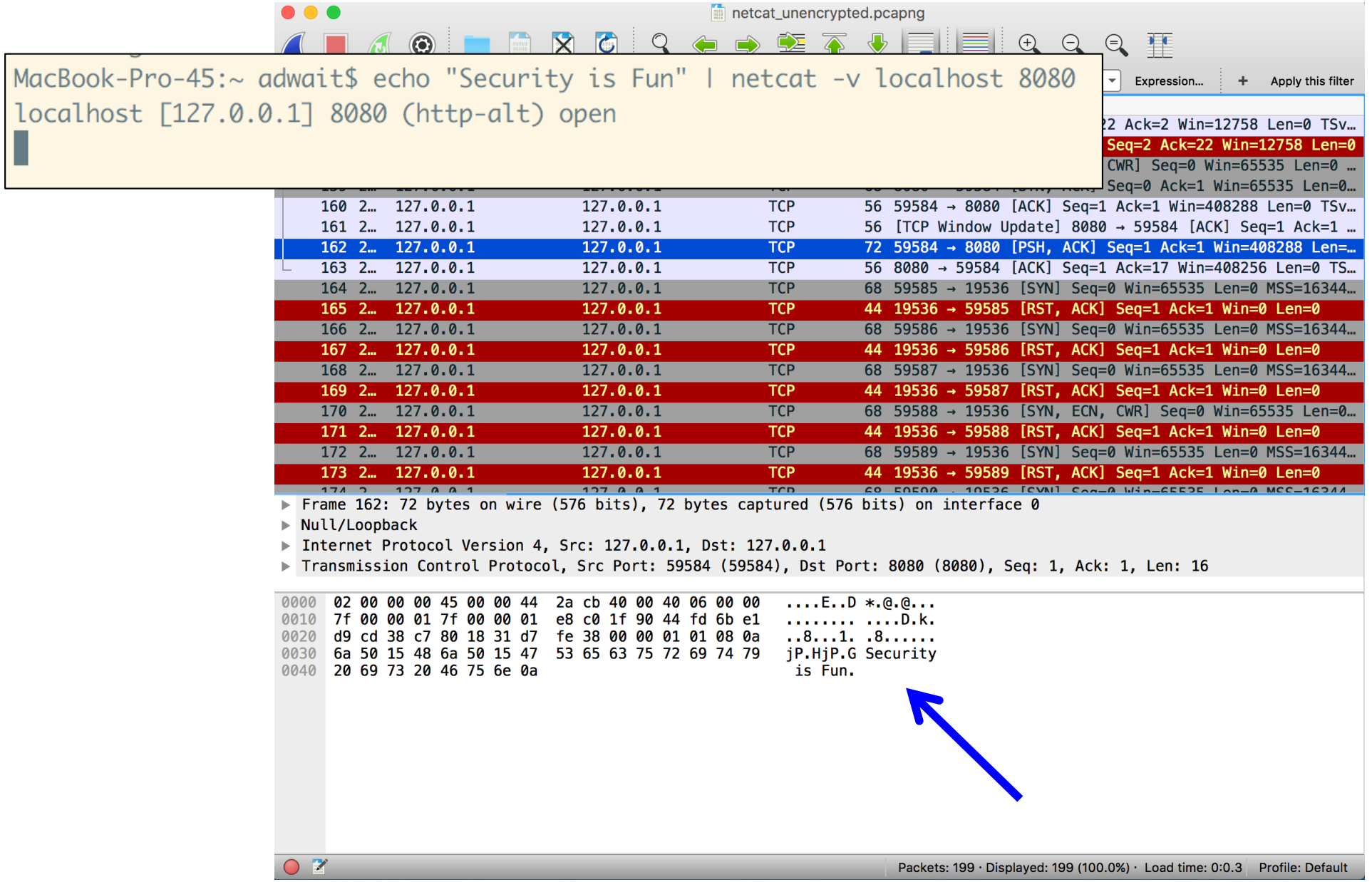
Eavesdropping







Why is crypto useful?



Why is this bad?

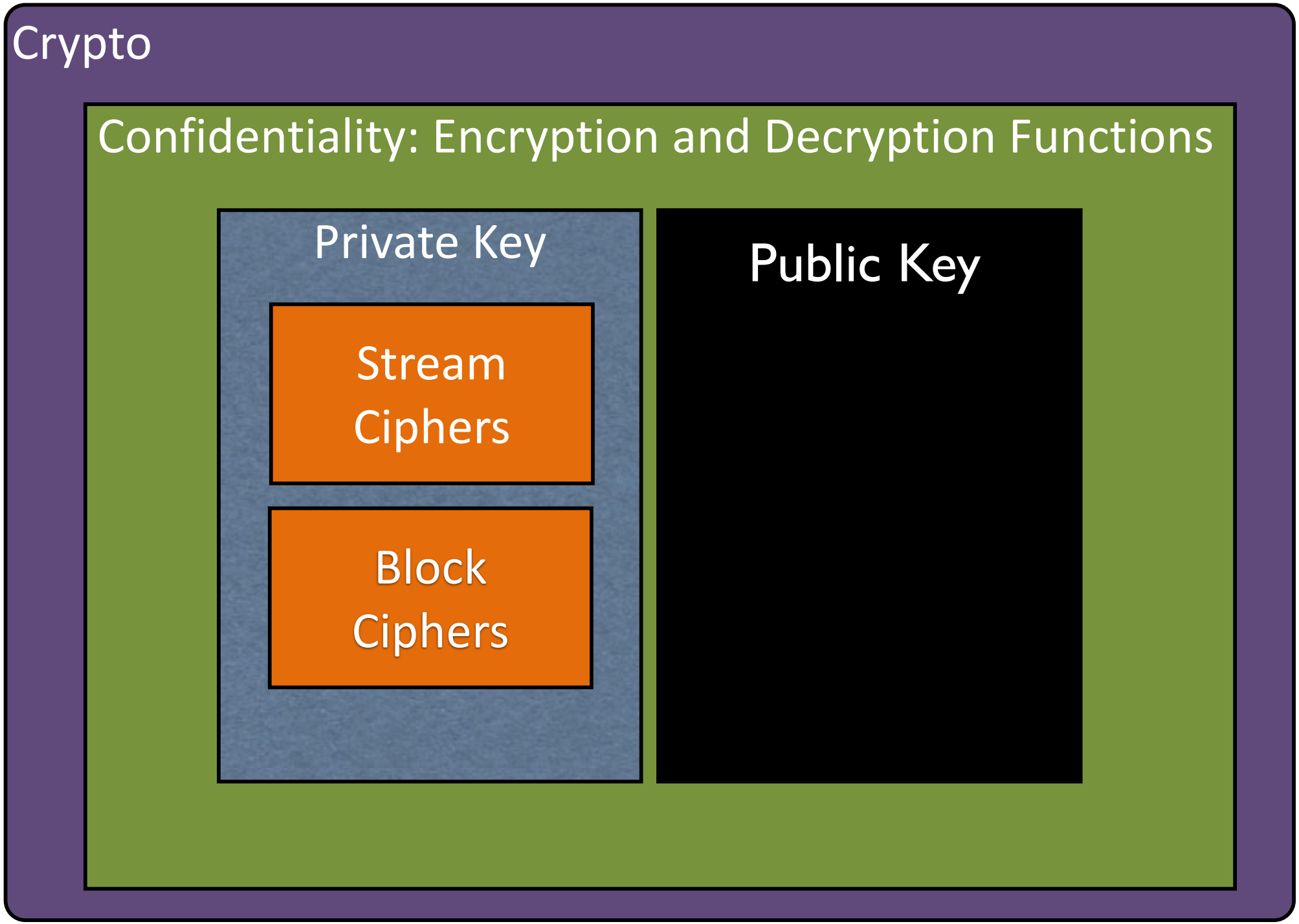
- Its just an instant message, right?

Alice uses the Internet for:

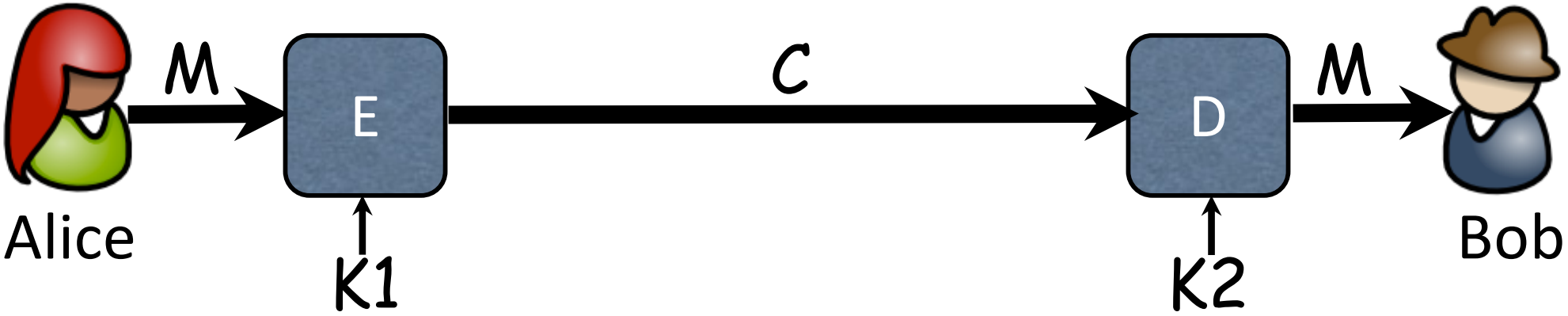
- Email
- Banking
- Online shopping
- Social networking
- …

cryptography < security

- Cryptography isn't the solution to security
  - Buffer overflows, worms, viruses, trojan horses, SQL injection attacks, cross-site scripting, bad programming practices, etc.
- It's a tool, not a solution
- Even when used, difficult to get right
  - Choice of encryption algorithms
  - Choice of parameters
  - Implementation
  - Hard to detect errors
    - Even when crypto fails, the program may still work
    - May not learn about crypto problems until after they've been exploited

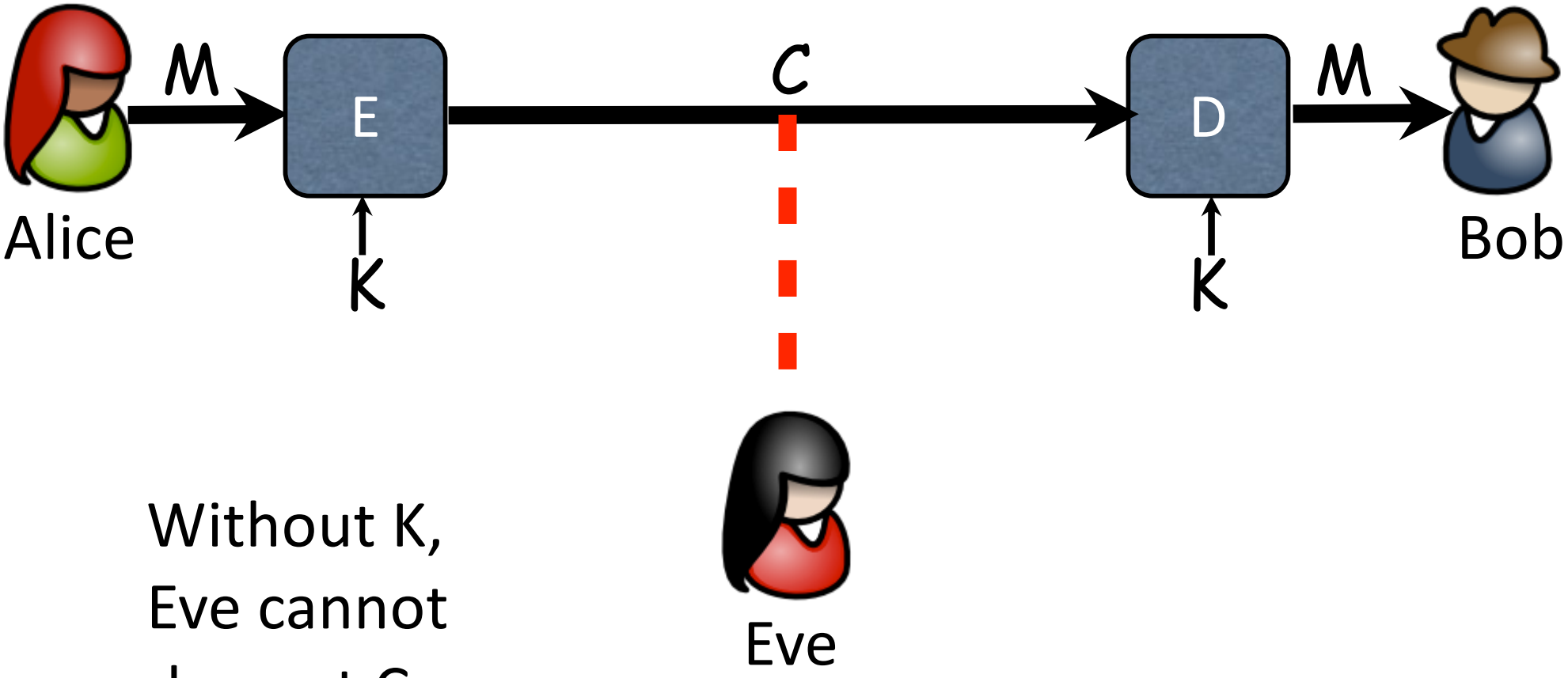

Crypto
Confidentiality: Encryption and Decryption Functions
Private Key
Stream Ciphers
Block Ciphers
Public Key

Symmetric and Asymmetric Crypto



- **Symmetric crypto:** (also called **private key crypto**)
  - Alice and Bob share the same key (K=K1=K2)
  - K used for both encrypting and decrypting
  - Doesn't imply that encrypting and decrypting are the same algorithm
  - Also called **private key** or **secret key** cryptography, since knowledge of the key reveals the plaintext
- **Asymmetric crypto:** (also called **public key crypto**)
  - Alice and Bob have different keys
  - Alice encrypts with K1 and Bob decrypts with K2
  - Also called **public key** cryptography, since Alice and Bob can publicly post their *public* keys

Secret/Symmetric/Private Key Crypto



Without K, Eve cannot decrypt C

Private-key crypto is like a door lock

Why?

Public Key Crypto
(10,000 ft view)

- Separate keys for encryption and decryption
  - Public key: anyone can know this
  - Private key: kept confidential
- Anyone can encrypt a message to you using your public key
- The private key (kept confidential) is required to decrypt the communication
- Alice and Bob no longer have to have *a priori* shared a secret key

Problem? YES. *How do we know if Bob's key is really Bob's?*

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***

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***

Encryption using private key

- Encryption and Decryption

  $E_{k-}(M)$: ciphertext = plaintext^{d} mod n

  D_{k+}(ciphertext) : plaintext = ciphertext^{e} mod n

- E.g.,
  - E({3,33},4) = 4^{3} mod 33 = 64 mod 33 = 31
  - D({7,33},31) = 31^{7} mod 33 = 27,512,614,111 mod 33 = 4
- Q: ***Why encrypt with private key?***

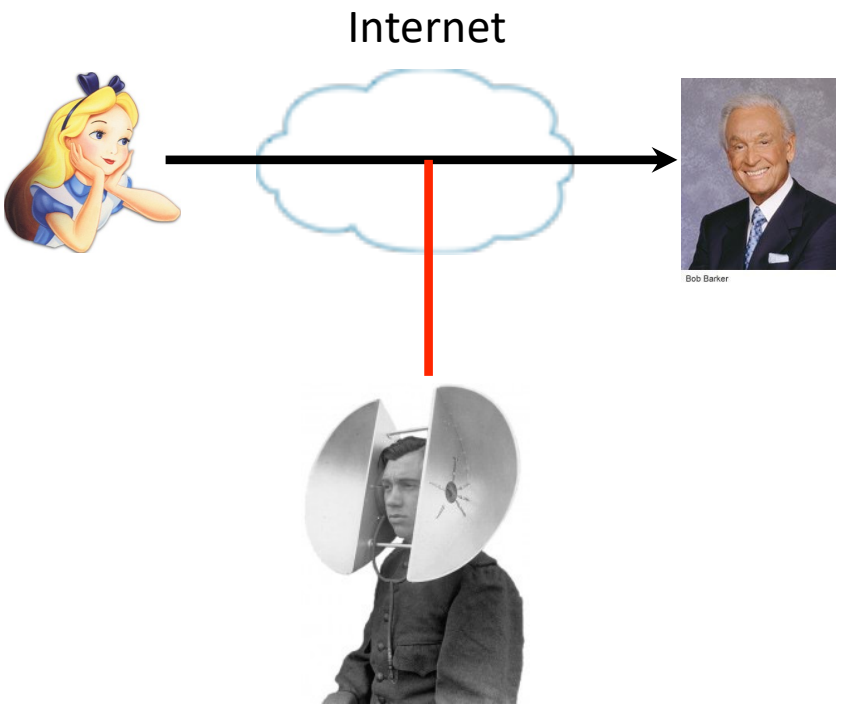
Digital Signatures

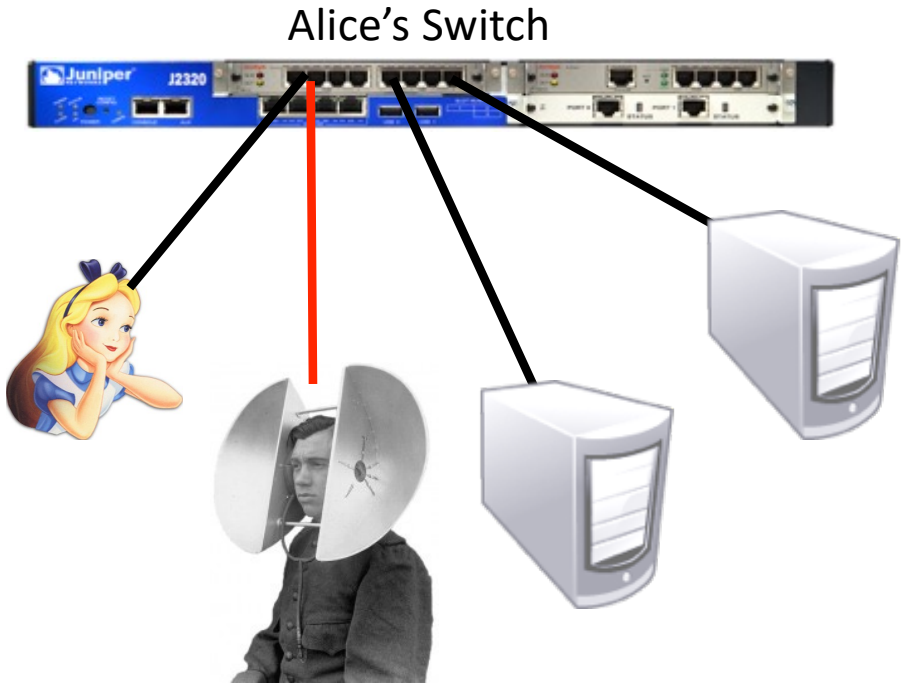
- A digital signature serves the same purpose as a real signature.
  - It is a mark that only sender can make
  - Other people can easily recognize it as belonging to the sender
- Digital signatures must be:
  - **Unforgeable**: If Alice signs message M with signature S, it is impossible for someone else to produce the pair (M, S).
  - **Authentic**: If Bob receives the pair (M, S) and knows Alice's public key, he can check ("verify") that the signature is really from Alice

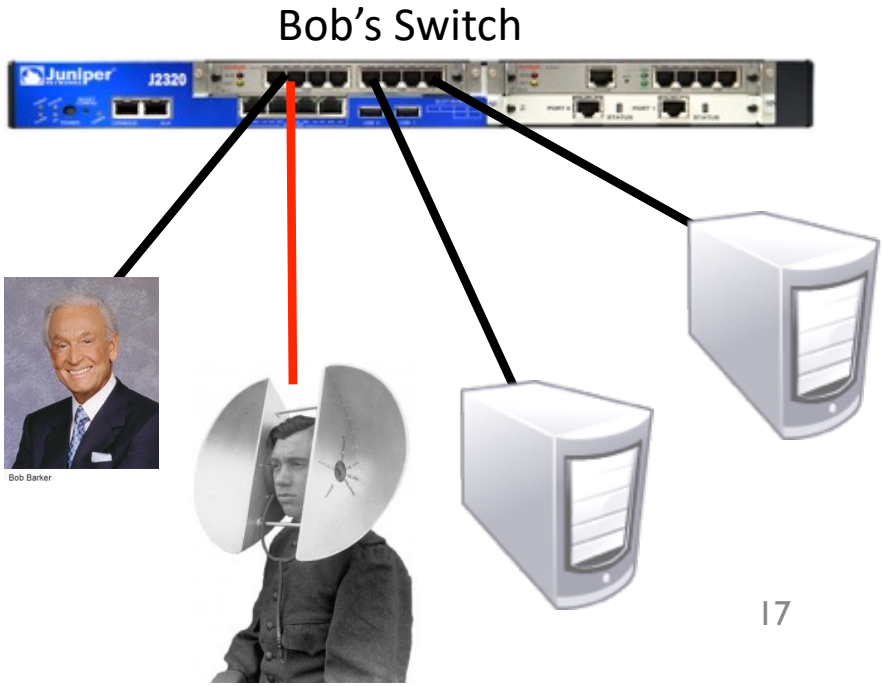
How can Alice *sign* a digital document?

- Digital document: M
- Since RSA is slow, hash M to compute digest: m = h(M)
- Signature: Sig(M) = $E_{k^-}(m) = m^d \bmod n$
  - Since only Alice knows k^-, only she can create the signature
- To verify: Verify(M,Sig(M))
  - Bob computes h(M) and compares it with $D_{k^+}(Sig(M))$
  - Bob can compute $D_{k^+}(Sig(M))$ since he knows k^+ (Alice's public key)
  - If and only if they match, the signature is verified (otherwise, verification fails)

Recall: Eavesdropping



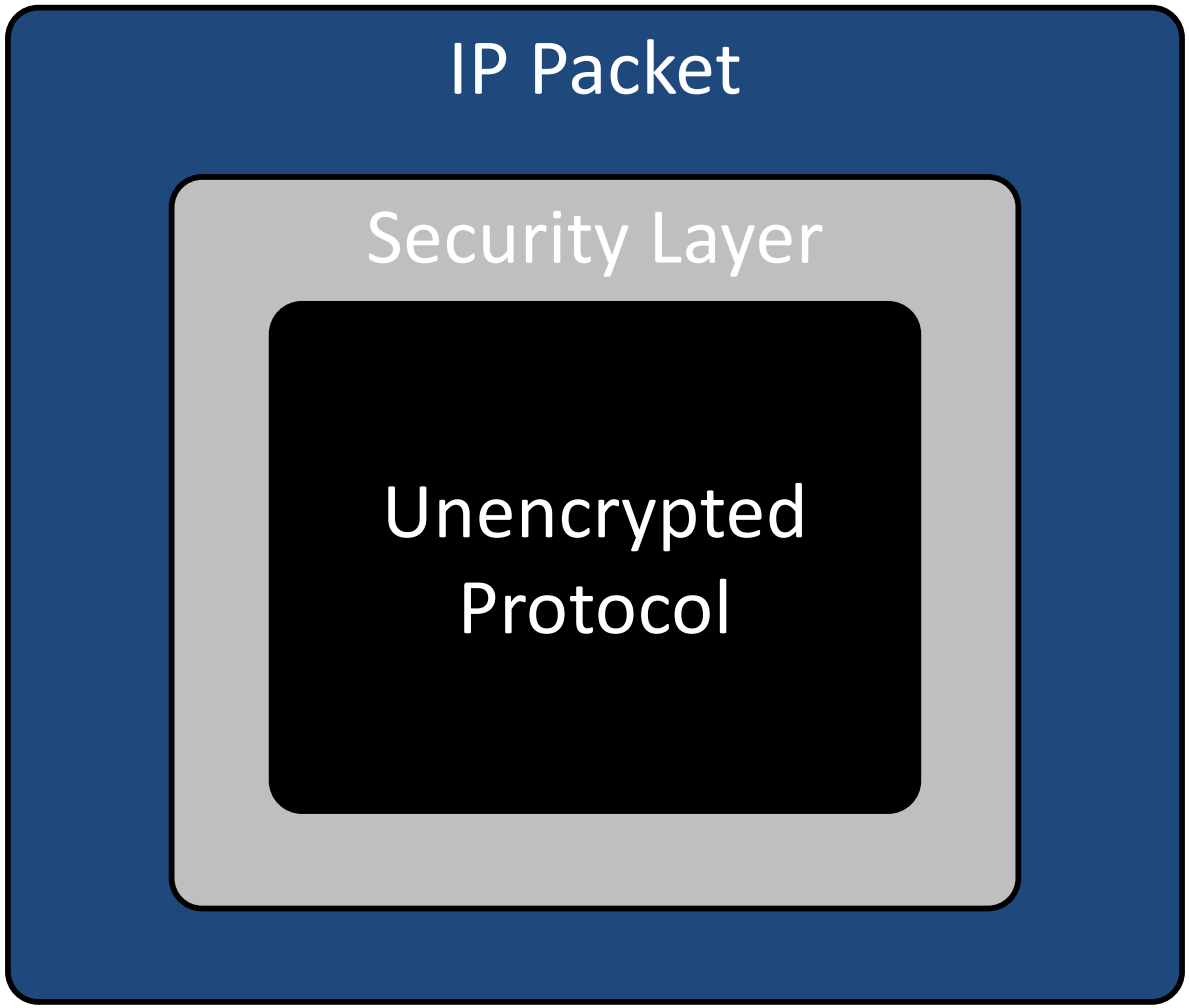




Let's use that crypto stuff

- Let's build some new protocols
  - HTTP → SecureHTTP
  - POP → POPSecure
  - IMAP → Cry[illegible]
  - SMTP → SMTP[illegible]
  - FTP → FTPS
  - Jabber → SecJabber
  - Telnet → TelCryptNet

Let's build a crypto-wrapper standard instead


IP Packet
Security Layer
Unencrypted
Protocol

What properties should this crypto-wrapper have?

- Confidentiality
- Integrity
- Authenticity
  - Server
  - Client
  - Mutual authentication
- Reliability

SSL / TLS

History

- **Secure Sockets Layer (SSL)** developed by Netscape (remember them?) in 1995
  - Version 1 never released
  - Version 2 incorporated into Netscape Navigator 1.1
  - Microsoft fixes vulnerabilities in SSLv2 and introduces Private Communications Technology (PCT) protocol
  - Netscape overhauls SSLv2, fixing some more security issues, and releases SSLv3
  - IETF takes over and releases **Transport Layer Security (TLS)**, a non-interoperable upgrade to SSLv3
    - current version is TLS version 1.3, RFC 8446 (Aug 2018)

Overview

- Alice (client) initiates conversation with Bob (server)
- Bob sends Alice his certificate
- Alice verifies certificate
- Alice picks a random number S and sends it to Bob, encrypted with Bob's public key
- Both parties derive key material from S
- Client and server exchange encrypted and integrity-protected data

SSLv2 Handshake

Alice

Bob

`ClientHello`, Version, Cipher list., R_{Alice}

`ServerHello`, Ver., Cert., Chosen cipher, R_{Bob}

E(Bob+,S)

E(K',Data)

Bob Barker

Alice computes **master secret k** as $K=f(S,R_{Alice},R_{Bob})$

Encryption and integrity keys derived from Master Secret

Alice randomly chooses S.

Bob computes **master secret k** as $K=f(S,R_{Alice},R_{Bob})$

Authentication

Alice

Bob

`ClientHello`, Version, Cipher list. R_{Alice}

`ServerHello`, Ver., Cert., Chosen cipher, R_{Bob}

E(Bob+,S)

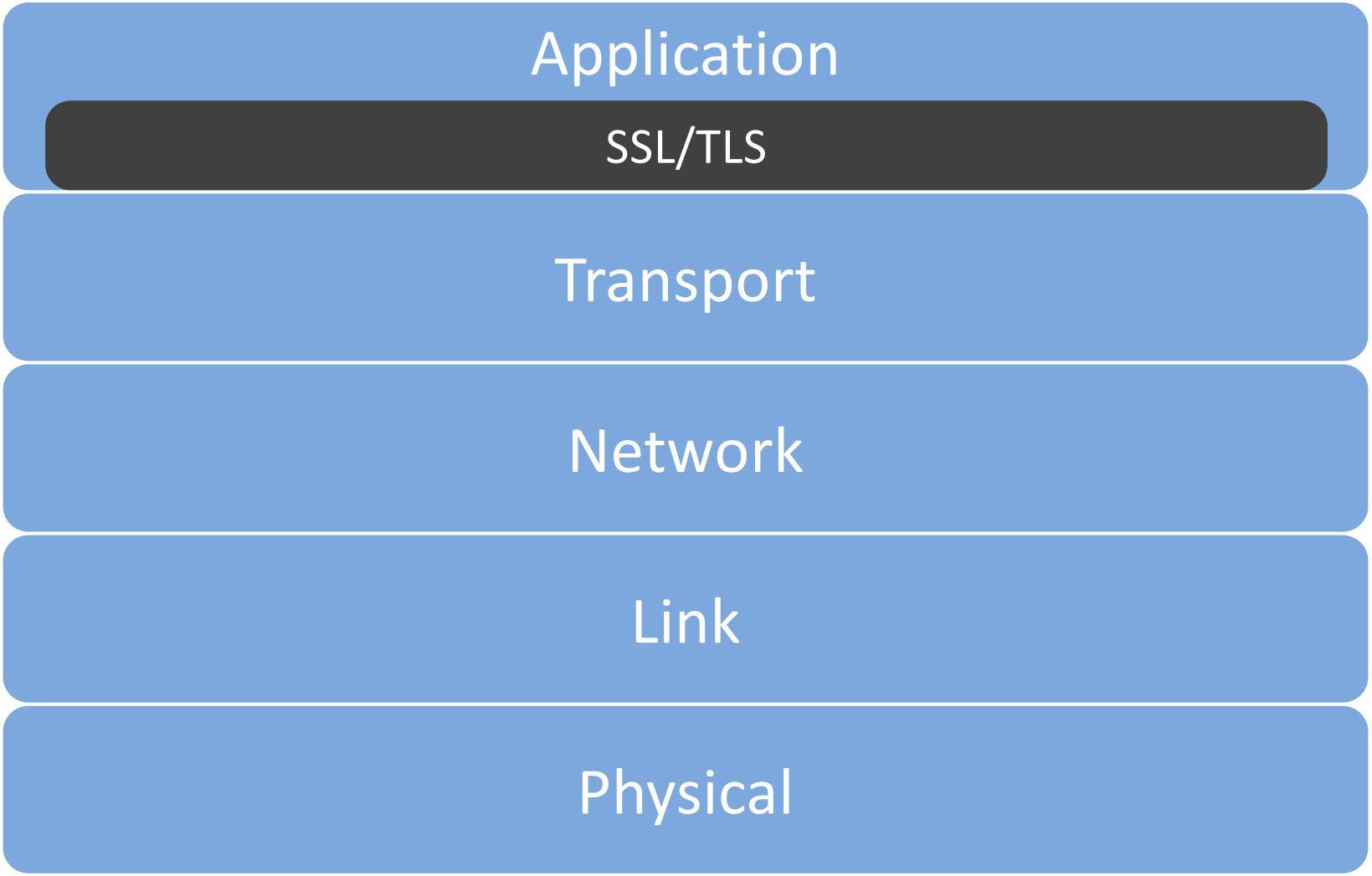
E(K',Data)

Q: Which parties are authenticated?

Bob Barker

SSL/TLS in the Real World

Network Stack, revisited

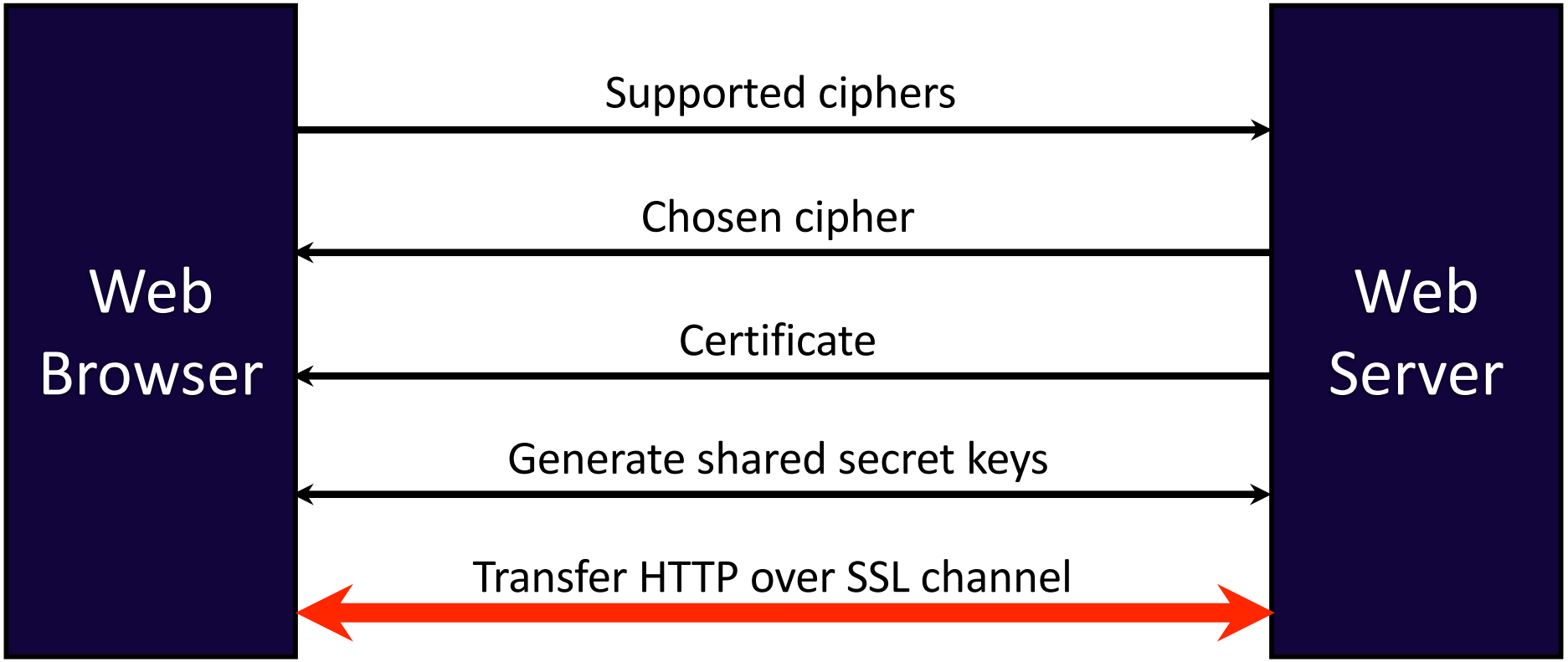


SSL/TLS in the Real World

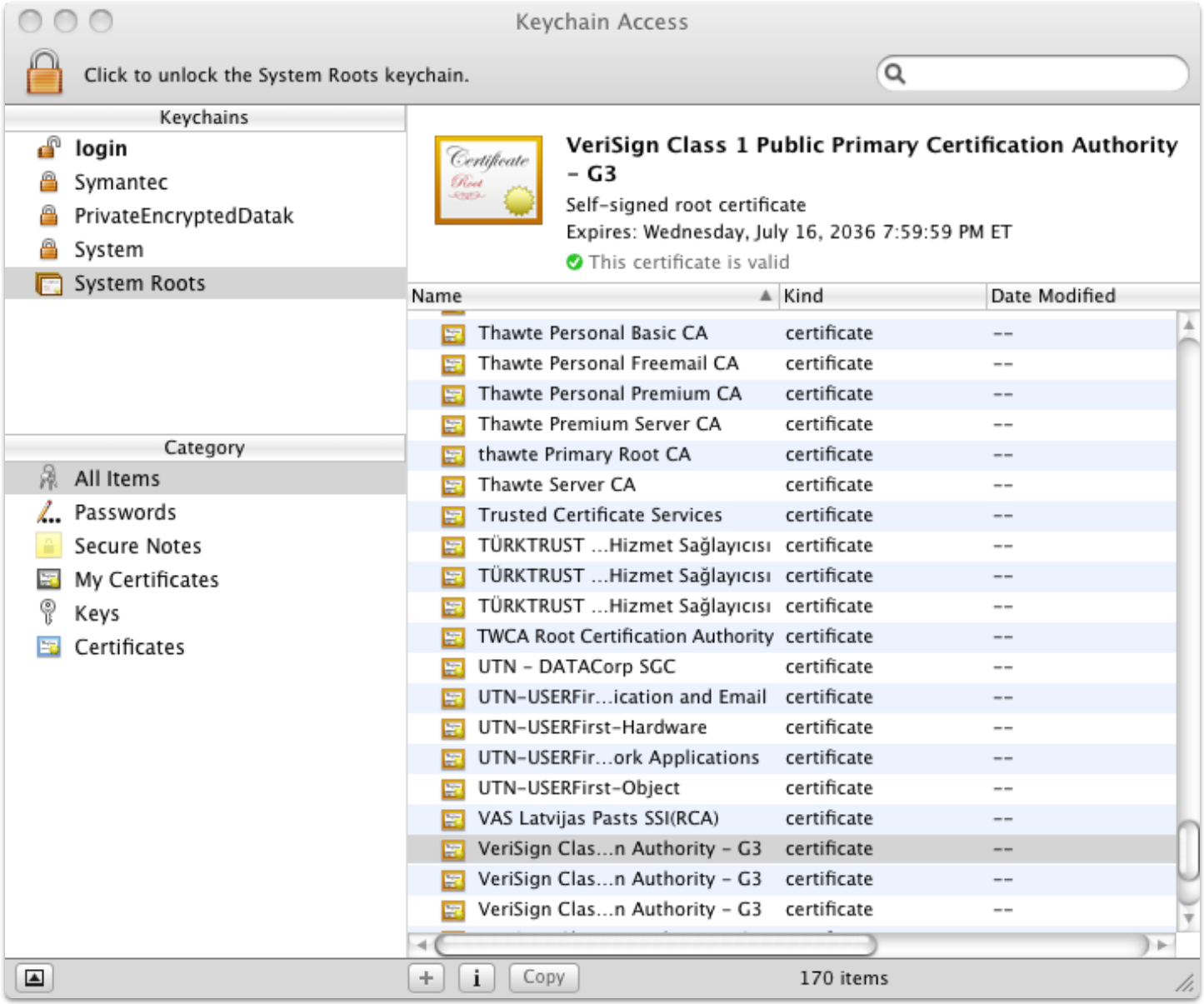
- Most (modern) browsers support SSLv3, TLS 1.2
- Client authentication very rare **-- WHY?**
- Implementations:
  - HTTP (80) → HTTPS (443)
  - POP (110) → POP3S (995)
  - IMAP (143) → IMAPS (993)
  - SMTP (25) → SMTP with SSL (465)
  - FTP (20,21)→ FTPS (989,990)
  - Telnet (23) → Telnets (992)

SSL/TLS and the Web

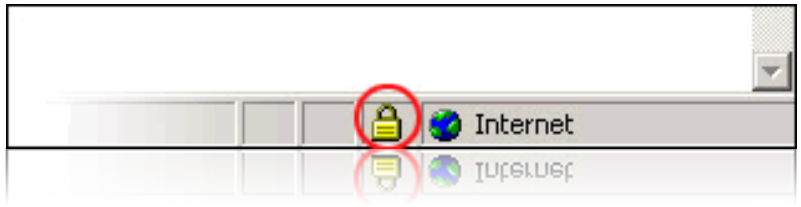
- HTTPS: Tunnel HTTP over SSL/TLS
- Add golden lock symbol 

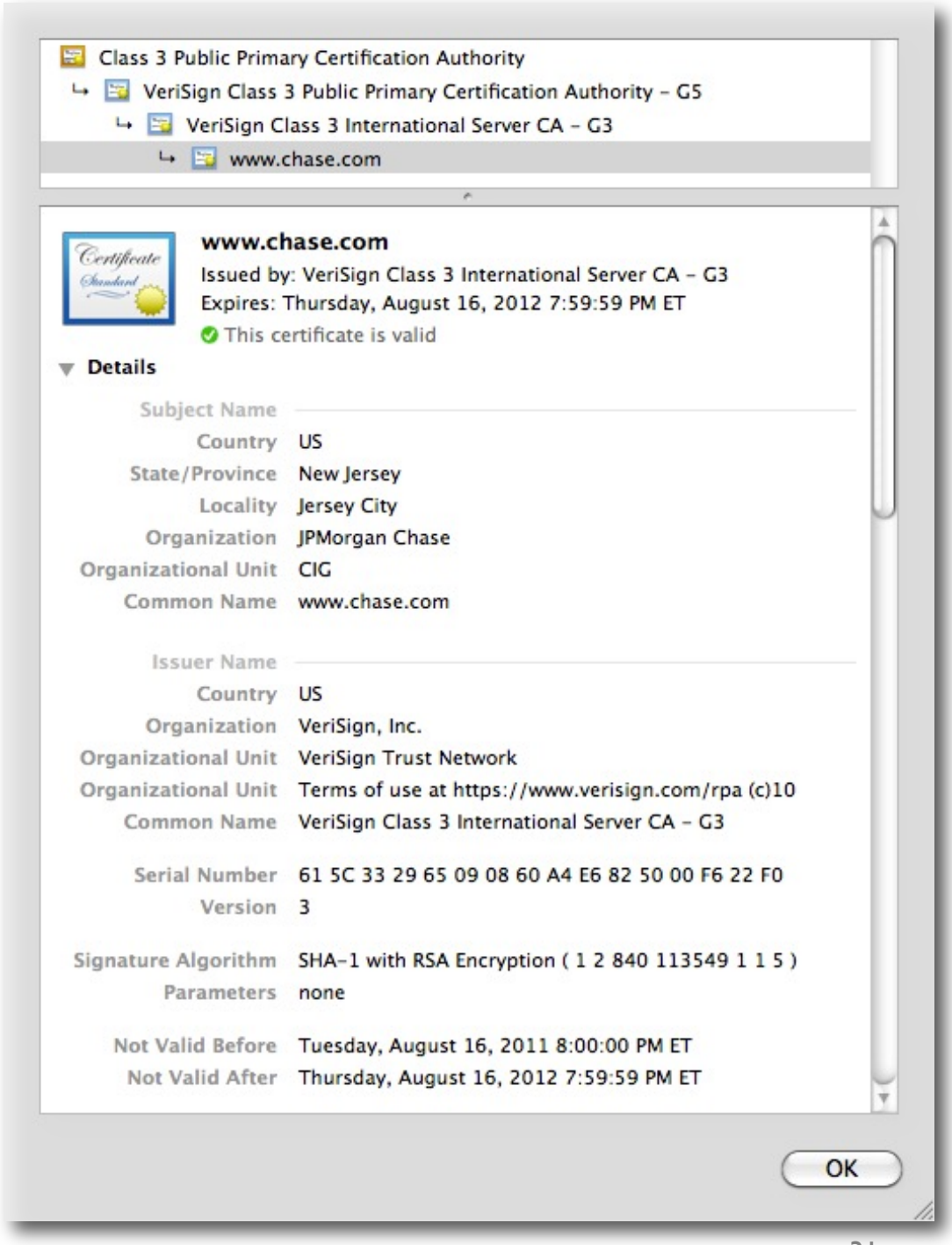


How to verify Bob's certificate? Use PKI



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

Why do I trust the certificate?

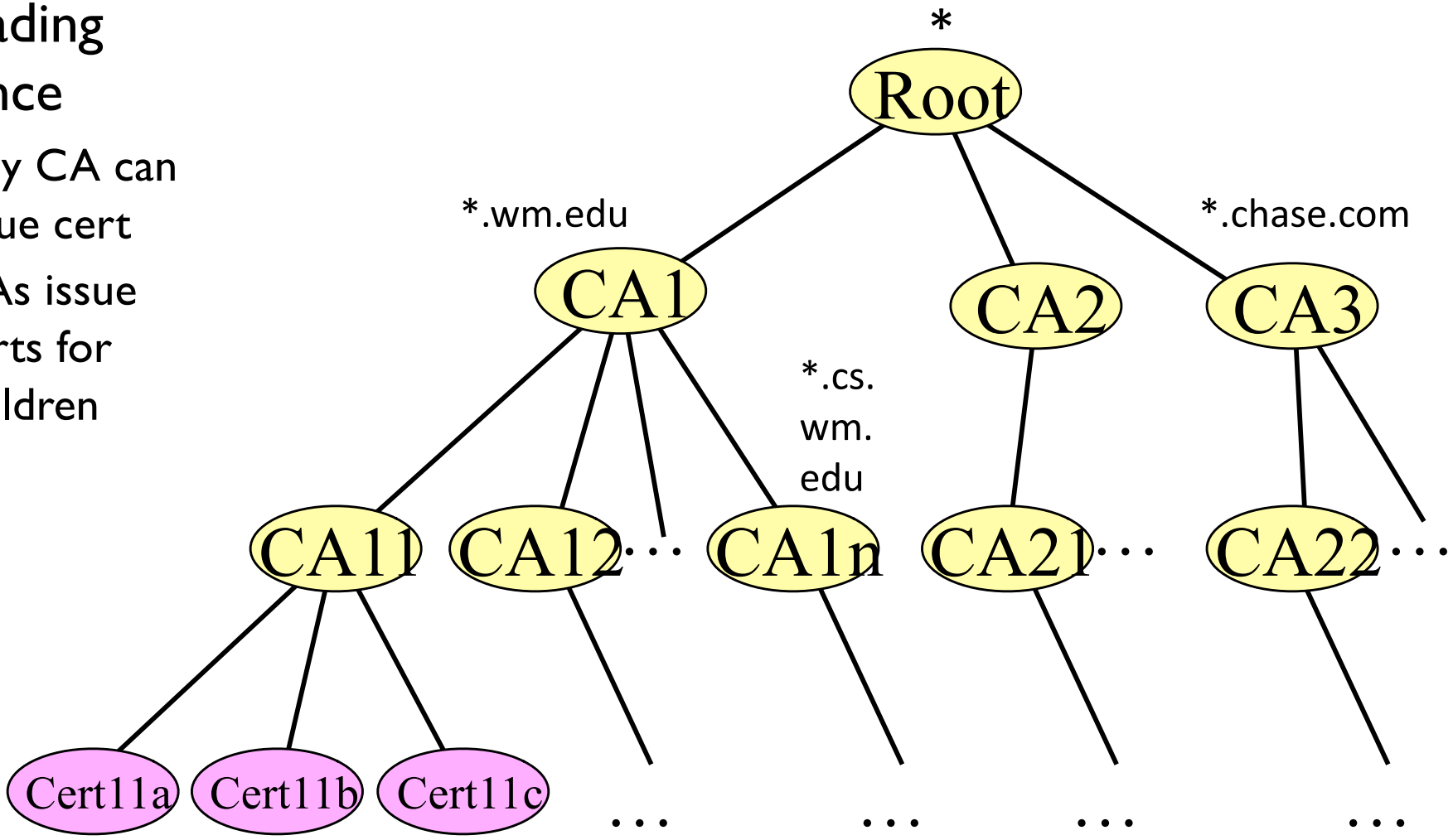
- A collections of "root" CA certificates
  - … baked into your browser
  - … vetted by the browser manufacturer
  - … supposedly closely guarded
- Root certificates used to validate certificate
  - Vouches for certificate's authenticity

Public Key Infrastructure

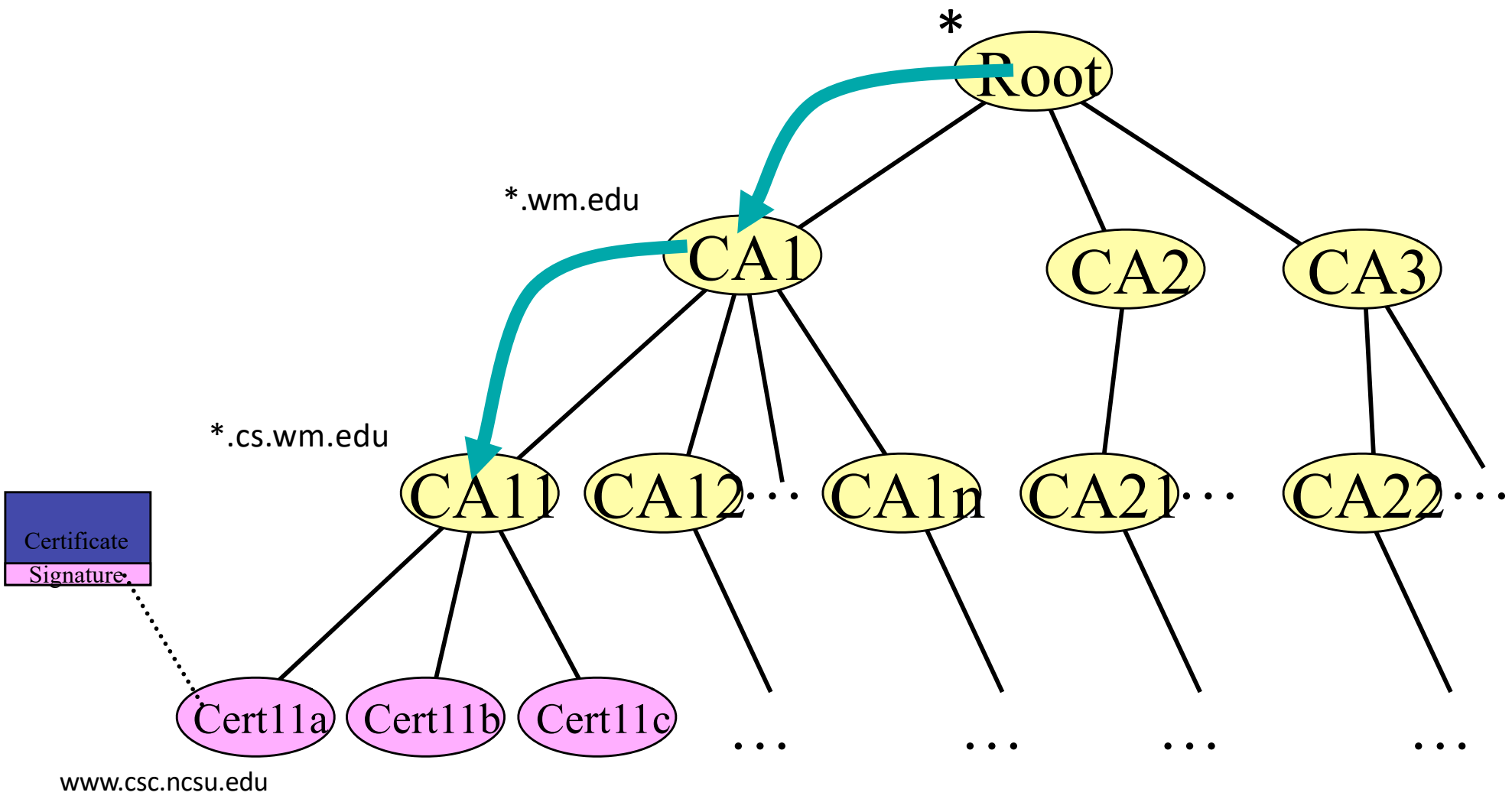
- Hierarchy of keys used to authenticate certificates
- Requires a **root of trust** (i.e., a **trust anchor**)

What is a PKI?

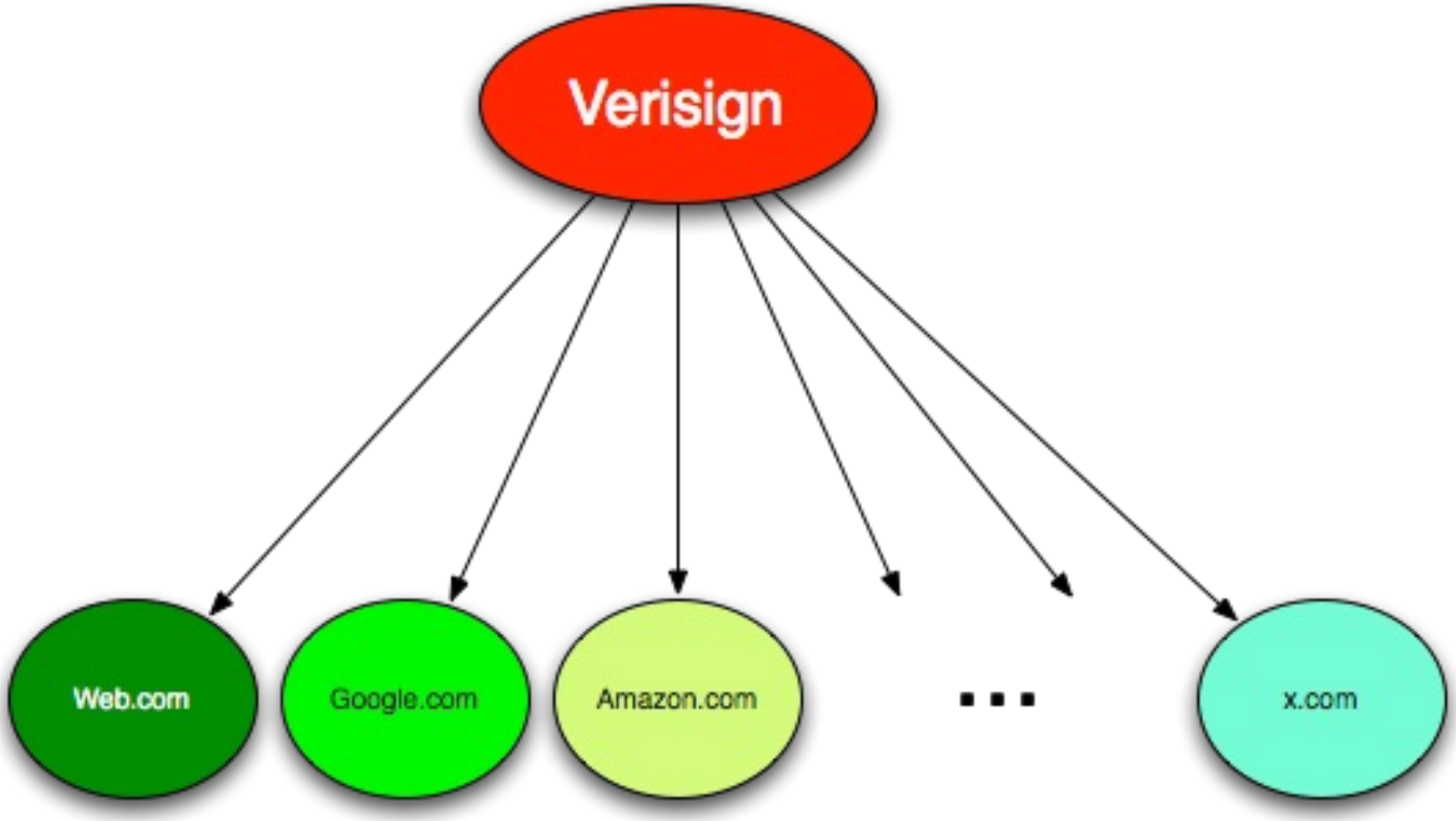
- Rooted tree of CAs
- Cascading issuance
  - Any CA can issue cert
  - CAs issue certs for children



Certificate Validation

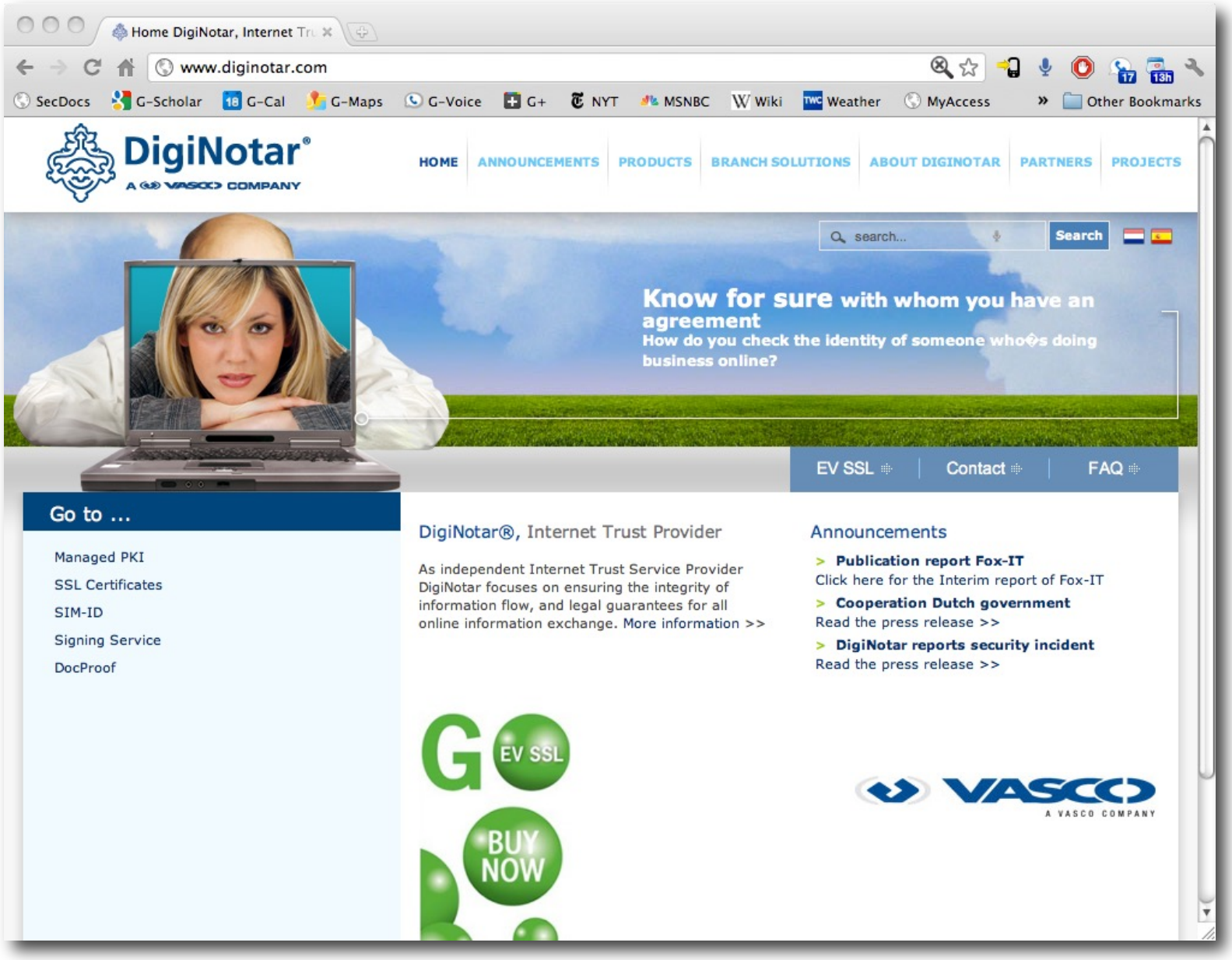


PKIs in Reality



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

The 2011 DigiNotar Incident



Certificate Verification

- SSL is an *application layer protocol*
  - Software developers must use it correctly
- Pre-Smartphone World
  - Small set of applications that use SSL (E.g., Web Browser)
  - Lots of attention to those apps
- Smartphone World
  - Hundreds of thousands of applications that use SSL
  - Many apps do not verify certificates correctly – **Implications?**
  - Developers change default configuration – **WHY?**