CSE 484 / CSE M 584: Applied Cryptography

Winter 2023

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements / Plan

- Through Wednesday (2/8): Applied Crypto
- Friday (2/10): Guest Lecture: Prof. Elissa Redmiles (MPI)
- Wednesday (2/22): Zoom
- Friday (2/24): Guest Lecture: Alex Gantman (Qualcomm) (On Zoom)

Review: Some Number Theory Facts

- Euler totient function $\varphi(n)$ $(n \geq 1)$ is the number of integers in the $[1,n]$ interval that are relatively prime to n
  - Two numbers are relatively prime if their greatest common divisor (gcd) is 1
  - Easy to compute for primes: $\varphi(p) = p-1$
  - Note that $\varphi(ab) = \varphi(a)\,\varphi(b)$ if a & b are relatively prime

Review: RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - Say, 2048 bits each (need primality testing, too)
  - Compute **n**=pq and φ(**n**)=(p-1)(q-1)
  - Choose small **e,** relatively prime to φ(n)
    - Typically, **e=3** or **e=2^{16}+1=65537**
  - Compute unique **d** such that ed ≡ 1 mod φ(n)
    - Modular inverse: d ≡ e^{-1} mod φ(n) ← How to compute?
  - Public key = (e,n); private key = (d,n)
- Encryption of m: c = m^e mod n
- Decryption of c: c^d mod n = $(m^e)^d$ mod n = m

How to compute?

- Extended Euclidian algorithm
- Wolfram Alpha ☺
- Brute force for small values

Review: Why is RSA Secure?

- RSA problem: given c, $n=pq$, and e such that $\gcd(e, \varphi(n))=1$, find m such that $m^e=c \bmod n$
  - In other words, recover m from ciphertext c and public key (n,e) by taking e^{th} root of c modulo n
  - There is no known efficient algorithm for doing this *without* knowing p and q
- Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e_1}p_2^{e_2}\ldots p_k^{e_k}$

- If factoring is easy, then RSA problem is easy
  (knowing factors means you can compute d = inverse of e mod (p-1)(q-1))
  - It may be possible to break RSA without factoring n – but if it is, we don't know how

Review: RSA Encryption Caveats

- Encrypted message needs to be interpreted as an integer less than n
- Don't use RSA **directly** for privacy – output is deterministic! Need to pre-process input somehow.
- Plain RSA also does not provide integrity
  - Can tamper with encrypted messages

In practice, OAEP is used: instead of encrypting M, encrypt

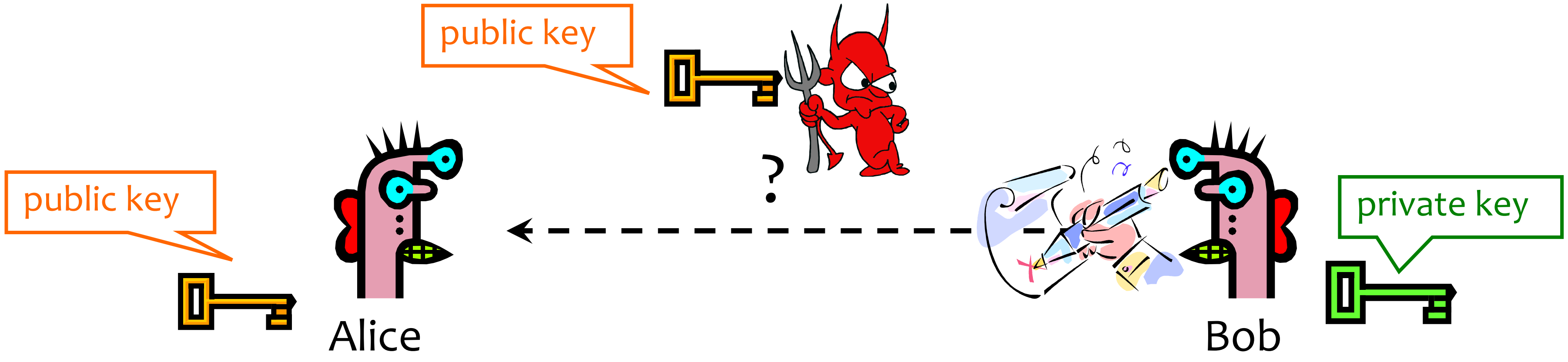
$M \oplus G(r) \,||\, r \oplus H(M \oplus G(r))$

- r is random and fresh, G and H are hash functions

Stepping Back: Asymmetric Crypto

- Last time we saw session key establishment (Diffie-Hellman)
  - Can then use shared key for symmetric crypto
- We just saw: public key encryption
  - For confidentiality
- Finally, now: digital signatures
  - For authenticity

Digital Signatures: Basic Idea



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goal: Bob sends a "digitally signed" message

1. To compute a signature, must know the private key
2. To verify a signature, only the public key is needed

RSA Signatures

- Public key is **(n,e)**, private key is **(n,d)**
- To sign message m: s = $m^d \bmod n$
  - Signing & decryption are same **underlying** operation in RSA
  - It's infeasible to compute **s** on **m** if you don't know **d**
- To verify signature s on message m: verify that $s^e \bmod n = (m^d)^e \bmod n = m$
  - Just like encryption (for RSA primitive)
  - Anyone who knows **n** and **e** (public key) can verify signatures produced with d (private key)
- In practice, also need padding & hashing
  - Without padding and hashing: Consider multiplying two signatures together
  - Standard padding/hashing schemes exist for RSA signatures

DSS Signatures

- Digital Signature Standard (DSS)
  - U.S. government standard (1991, most recent rev. 2013)
- Public key: $(p, q, g, y=g^x \bmod p)$, private key: x
- Security of DSS requires hardness of discrete log
  - If could solve discrete logarithm problem, would extract x (private key) from $g^x \bmod p$ (public key)
- Again: We've discussed discrete logs modulo integers; significant advantages to using elliptic curve groups instead.

Post-Quantum Cryptography

- If quantum computers become a reality
  - It becomes much more efficient to break conventional asymmetric encryption schemes (e.g., factoring becomes "easy")
  - For block ciphers (symmetric encryption), use 128-bit keys for 256-bits of security
- There exists efforts to make quantum-resilient asymmetric encryption schemes

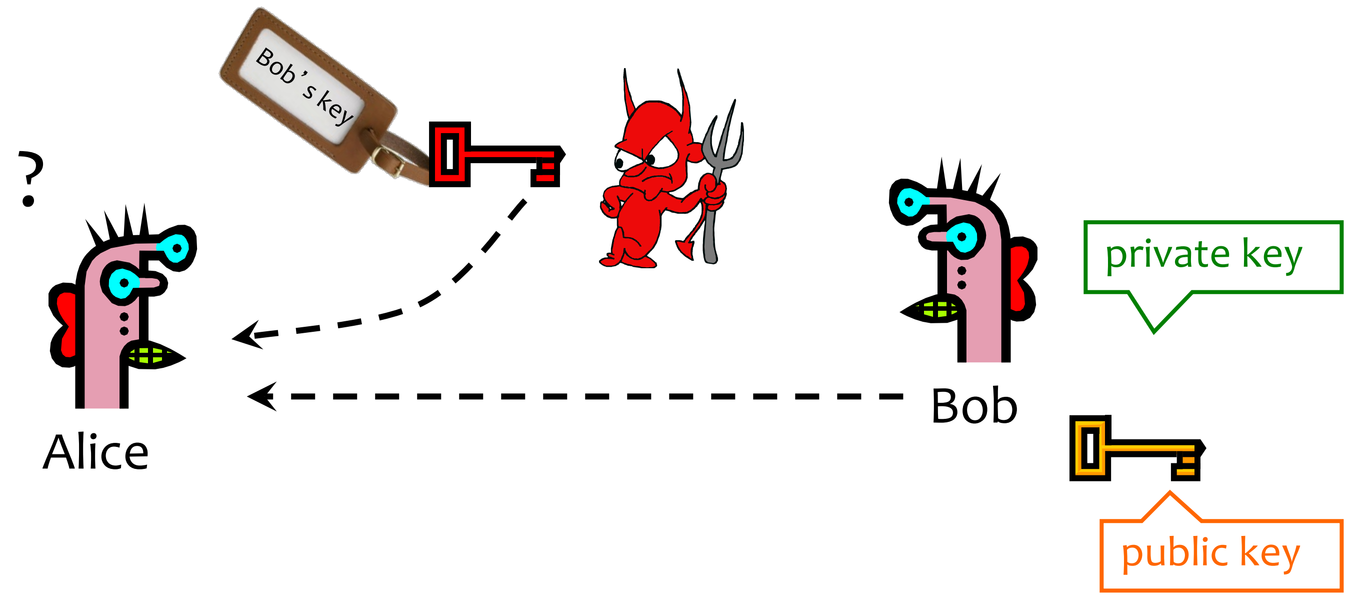
Cryptography Summary

- Goal: Privacy
  - Symmetric keys:
    - One-time pad, Stream ciphers
    - Block ciphers (e.g., DES, AES) → modes: EBC, CBC, CTR
  - Public key crypto (e.g., Diffie-Hellman, RSA)
- Goal: Integrity
  - MACs, often using hash functions (e.g, SHA-256)
- Goal: Privacy and Integrity ("authenticated encryption")
  - Encrypt-then-MAC
- Goal: Authenticity (and Integrity)
  - Digital signatures (e.g., RSA, DSS)

Want More Crypto?

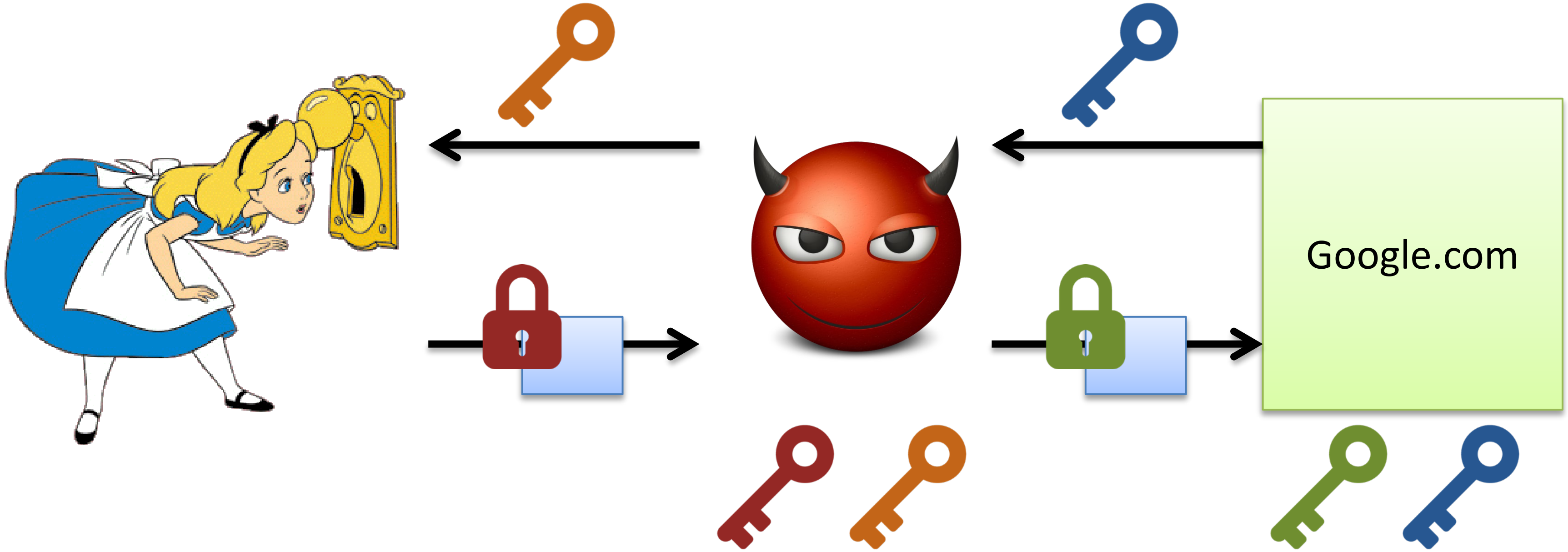
- Some suggestions:
  - Cryptography course
  - Stanford Coursera (Dan Boneh): https://www.coursera.org/learn/crypto

Authenticity of Public Keys



Problem: How does Alice know that the public key she received is really Bob's public key?

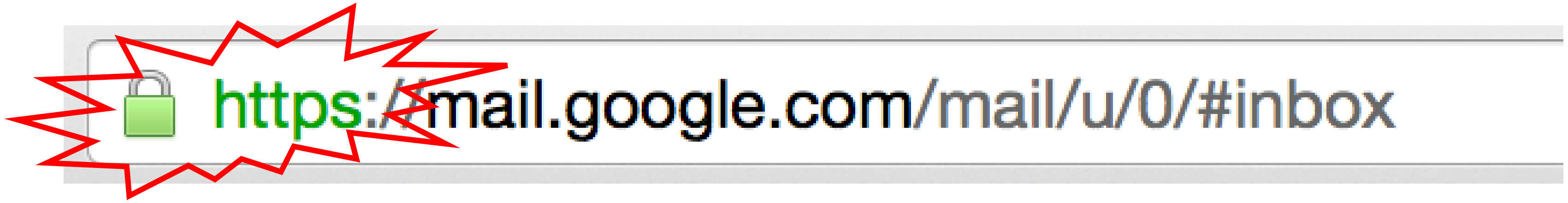
Threat: Person-in-the Middle



Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - sig_{CA}(“Bob”, PK_B)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating a private/public key pair, user proves his identity and knowledge of the private key to obtain CA’s certificate for the public key (offline)
  - Every computer is pre-configured with CA’s public key

You encounter this every day…



SSL/TLS: Encryption & authentication for connections

SSL/TLS High Level

- SSL/TLS consists of two protocols
  - Familiar pattern for key exchange protocols
- Handshake protocol
  - Use public-key cryptography to establish a shared secret key between the client and the server
- Record protocol
  - Use the secret symmetric key established in the handshake protocol to protect communication between the client and the server

Example of a Certificate

GeoTrust Global CA
↳ Google Internet Authority G2
↳ *.google.com

***.google.com**
Issued by: Google Internet Authority G2
Expires: Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time
✓ This certificate is valid

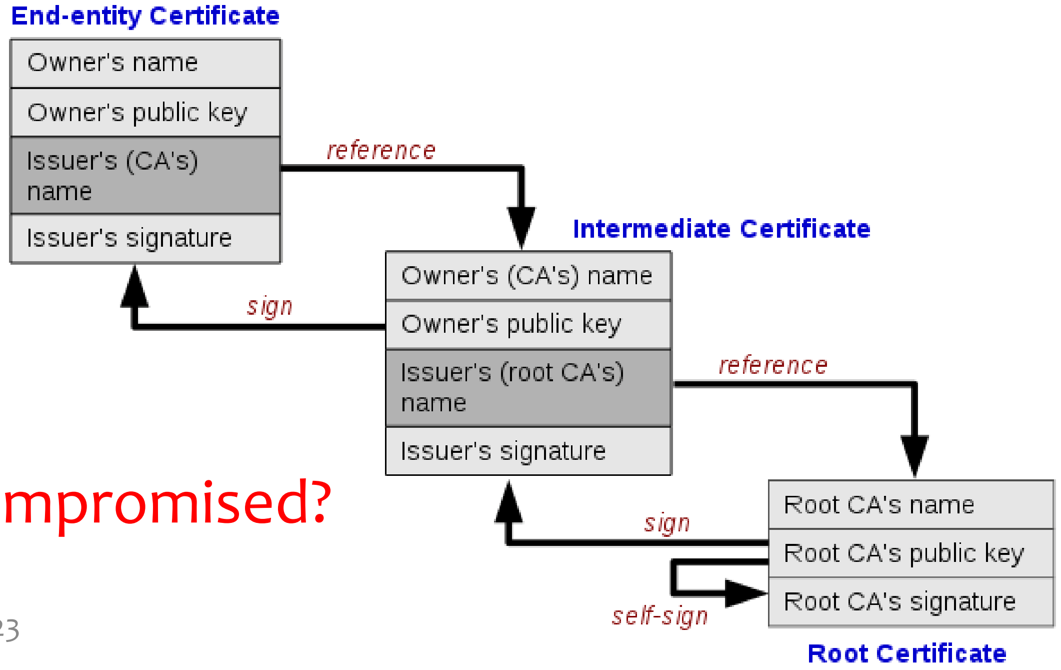
Details

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	*.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2
Serial Number	6082711391012222858
Version	3

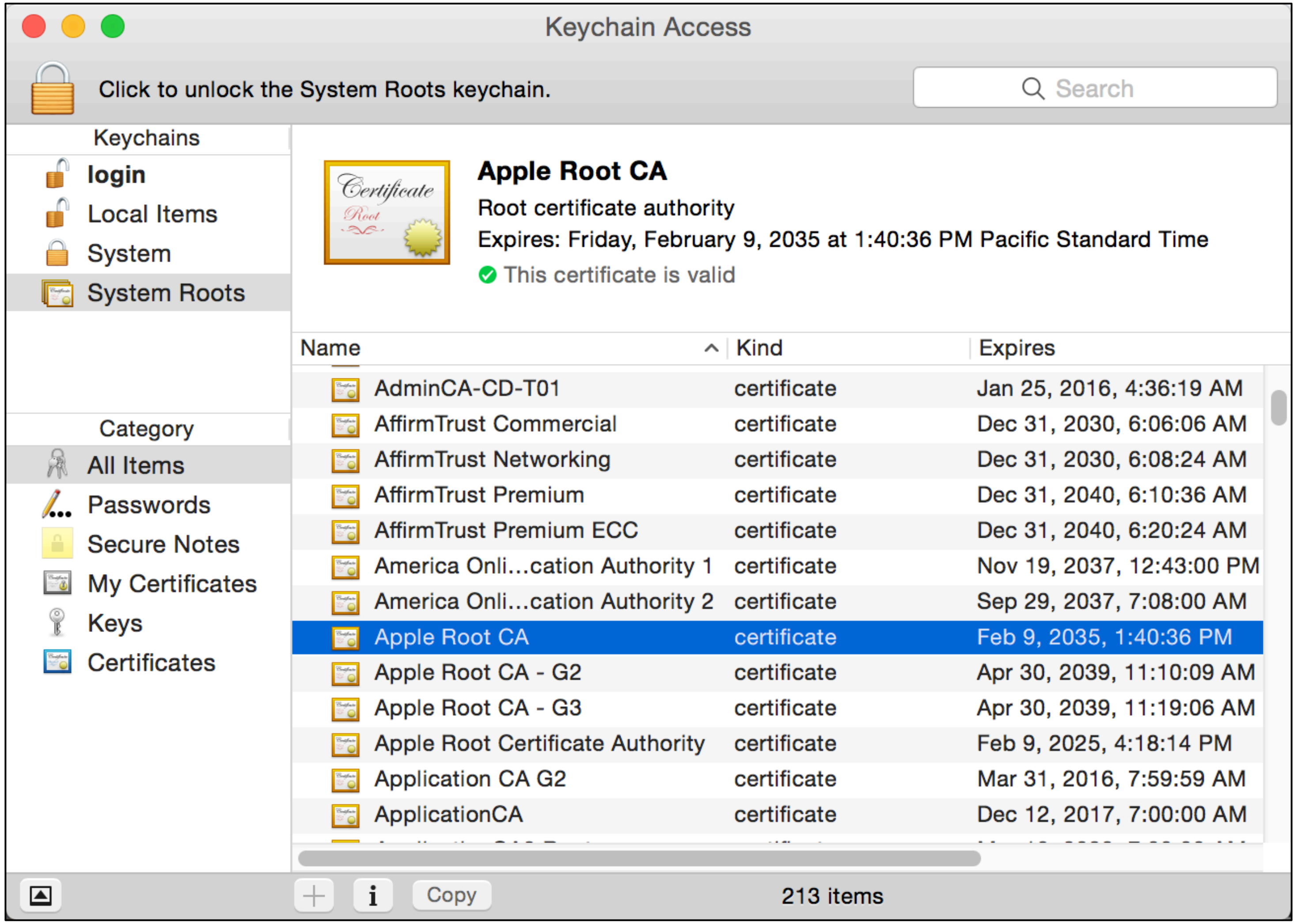
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, April 8, 2015 at 6:40:10 AM Pacific Daylight Time
Not Valid After	Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 CB DD C1 CE AC D6 20 …
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 34 8B 7D 64 5A 64 08 5B …

Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, use a trusted root authority (e.g., Verisign)
  - Everybody must know the root's public key
  - Instead of single cert, use a certificate chain
    - $sig_{Verisign}$("AnotherCA", $PK_{AnotherCA}$), $sig_{AnotherCA}$("Alice", PK_A)
  - Not shown in figure but important:
    - Signed as part of each cert is whether party is a CA or not
  - What happens if root authority is ever compromised?



Trusted(?) Certificate Authorities



Turtles All The Way Down…

The saying holds that the world is supported by a chain of increasingly large turtles. Beneath each turtle is yet another: it is "turtles all the way down".

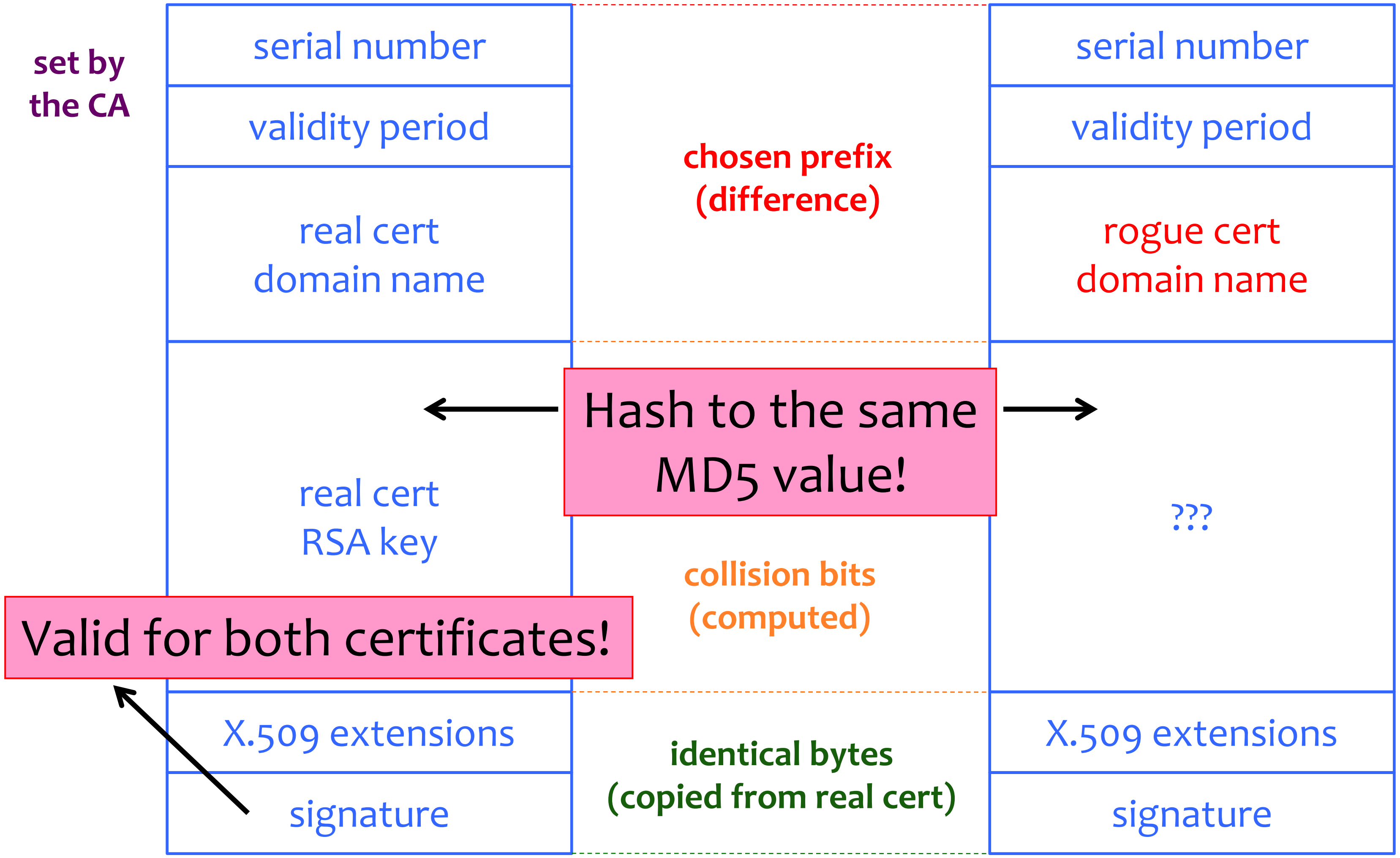
[Image from Wikipedia]

Many Challenges…

- Hash collisions
- Weak security at CAs
  - Allows attackers to issue rogue certificates
- Users don’t notice when attacks happen
  - We’ll talk more about this later in the course
- How do you revoke certificates?

[Sotirov et al. "Rogue Certificates"]

Colliding Certificates



DigiNotar Hacked by Black.Spook and Iranian Hackers Posted by Mikko @ 09:05 GMT | Comments

DigiNotar is a Dutch Certificate Authority. They sell SSL certificates.

Attacking CAs

Security of DigiNotar servers:

- All core certificate servers controlled by a single admin password (Prod@dm1n)
- Software on public-facing servers out of date, unpatched
- No anti-virus (could have detected attack)

Somehow, somebody managed to get a rogue SSL certificate from them on **July 10th, 2011**. This certificate was issued for domain name **.google.com**.

What can you do with such a certificate? Well, you can impersonate Google — assuming you can first reroute Internet traffic for google.com to you. This is something that can be done by a government or by a rogue ISP. Such a reroute would only affect users within that country or under that ISP.

Consequences

- Attacker needs to first divert users to an attacker-controlled site instead of Google, Yahoo, Skype, but then…
  - For example, use DNS to poison the mapping of mail.yahoo.com to an IP address
- … "authenticate" as the real site
- … decrypt all data sent by users
  - Email, phone conversations, Web browsing

More Rogue Certs

- In Jan 2013, a rogue *.google.com certificate was issued by an intermediate CA that gained its authority from the Turkish root CA TurkTrust
  - TurkTrust accidentally issued intermediate CA certs to customers who requested regular certificates
  - Ankara transit authority used its certificate to issue a fake *.google.com certificate in order to filter SSL traffic from its network
- This rogue *.google.com certificate was trusted by every browser in the world
- There are plenty more stories like this…

Certificate Revocation

- Revocation is very important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying their certification fee to this CA and CA no longer wishes to certify them
  - CA's private key has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

Certificate Revocation Mechanisms

- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates
    - Credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a "delta CRL" containing only updates
- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor

Attempt to Fix CA Problems:

Certificate Transparency

- **Problem:** browsers will think nothing is wrong with a rogue certificate until revoked
- **Goal:** make it impossible for a CA to issue a bad certificate for a domain *without the owner of that domain knowing*
- **Approach:** auditable certificate logs
  – Certificates published in public logs
  – Public logs checked for unexpected certificates

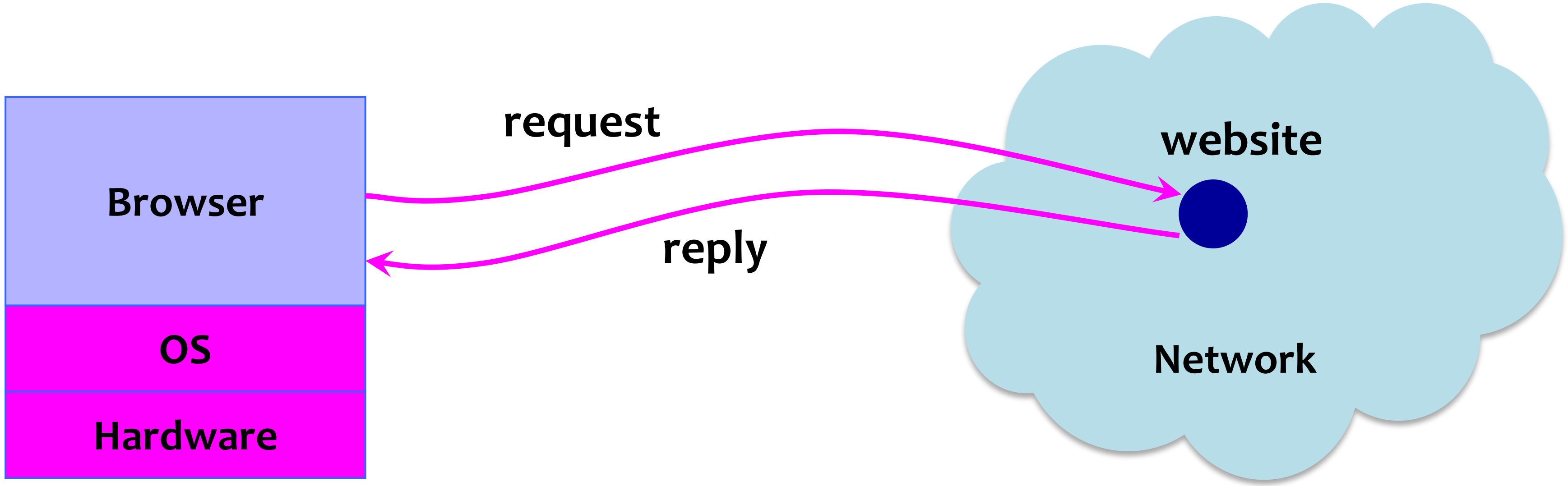
www.certificate-transparency.org

Attempt to Fix CA Problems:

Certificate Pinning

- Trust on first access: tells browser how to act on subsequent connections
- HPKP – HTTP Public Key Pinning
  - Use these keys!
  - HTTP response header field "`Public-Key-Pins`"
- HSTS – HTTP Strict Transport Security
  - Only access server via HTTPS
  - HTTP response header field "`Strict-Transport-Security`"

Big Picture: Browser and Network



Where Does the Attacker Live?