

CSE 484 : Computer Security and Privacy

# Cryptography with Hints Toward Web Security

Spring 2021

Tadayoshi Kohno

yoshi@cs

Thanks to Franz Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, David Kohlbrenner, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

# Admin

- HW 2 out, due in two weeks
- Lab 1 due today
  
- Aaron Alva (FTC) next Wednesday – not recorded

# 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  $\varphi(n)=(p-1)(q-1)$
- Choose small  $e$ , relatively prime to  $\varphi(n)$ 
  - Typically,  $e=3$  or  $e=2^{16}+1=65537$

- Compute unique  $d$  such that  $ed \equiv 1 \pmod{\varphi(n)}$ 
  - Modular inverse:  $d \equiv e^{-1} \pmod{\varphi(n)}$

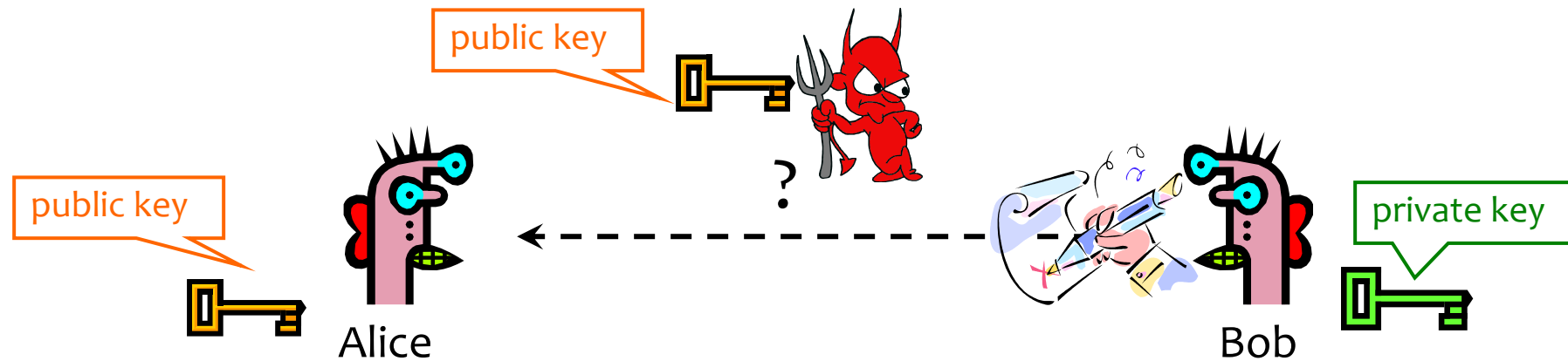
How to compute?

- Public key =  $(e, n)$ ; private key =  $(d, n)$

- Encryption of  $m$ :  $c = m^e \pmod n$

- Decryption of  $c$ :  $c^d \pmod n = (m^e)^d \pmod n = m$

# 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

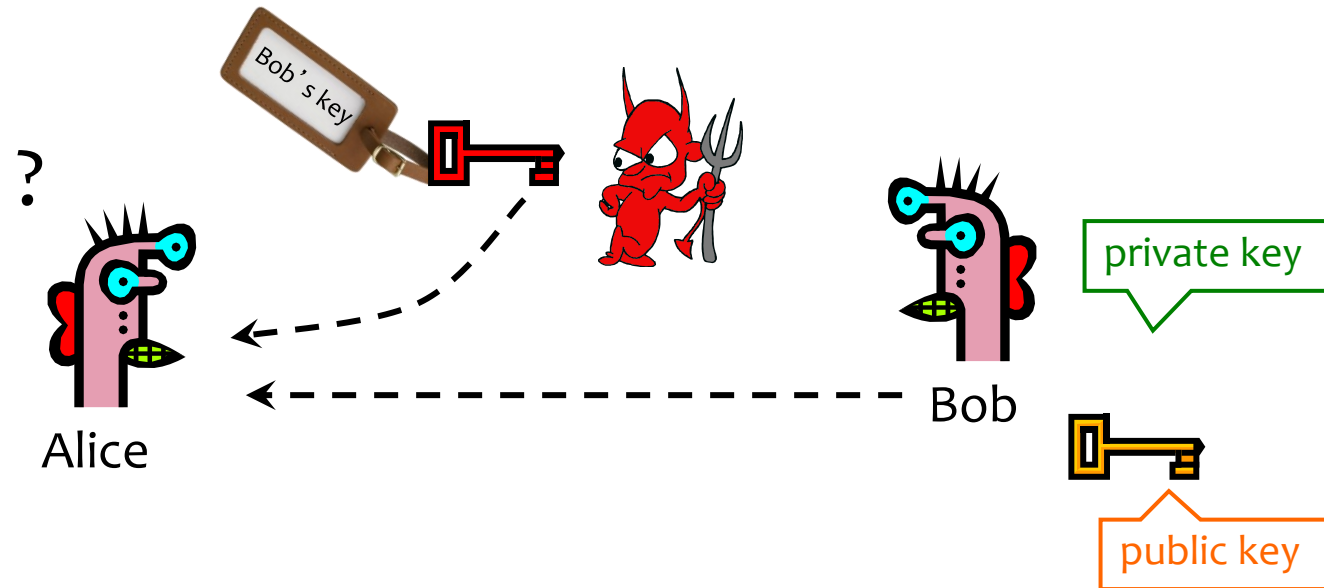
# DSS Signatures

- 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:  $x$
- Each signing operation picks a new random value, to use during signing. Security breaks if two messages are signed with that same value.
- 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

- If quantum computer 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

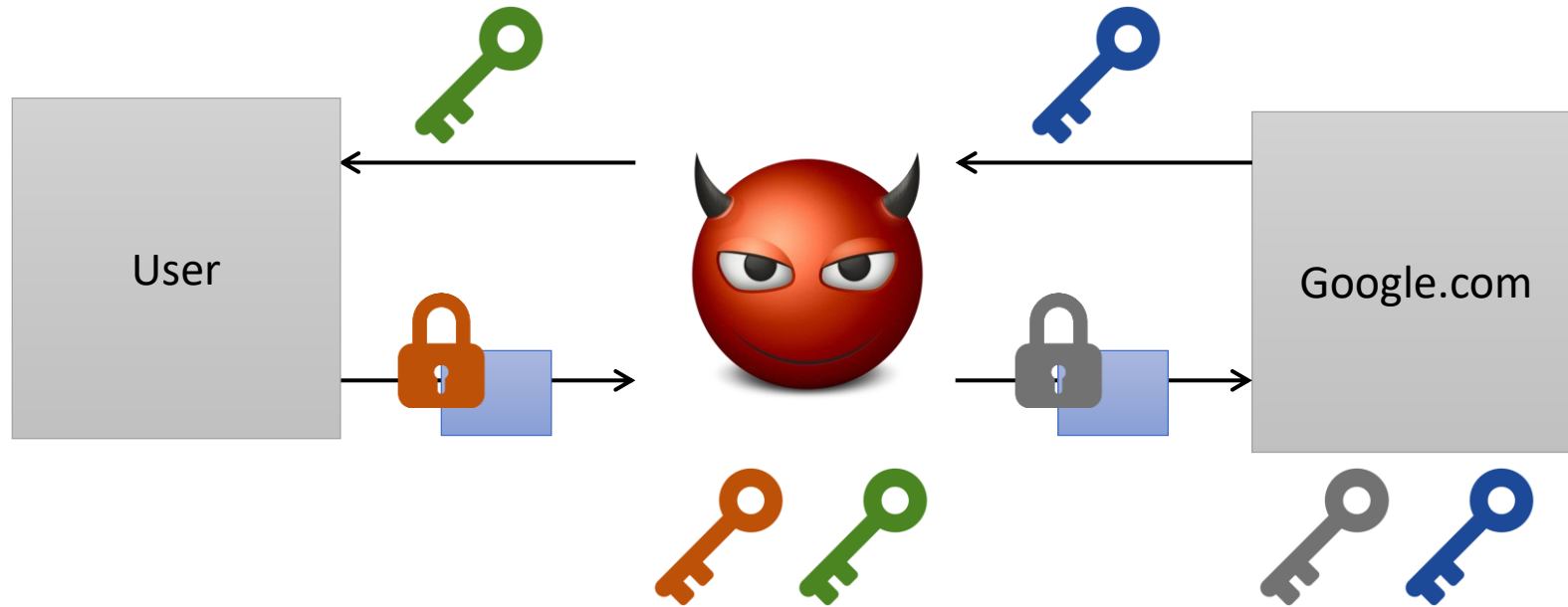
# Authenticity of Public Keys



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



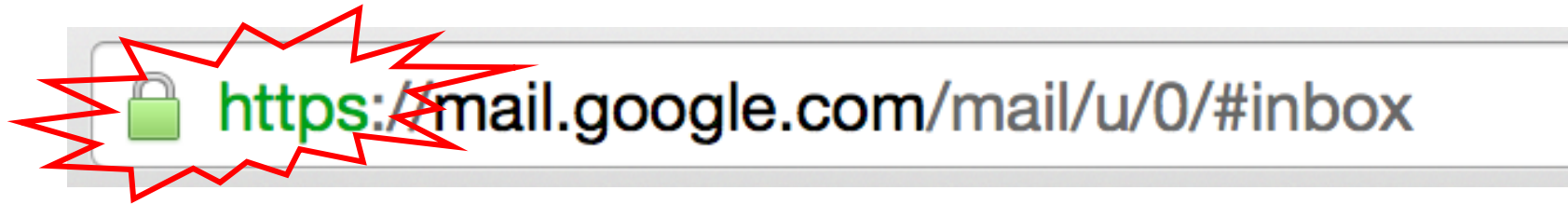
# Threat: Person-in-the Middle



# Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - $\text{sig}_{\text{CA}}(\text{"Bob"}, \text{PK}_B)$
    - Additional information often signed as well (e.g., expiration date)
- Common approach: certificate authority (CA)
  - Single agency responsible for certifying public keys
  - After generating a private/public key pair, user proves their 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...

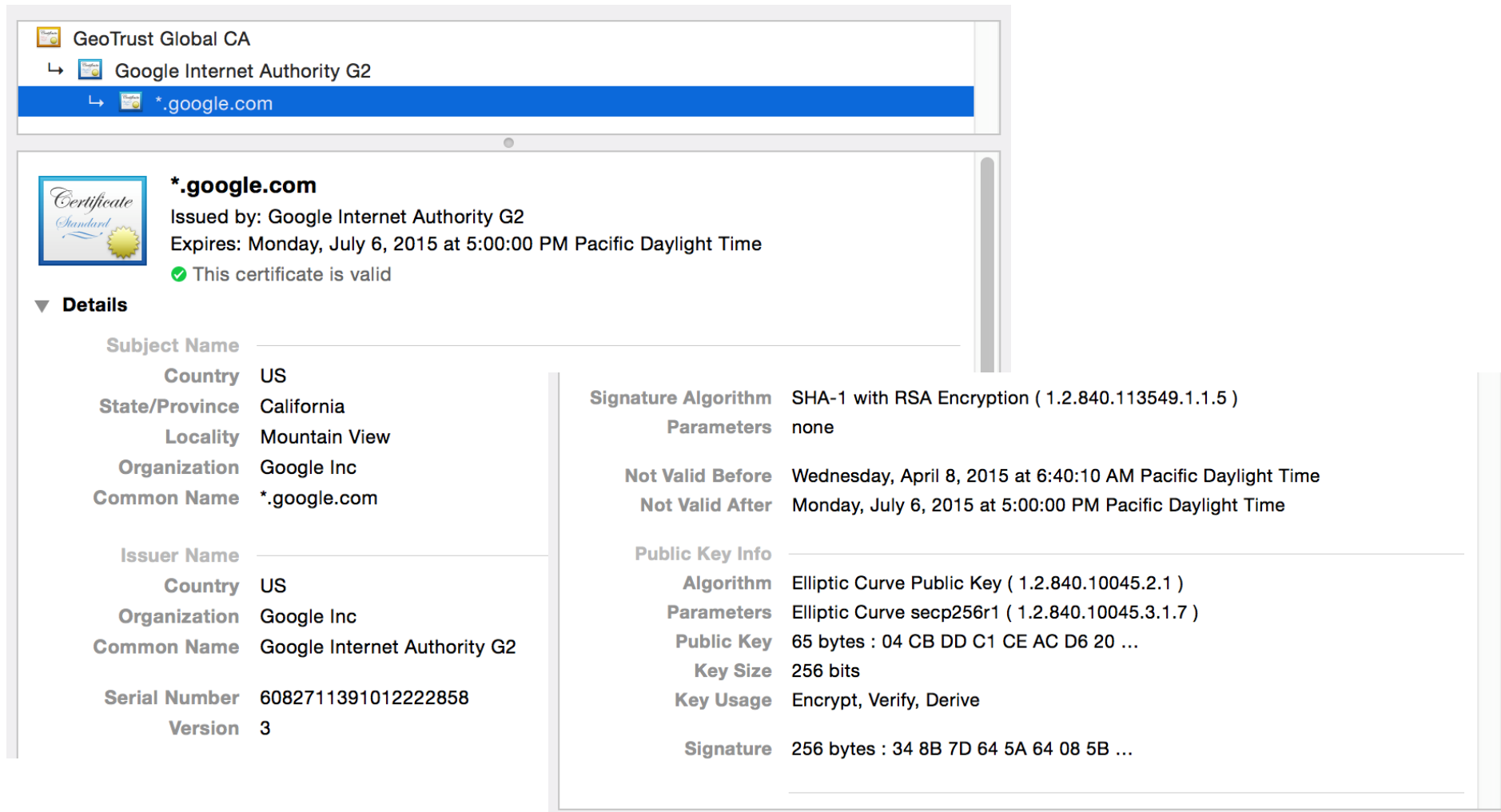


**SSL/TLS:** Encryption & authentication for connections

# 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



The screenshot shows a browser's certificate viewer interface. At the top, the certificate path is displayed: GeoTrust Global CA, Google Internet Authority G2, and \*.google.com. The main section features a 'Certificate Standard' icon, the domain name '\*.google.com', and the issuer 'Google Internet Authority G2'. It also shows the expiration date and time: 'Expires: Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time'. A green checkmark indicates 'This certificate is valid'. Below this, a 'Details' section is expanded, showing two columns of information: 'Subject Name' and 'Issuer Name'. The 'Subject Name' column lists: Country (US), State/Province (California), Locality (Mountain View), Organization (Google Inc), and Common Name (\*.google.com). The 'Issuer Name' column lists: Country (US), Organization (Google Inc), and Common Name (Google Internet Authority G2). To the right of these details, another section provides technical specifications: 'Signature Algorithm' (SHA-1 with RSA Encryption), '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, Parameters: Elliptic Curve secp256r1, Public Key: 65 bytes, Key Size: 256 bits, Key Usage: Encrypt, Verify, Derive), and 'Signature' (256 bytes).

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

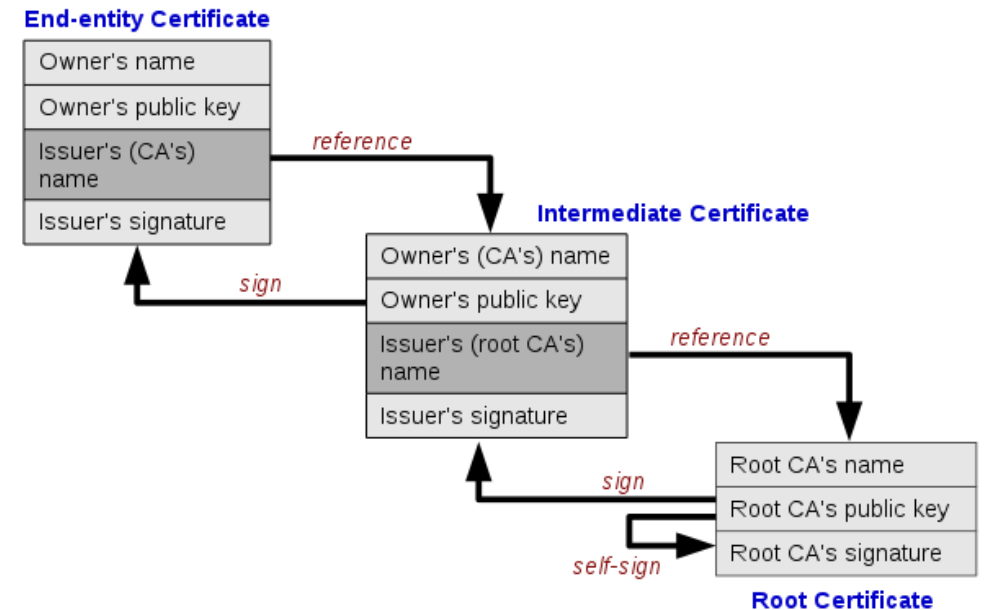
▼ **Details**

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

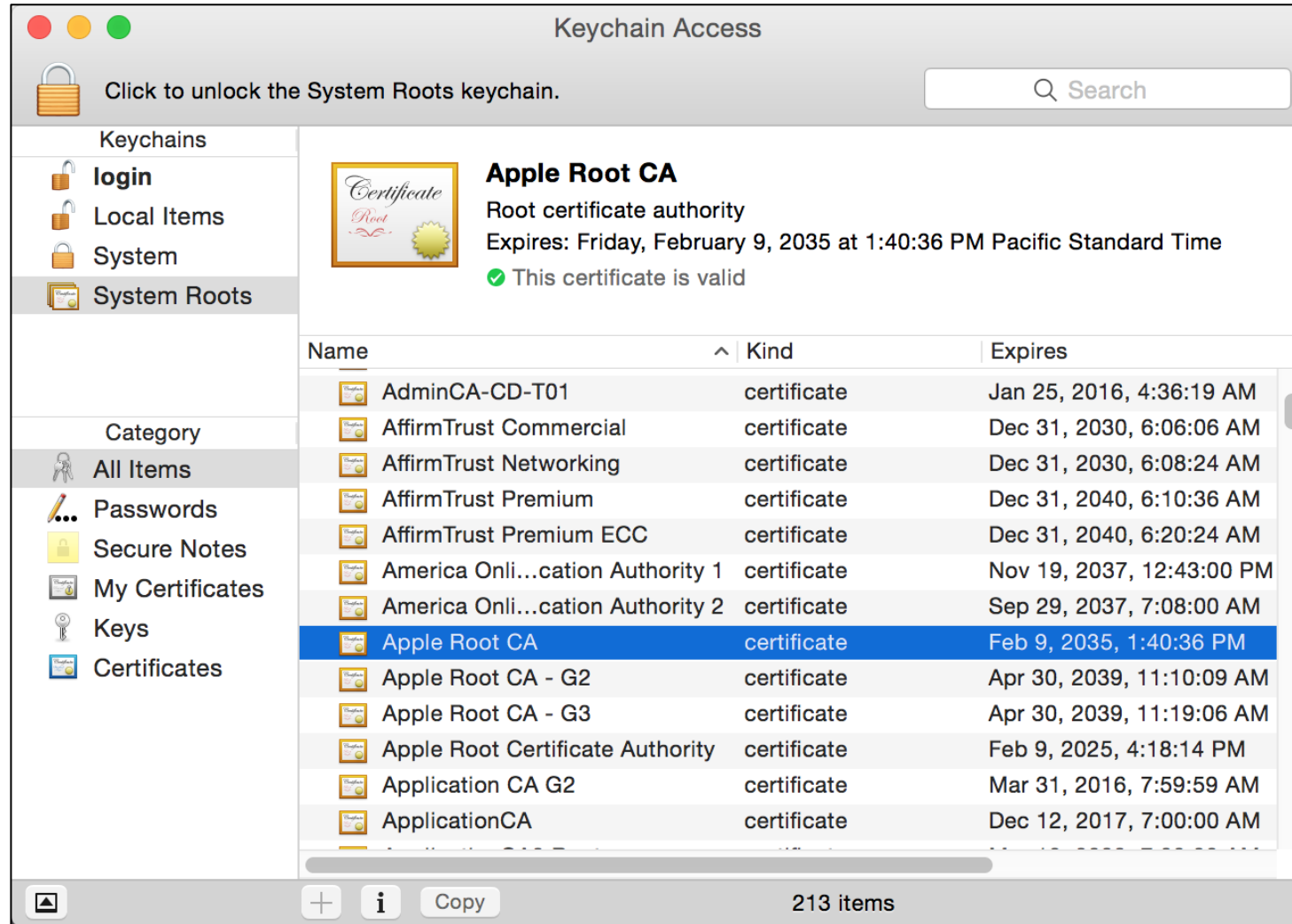
<b>Signature Algorithm</b>	SHA-1 with RSA Encryption ( 1.2.840.113549.1.1.5 )
<b>Parameters</b>	none
<b>Not Valid Before</b>	Wednesday, April 8, 2015 at 6:40:10 AM Pacific Daylight Time
<b>Not Valid After</b>	Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time
<b>Public Key Info</b>	
<b>Algorithm</b>	Elliptic Curve Public Key ( 1.2.840.10045.2.1 )
<b>Parameters</b>	Elliptic Curve secp256r1 ( 1.2.840.10045.3.1.7 )
<b>Public Key</b>	65 bytes : 04 CB DD C1 CE AC D6 20 ...
<b>Key Size</b>	256 bits
<b>Key Usage</b>	Encrypt, Verify, Derive
<b>Signature</b>	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**
    - $\text{sig}_{\text{Verisign}}(\text{"AnotherCA"}, \text{PK}_{\text{AnotherCA}})$ ,  
 $\text{sig}_{\text{AnotherCA}}(\text{"Alice"}, \text{PK}_A)$
  - 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



Keychain Access

Click to unlock the System Roots keychain.

Keychains

- login
- Local Items
- System
- System Roots**

Category

- All Items
- Passwords
- Secure Notes
- My Certificates
- Keys
- Certificates

**Apple Root CA**  
Root certificate authority  
Expires: Friday, February 9, 2035 at 1:40:36 PM Pacific Standard Time  
✔ This certificate is valid

Name	Kind	Expires
AdminCA-CD-T01	certificate	Jan 25, 2016, 4:36:19 AM
AffirmTrust Commercial	certificate	Dec 31, 2030, 6:06:06 AM
AffirmTrust Networking	certificate	Dec 31, 2030, 6:08:24 AM
AffirmTrust Premium	certificate	Dec 31, 2040, 6:10:36 AM
AffirmTrust Premium ECC	certificate	Dec 31, 2040, 6:20:24 AM
America Onli...cation Authority 1	certificate	Nov 19, 2037, 12:43:00 PM
America Onli...cation Authority 2	certificate	Sep 29, 2037, 7:08:00 AM
<b>Apple Root CA</b>	certificate	<b>Feb 9, 2035, 1:40:36 PM</b>
Apple Root CA - G2	certificate	Apr 30, 2039, 11:10:09 AM
Apple Root CA - G3	certificate	Apr 30, 2039, 11:19:06 AM
Apple Root Certificate Authority	certificate	Feb 9, 2025, 4:18:14 PM
Application CA G2	certificate	Mar 31, 2016, 7:59:59 AM
ApplicationCA	certificate	Dec 12, 2017, 7:00:00 AM

213 items

# 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".

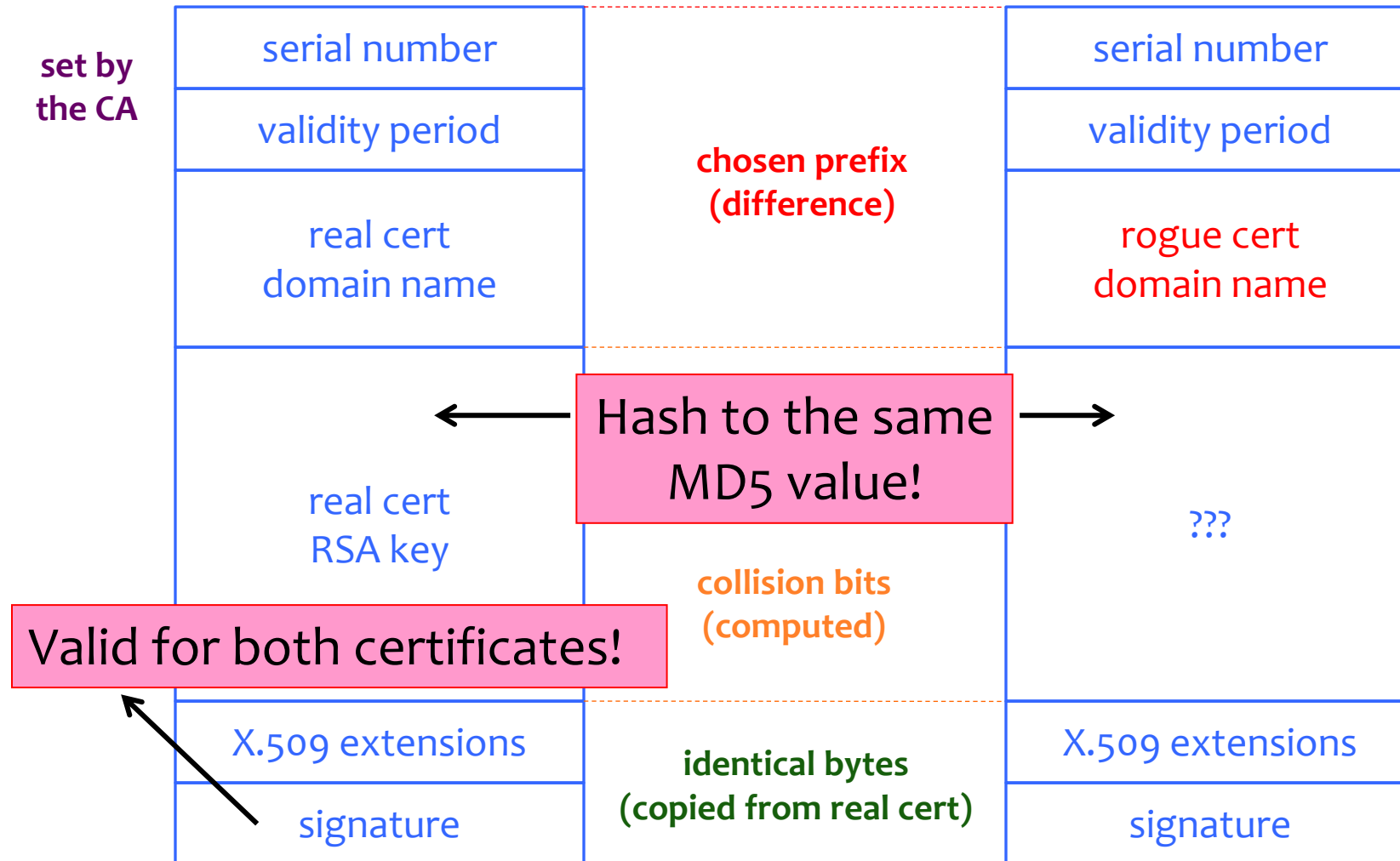
[Image from Wikipedia]



# 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?

# Colliding Certificates



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



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.

## 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)

# 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

# Bad CAs

- **DarkMatter** (<https://groups.google.com/g/mozilla.dev.security.policy/c/nnLVNfqgz7g/m/TseYqDzaDAAJ> and [https://bugzilla.mozilla.org/show\\_bug.cgi?id=1427262](https://bugzilla.mozilla.org/show_bug.cgi?id=1427262))
  - Security company wanted to get CA status
  - Questionable practices
- **Symantec!** ([https://wiki.mozilla.org/CA:Symantec\\_Issues](https://wiki.mozilla.org/CA:Symantec_Issues))
  - Major company, regular participant in standards
  - Poor practices, mismanagement 2013-2017
  - CA distrusted in Oct 2018
- Recall: Turtles all the way down. How can we trust the CAs? What happens if we can't?