

CSE 484 / CSE M 584: Computer Security and Privacy

Certificate Authorities and SSL/TLS/HTTPS

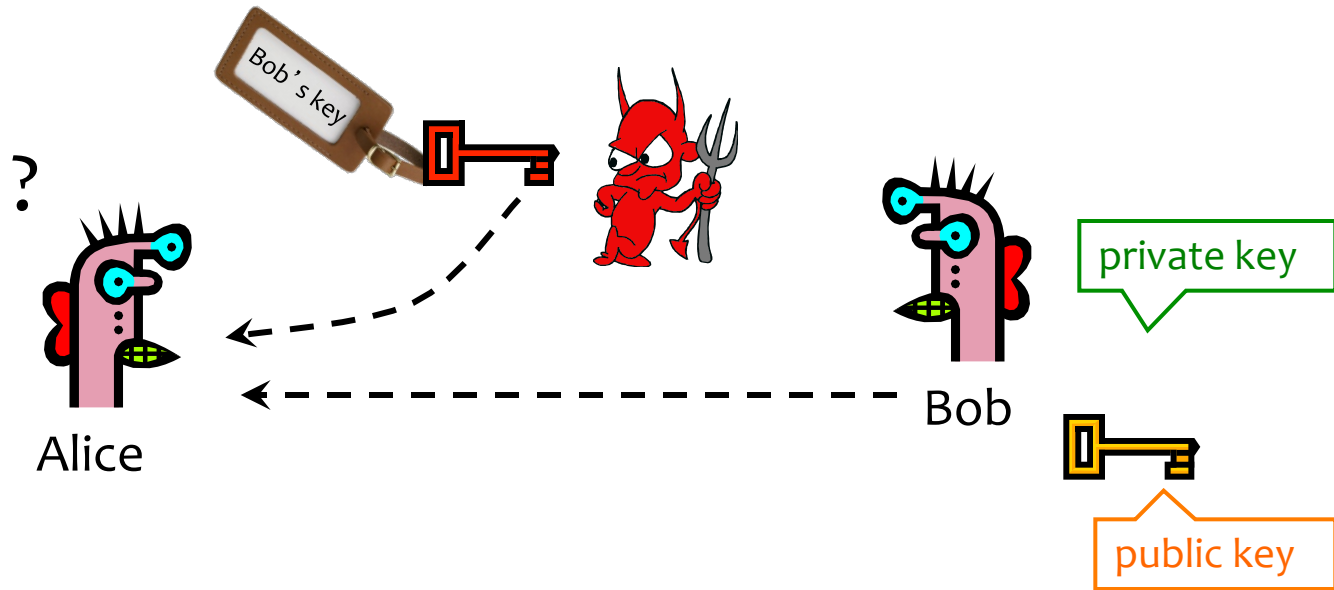
Fall 2016

Ada (Adam) Lerner

lerner@cs.washington.edu

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

Authenticity of Public Keys



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

Announcements

- Lab 2 (web security) will be coming out next Tuesday

RSA decryption

- Based on feedback and interest, not in lecture
- I've added a slide to lecture 12's slides which explains it (it's slide 18)

RSA decryption

- On the interest scale of 1-5...
 - ... someone answered 0
 - ... someone answered 6
 - ... someone answered π
 - ... someone answered 2^5

Security mindset anecdote – Mining Your Ps and Qs

- A 2012 study titled
“Mining your Ps and Qs: Detection of
Widespread Weak Keys in Network Devices”

Scanned the entire internet to look for weak
public keys

Mining Your Ps and Qs

- They were able to determine the RSA **private** key for 0.5% of HTTPS servers and 0.03% of SSH servers
- How? Insufficient randomness. 0.5% of keys shared a p or q with at least one other key (but not both).

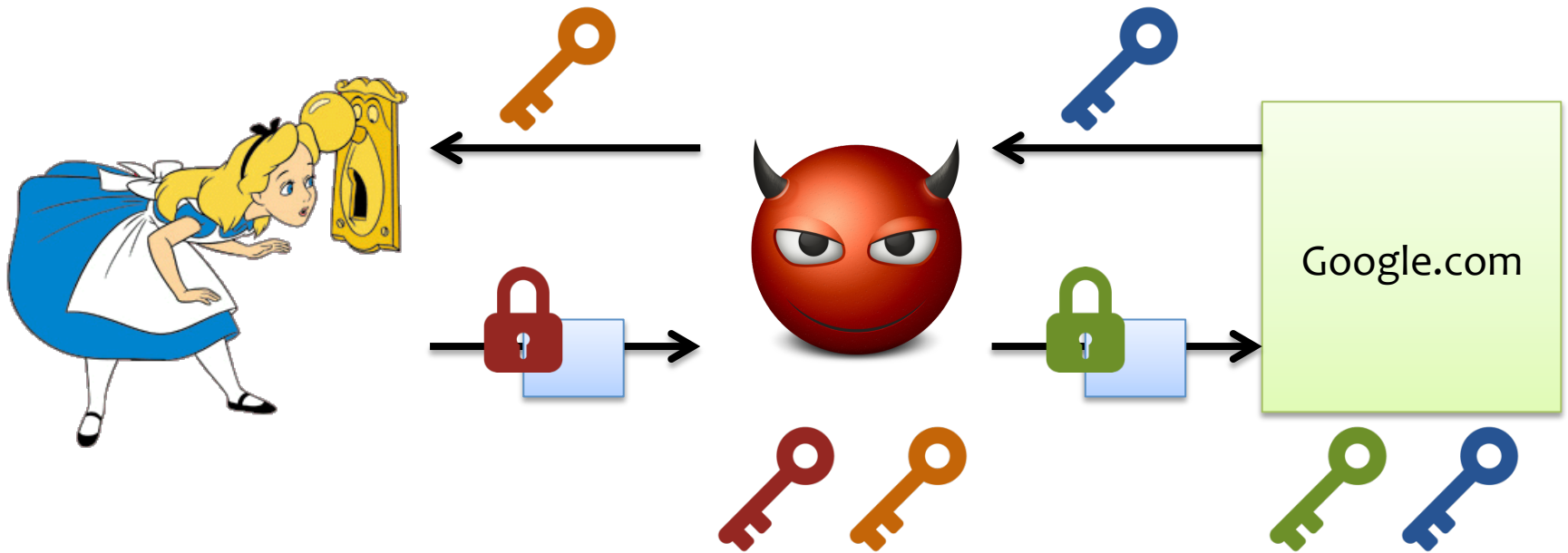
RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Key generation:
 - Generate random large primes p, q
 - Say, 1024 bits each
 - Compute $n=pq$ and $\varphi(n)=(p-1)(q-1)$
 - Choose small e , relatively prime to $\varphi(n)$
 - Typically, $e=2^{16}+1=65537$
 - Compute unique d such that $ed = 1 \pmod{\varphi(n)}$
 - Modular inverse: $d = e^{-1} \pmod{\varphi(n)}$
 - Public key = (e,n) ; private key = (d,n)
- Encryption of m : $c = m^e \pmod n$
- Decryption of c : $c^d \pmod n = (m^e)^d \pmod n = m$

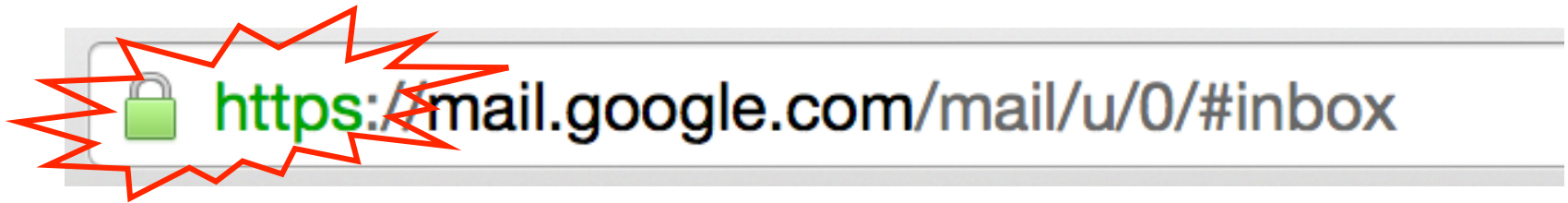
Certificates

- Public-key certificate
 - Signed statement specifying the key and identity
 - $\text{sig}_{CA}(\text{“Bob”}, PK_B)$

Threat: Man-In-The-Middle (MITM)



You encounter this every day...



SSL/TLS: Encryption & authentication for connections

(More on this later!)

Certificate Authority

- Trusted organization that **verifies** who owns what keys out of band and tells everyone else whose keys are whose

Strawman CA design

1. You browse to www.cs.washington.edu
2. www.cs.washington.edu sends its key K
3. Your browser asks a trusted CA: “hey, key K the right key for UW CSE?”
4. CA replies “yes” or “no”

Why is this a bad idea? (Q1)

Real CA design

- Think of a certificate as a cryptographically hard-to-forge piece of ID



<proof that I'm UWCSE and
 PK_{UWCSE} is my key>

www.cs.washington.edu

Certificate
authority
(e.g., Verisign
or Let's
Encrypt)



$sig_{CA}(\text{"UWCSE"}, PK_{UWCSE})$

Example Certificate

The screenshot shows a Windows Certificate dialog box. The top section is a tree view of certificates:

- AddTrust External CA Root
 - ↳ USERTrust RSA Certification Authority
 - ↳ InCommon RSA Server CA
 - ↳ ***.cs.washington.edu**

The bottom section displays details for the selected certificate:

-  ***.cs.washington.edu**
- Issued by: InCommon RSA Server CA
- Expires: Sunday, April 15, 2018 at 4:59:59 PM Pacific Daylight Time
-  This certificate is valid

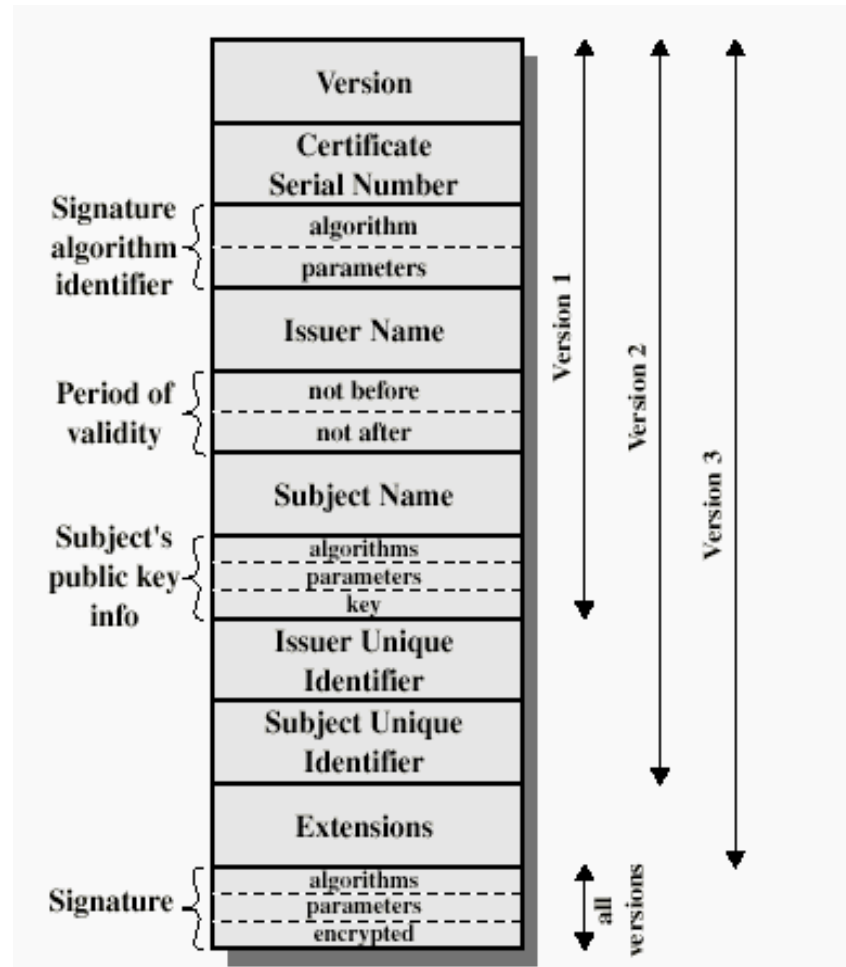
At the bottom left, there is a **Details** link with a right-pointing triangle. At the bottom right, there is an **OK** button.

Example Certificate

Public Key Info

Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : C8 30 FE 26 A7 36 CB 6D ...
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes : 40 3B FA E9 66 A3 4B 99 ...

X.509 Certificate



Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, one or more trusted **root authorities**
 - Everybody must know the public key for verifying root authority's signatures
- CAs delegate to other authorities
 - What happens if root authority is ever compromised?

Hierarchical Approach

- Single CA certifying every public key is impractical
- Instead, use a trusted **root authority**
 - For example, Verisign
 - Everybody must know the public key for verifying root authority's signatures
- Root authority signs certificates for lower-level authorities, lower-level authorities sign certificates for individual networks, and so on
 - Instead of a single certificate, use a **certificate chain**
 - $\text{sig}_{\text{Verisign}}(\text{"AnotherCA"}, \text{PK}_{\text{AnotherCA}}), \text{sig}_{\text{AnotherCA}}(\text{"Alice"}, \text{PK}_A)$
 - What happens if root authority is ever compromised?

Many Challenges...

- CAs make serious mistakes
 - Bad security practices, bad operational practices
- Revocation is hard...
- Users don't notice when attacks happen
 - We'll talk more about this later



<https://mail.google.com/mail/u/0/#inbox>

Mining Your Ps and Qs

- Apache ships with a “snake-oil” certificate -- an example certificate for demonstrating how to set up HTTPS
- A study found >85k hosts on the internet (0.66% of all TLS hosts on the internet) actively using these keys!
- 22 hosts had certificates using these keys **THAT WERE SIGNED BY A CA!**

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



Attacking CAs

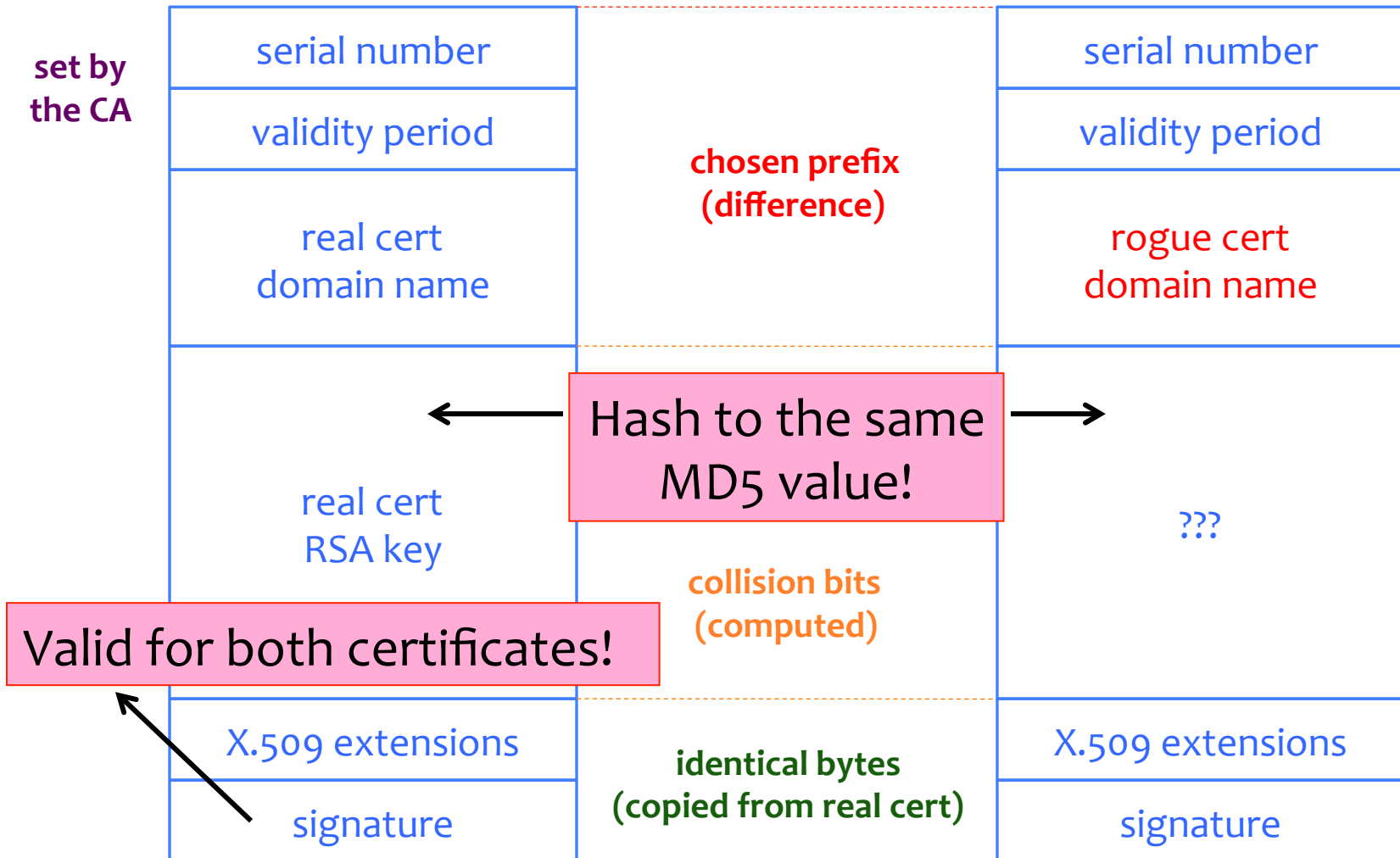
Security of DigiNotar servers:

- 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.

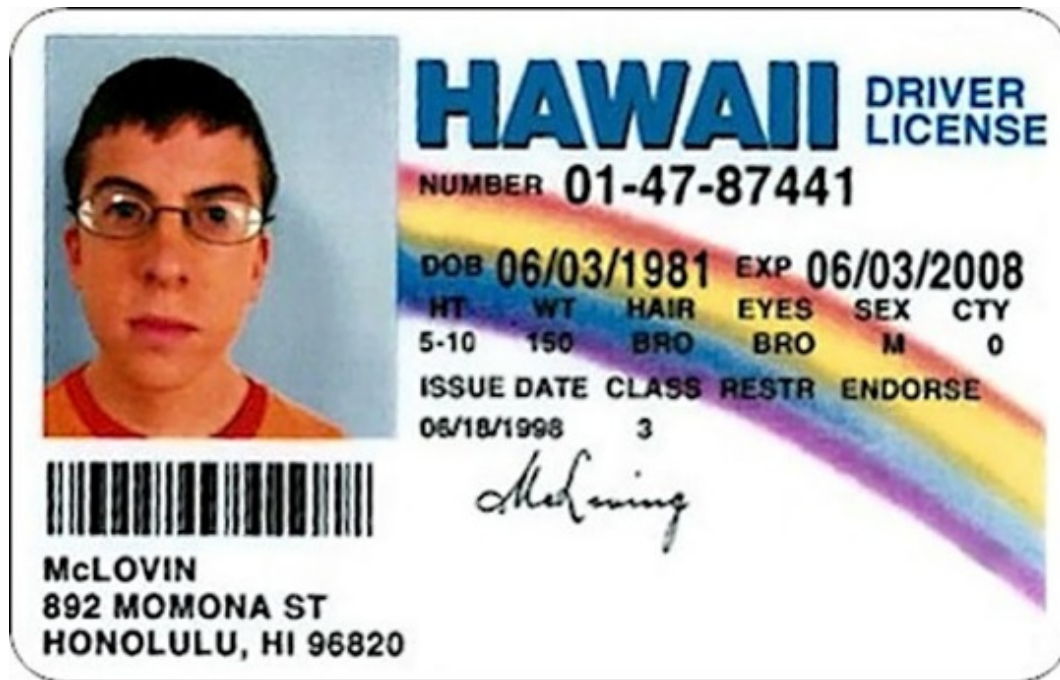
Colliding Certificates



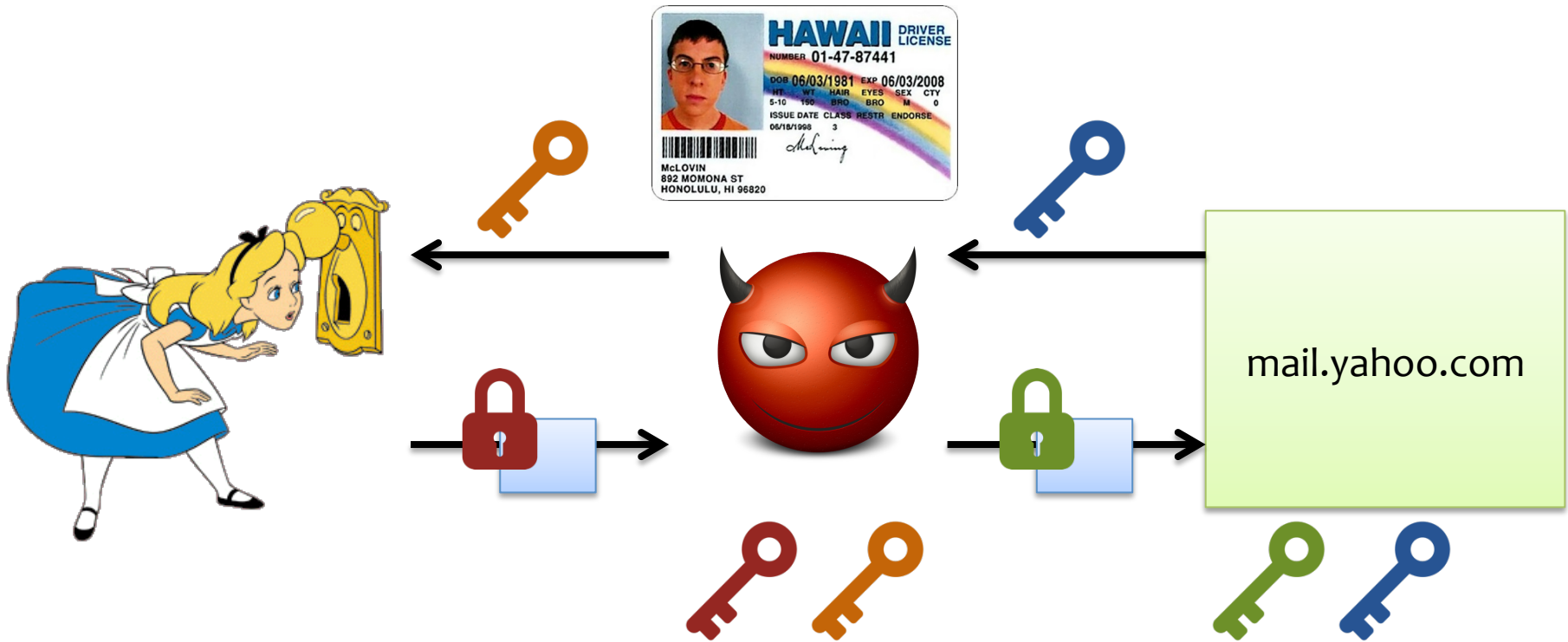
Consequences of Hacking a CA

- Attacker makes themselves a fake certificate for a site (say, mail.yahoo.com):

$\text{fakeCert} = \text{sig}_{\text{CA}}(\text{"Yahoo"}, \text{<attacker's key>})$



Q2: Man-In-The-Middle (MITM)



Consequences of Hacking a CA

- Attacker makes themselves a fake certificate for a site (say, mail.yahoo.com):
$$\text{fakeCert} = \text{sig}_{\text{CA}}(\text{“Yahoo”}, \text{<attacker’s key>})$$
- An attacker can pretend to be any real site
 - 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

Many Challenges...

- CAs make serious mistakes
 - Bad security practices, bad operational practices
- Revocation is hard...
- Users don't notice when attacks happen
 - We'll talk more about this later



<https://mail.google.com/mail/u/0/#inbox>

Certificate Revocation (Q3)

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 him
 - 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


Keybase

- Basic idea:
 - Rely on existing trust of a person's ownership of other accounts (e.g., Twitter, GitHub, website)
 - Each user publishes signed proofs to their linked account



<https://keybase.io/>

SSL/TLS

 <https://mail.google.com/mail/u/0/#inbox>

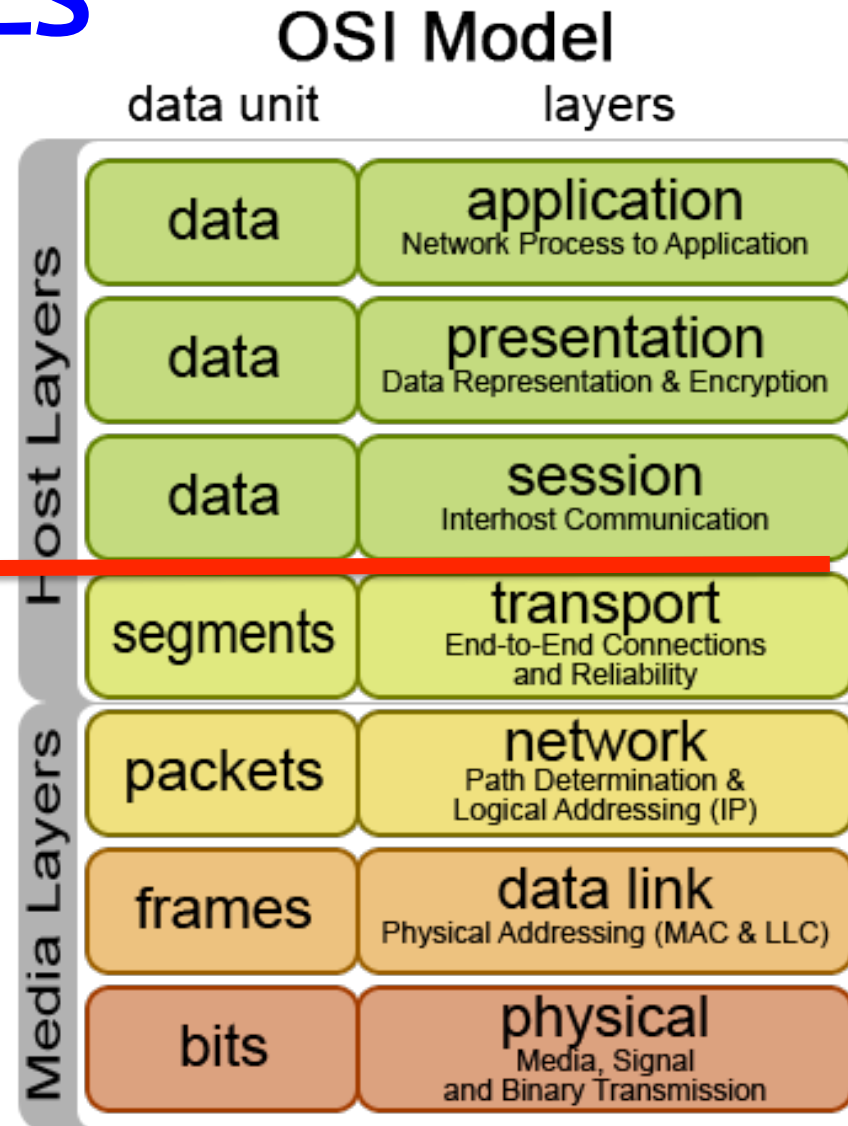
- Secure Sockets Layer and Transport Layer Security
 - Same protocol, new version (TLS is current)
- De facto standard for Internet security
 - “The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications”
- Deployed in every Web browser; also VoIP, payment systems, distributed systems, etc.

SSL/TLS

- TLS is typically used on top of a TCP connection

TLS

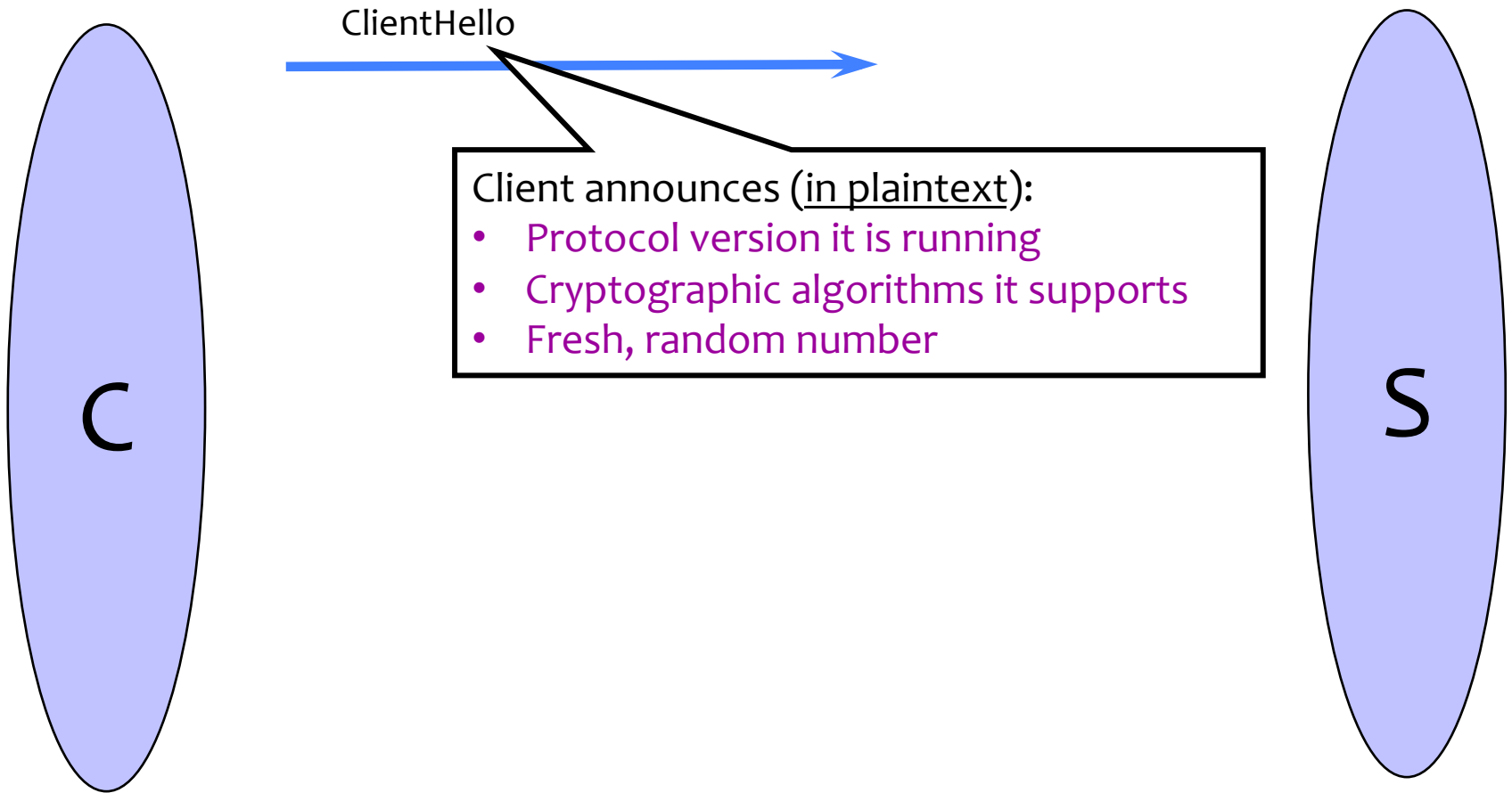
- Can be used over other transport protocols



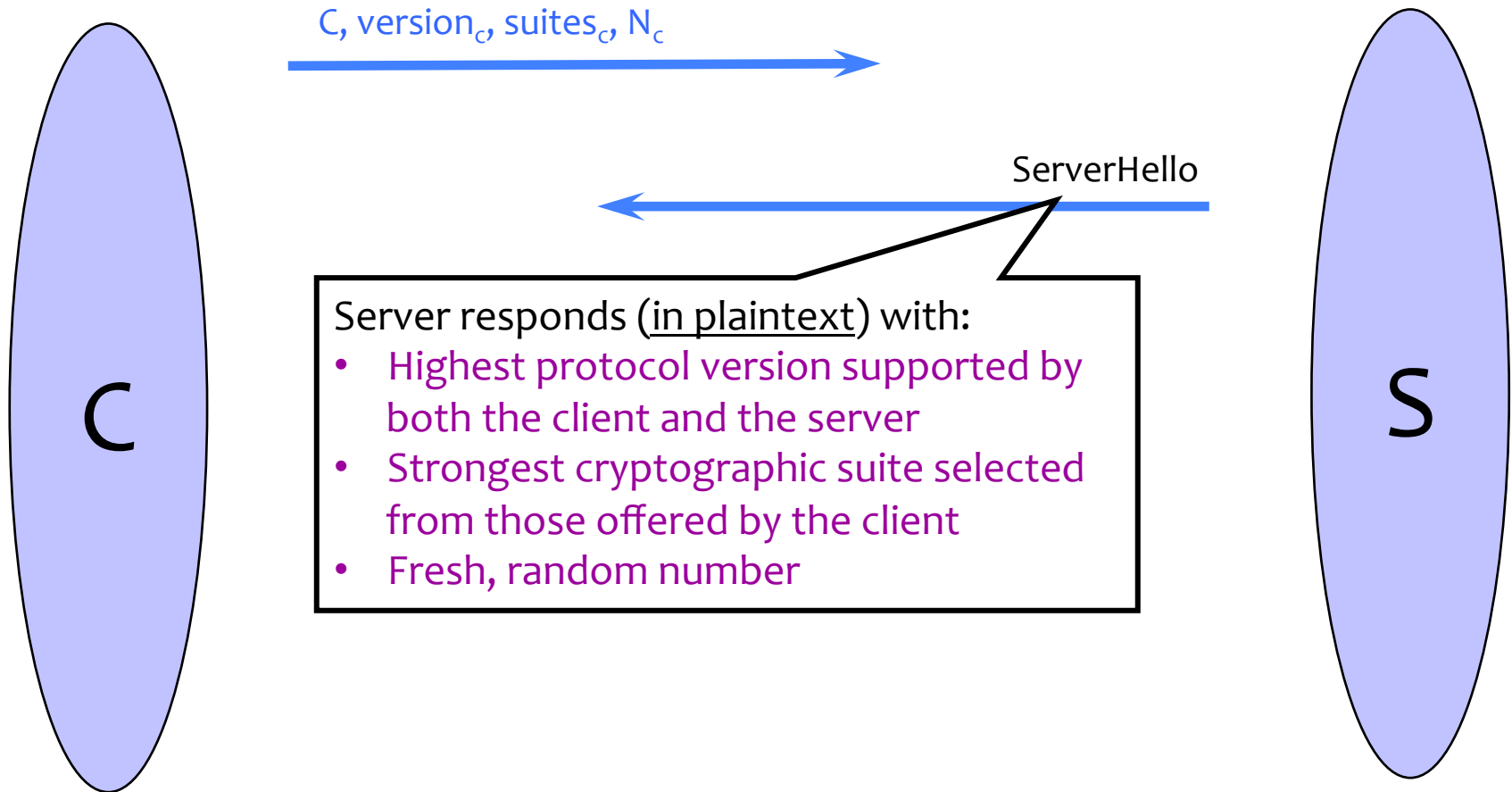
TLS Basics

- 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

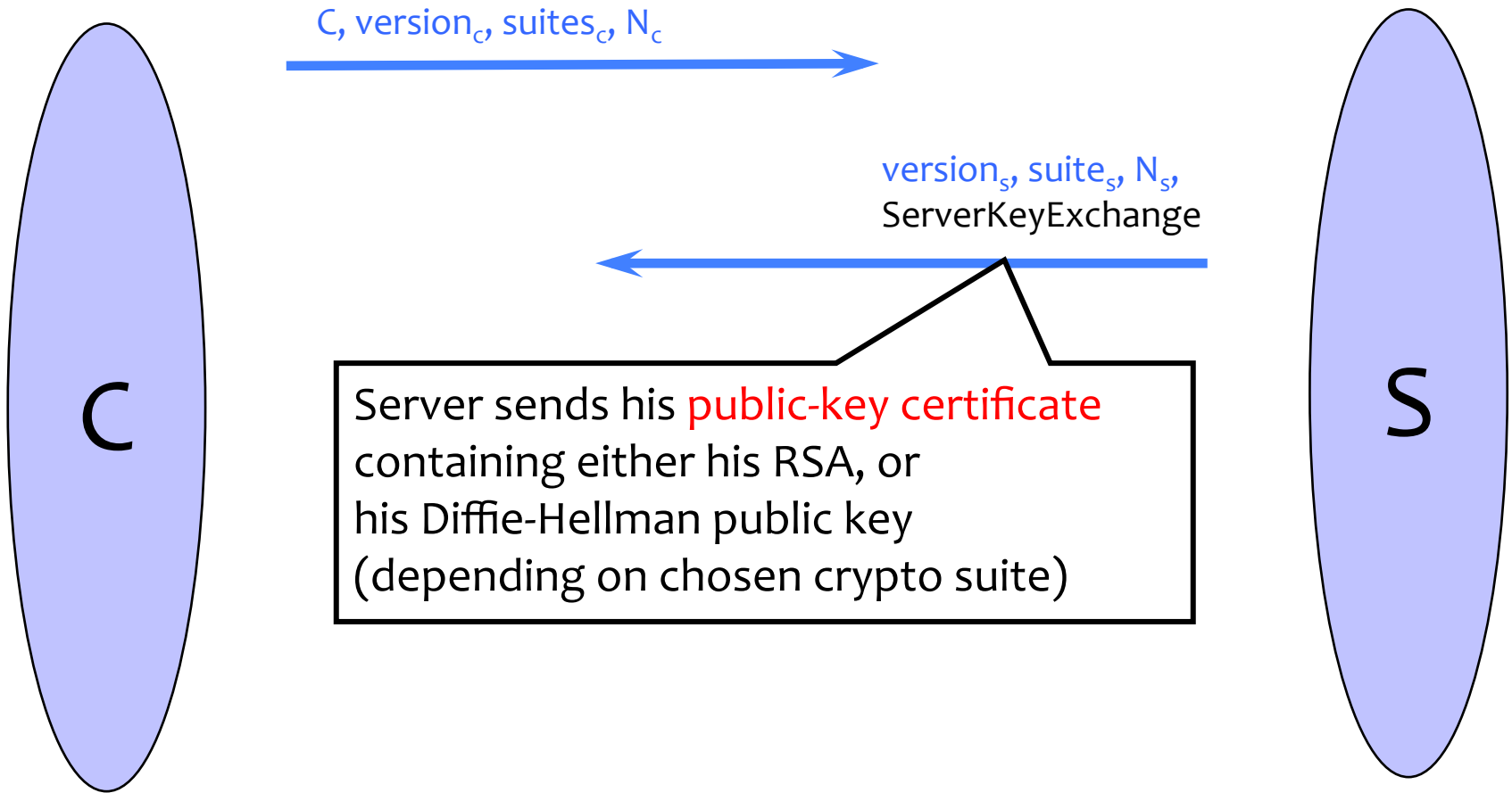
Basic Handshake Protocol



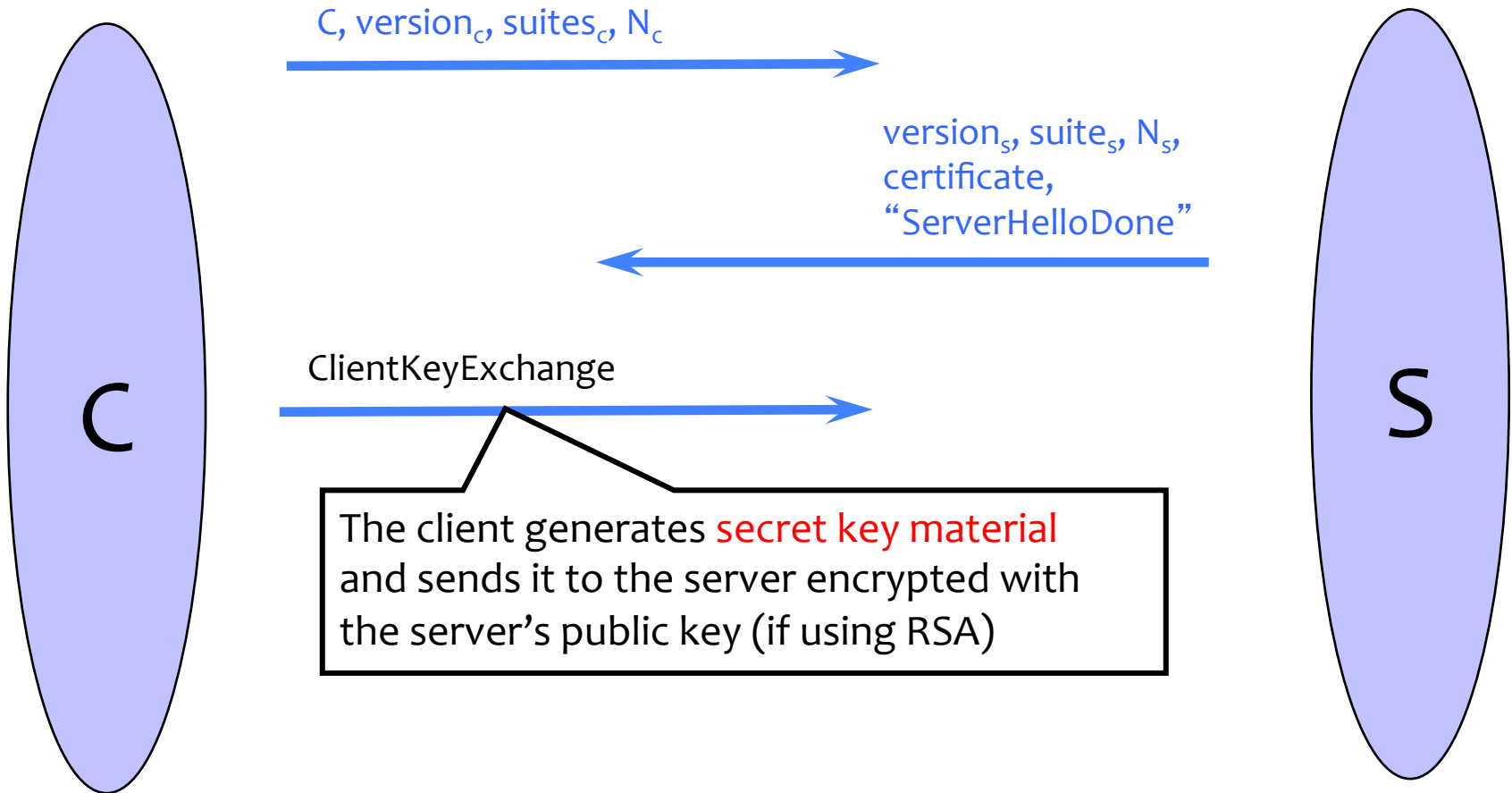
Basic Handshake Protocol



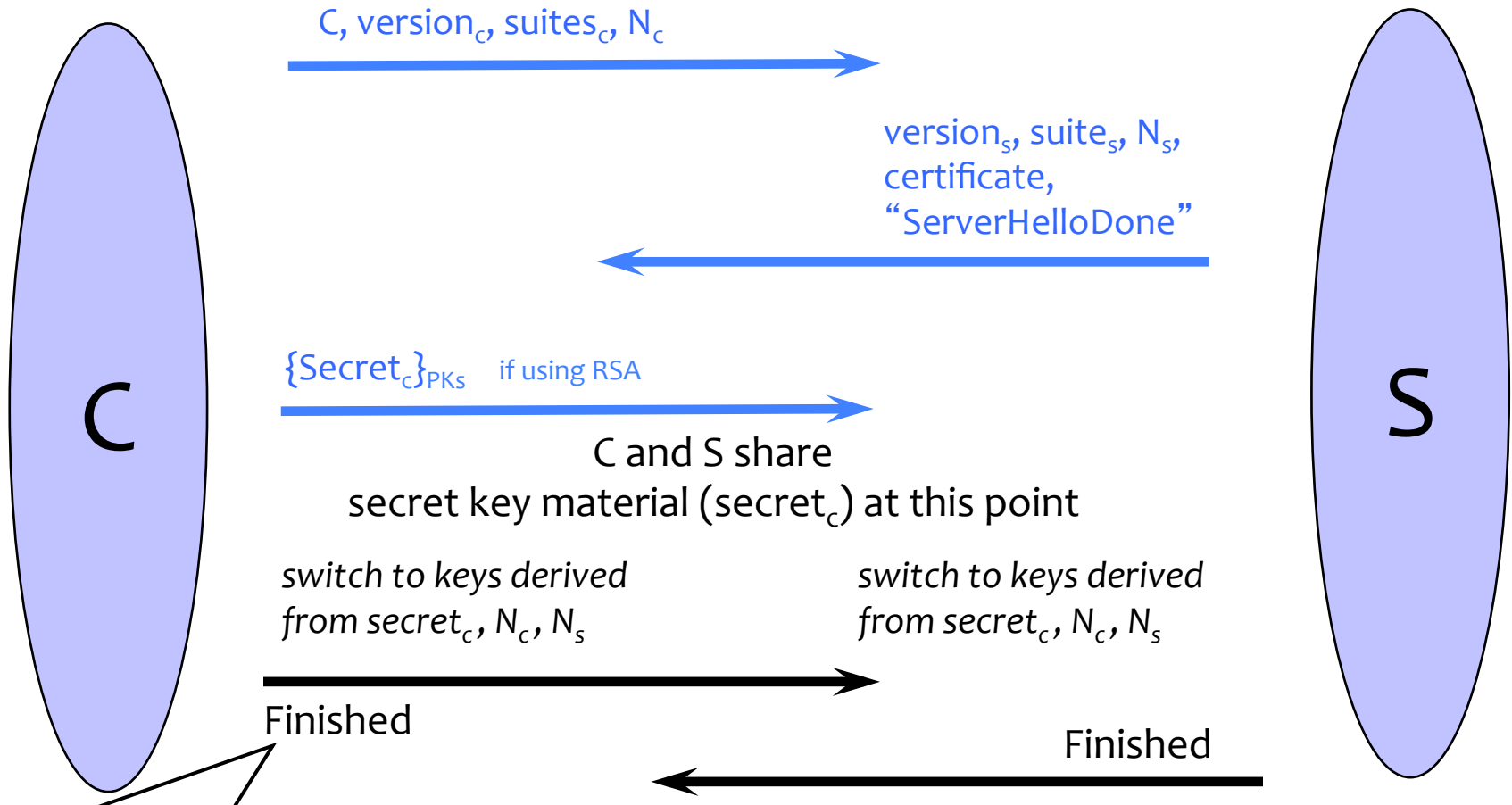
Basic Handshake Protocol



Basic Handshake Protocol

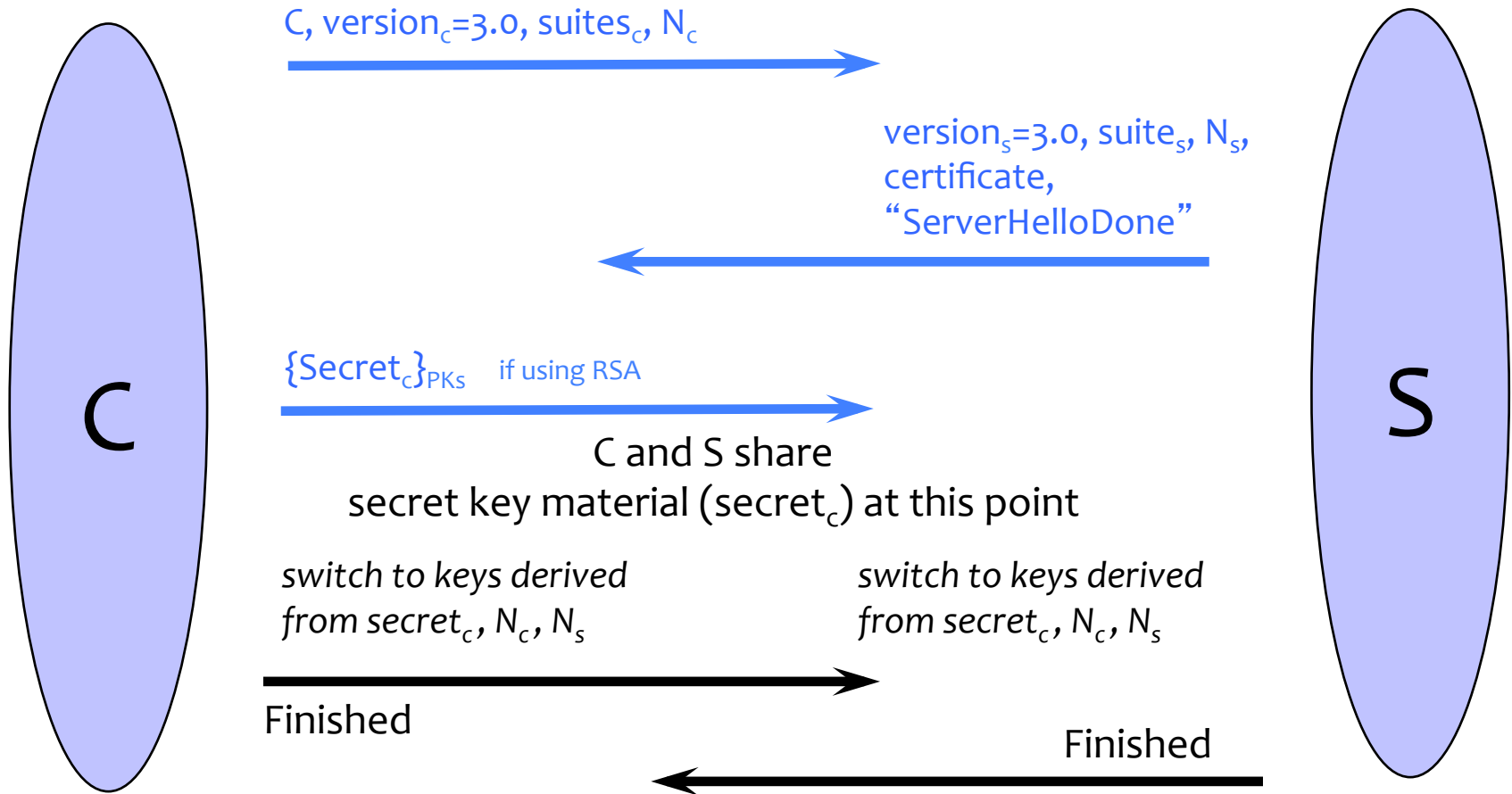


Basic Handshake Protocol

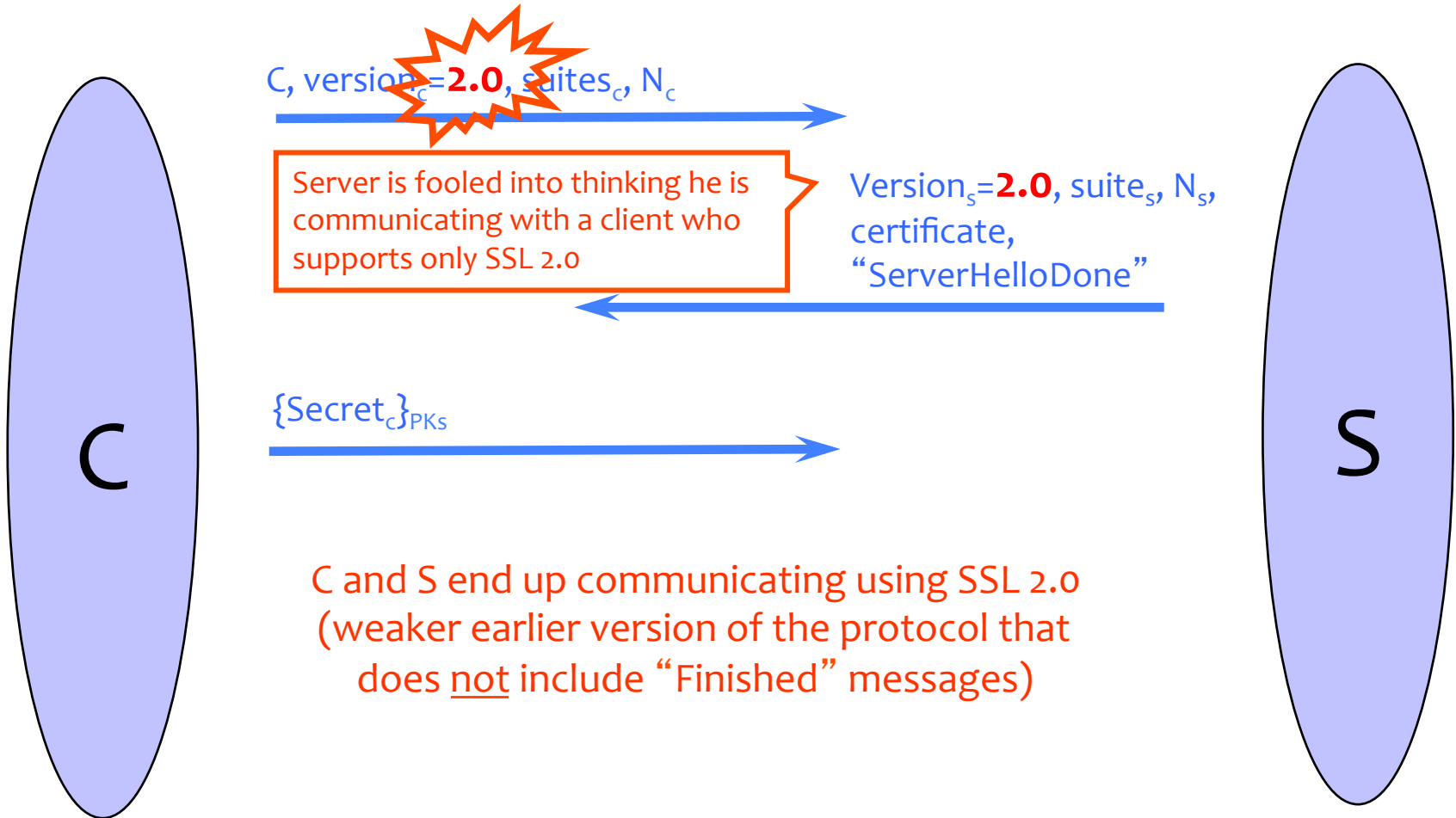


Record of all sent and received handshake messages

“Core” SSL 3.0 Handshake (Not TLS)



Version Rollback Attack



“Chosen-Protocol” Attacks

- Why do people release new versions of security protocols?
Because the old version got broken!
- New version must be **backward-compatible**
 - Not everybody upgrades right away
- Attacker can fool someone into using the old, broken version and exploit known vulnerability
 - Similar: fool victim into using weak crypto algorithms
- Defense is hard: must authenticate version in early designs
- Many protocols had “version rollback” attacks
 - SSL, SSH, GSM (cell phones)

Version Check in SSL 3.0

