CSE 484 / CSE M 584: Computer Security and Privacy

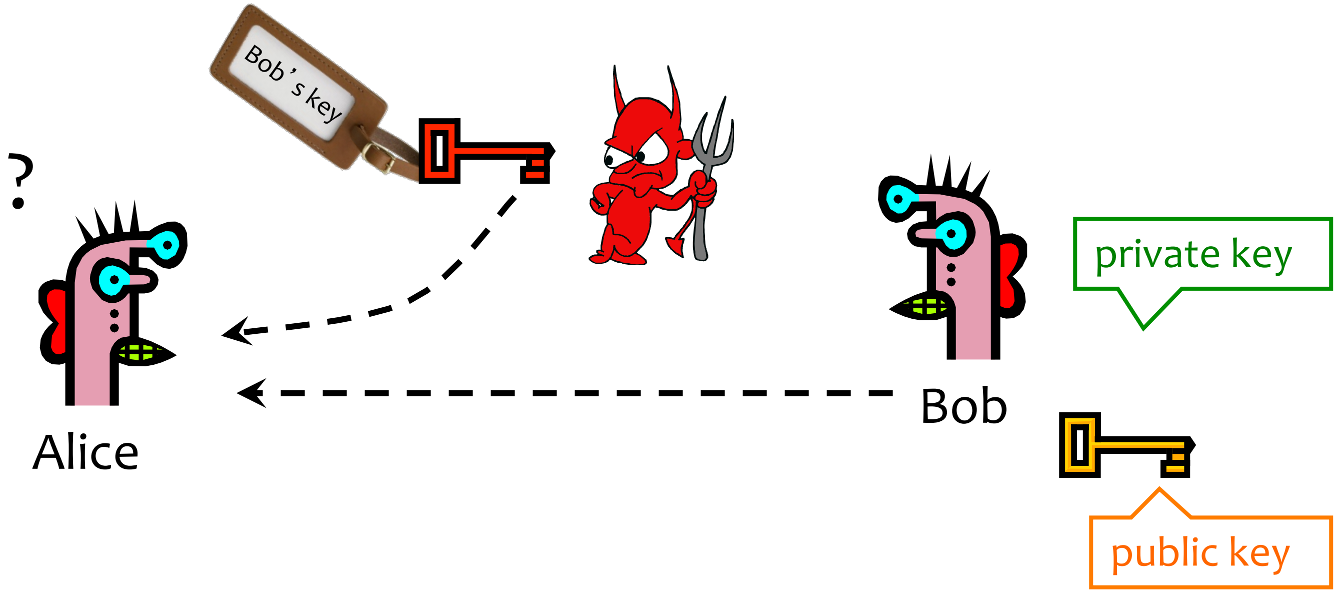
Crypto meets Web Security: Certificates and SSL/TLS

Spring 2016

Franziska (Franzi) Roesner
franzi@cs.washington.edu

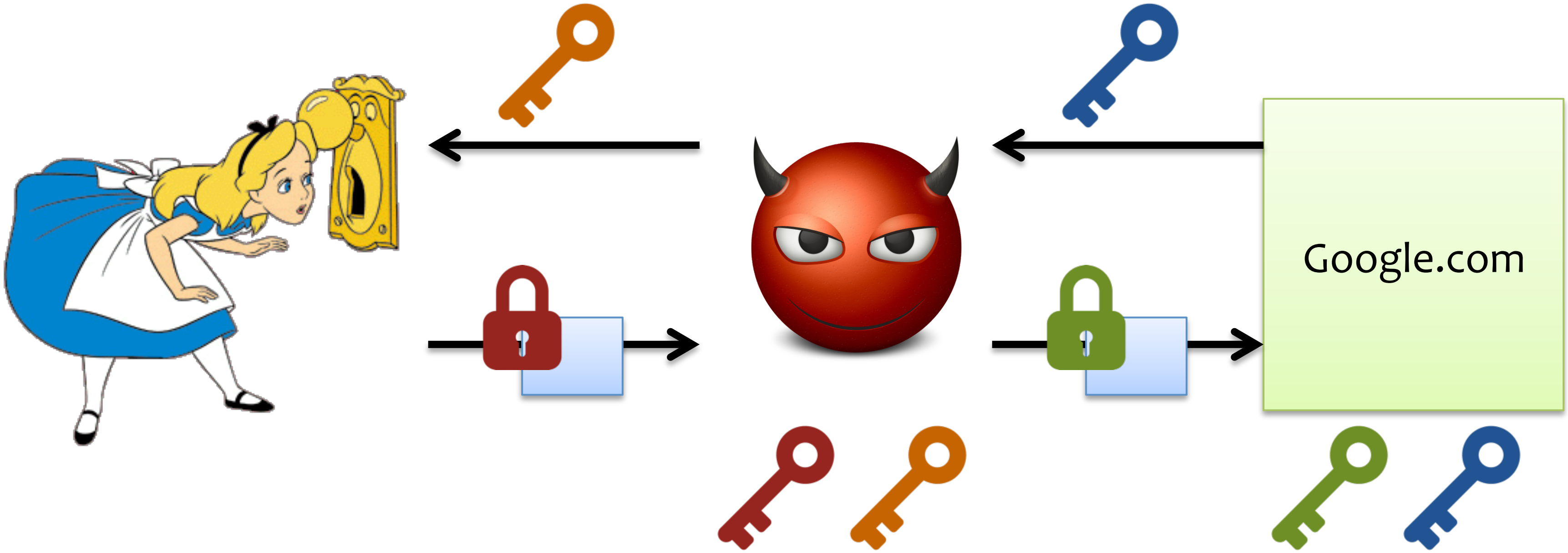
Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Authenticity of Public Keys



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

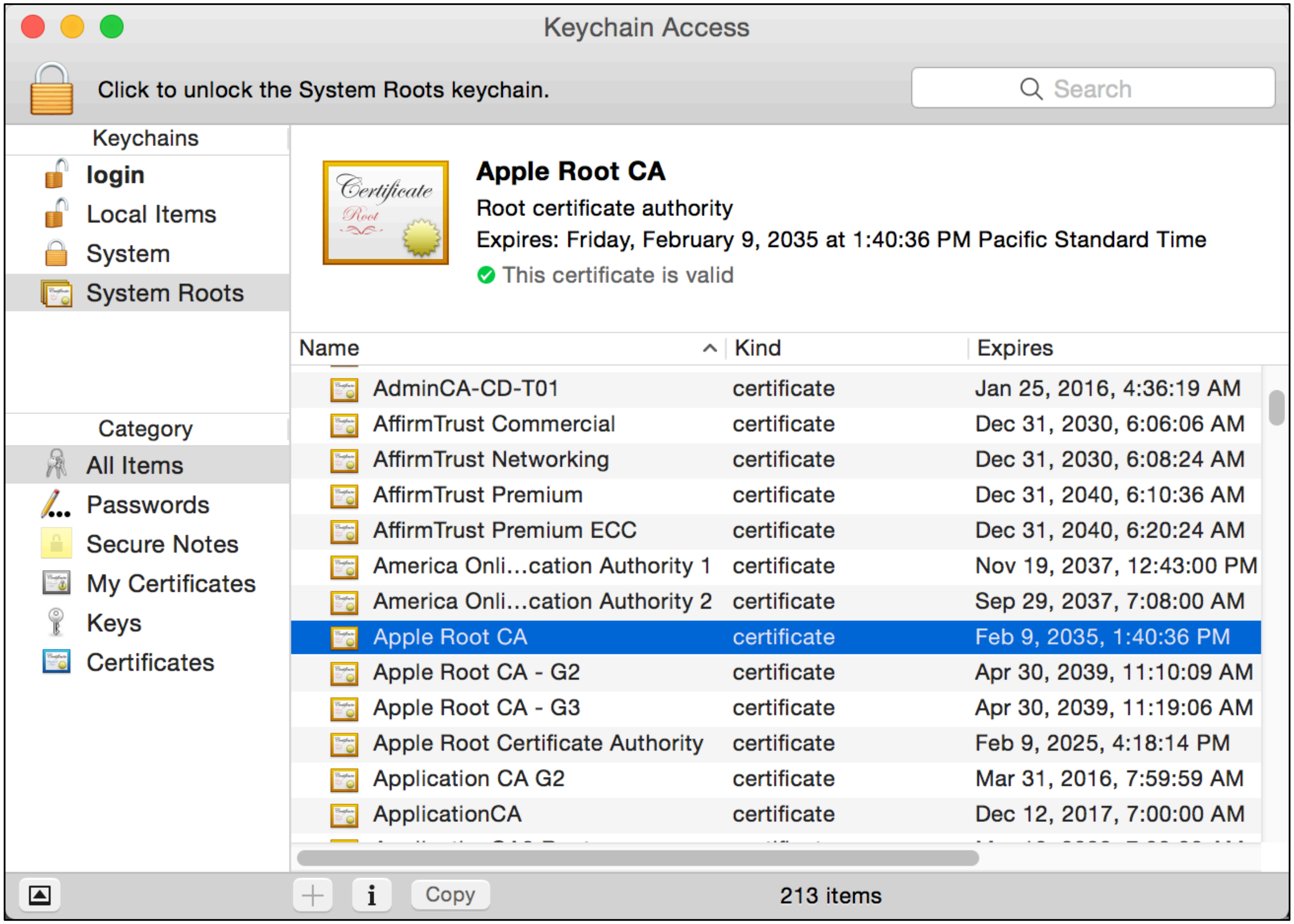
Threat: Man-In-The-Middle (MITM)



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

Trusted Certificate Authorities



Click to unlock the System Roots keychain.

Apple Root CA

Root certificate authority

Expires: Friday, February 9, 2035 at 1:40:36 PM Pacific Standard Time

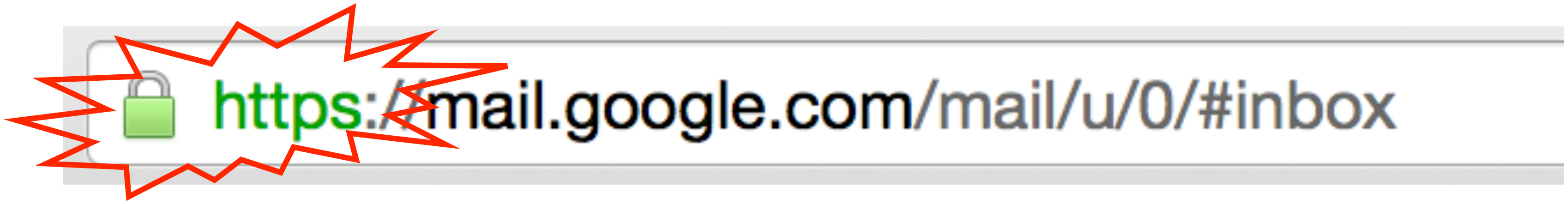
✓ This certificate is valid

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
Apple Root CA	certificate	Feb 9, 2035, 1:40:36 PM
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

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
    - $\text{sig}_{\text{Verisign}}$(“AnotherCA”, $\text{PK}_{\text{AnotherCA}}$), $\text{sig}_{\text{AnotherCA}}$(“Alice”, PK_{A})
  - What happens if root authority is ever compromised?

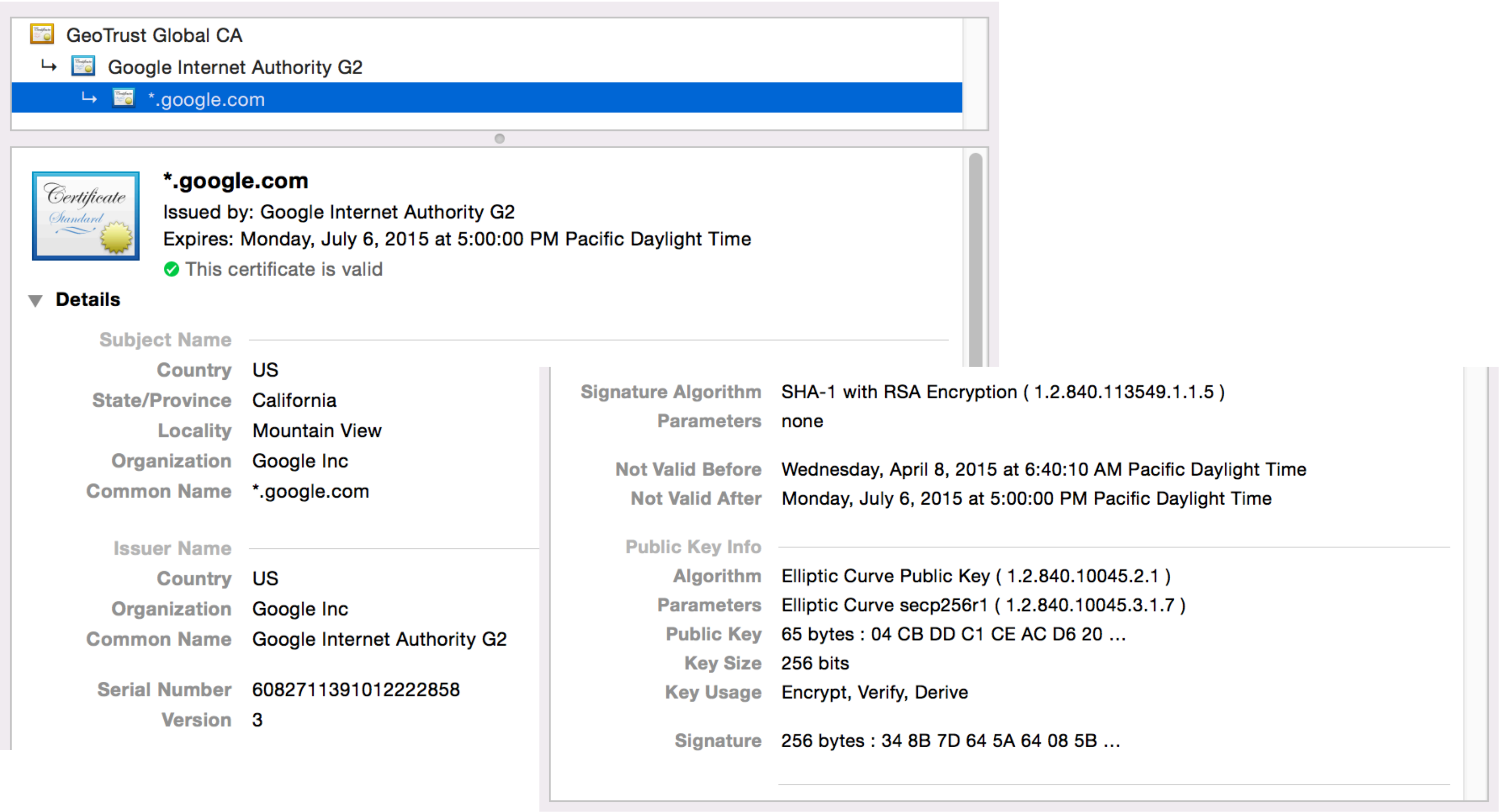
You encounter this every day…


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

SSL/TLS: Encryption & authentication for connections

(More on this later!)

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

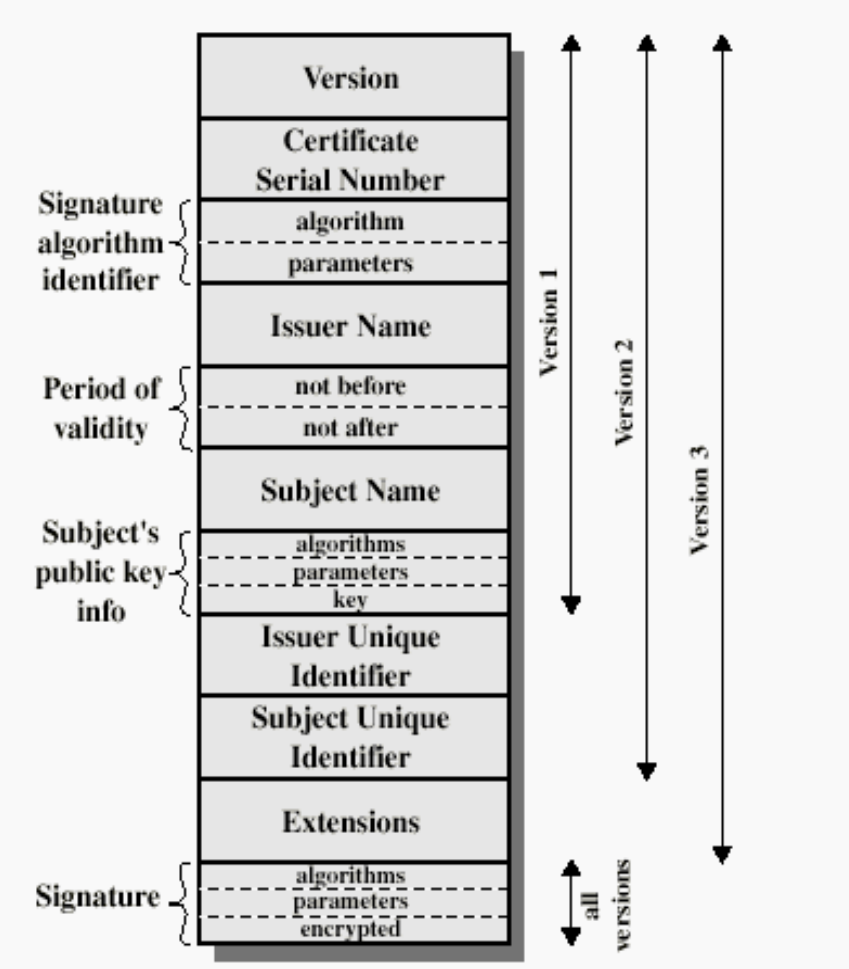
✓ This certificate is valid

▼ **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	608271139101222858
Version	3

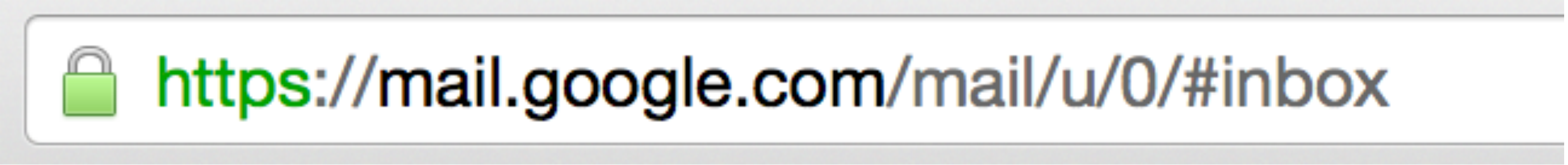
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, April 8, 2015 at 6:40:10 AM Pacific Daylight Time
Not Valid After	Monday, July 6, 2015 at 5:00:00 PM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
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 …

X.509 Certificate



Many Challenges…
[more examples in section]

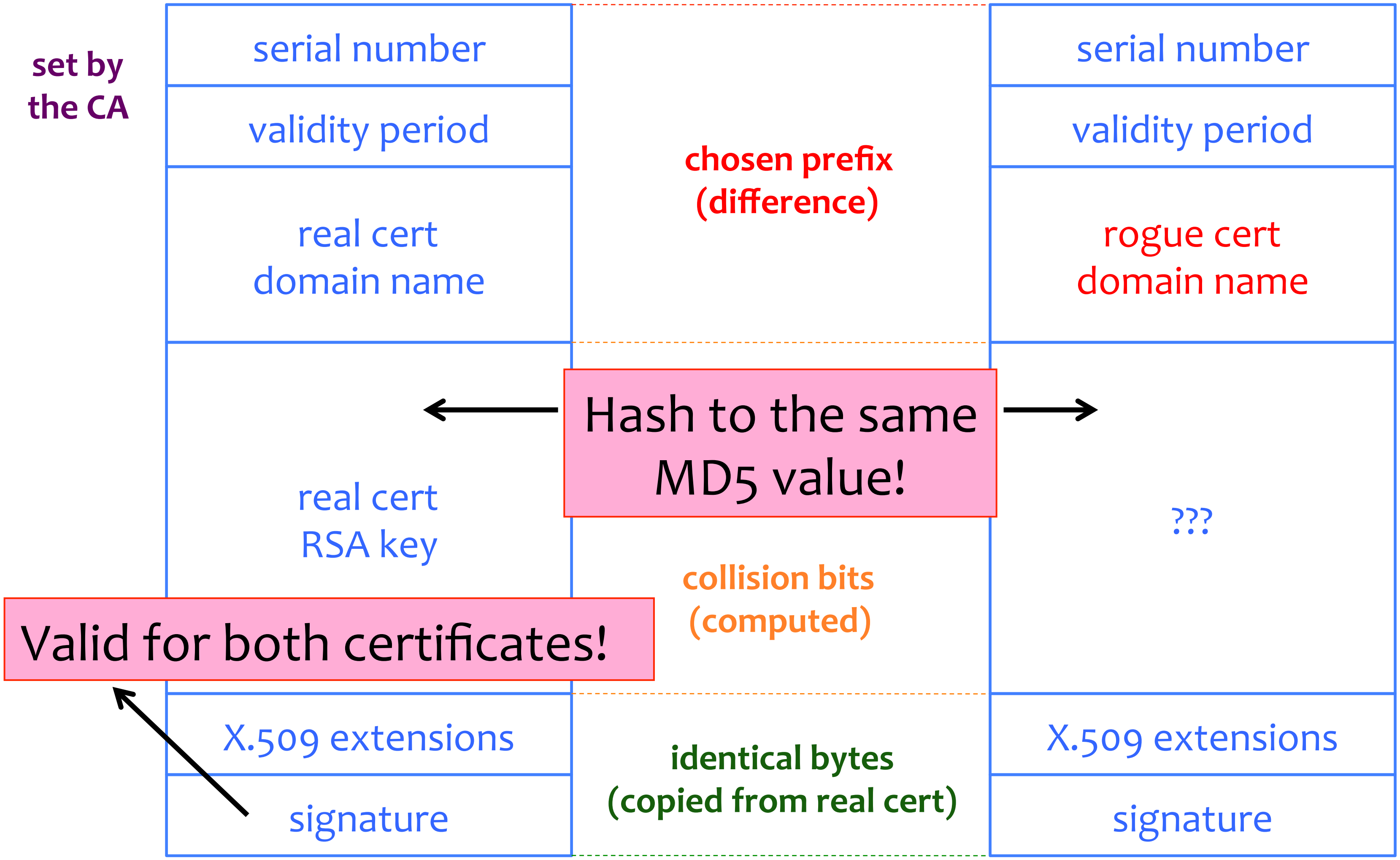
- Hash collisions
- Weak security at CAs
  - Allows attackers to issue rogue certificates
- Users don’t notice when attacks happen
  - We’ll talk more about this later
- Etc…



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

[Sotirov et al. "Rogue Certificates"]

Colliding Certificates



set by the CA

serial number | serial number

validity period | validity period

real cert domain name | rogue cert domain name

chosen prefix (difference)

real cert RSA key | ???

Hash to the same MD5 value!

collision bits (computed)

Valid for both certificates!

X.509 extensions | X.509 extensions

signature | signature

identical bytes (copied from real cert)

Attacking CAs

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.

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.

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

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 his 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

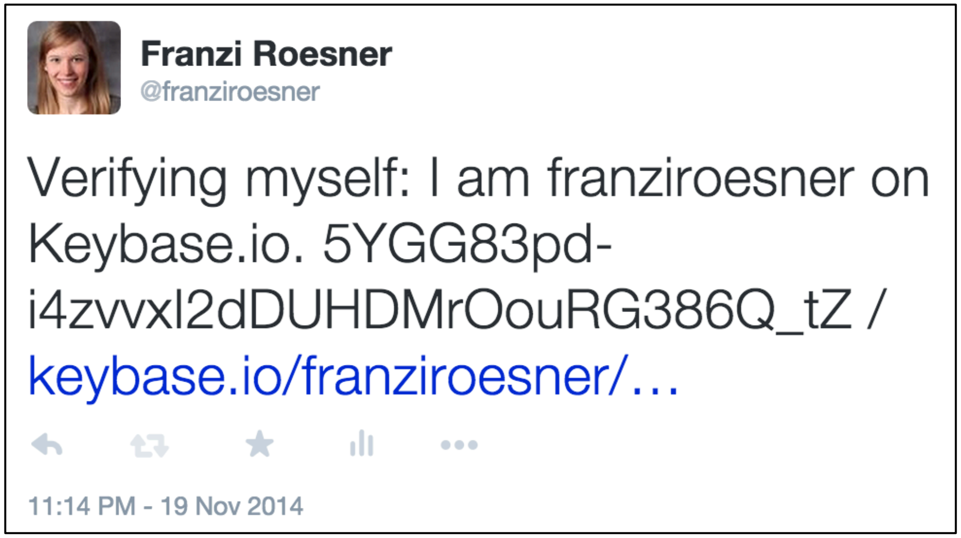
Attempt to Fix CA Problems: Convergence

- Background observation:
  - Attacker will have a hard time mounting man-in-the-middle attacks against **all** clients around the world
- Basic idea:
  - Lots of nodes around the world obtaining SSL/TLS certificates from servers
  - Check responses across servers, and also observe unexpected changes from existing certificates

http://convergence.io/

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



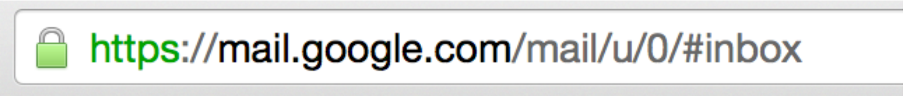
Franzi Roesner
@franziroesner

Verifying myself: I am franziroesner on Keybase.io. 5YGG83pd-i4zvvxl2dDUHDMrOouRG386Q_tZ / keybase.io/franziroesner/…

11:14 PM - 19 Nov 2014

https://keybase.io/

SSL/TLS



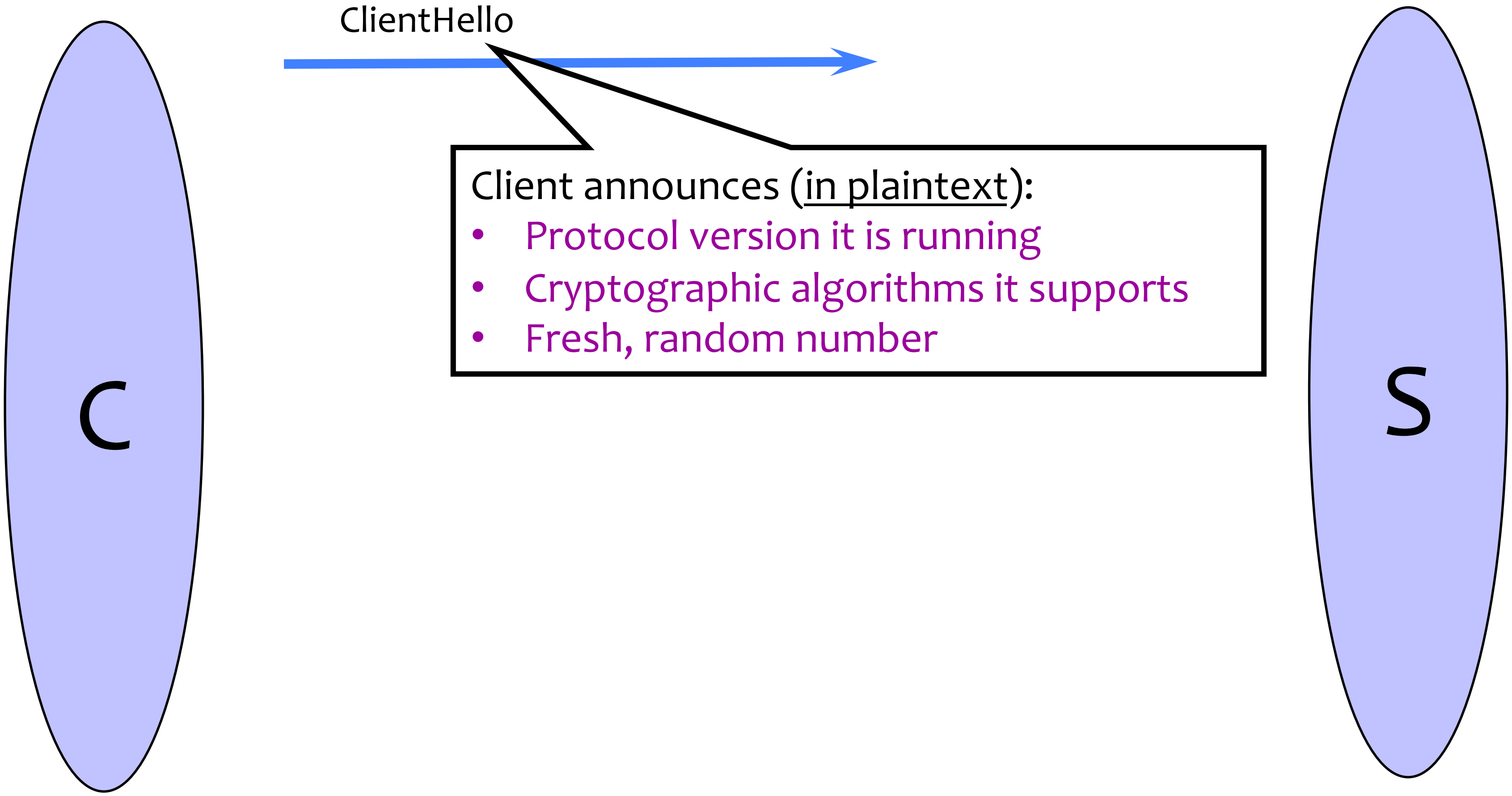
https://mail.google.com/mail/u/0/#inbox

- Secure Sockets Layer and Transport Layer Security protocols
  - Same protocol design, different crypto algorithms
- 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.

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

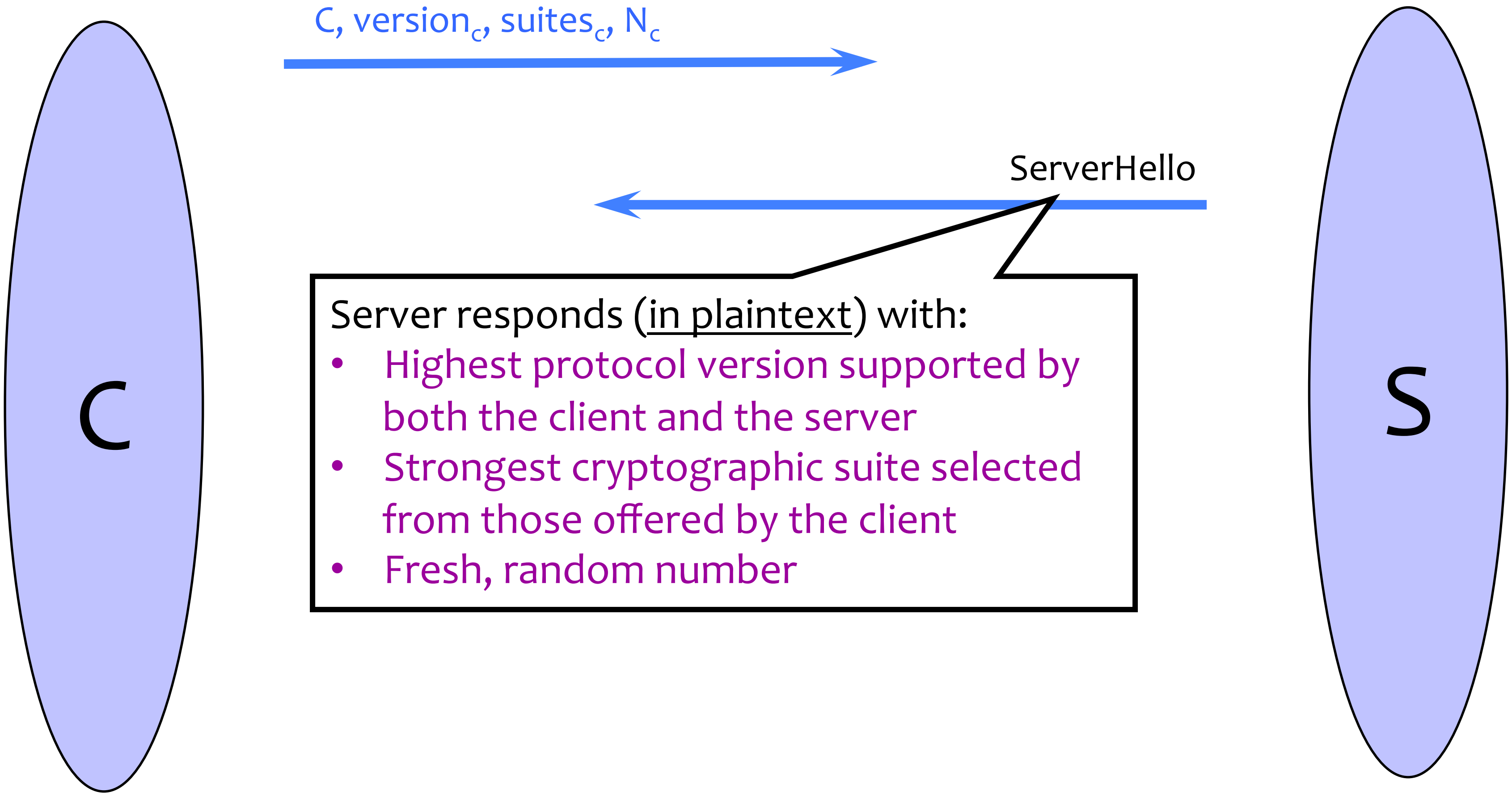
Basic Handshake Protocol



Client announces (in plaintext):

- Protocol version it is running
- Cryptographic algorithms it supports
- Fresh, random number

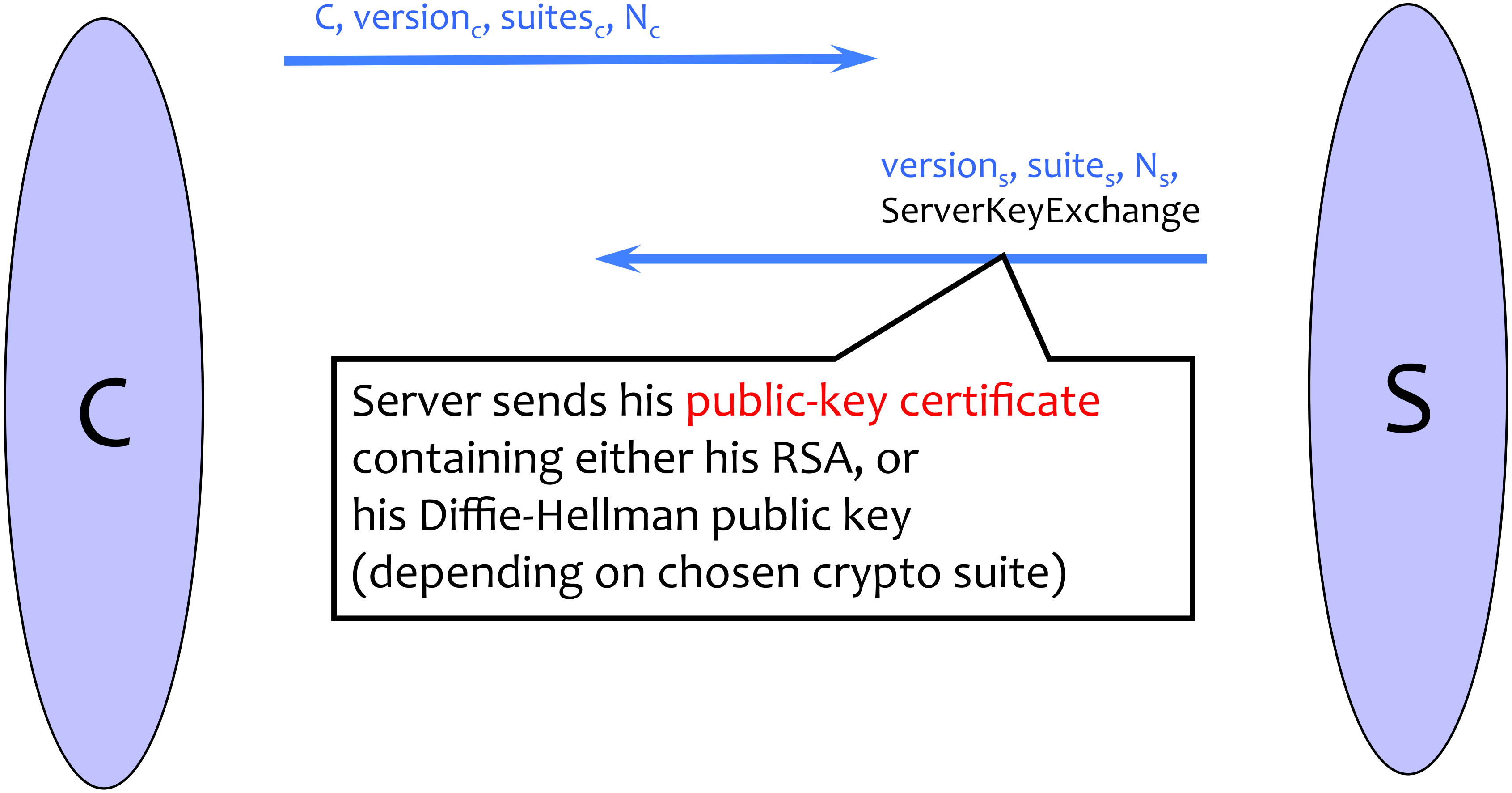
Basic Handshake Protocol



Server responds (in plaintext) with:

- Highest protocol version supported by both the client and the server
- Strongest cryptographic suite selected from those offered by the client
- Fresh, random number

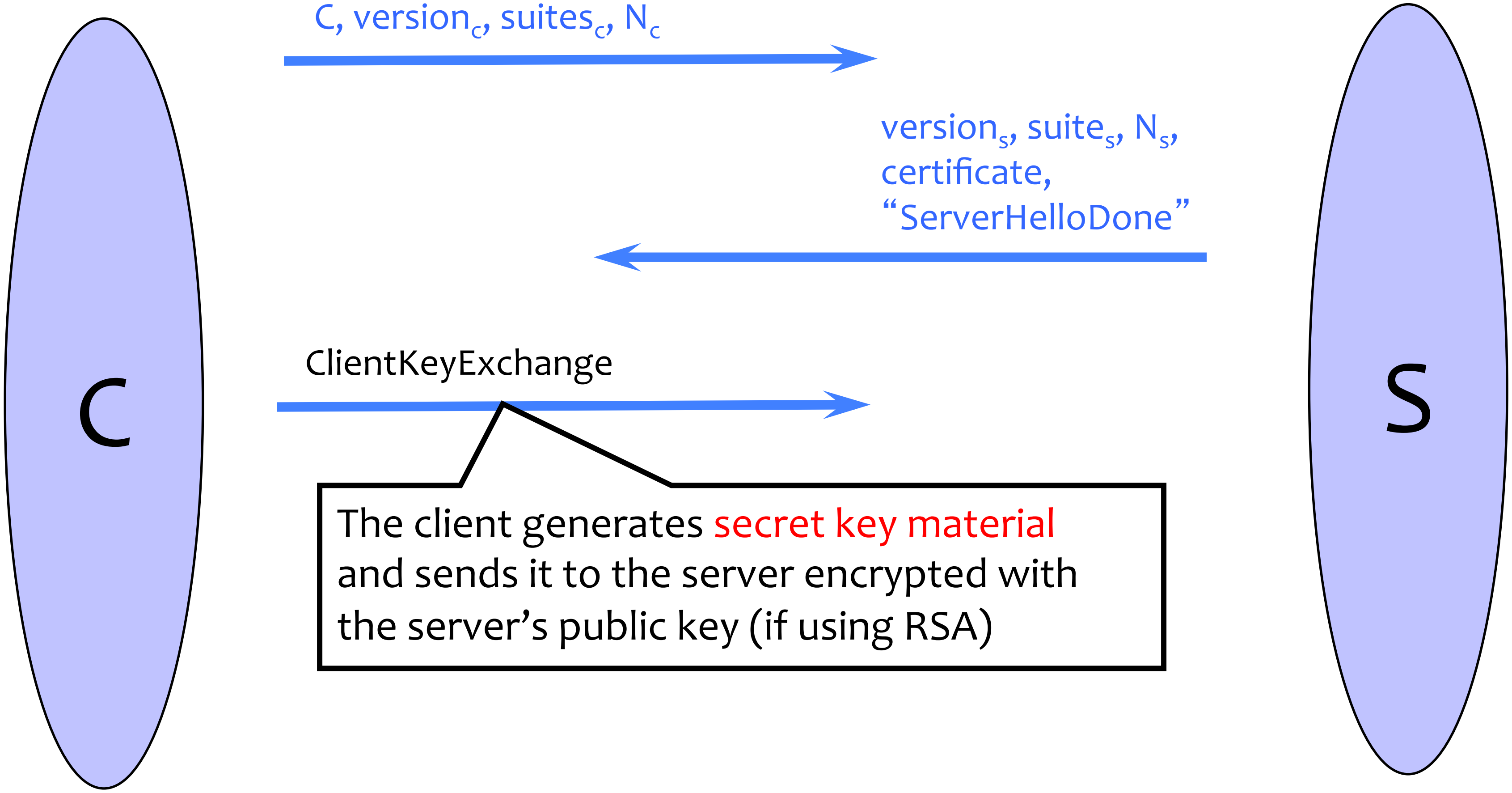
Basic Handshake Protocol



C, version$_c$, suites$_c$, N$_c$
version$_s$, suite$_s$, N$_s$,
ServerKeyExchange
C
S

Server sends his public-key certificate containing either his RSA, or his Diffie-Hellman public key (depending on chosen crypto suite)

Basic Handshake Protocol



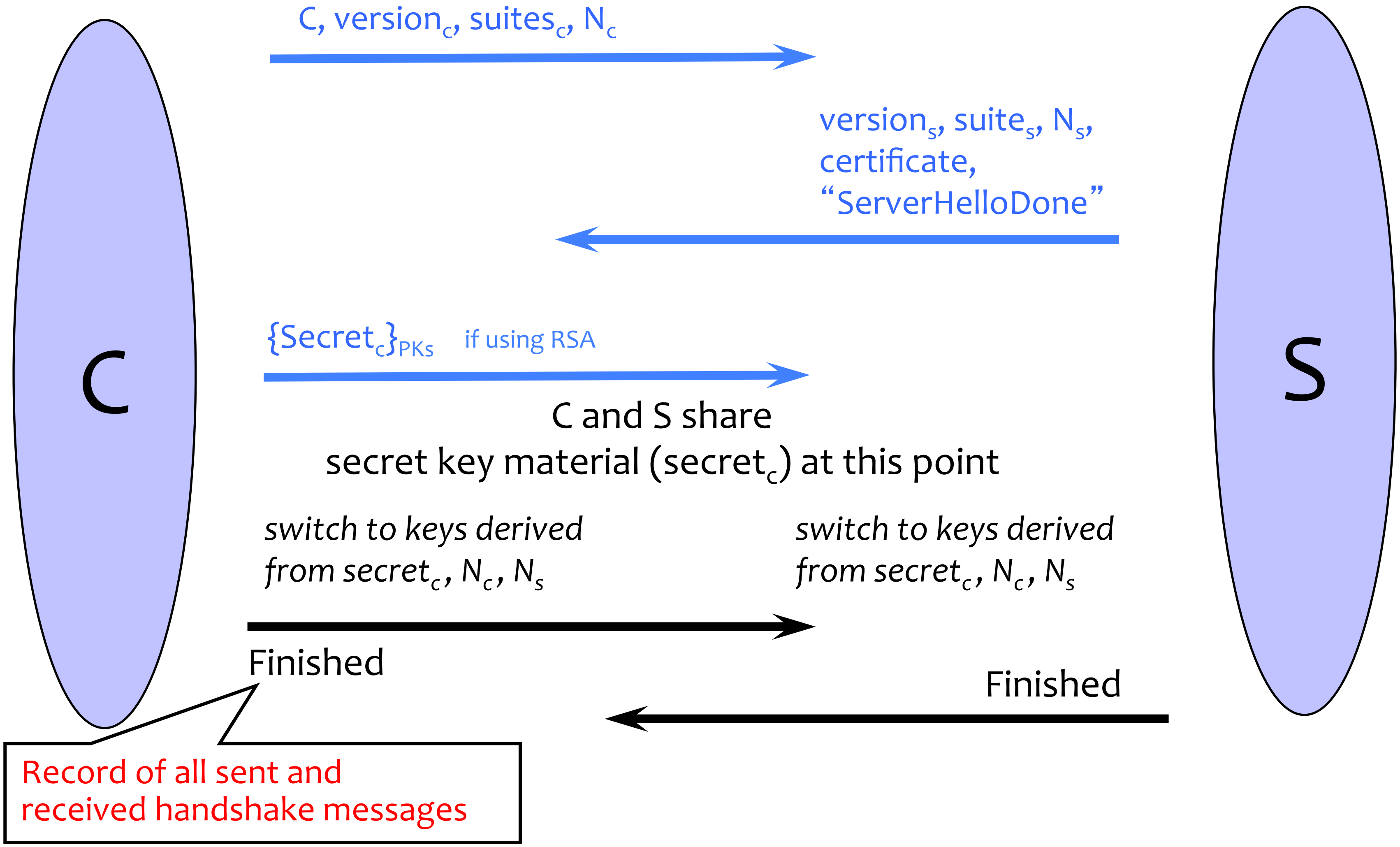
C, version_c, suites_c, N_c

version_s, suite_s, N_s,
certificate,
"ServerHelloDone"

ClientKeyExchange

The client generates secret key material and sends it to the server encrypted with the server's public key (if using RSA)

Basic Handshake Protocol



C, version_c, suites_c, N_c

version_s, suite_s, N_s,
certificate,
"ServerHelloDone"

$\{\text{Secret}_c\}_{PKs}$ if using RSA

C and S share
secret key material (secret_c) at this point

switch to keys derived
from $secret_c$, N_c, N_s

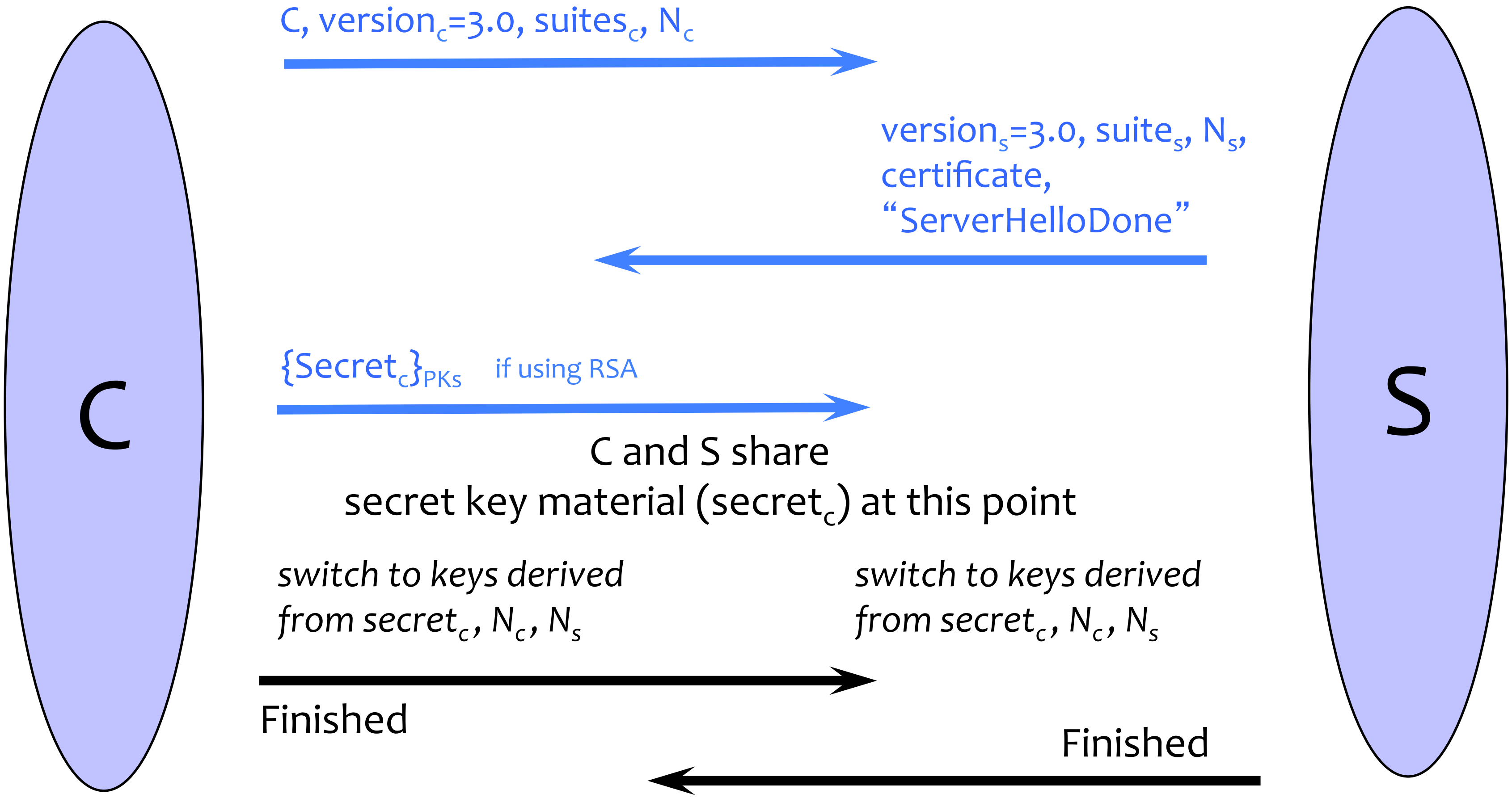
switch to keys derived
from $secret_c$, N_c, N_s

Finished

Finished

Record of all sent and
received handshake messages

"Core" SSL 3.0 Handshake (Not TLS)



C

C, version$_c$=3.0, suites$_c$, N_c

version$_s$=3.0, suite$_s$, N_s,
certificate,
"ServerHelloDone"

$\{\text{Secret}_c\}_{PKs}$ if using RSA

C and S share
secret key material (secret$_c$) at this point

*switch to keys derived
from secret$_c$, N_c, N_s*

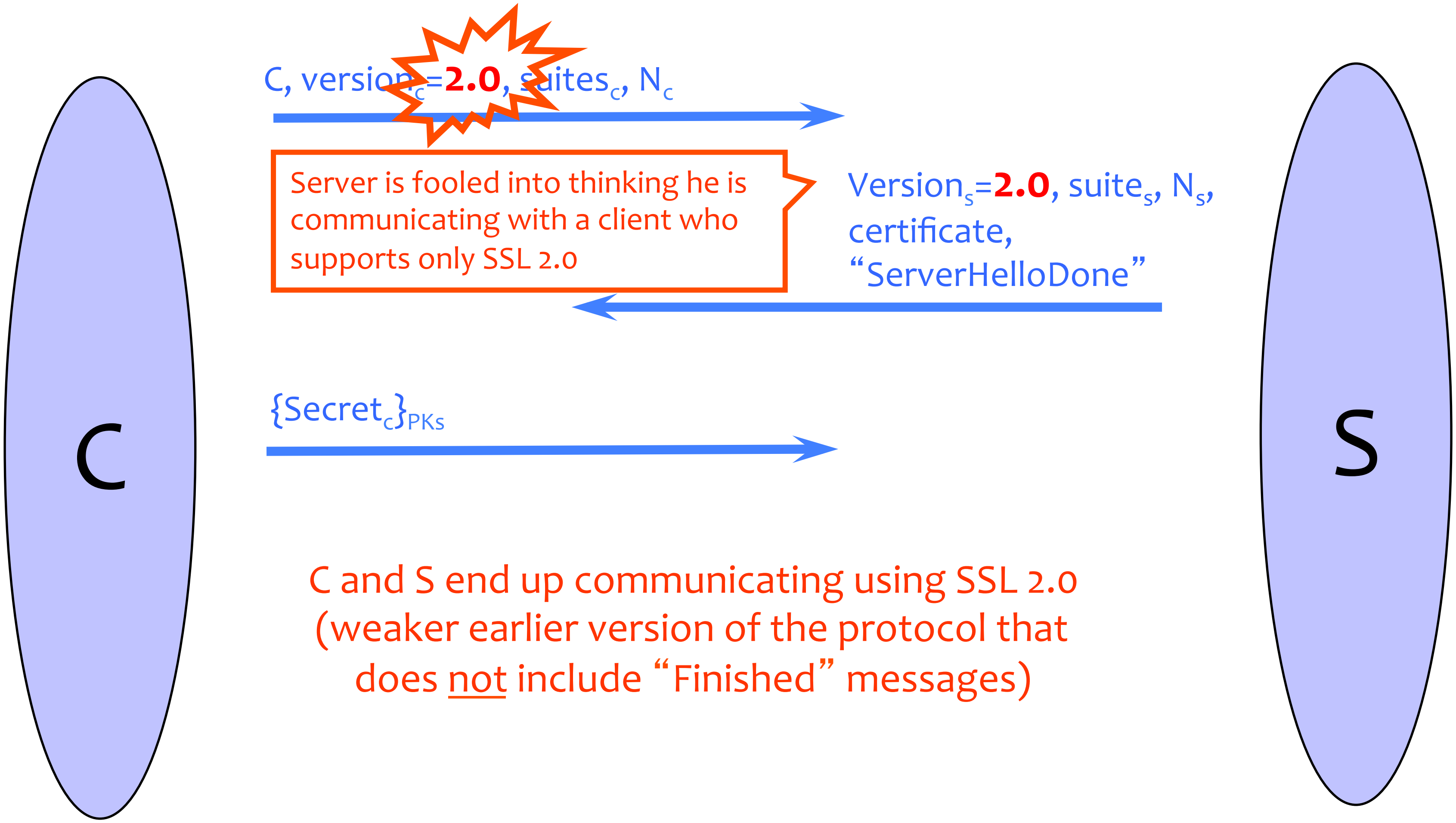
*switch to keys derived
from secret$_c$, N_c, N_s*

Finished

Finished

S

Version Rollback Attack



C, version$_c$=**2.0**, suites$_c$, N$_c$

Server is fooled into thinking he is communicating with a client who supports only SSL 2.0

Version$_s$=**2.0**, suite$_s$, N$_s$, certificate, "ServerHelloDone"

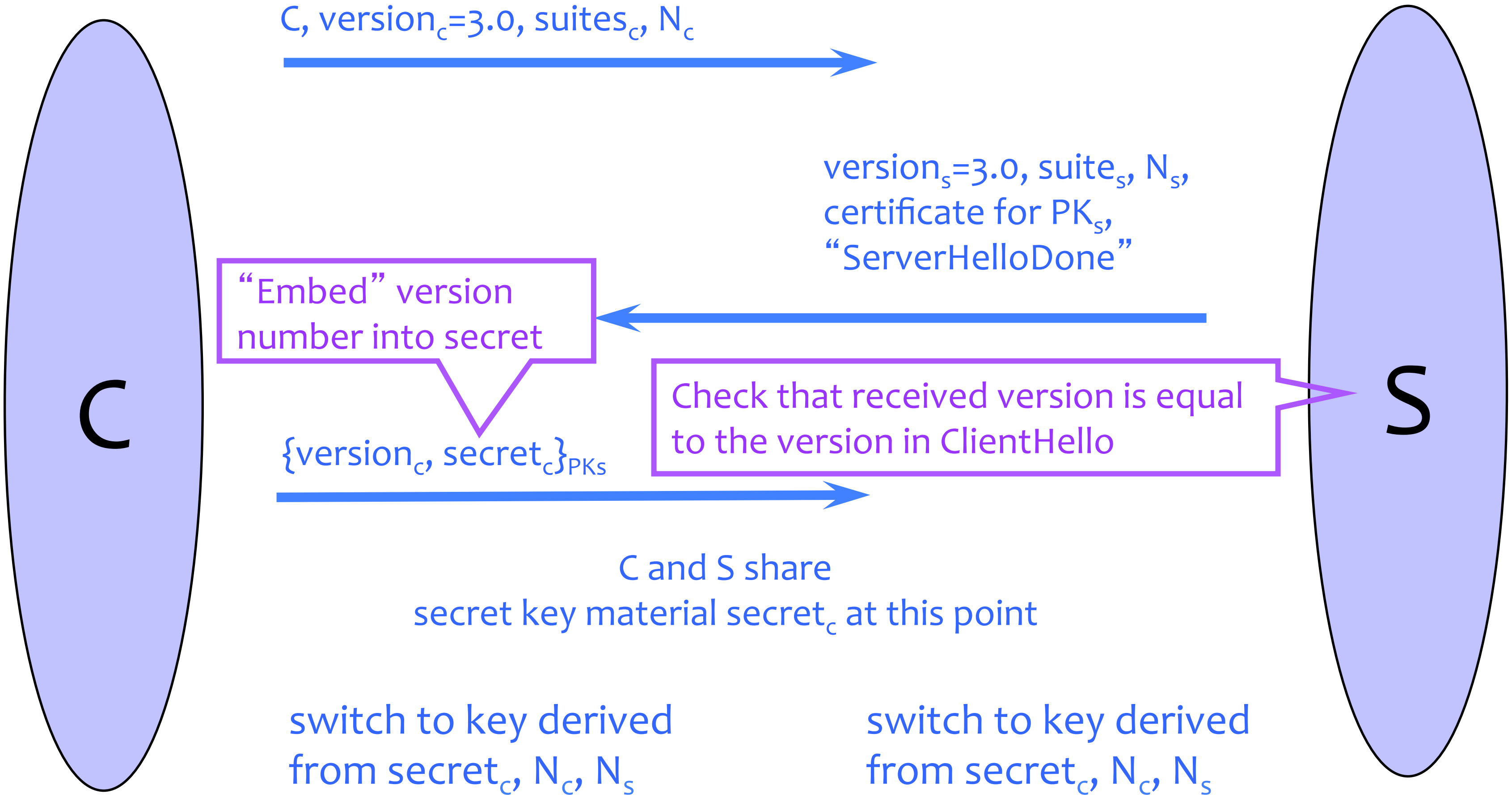
{Secret$_c$}$_{PKs}$

C and S end up communicating using SSL 2.0 (weaker earlier version of the protocol that does <ins>not</ins> include "Finished" messages)

“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



C, version$_c$=3.0, suites$_c$, N$_c$

version$_s$=3.0, suite$_s$, N$_s$,
certificate for PK$_s$,
"ServerHelloDone"

"Embed" version number into secret

{version$_c$, secret$_c$}$_{PKs}$

Check that received version is equal to the version in ClientHello

C and S share
secret key material secret$_c$ at this point

switch to key derived
from secret$_c$, N$_c$, N$_s$

switch to key derived
from secret$_c$, N$_c$, N$_s$