CSE 484 (Winter 2010)

Asymmetric Cryptography

Tadayoshi Kohno

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

Goals for Today

- HW3 up
- SSL
- User authentication
- SF Prototyping Today, 1pm (Gates Commons, CSE 691): Dedicated time to interact with other groups.
- Wednesday: Guest lectures on Research (Roxana Geambasu and Karl Koscher)

ExtendedGCD algorithm

function EXTENDEDGCD
input: a Positive integer argument.
b Positive integer argument.
output: k The greatest common divisor of a and b.
(u, v) Integers such that $ua + vb = k$.

assert $a \geq 0 \wedge b \geq 0$
$(c, d) \leftarrow (a, b)$
$(u_c, v_c, u_d, v_d) \leftarrow (1, 0, 0, 1)$
while $c \neq 0$ **do**
Invariant: $u_c a + v_c b = c \wedge u_d a + v_d b = d$
$q \leftarrow \lfloor d/c \rfloor$
$(c, d) \leftarrow (d - qc, c)$
$(u_c, v_c, u_d, v_d) \leftarrow (u_d - qu_c, v_d - qv_c, u_c, v_c)$
od
return $d, (u_d, v_d)$

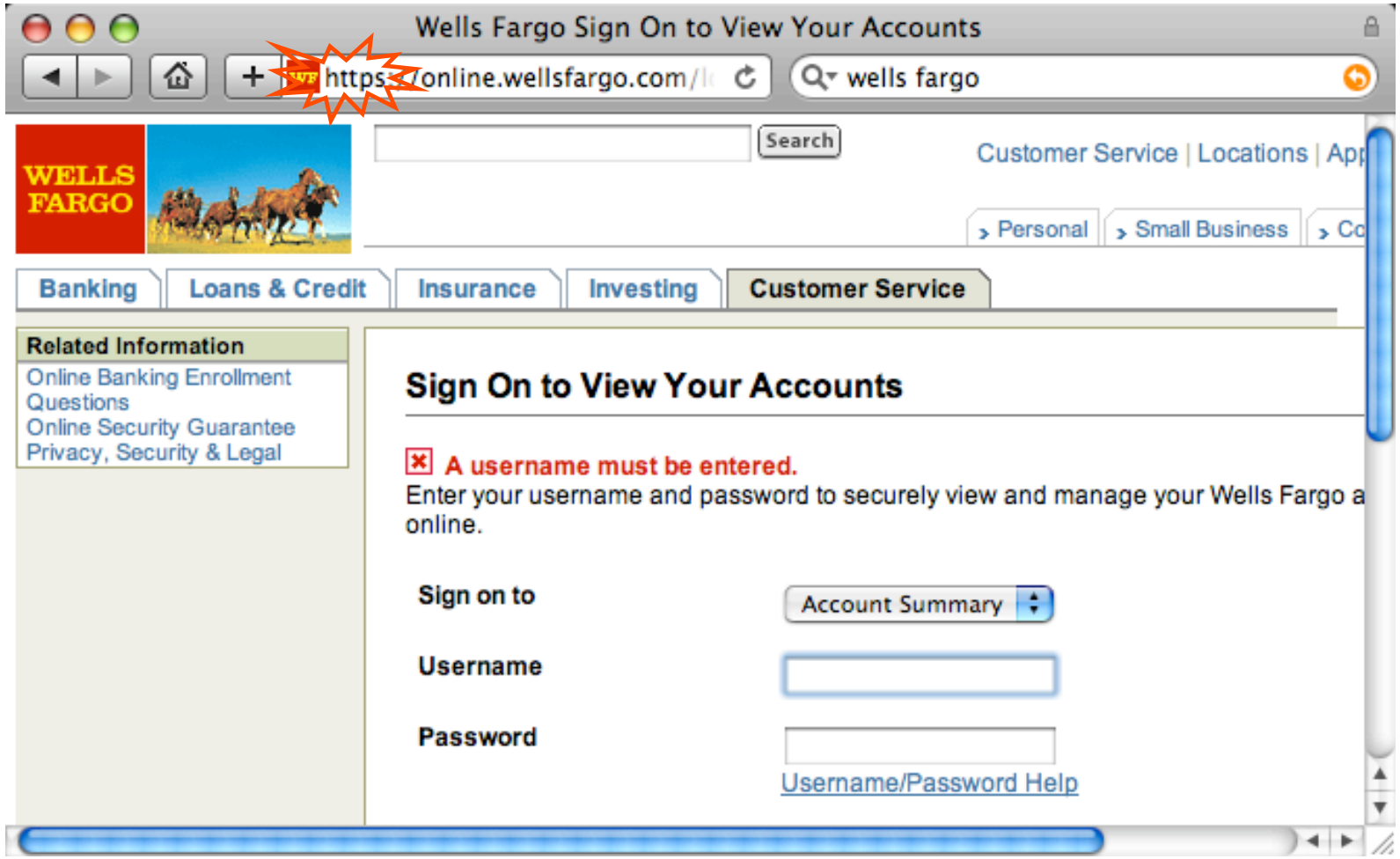
- If a and b are relatively prime (gcd = 1), then u is multiplicative inverse of a modulo b.

SSL

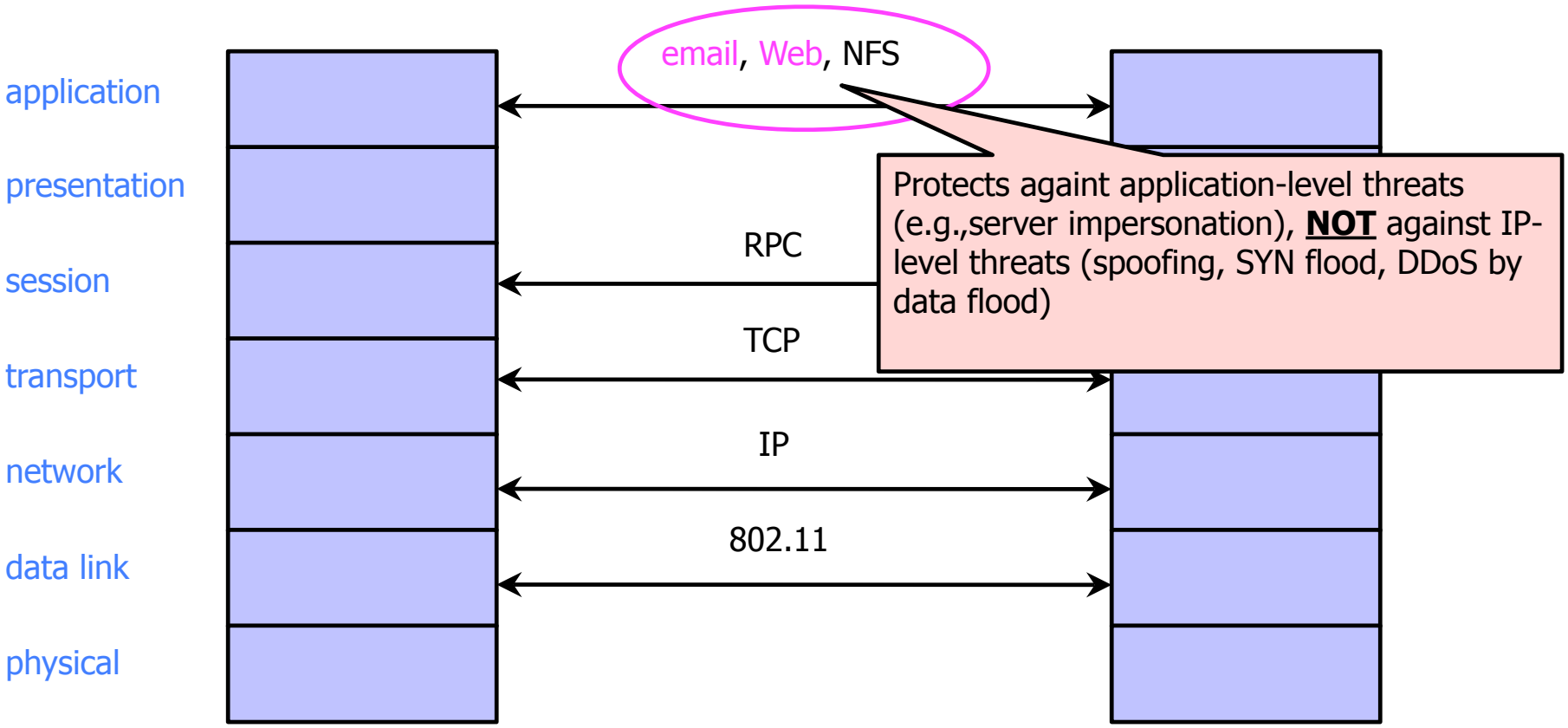
What is SSL / TLS?

- Transport Layer Security (TLS) protocol, version 1.2
  - De facto standard for Internet security
  - "The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications"
  - In practice, used to protect information transmitted between browsers and Web servers (and mail readers and ...)
- Based on Secure Sockets Layers (SSL) protocol, version 3.0
  - Same protocol design, different algorithms
- Deployed in nearly every Web browser

SSL / TLS in the Real World



Application-Level Protection



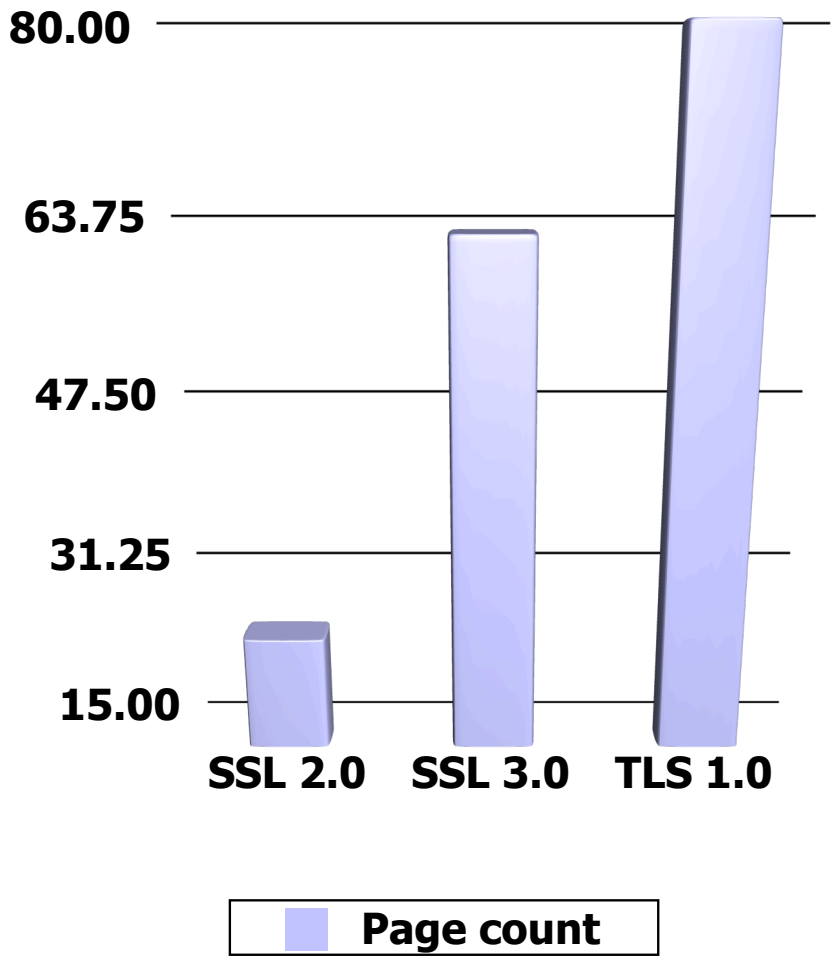
History of the Protocol

- SSL 1.0
  - Internal Netscape design, early 1994?
  - Lost in the mists of time
- SSL 2.0
  - Published by Netscape, November 1994
  - Several weaknesses
- SSL 3.0
  - Designed by Netscape and Paul Kocher, November 1996
- TLS 1.0
  - Internet standard based on SSL 3.0, January 1999
  - Not interoperable with SSL 3.0
    - TLS uses HMAC instead of earlier MAC; can run on any port
- TLS 1.2
  - Remove dependencies to MD5 and SHA1

“Request for Comments”

- Network protocols are usually disseminated in the form of an RFC
- TLS version 1.0 is described in RFC 5246
- Intended to be a self-contained definition of the protocol
  - Describes the protocol in sufficient detail for readers who will be implementing it and those who will be doing protocol analysis
  - Mixture of informal prose and pseudo-code

Evolution of the SSL/TLS RFC



104 pages for TLS 1.2

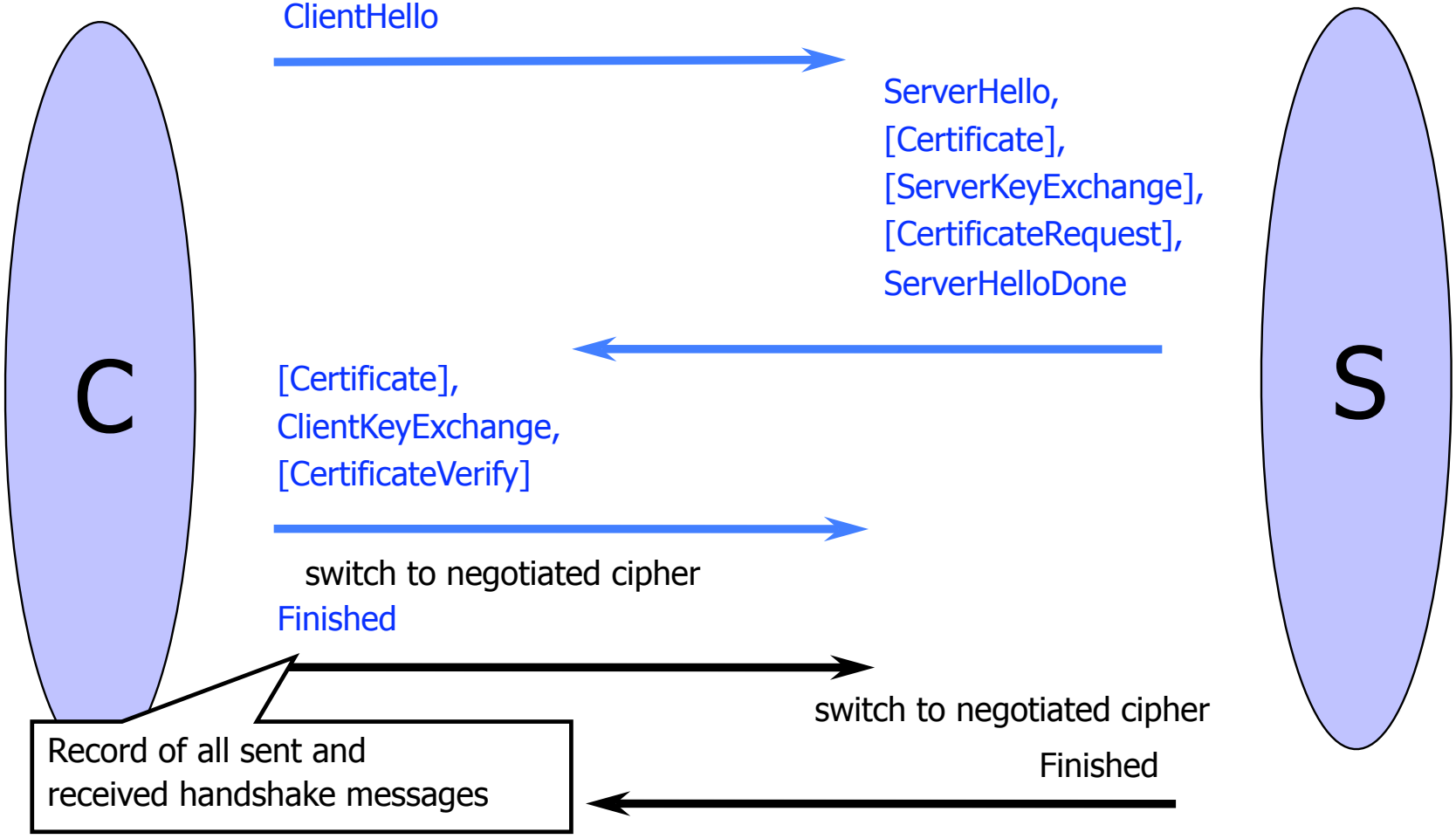
TLS Basics

- 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 key established in the handshake protocol to protect communication between the client and the server
- We will focus on the handshake protocol

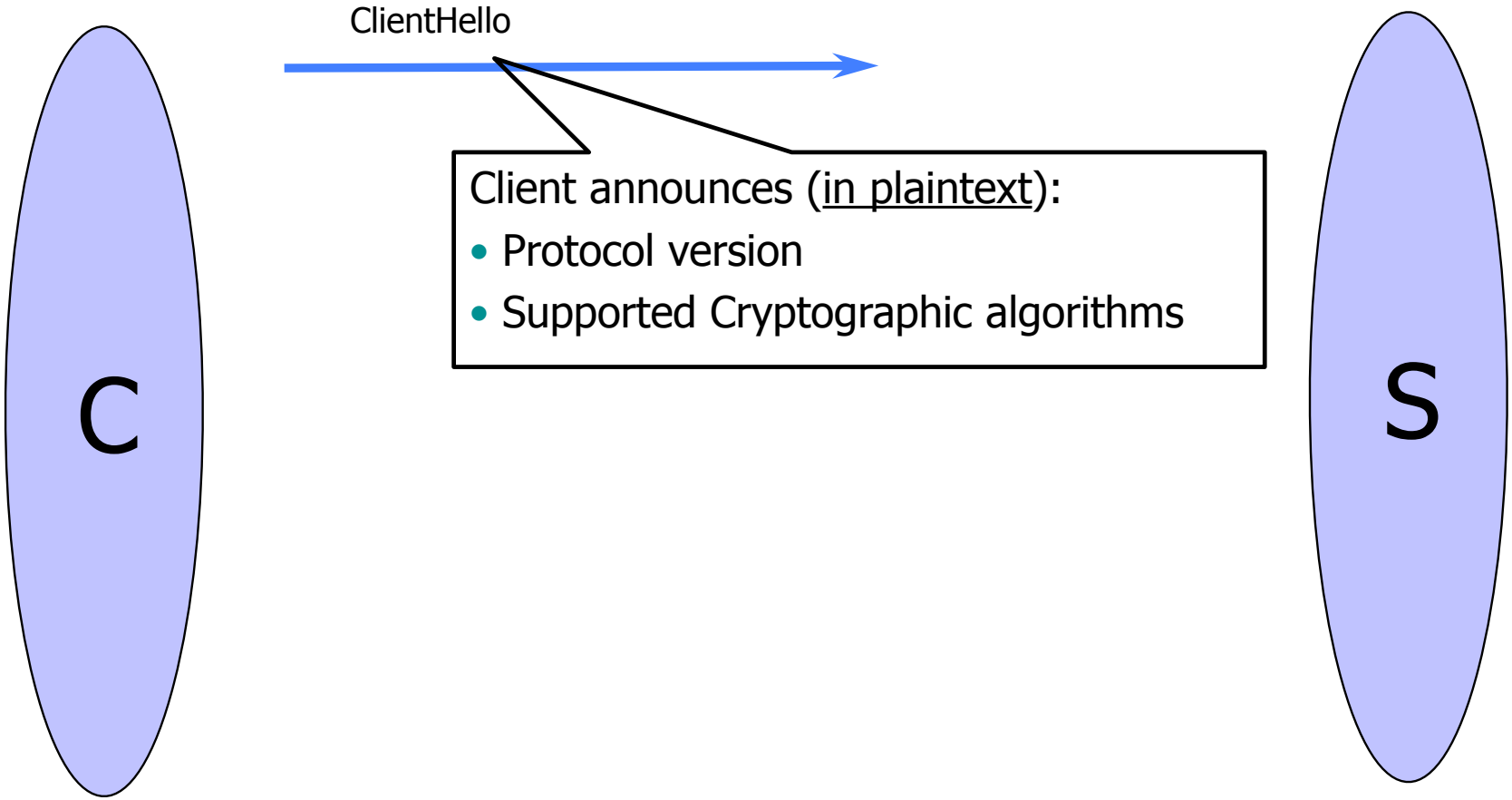
TLS Handshake Protocol

- Two parties: client and server
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations of the protocol
- Authenticate client and server (optional)
  - Use digital certificates to learn each other's public keys and verify each other's identity
- Use public keys to establish a shared secret

Handshake Protocol Structure



ClientHello



ClientHello (RFC)

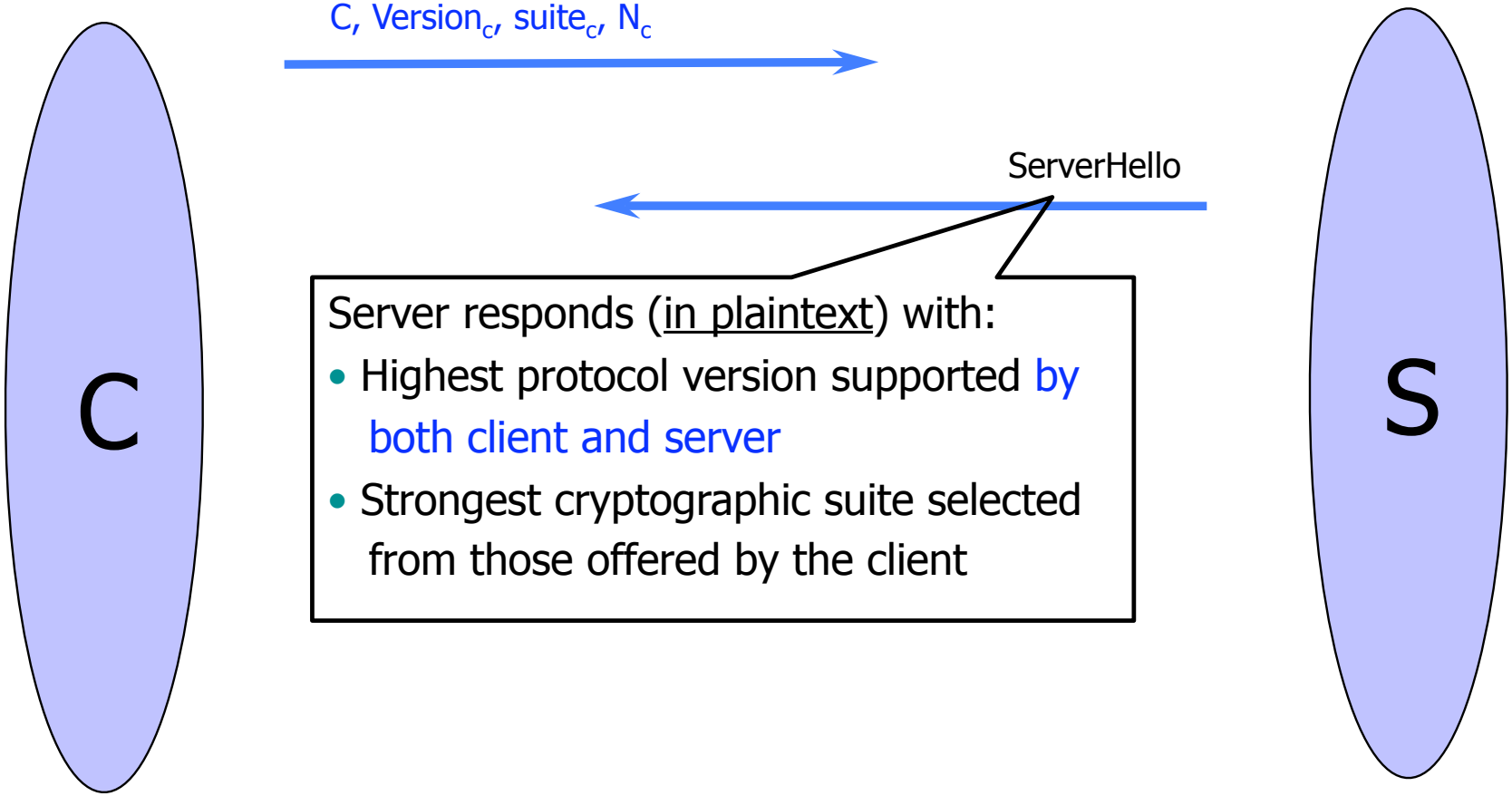
```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
} ClientHello
```

Highest version of the protocol supported by the client

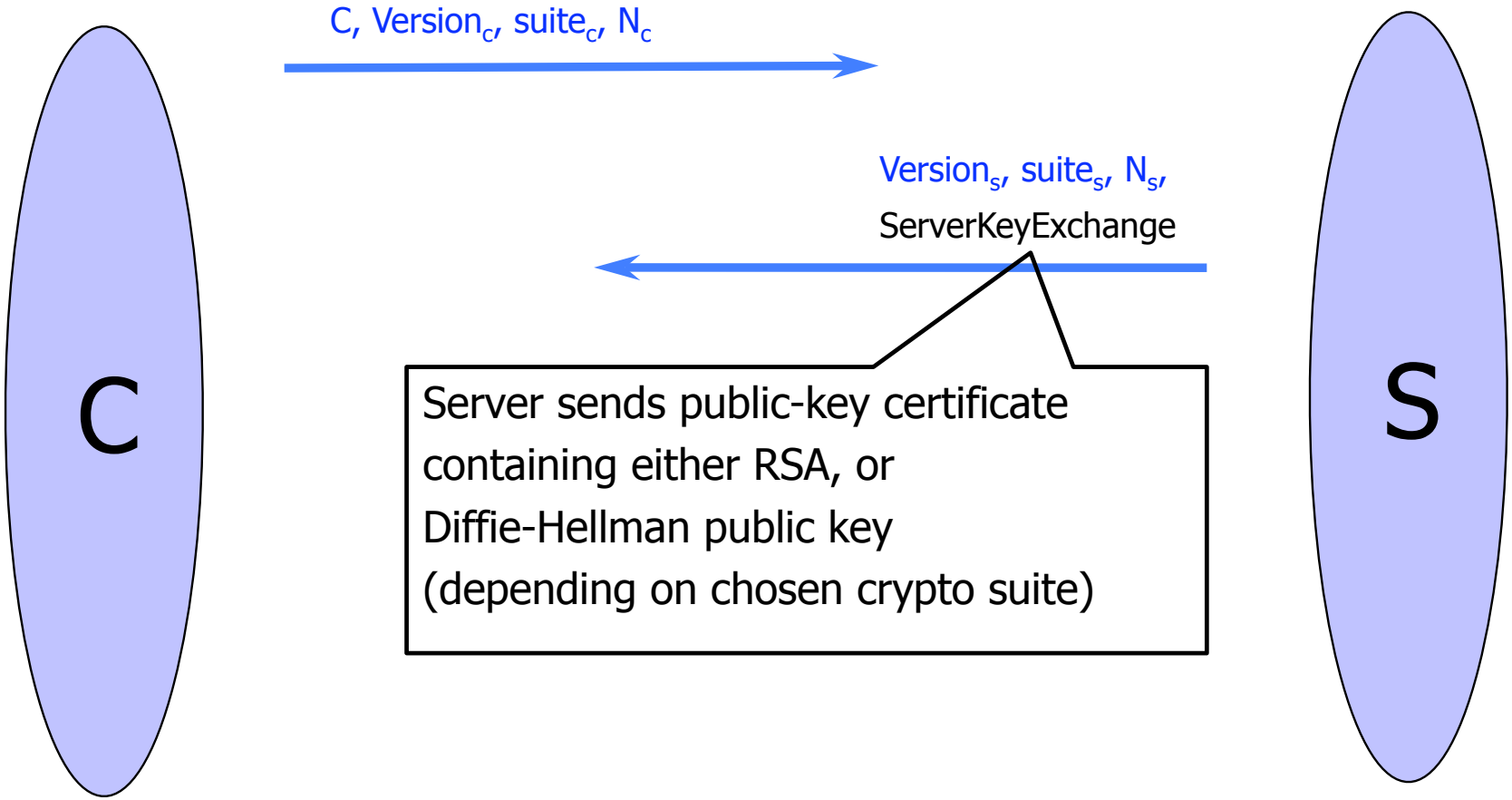
Session id (if the client wants to resume an old session)

Set of cryptographic algorithms supported by the client (e.g., RSA or Diffie-Hellman)

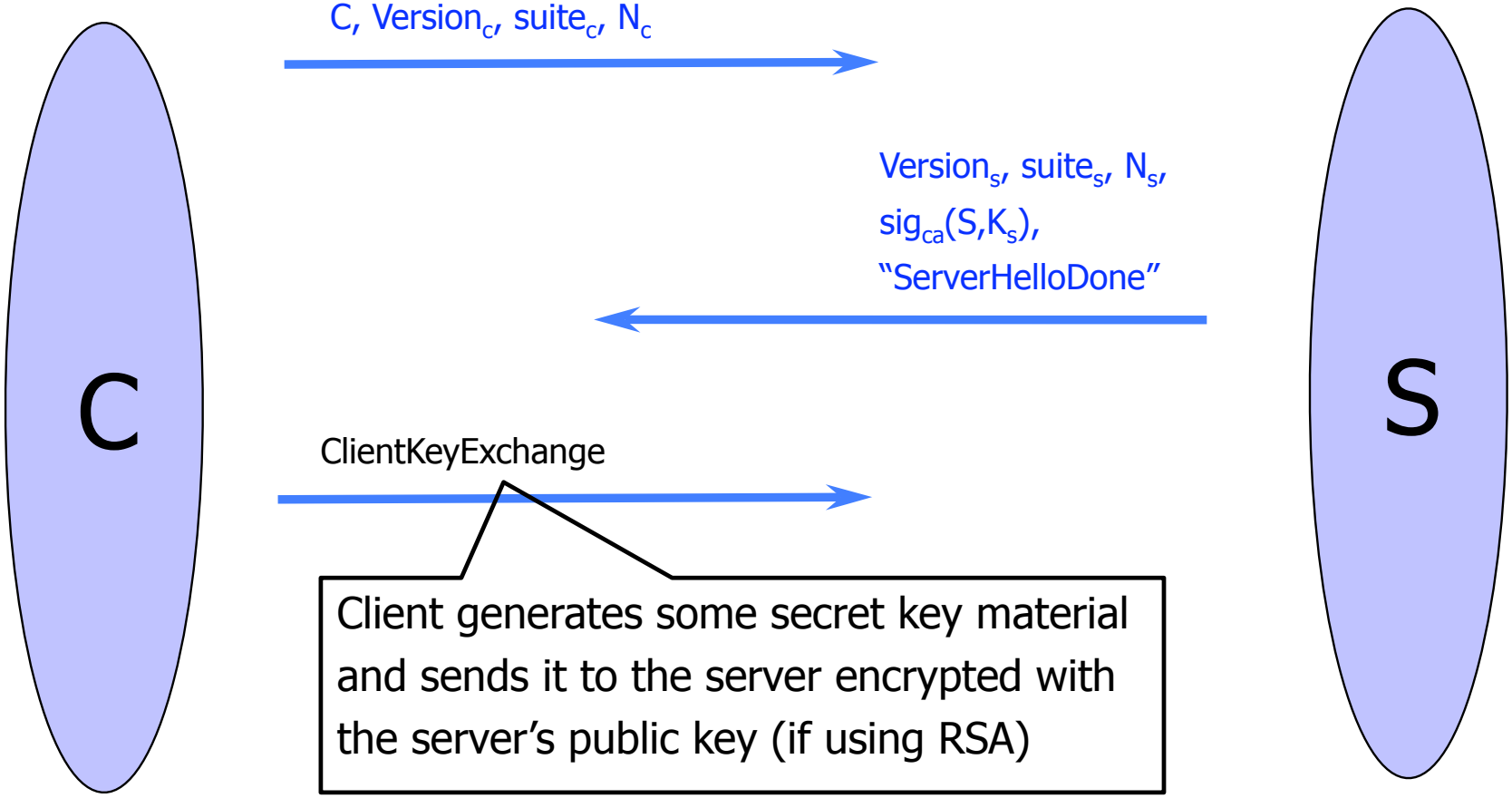
ServerHello



ServerKeyExchange



ClientKeyExchange

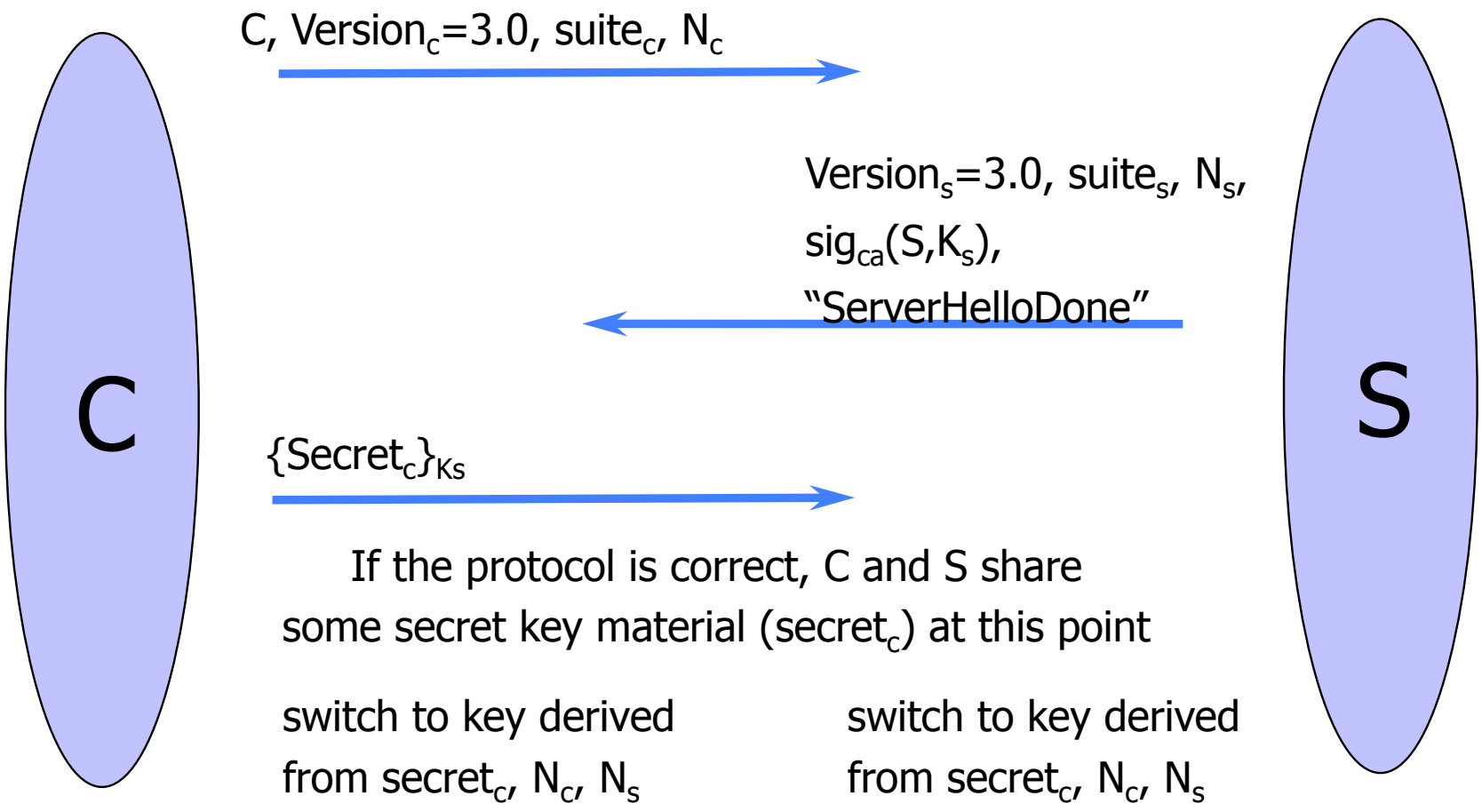


ClientKeyExchange (RFC)

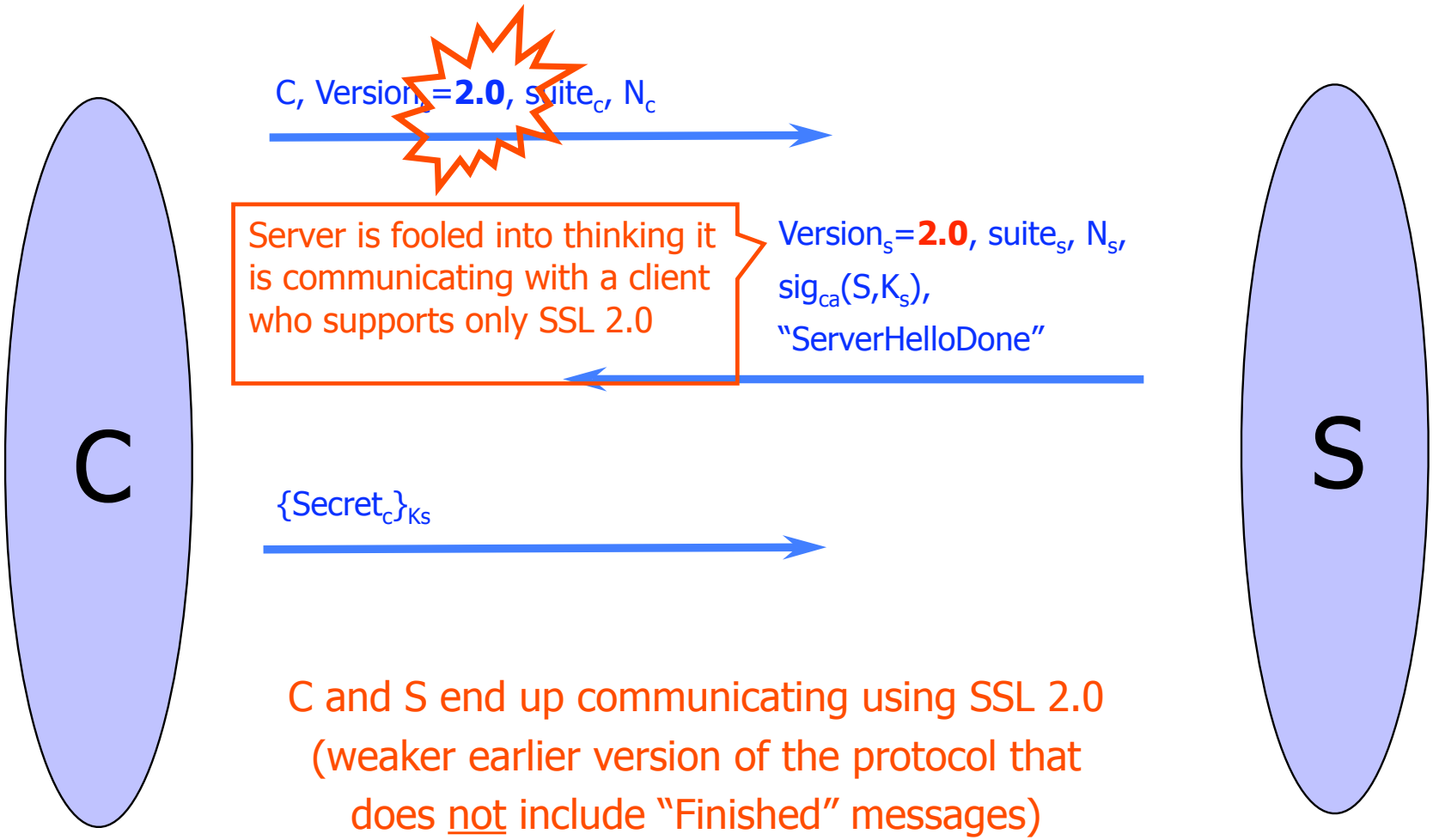
```
struct {
   select (KeyExchangeAlgorithm) {
      case rsa: EncryptedPreMasterSecret;
      case diffie_hellman: ClientDiffieHellmanPublic;
   } exchange_keys
} ClientKeyExchange
struct {

   ProtocolVersion client_version;
   opaque random[46];
} PreMasterSecret
```

Random bits from which symmetric keys will be derived (by hashing them with nonces)

"Core" SSL 3.0 Handshake (Not TLS)



Version Rollback Attack



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

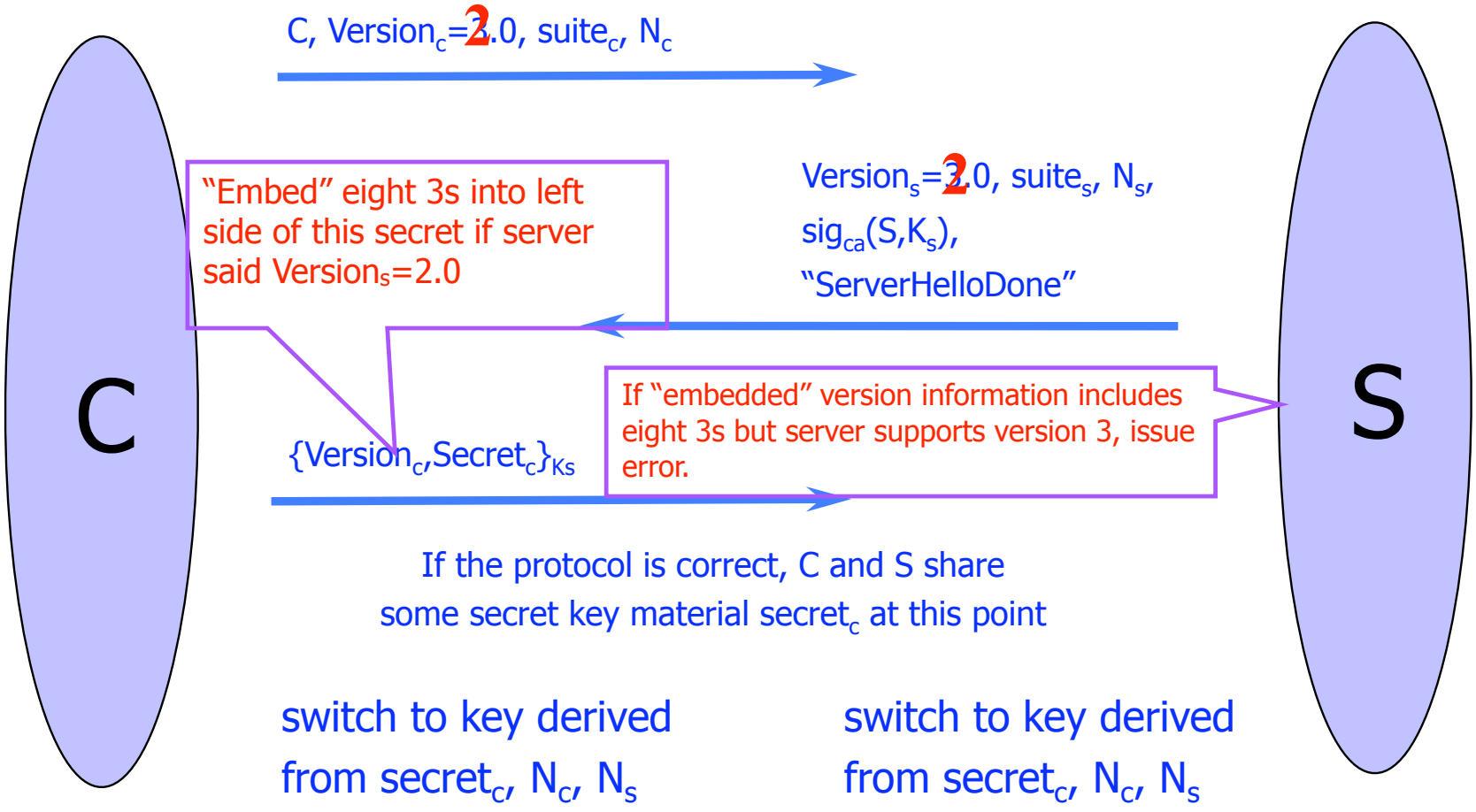
SSL 2.0 Weaknesses (Fixed in 3.0)

- Cipher suite preferences are not authenticated
  - "Cipher suite rollback" attack is possible
- SSL 2.0 uses padding when computing MAC in block cipher modes, but padding length field is not authenticated
  - Attacker can delete bytes from the end of messages
- MAC hash uses only 40 bits in export mode
- No support for certificate chains or non-RSA algorithms, no handshake while session is open

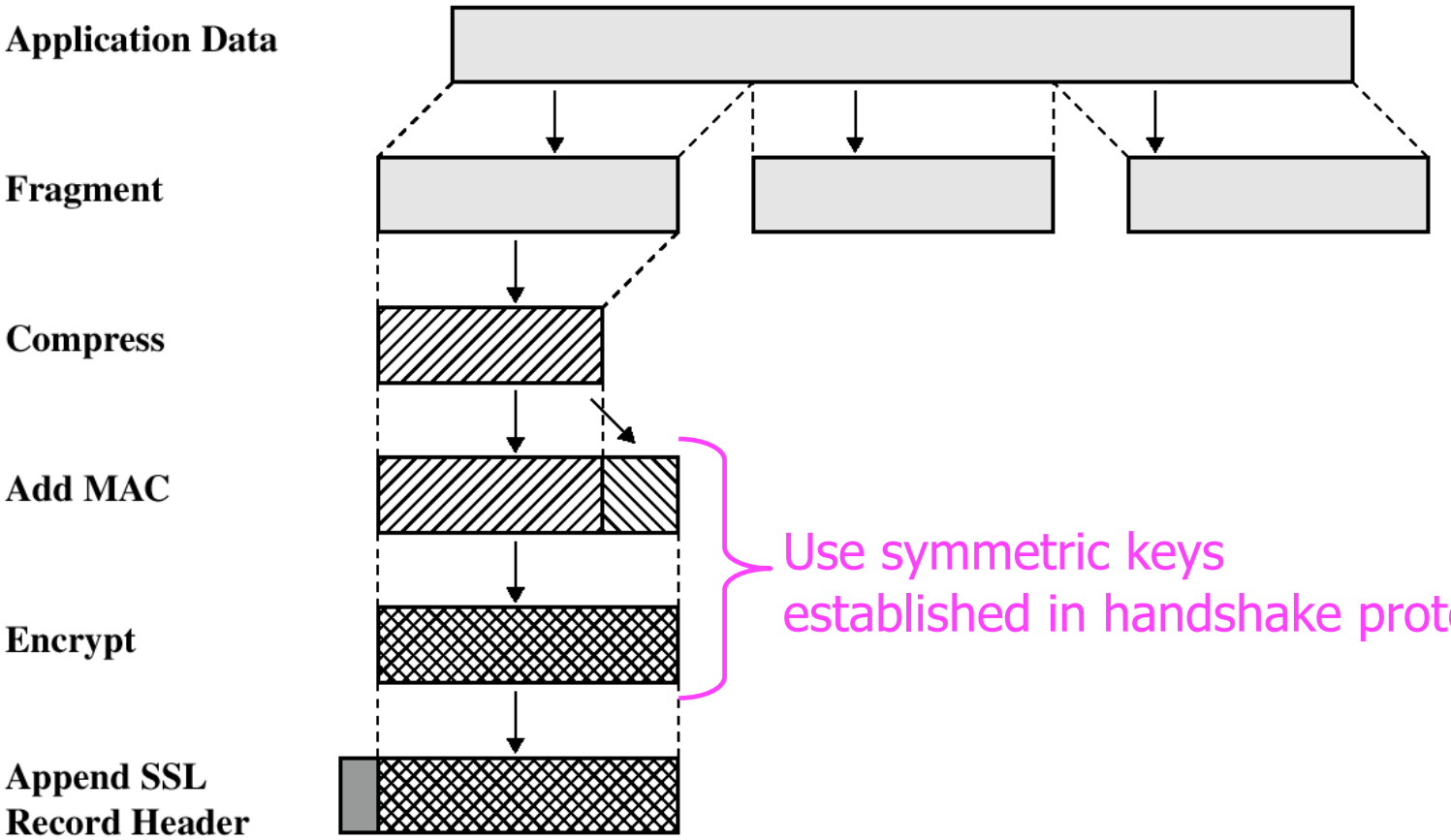
Protocol Rollback Attacks

- 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 (Approximate)

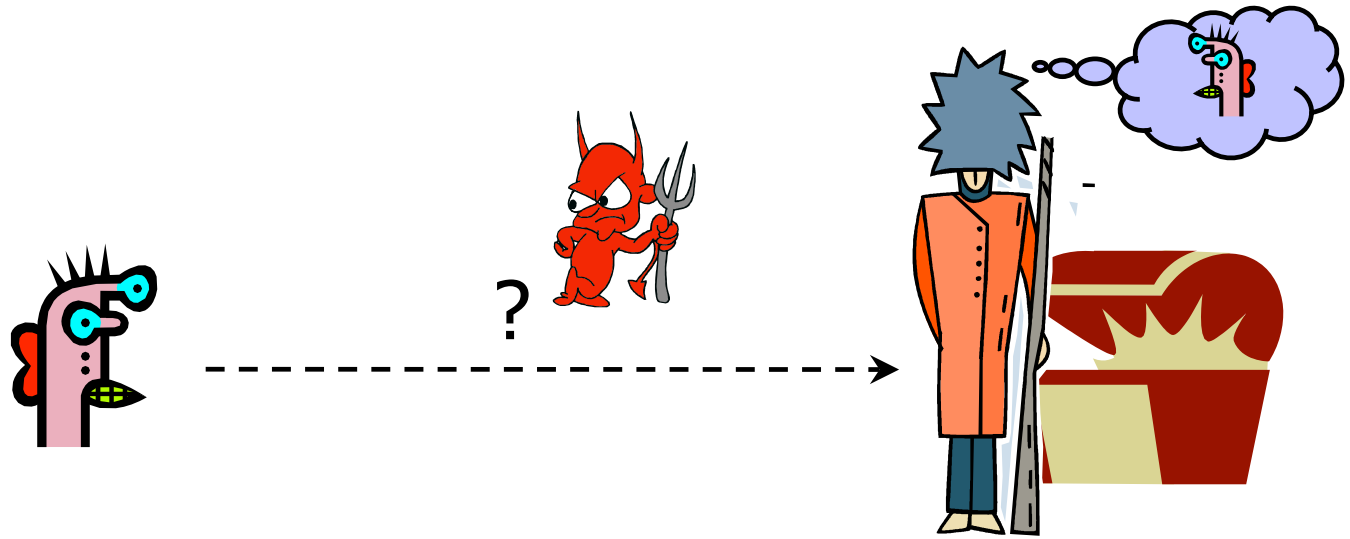


SSL/TLS Record Protection



User Authentication

Basic Problem



How do you prove to someone that
you are who you claim to be?

Any system with access control must solve this problem

Many Ways to Prove Who You Are

- What you know
  - Passwords
  - Secret key
- Where you are
  - IP address
  - Physical location
- What you are
  - Biometrics
- What you have
  - Secure tokens
- All have advantages and disadvantages

Why Authenticate?

- To prevent an attacker from breaking into our account
  - Co-worker, family member, ...
- To prevent an attacker from breaking into any account on our system
  - Unix system
    - Break into single account, then exploit local vulnerability or mount a "stepping stones" attack
  - Calling cards
  - Building
- To prevent an attacker from breaking into any account on any system

Also Need

- Usability!
  - Remember password?
  - Have to bring physical object with us all the time?
- Denial of service
  - Stolen wallet
  - Try to authenticate as you until your account becomes locked
  - What about a military or other mission critical scenario
    - Lock all accounts - system unusable

Password-Based Authentication

- User has a secret password.

  System checks it to authenticate the user.
  - May be vulnerable to eavesdropping when password is communicated from user to system
- How is the password stored?
- How does the system check the password?
- How easy is it to remember the password?
- How easy is it to guess the password?
  - Easy-to-remember passwords tend to be easy to guess
  - Password file is difficult to keep secret

Common usage modes

Amazon = t0p53cr37

UWNetID = f0084r#1

Bank = a2z@m0$;

Image from http://www.interactivetools.com/staff/dave/damons_office/

Common usage modes

- Write down passwords
- Share passwords with others
- Use a single password across multiple sites
  - Amazon.com and Bank of America?
  - UW CSE machines and Facebook?
  - GMail and Facebook?
- Use easy to remember passwords
  - Favorite <something>?
  - Name + <number>?
- Other "authentication" questions
  - Mother's maiden name?

Some anecdotes [Dhamija and Perrig]

- Users taught how to make secure passwords, but chose not to do so

- Reasons:
  - Awkward or difficult
  - No accountability
  - Did not feel that it was important

Social Engineering

- "Hi, I'm the CEO's assistant. I need you to reset his password right away. He's stuck in an airport and can't log in! He lost the paper that he wrote the password on.
- "What do you mean you can't do it!? Do you really want me to tell him that you're preventing him from closing this major deal?
- "Great! That's really helpful. You have no idea how important this is. Please set the password to ABCDEFG. He'll reset it again himself right away.
- "Thanks!"

University of Sydney Study [Greening '96]

- 336 CS students emailed message asking them to supply their password
  - Pretext: in order to "validate" the password database after a suspected break-in
- 138 students returned their password
- 30 returned invalid password
- 200 changed their password
- (Not disjoint)

- Still, 138 is a lot!

Awkward

- How many times do you have to enter your password before it actually works?
  - Sometimes quite a few for me! (Unless I type extra slowly.)
- Interrupts normal activity
  - Do you lock your computer when you leave for 5 minutes?
  - Do you have to enter a password when your computer first boots? (Sometimes it's an option.)
- And memorability is an issue!

Memorability [Anderson]

- Hard to remember many PINs and passwords
- One bank had this idea
  - If pin is 2256, write your favorite 4-letter word in this grid
  - Then put random letters everywhere else

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

Memorability [Anderson]

- Problem!
- Normally 10000 choices for the PIN --- hard to guess on the first try
- Now, only a few dozen possible English words --- easy to guess on first try!

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				