CSE 484 and CSE M 584 (Winter 2009)

Public key 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 ...

Exponentiation

- How to compute g^x mod N?

Exponentiation

- How to compute M^x mod N?
- Say, x = 13
- Sums of power of 2, x = 8+4+1 = $2^3+2^2+2^0$
- Can also write x in binary, e.g., x = 1101
- Can solve by repeated squaring
  - y = 1;
  - y = y^2 * M mod N // y = M
  - y = y^2 * M mod N // y = M^2 *M = M^{2+1} = M^3
  - y = y^2 mod N // y = $(M^3)^2$ = M^6
  - y = y^2 * M mod N // y = $(M^6)^2$ *M = M^{12+1} = M^{13} = M^x

Timing attacks

Collect timings for exponentiation with a bunch of messages M1, M2, ... (e.g., RSA signing operations with a private exponent)

i	$b_i = 0$	$b_i = 1$	Comp	Meas
3	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$		
2	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$		
1	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$	X1 secs	
0	$y = y^2 \bmod N$	$y = y^2 * M1 \bmod N$		Y1 secs

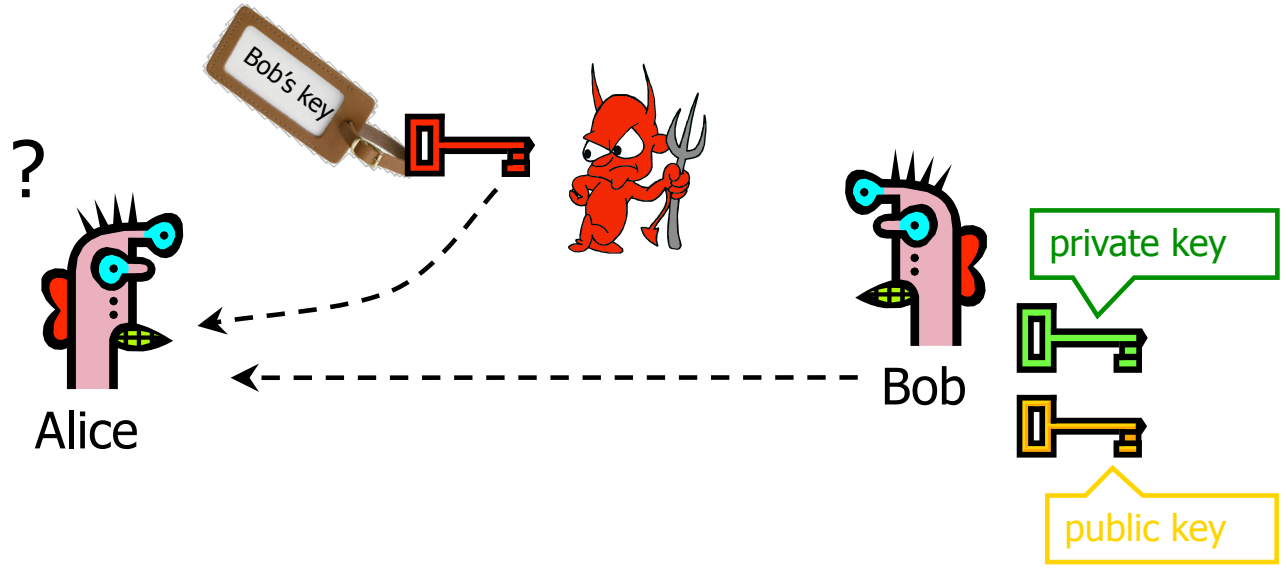
i	$b_i = 0$	$b_i = 1$	Comp	Meas
3	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$		
2	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$		
1	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$	X2 secs	
0	$y = y^2 \bmod N$	$y = y^2 * M2 \bmod N$		Y2 secs

Timing attacks

- If $b_1 = 1$, then set of { Yj - Xj | j in {1,2, ..} } has distribution with "small" variance (due to time for final step, i=0)
  - "Guess" was correct when we computed X1, X2, ...
- If $b_1 = 0$, then set of { Yj - Xj | j in {1,2, ..} } has distribution with "large" variance (due to time for final step, i=0)
  - "Guess" was incorrect when we computed X1, X2, ...
- Strategy: Force user to sign large number of messages M1, M2, ... Record timings for signing.
- Iteratively learn bits of key by using above property.

PKI

Authenticity of Public Keys

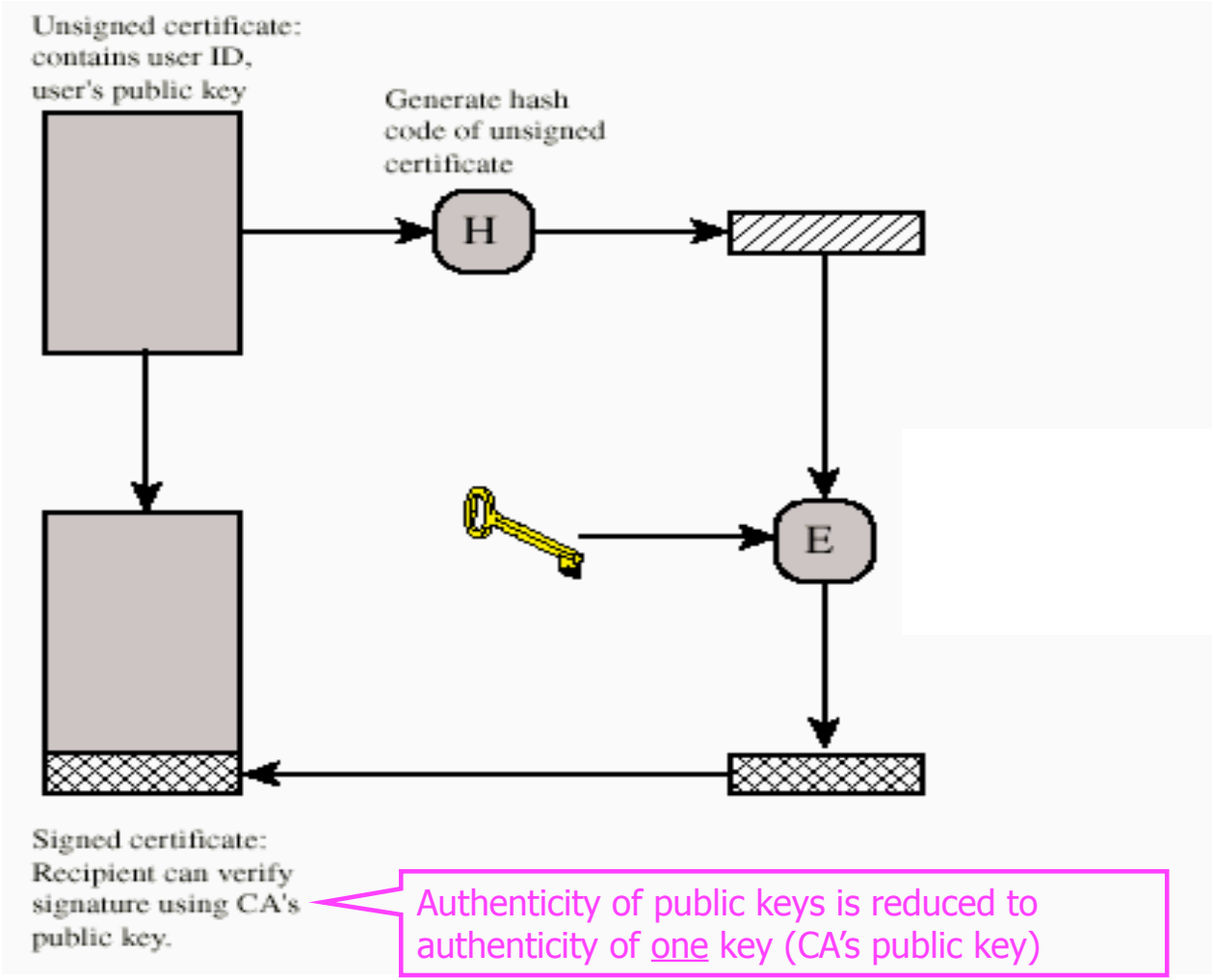


Problem: How does Alice know that the public key she received is really Bob's public key?

Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery and tampering
- Public-key certificate
  - Signed statement specifying the key and identity
    - sig_{Alice}("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

Using Public-Key Certificates



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
    - $sig_{Verisign}(\text{“UW”}, PK_{UW})$, $sig_{UW}(\text{“Alice”}, PK_A)$
  - What happens if root authority is ever compromised?

Many Challenges

Spoofing URLs With Unicode

Posted by timothy **on Mon May 27, '02 09:48 PM**
from the **there-is-a-problem-with-this-certificate** dept.

Embedded Geek writes:

> "Scientific American has an interesting article about how a pair of students at the Technion-Israel Institute of Technology registered "microsoft.com" with Verisign, using the Russian Cyrillic letters "c" and "o". Even though it is a completely different domain, the two display identically (the article uses the term "homograph"). The work was done for a paper in the **Communications of the ACM** (the paper itself is not online). The article characterizes attacks using this spoof as "scary, if not entirely probable," assuming that a hacker would have to first take over a page at another site. I disagree: sending out a mail message with the URL waiting to be clicked ("Bill Gates will send you ten dollars!") is just one alternate technique. While security problems with Unicode have been noted here before, this might be a new twist."

Many Challenges

Shmoo Group Finds Exploit For non-IE Browsers

Posted by Hemos on Mon Feb 07, '05 11:30 AM
from the **even-mozilla-is-guilty** dept.

shut_up_man writes

> "Saw this on Boing Boing: East coast hacker con Shmoocon ended today and they had a nasty browser exploit to show off... using International Domain Name (IDN) character support to display fake domain names in links and the address bar. Their examples use Paypal (with SSL too) and this looks very useful for phishing attacks. Interesting note that it works in every browser *except* IE (which makes this exploit a lot less dangerous in the end, I suppose)."

v The reason IE isn't vulnerable is because it doesn't natively support IDN; with the right plug-in, it too is vulnerable.

Many Challenges

CCC Create a Rogue CA Certificate

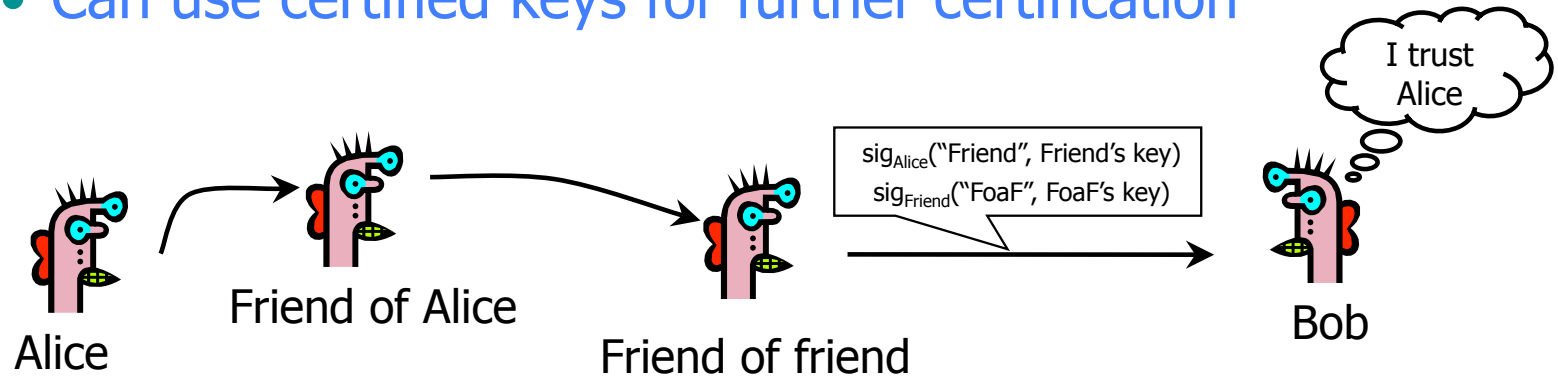
Posted by CmdrTaco on Tue Dec 30, 2008 12:14 PM
from the **they-even-faked-this-dept** dept.

t3rmin4t0r writes

> "Just when you were breathing easy about Kaminsky, DNS and the word hijacking, by repeating the word SSL in your head, the hackers at CCC were busy at work making a hash of SSL certificate security. Here's the scoop on how they set up their own rogue CA, by (from what I can figure) reversing the hash and engineering a collision up in MD5 space. Until now, MD5 collisions have been ignored because nobody would put in that much effort to create a useful dummy file, but a CA certificate for phishing seems juicy enough to be fodder for the botnets now."

Alternative: "Web of Trust"

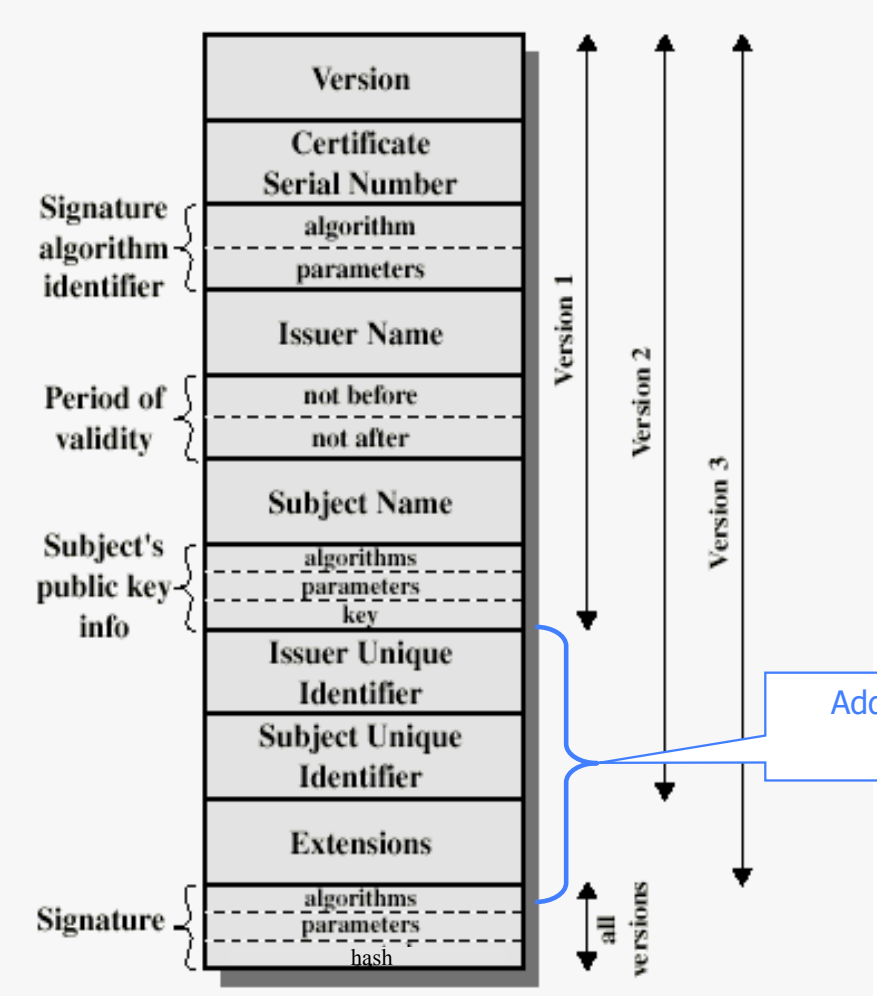
- Used in PGP (Pretty Good Privacy)
- Instead of a single root certificate authority, each person has a set of keys they "trust"
  - If public-key certificate is signed by one of the "trusted" keys, the public key contained in it will be deemed valid
- Trust can be transitive
  - Can use certified keys for further certification



X.509 Authentication Service

- Internet standard (1988-2000)
- Specifies certificate format
  - X.509 certificates are used in IPSec and SSL/TLS
- Specifies certificate directory service
  - For retrieving other users' CA-certified public keys
- Specifies a set of authentication protocols
  - For proving identity using public-key signatures
- Does not specify crypto algorithms
  - Can use it with any digital signature scheme and hash function, but hashing is required before signing

X.509 Certificate



Added in X.509 versions 2 and 3 to address usability and security problems

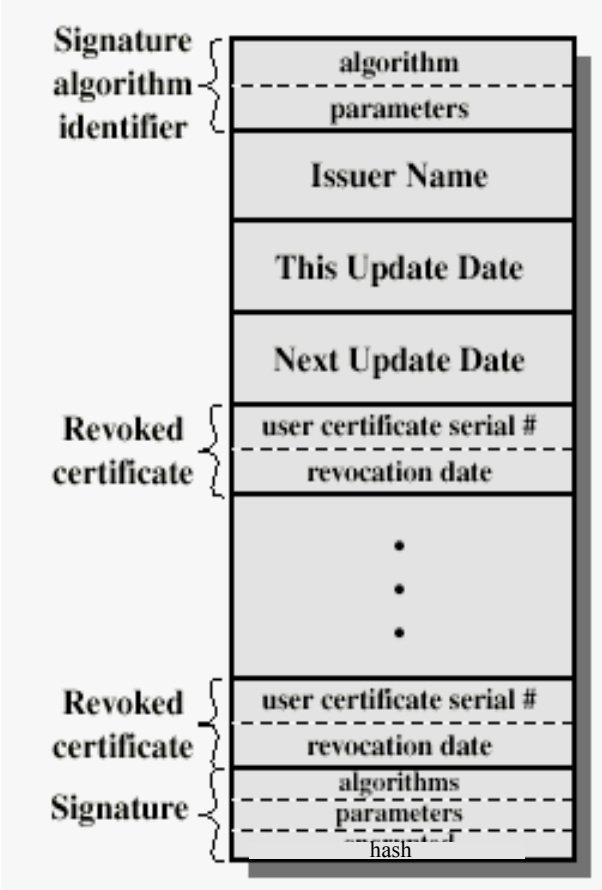
Certificate Revocation

- 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 certificate 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

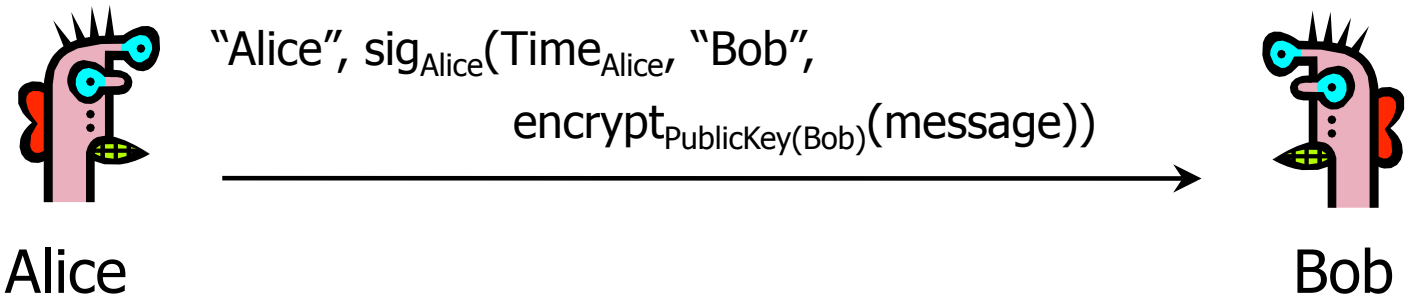
- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor
- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates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

X.509 Certificate Revocation List



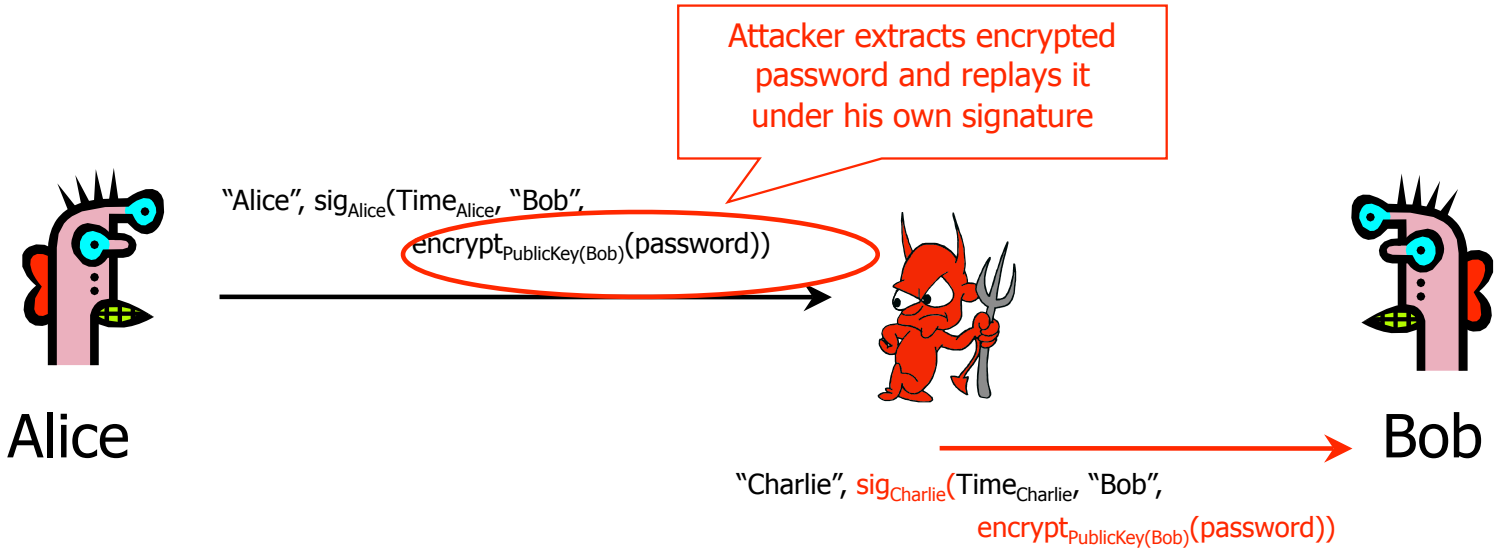
Because certificate serial numbers must be unique within each CA, this is enough to identify the certificate

X.509 Version 1



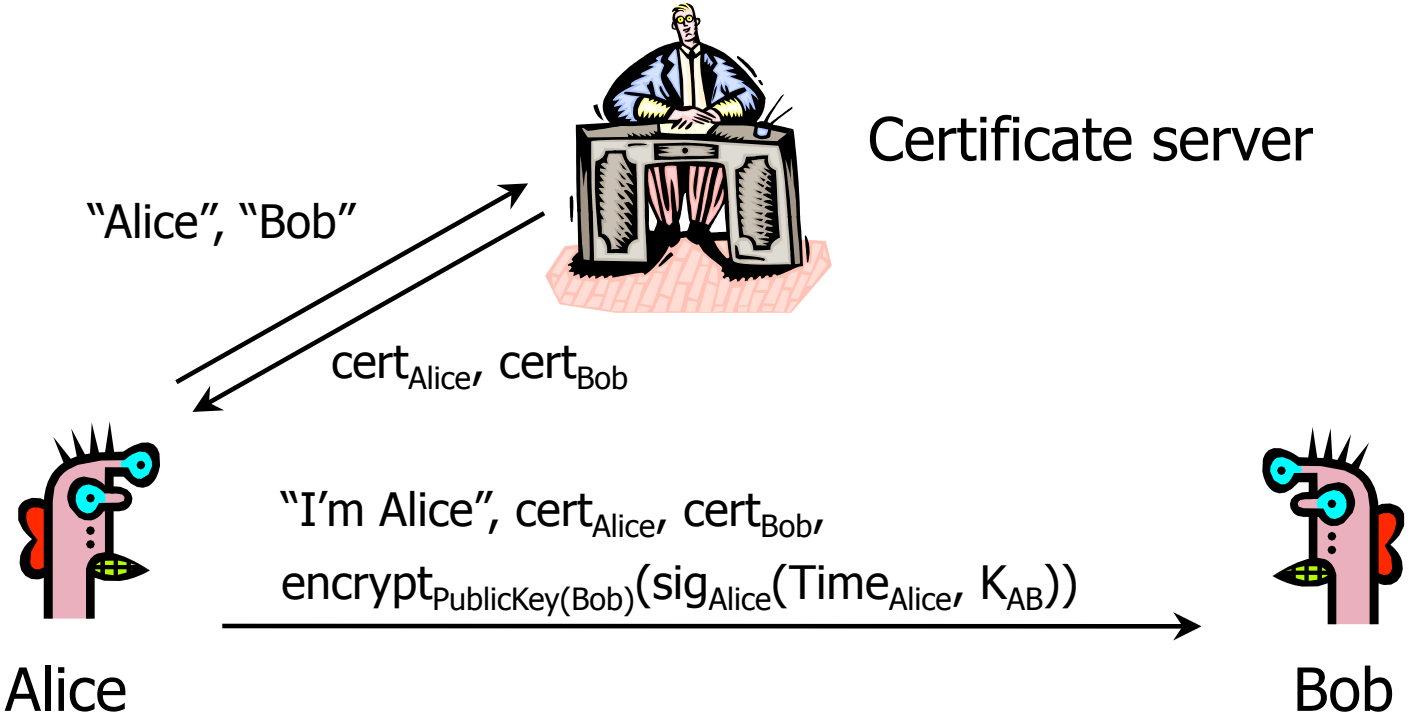
- Encrypt, then sign for authenticated encryption
  - Goal: achieve both confidentiality and authentication
  - E.g., encrypted, signed password for access control
- Does this work?

Attack on X.509 Version 1



- Receiving encrypted password under signature does not mean that the sender actually knows the password!

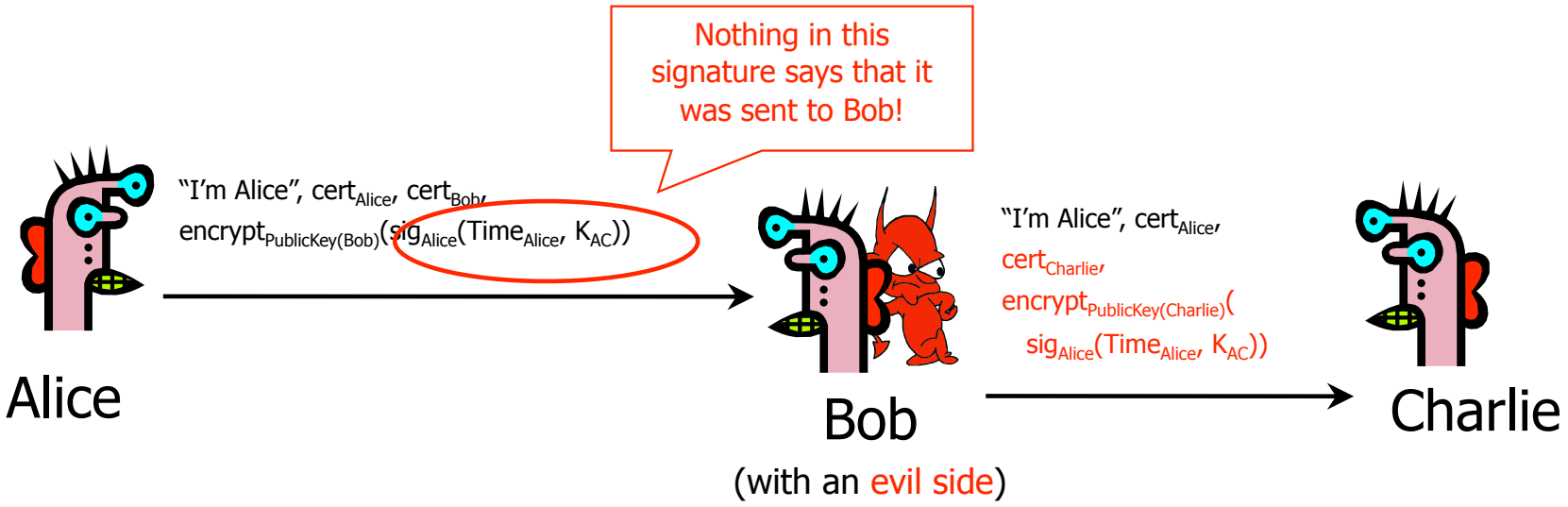
Denning-Sacco Protocol



- Goal: establish a new shared key K_{AB} with the help of a trusted certificate service

slide

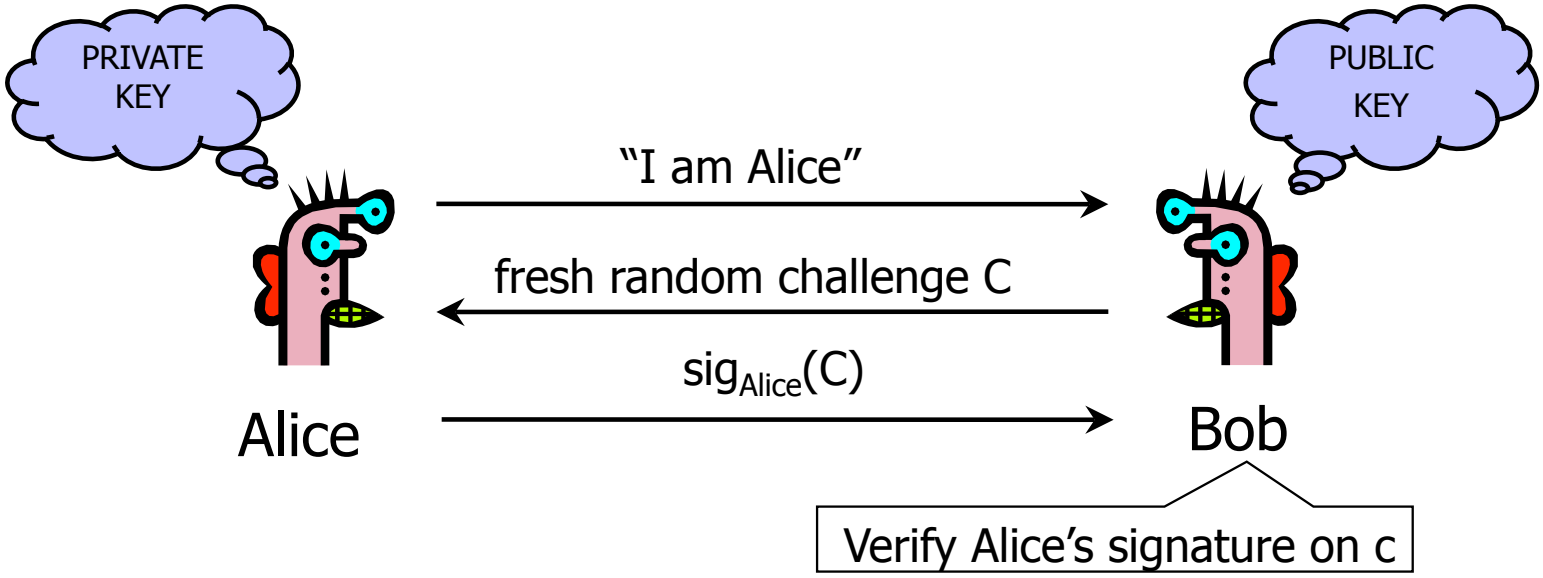
Attack on Denning-Sacco



- Alice's signature is insufficiently explicit
  - Does not say to whom and why it was sent
- Alice's signature can be used to impersonate her

slide

Authentication with Public Keys

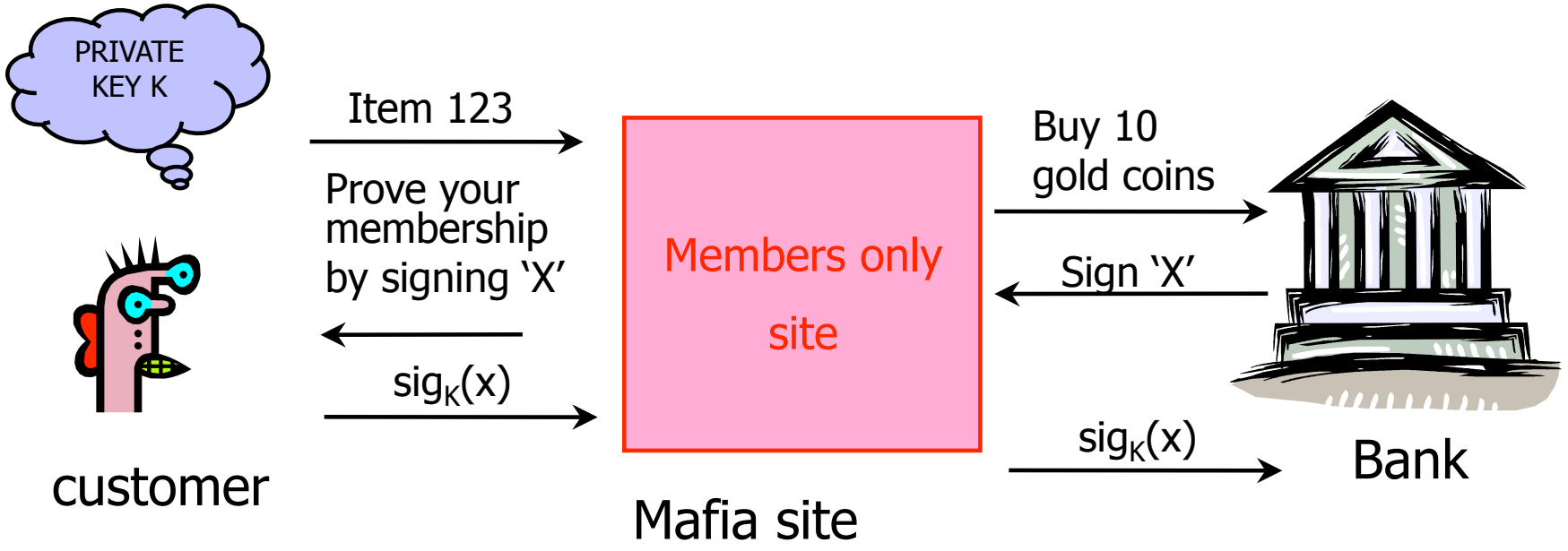


1. Only Alice can create a valid signature
2. Signature is on a fresh, unpredictable challenge

Potential problem: Alice will sign anything

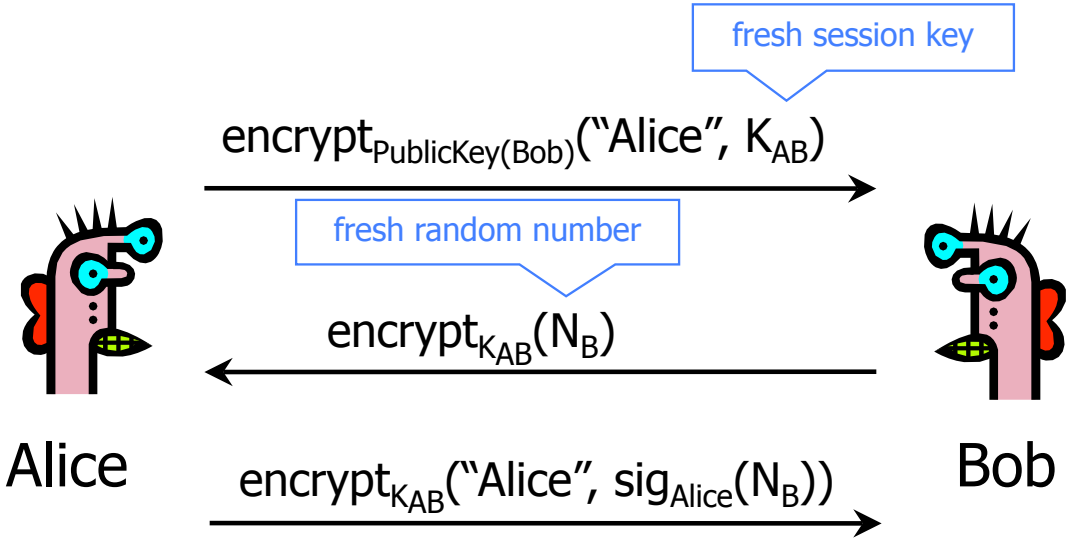
slide

Mafia-in-the-Middle Attack [from Anderson's book]



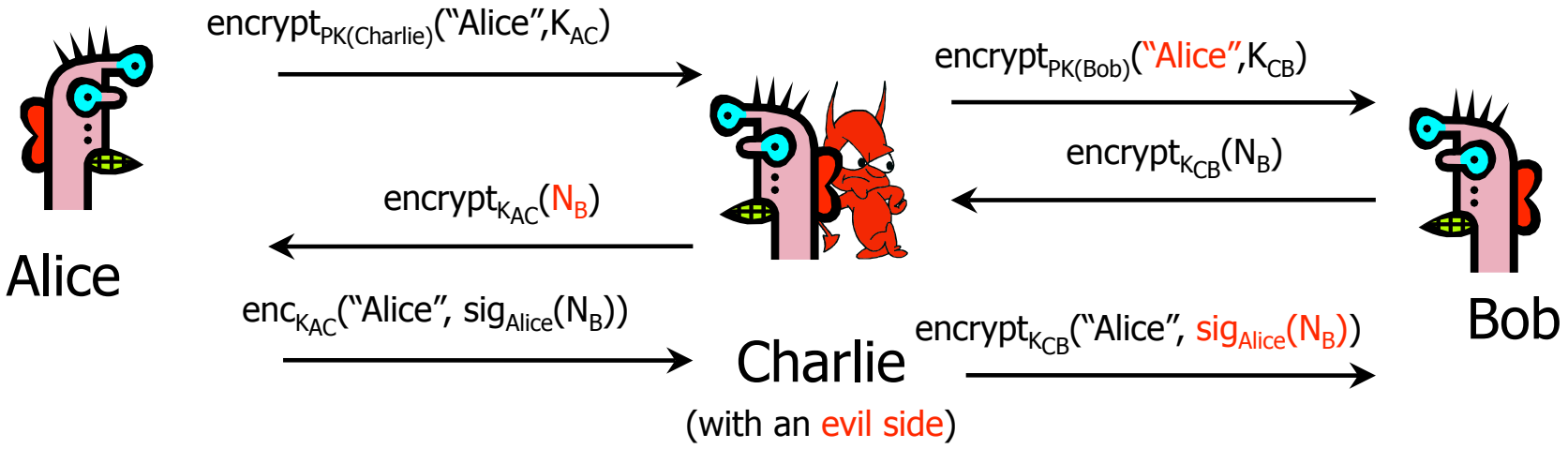
slide

Early Version of SSL (Simplified)



- Bob's reasoning: I must be talking to Alice because...
  - Whoever signed N_B knows Alice's private key... Only Alice knows her private key... Alice must have signed N_B... N_B is fresh and random and I sent it encrypted under K_{AB}... Alice could have learned N_B only if she knows K_{AB}... She must be the person who sent me K_{AB} in the first message...

Breaking Early SSL



- Charlie uses his legitimate conversation with Alice to impersonate Alice to Bob
  - Information signed by Alice is not sufficiently explicit

Secure Sessions

- Secure sessions are among the most important applications in network security
  - Enable us to talk securely on an insecure network
- Goal: secure bi-directional communication channel between two parties
  - The channel must provide confidentiality
    - Third party cannot read messages on the channel
  - The channel must provide authentication
    - Each party must be sure who the other party is
  - Other desirable properies: integrity, protection against denial of service, anonymity against eavesdroppers

slide

Key Establishment Protocols

- Common implementation of secure sessions: establish a secret key known only to two parties
  - Can then use block ciphers for confidentiality, HMAC for authentication, and so on
- Challenge: how to establish a secret key using only public information
- Even if the two parties share a long-term secret, a fresh key should be created for each session
  - Long-term secrets are valuable; want to use them as sparingly as possible to limit exposure and the damage if the key is compromised

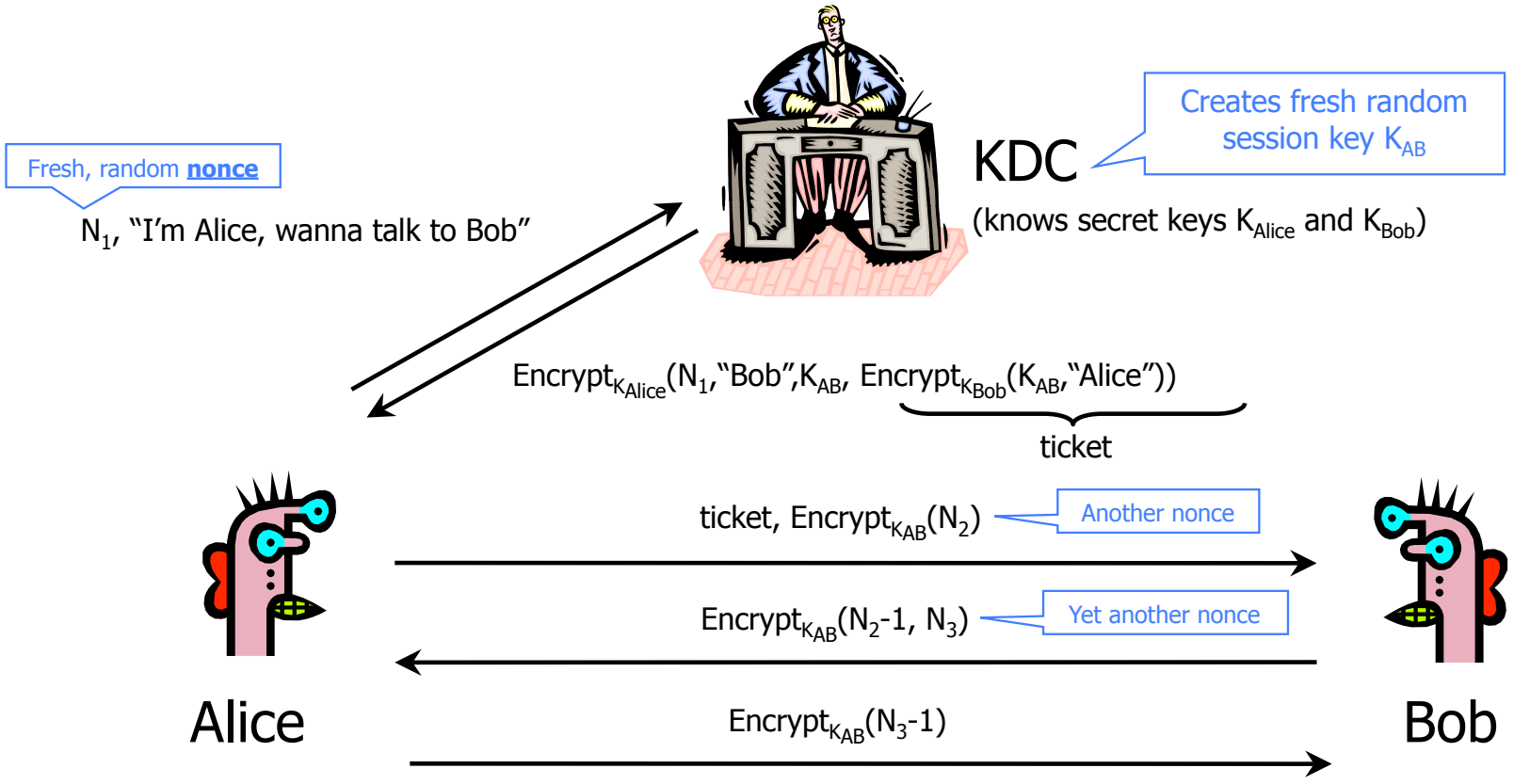
slide

Key Establishment Techniques

- Use a trusted key distribution center (KDC)
  - Every party shares a pairwise secret key with KDC
  - KDC creates a new random session key and then distributes it, encrypted under the pairwise keys
    - Example: Kerberos
- Use public-key cryptography
  - Diffie-Hellman authenticated with signatures
    - Example: IKE (Internet Key Exchange)
  - One party creates a random key, sends it encrypted under the other party's public key
    - Example: TLS (Transport Layer Security)

slide

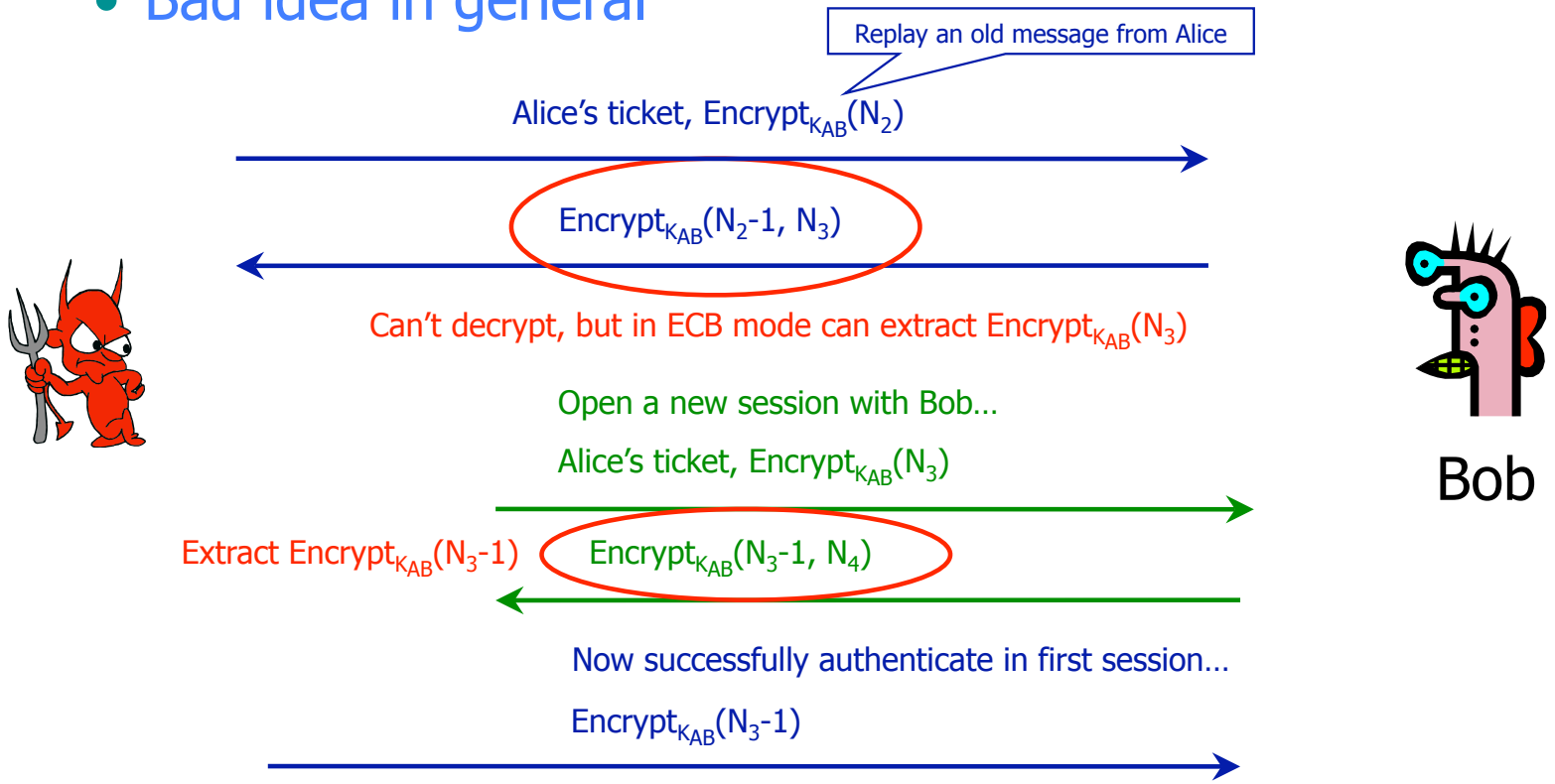
Private-Key Needham-Schroeder



slide

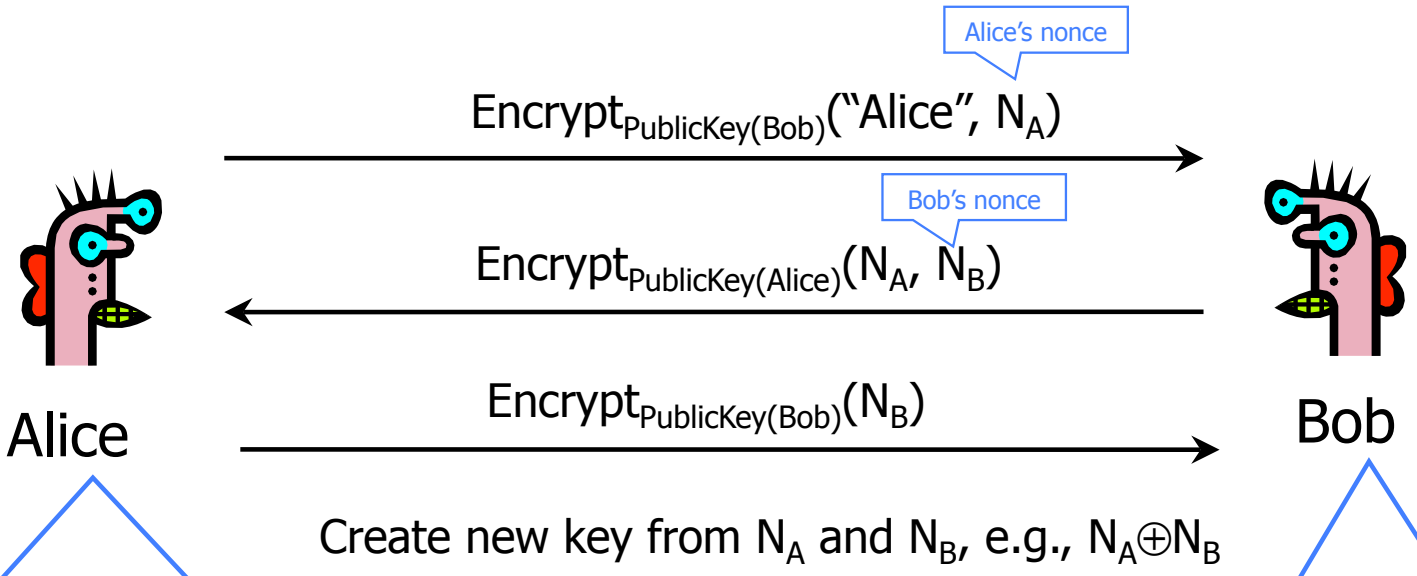
Weird Reflection Attack

- Suppose symmetric encryption is in ECB mode…
  - Bad idea in general



slide

Public-Key Needham-Schroeder



Alice's reasoning:

- The only person who could know N_A is the person who decrypted 1st message
- Only Bob can decrypt message encrypted with Bob's public key
- Therefore, Bob is on the other end of the line

Bob is authenticated!

Bob's reasoning:

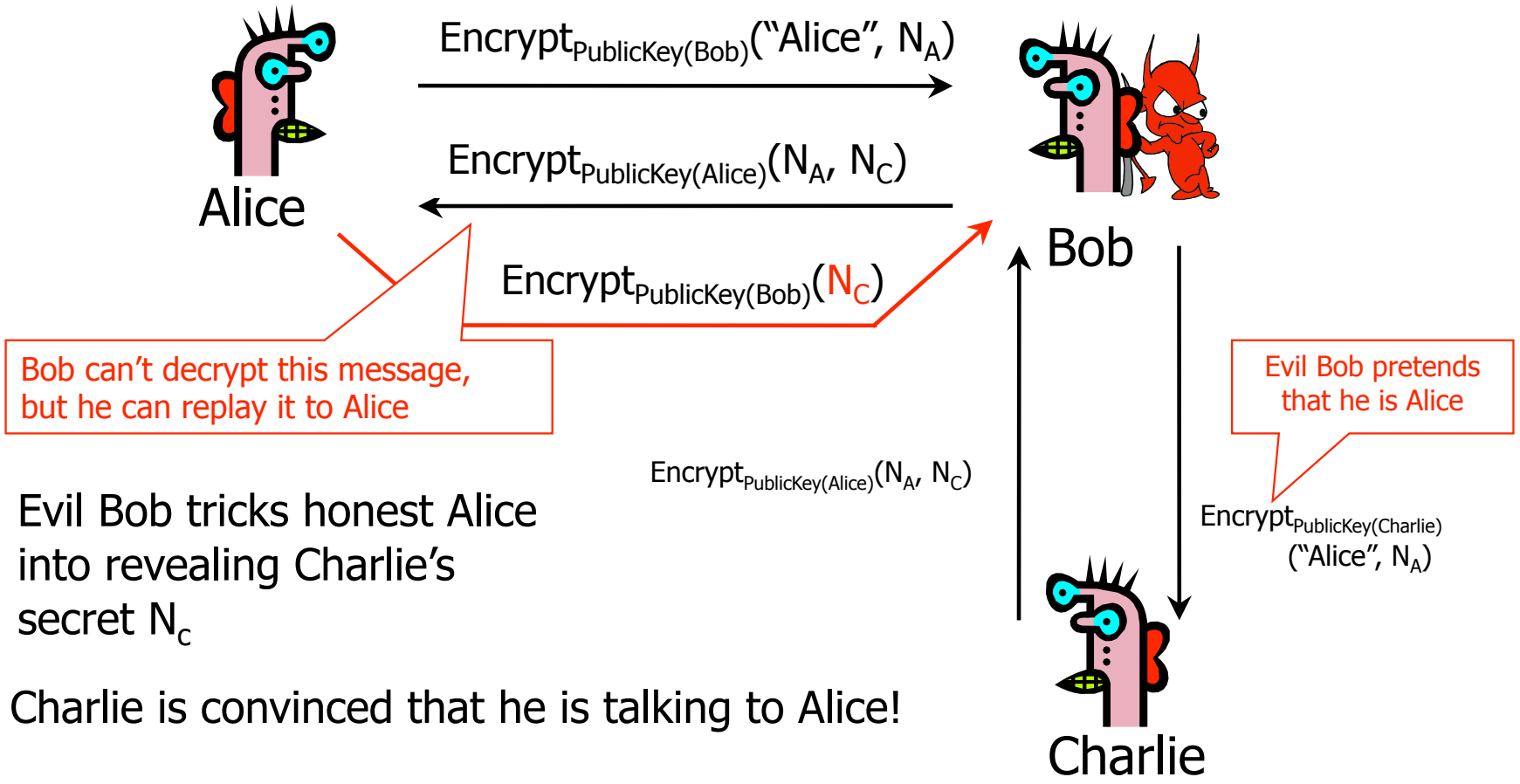
- The only way to learn N_B is to decrypt 2nd message
- Only Alice can decrypt 2nd message
- Therefore, Alice is on the other end

Alice is authenticated!

slide

Attack on Needham-Schroeder

[published by Gavin Lowe]



Evil Bob tricks honest Alice into revealing Charlie's secret N_c

Charlie is convinced that he is talking to Alice!

slide

Lessons of Needham-Schroeder

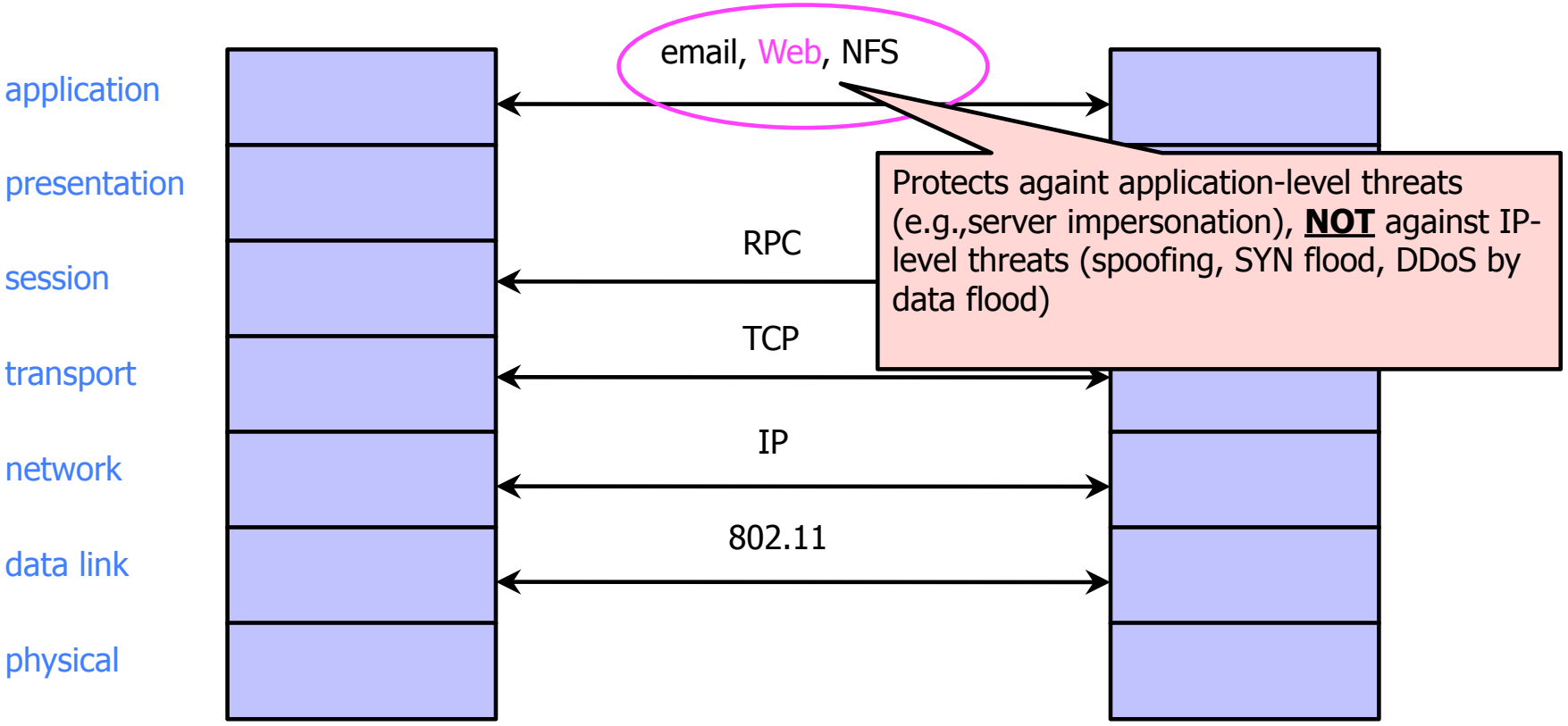
- Yet another example of faulty reasoning
  - Alice is correct that Bob must have decrypted $Encrypt_{PublicKey(Bob)}$("Alice", N_A), but this does not mean that $Encrypt_{PublicKey(Alice)}$($N_A$, N_B) came from Bob
- It is important to realize limitations of protocols
  - The attack requires that Alice willingly talk to attacker
    - Attacker uses a legitimate conversation with Alice to impersonate Alice to Charlie
  - Needham and Schroeder intended this protocol to be used by well-behaved workstations on an insecure network. In their setting, the protocol is correct!

slide

What is SSL / TLS?

- Transport Layer Security protocol, version 1.0
  - 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
- Based on Secure Sockets Layers protocol, ver 3.0
  - Same protocol design, different algorithms
- Deployed in nearly every Web browser

Application-Level Protection



slide

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 MAC; can run on any port

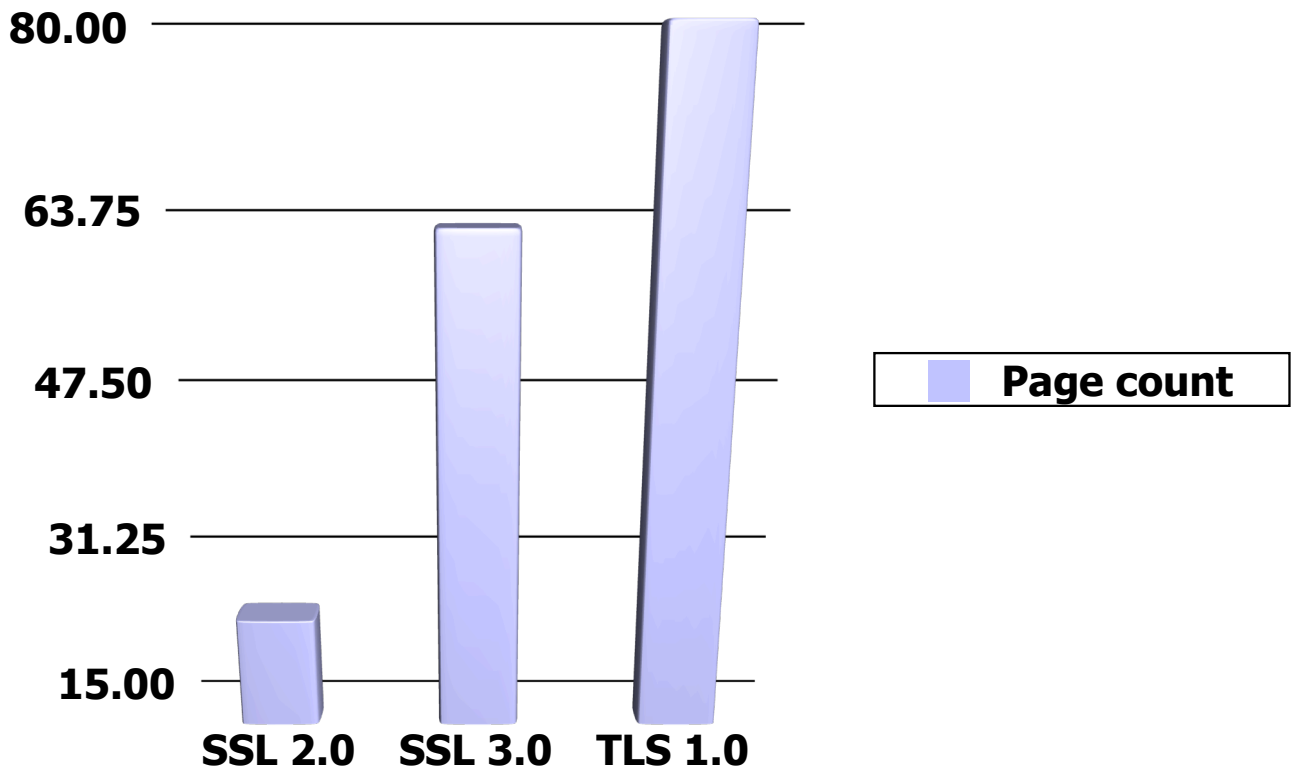
slide

"Request for Comments"

- Network protocols are usually disseminated in the form of an RFC
- TLS version 1.0 is described in RFC 2246
- 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

slide

Evolution of the SSL/TLS RFC



slide

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

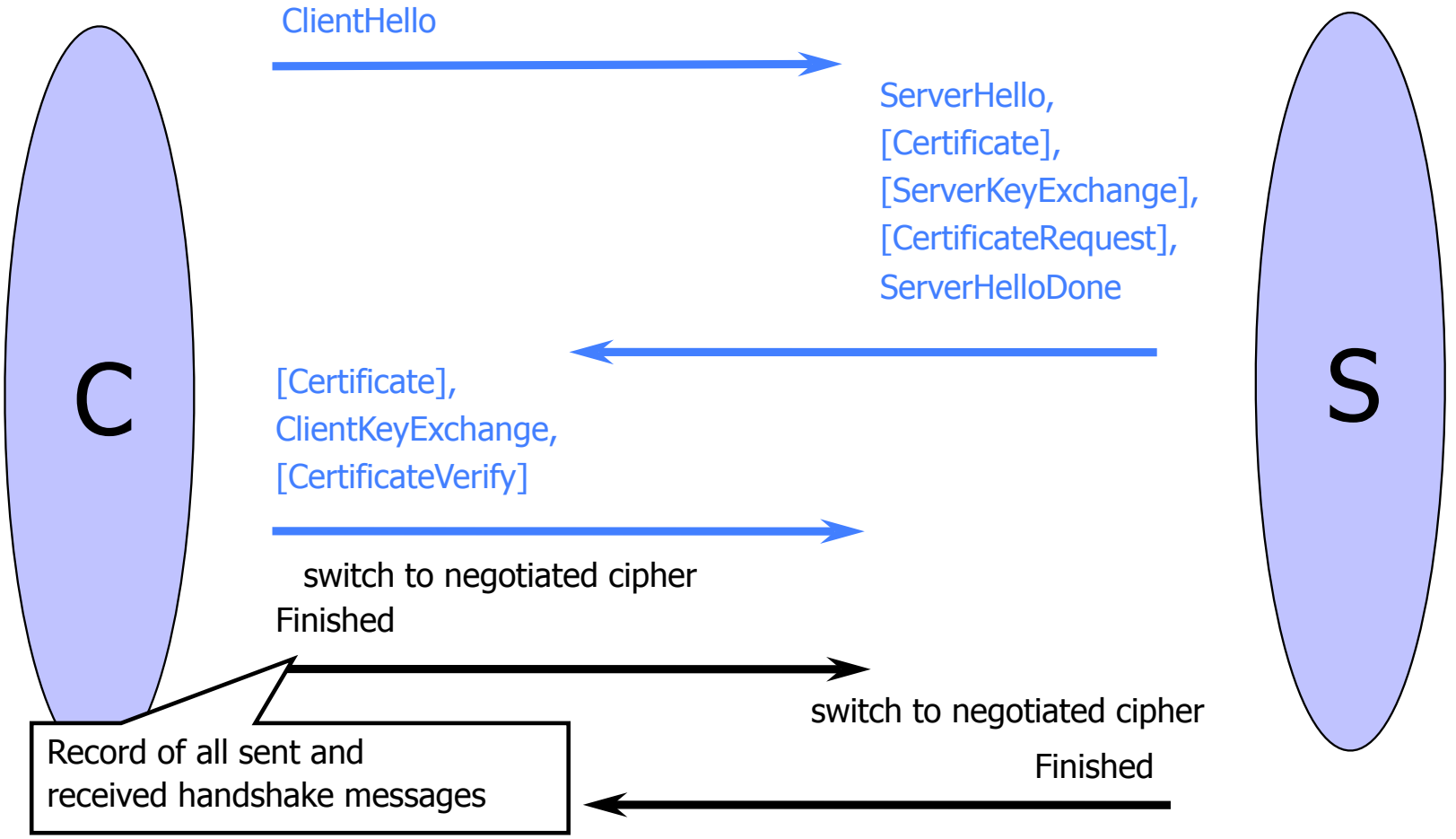
slide

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

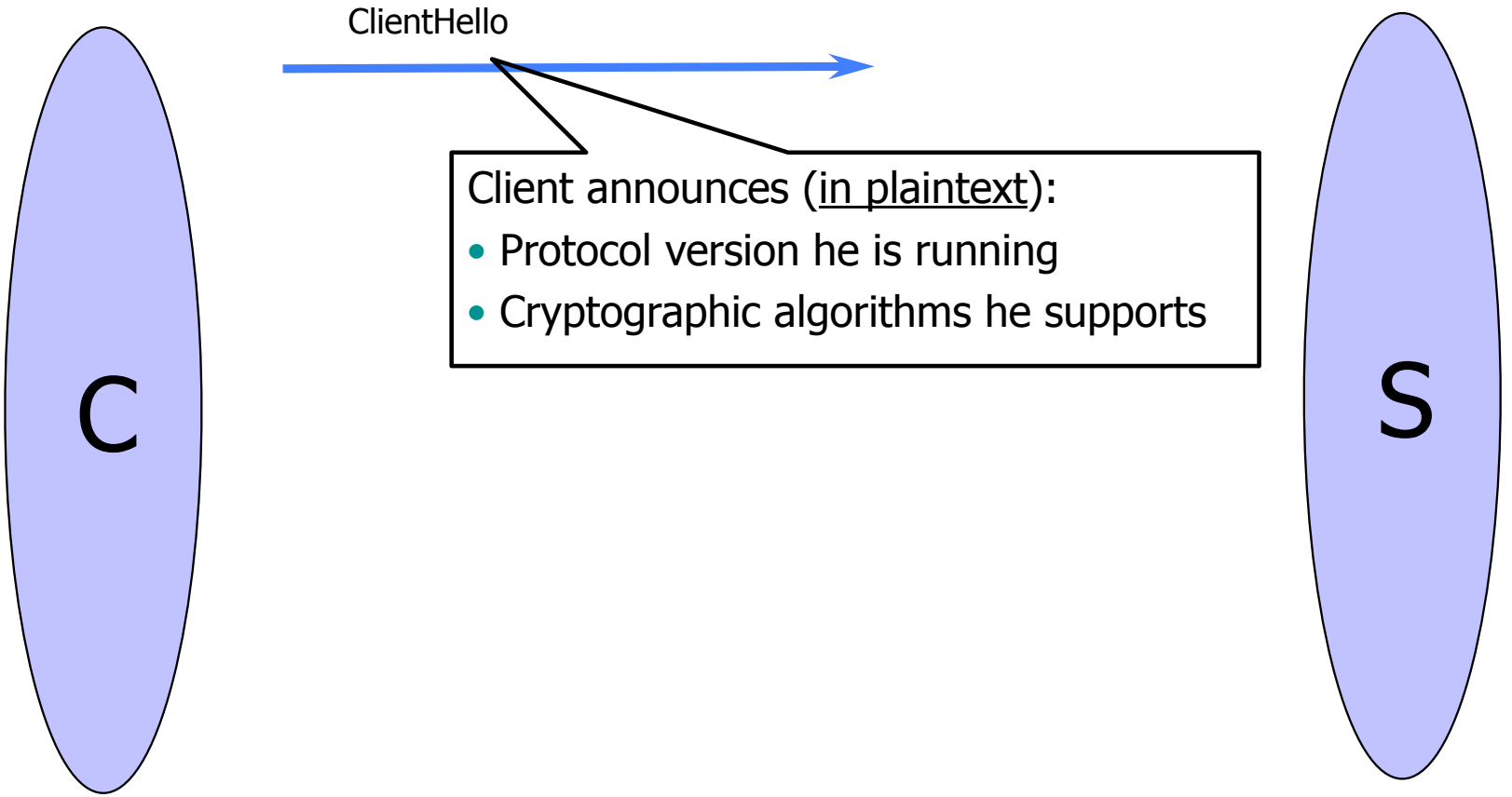
slide

Handshake Protocol Structure



slide

ClientHello



slide

ClientHello (RFC)

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

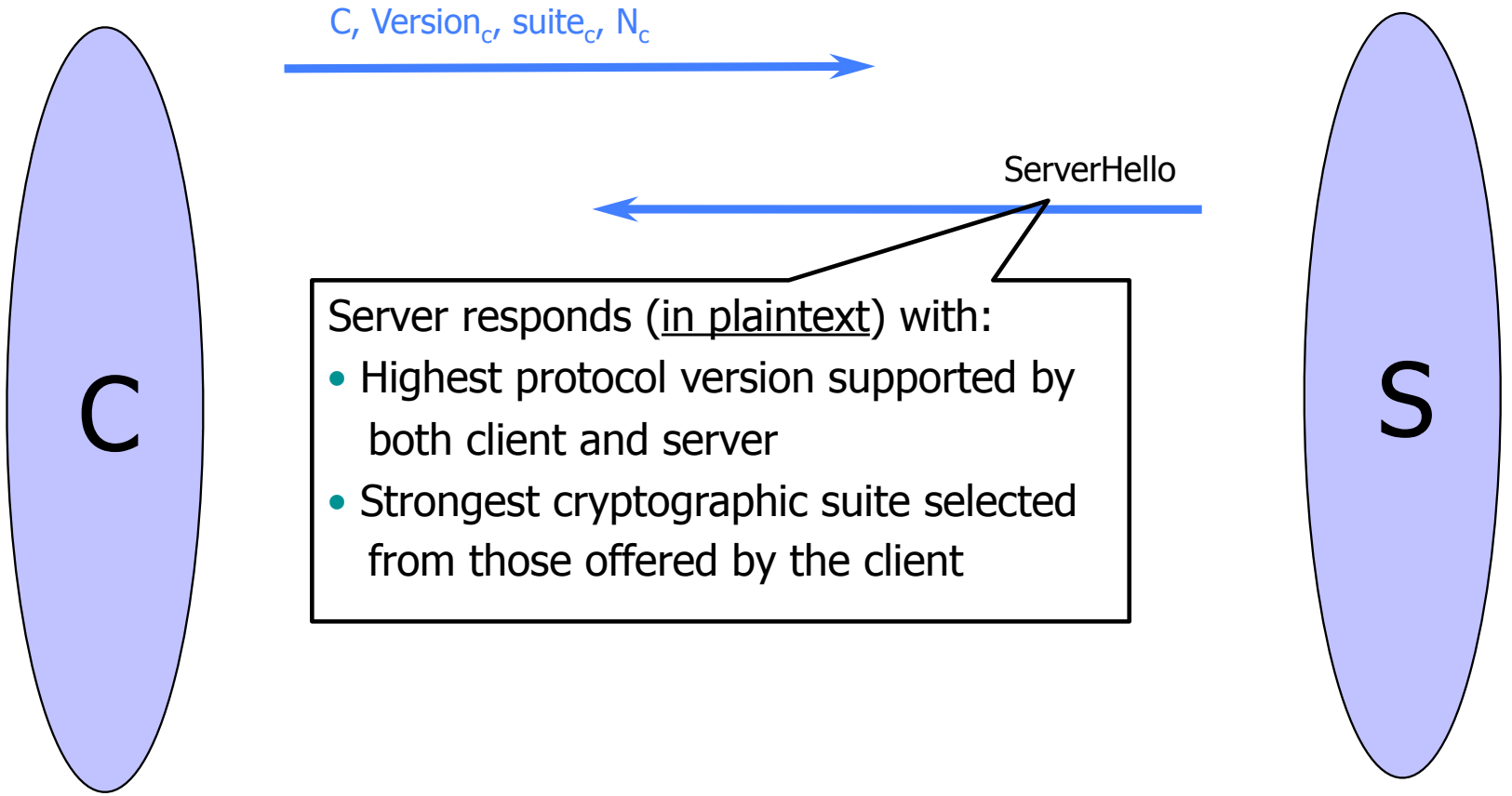
Highest version of the protocol supported by the client

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)

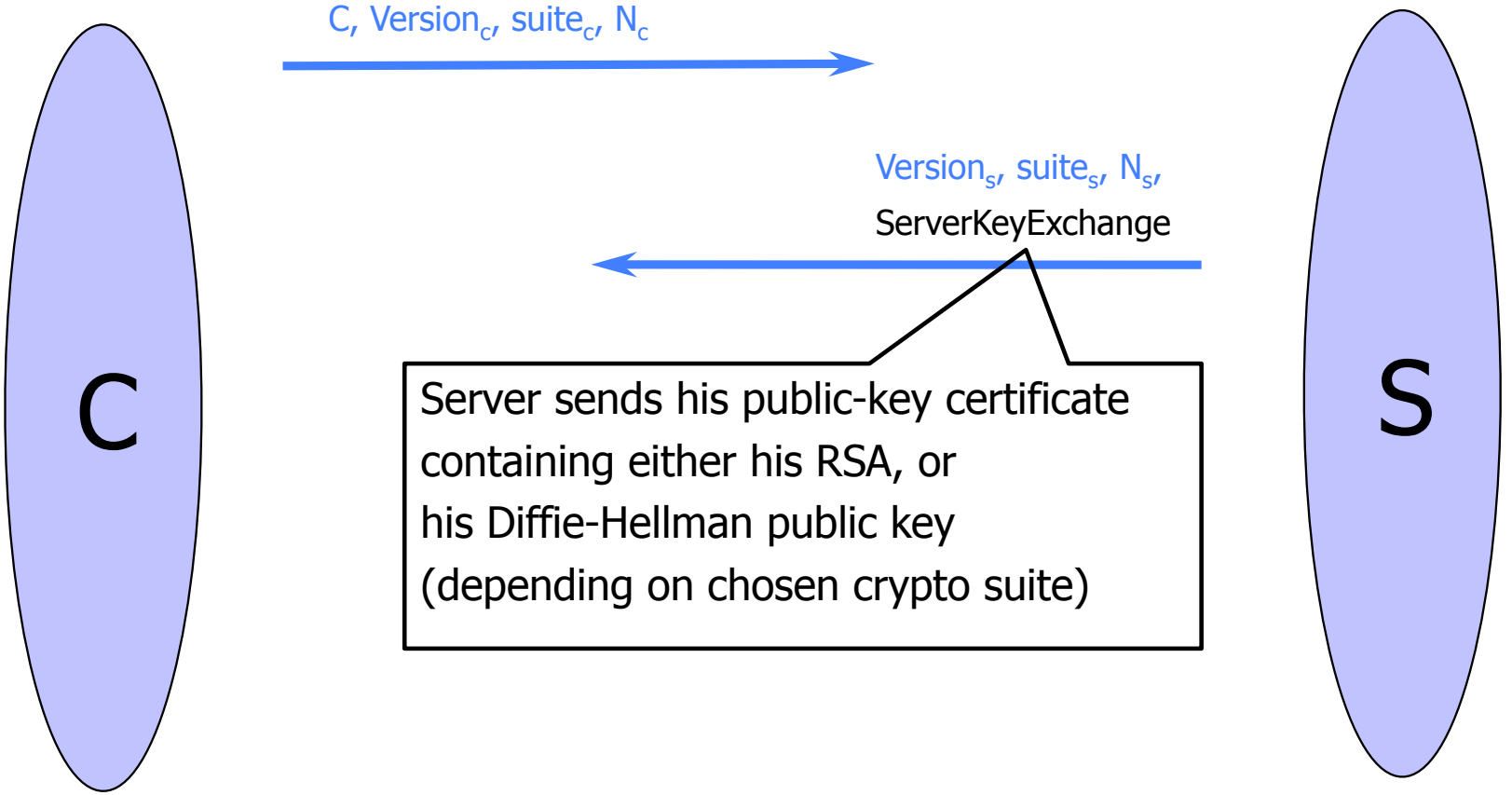
slide

ServerHello



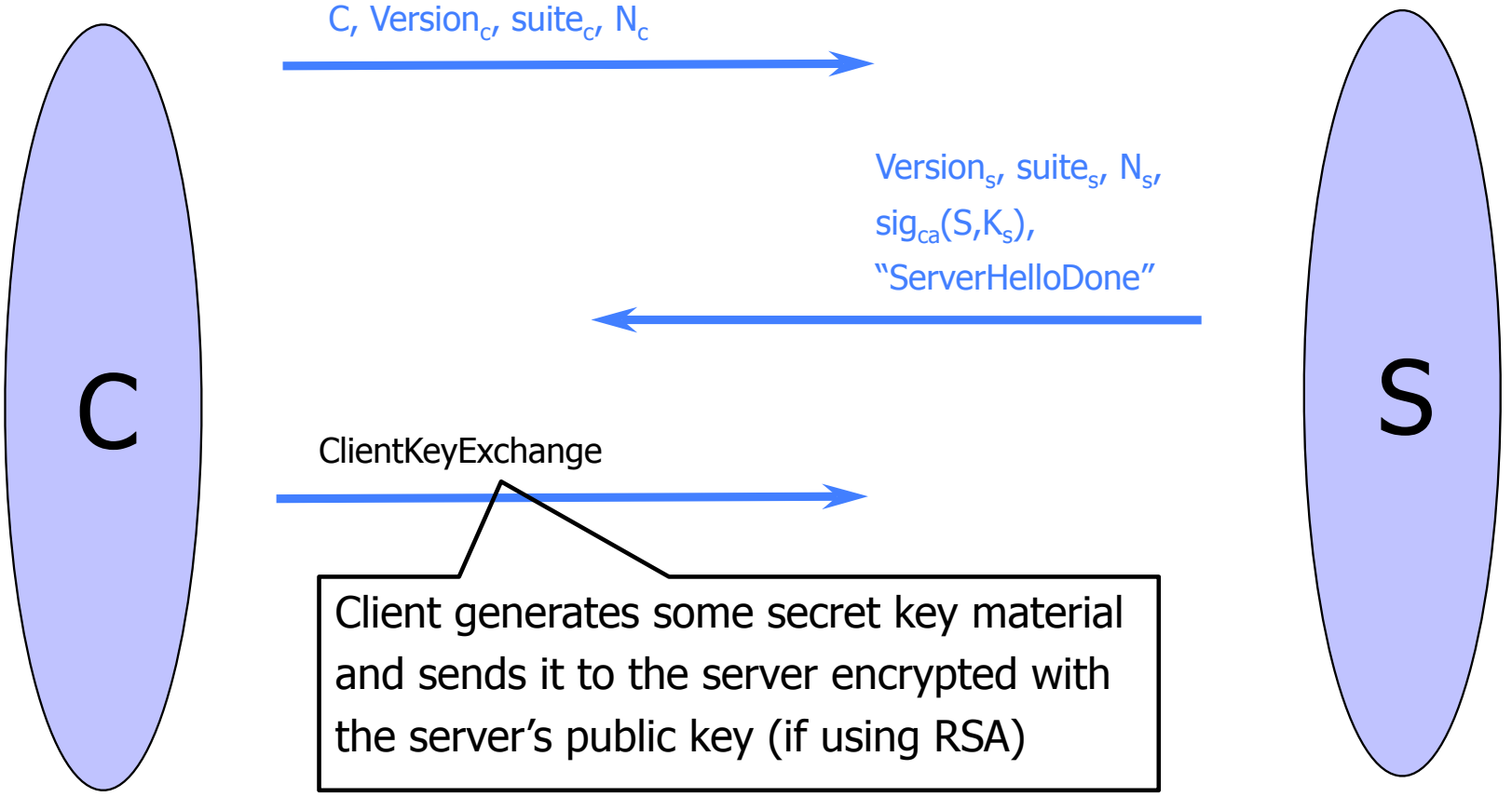
slide

ServerKeyExchange



slide

ClientKeyExchange



slide

ClientKeyExchange (RFC)

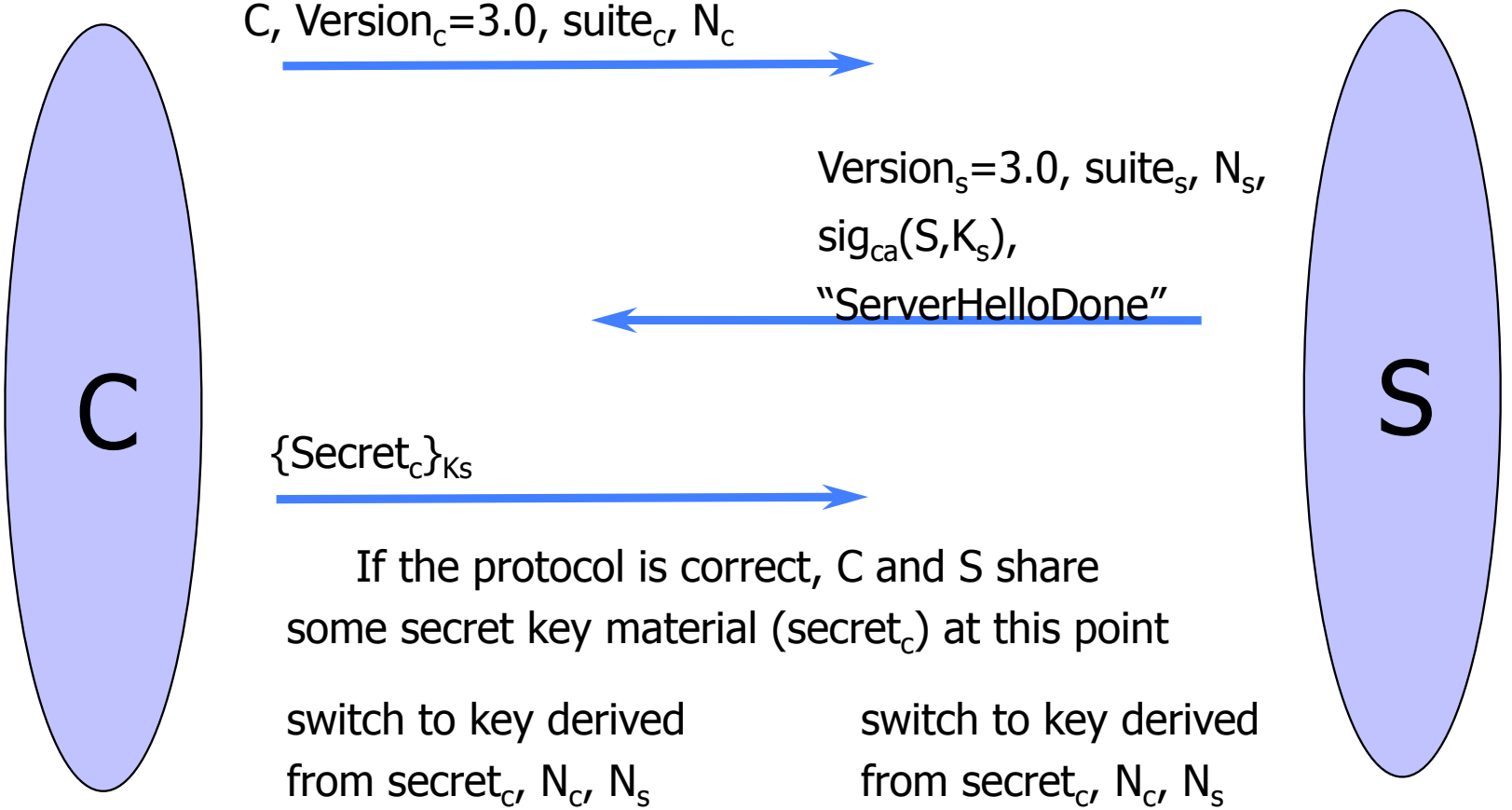
```
struct {
   select (KeyExchangeAlgorithm) {
      case rsa: EncryptedPreMasterSecret;
      case diffie_hellman: ClientDiffieHellmanPublic;
   } exchange_keys
} ClientKeyExchange
struct {

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

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

slide

"Core" SSL 3.0 Handshake



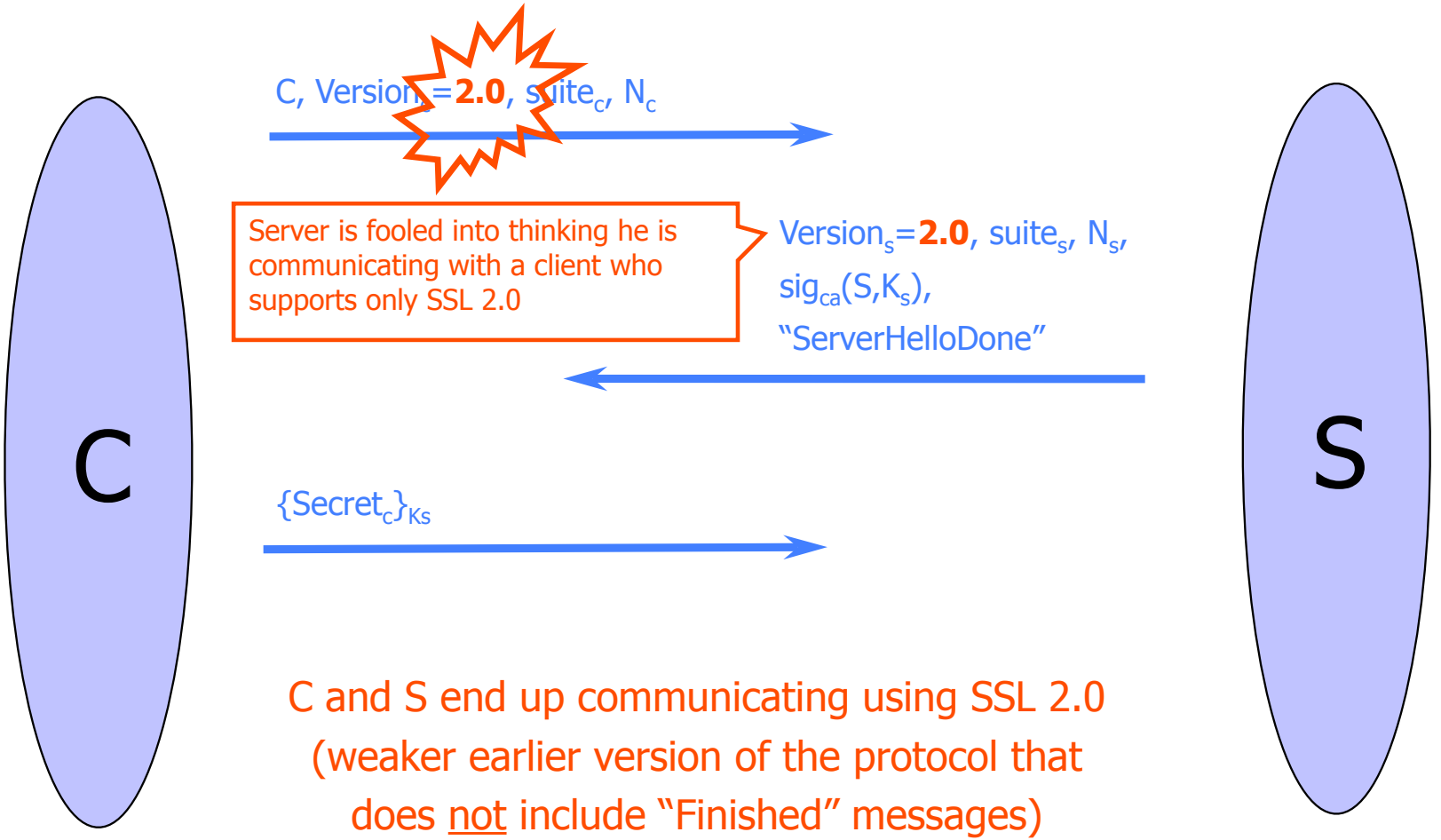
If the protocol is correct, C and S share some secret key material (secret$_c$) at this point

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

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

slide

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)

slide

SSL 2.0 Weaknesses (Fixed in 3.0)

- Cipher suite preferences are not authenticated
  - "Cipher suite rollback" attack is possible
- Weak MAC construction
- 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

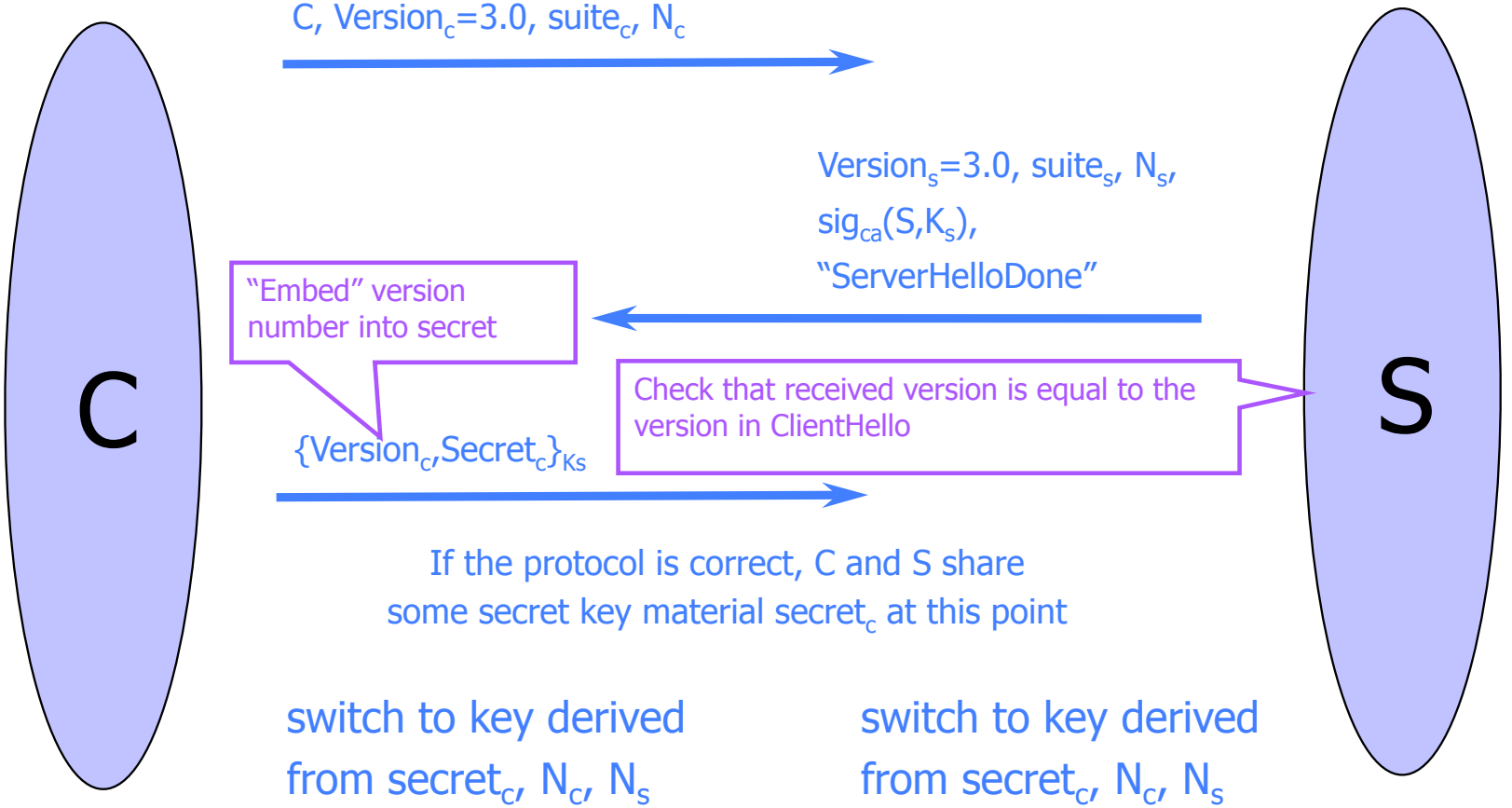
slide

"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 early
- Many protocols had "version rollback" attacks
  - SSL, SSH, GSM (cell phones)

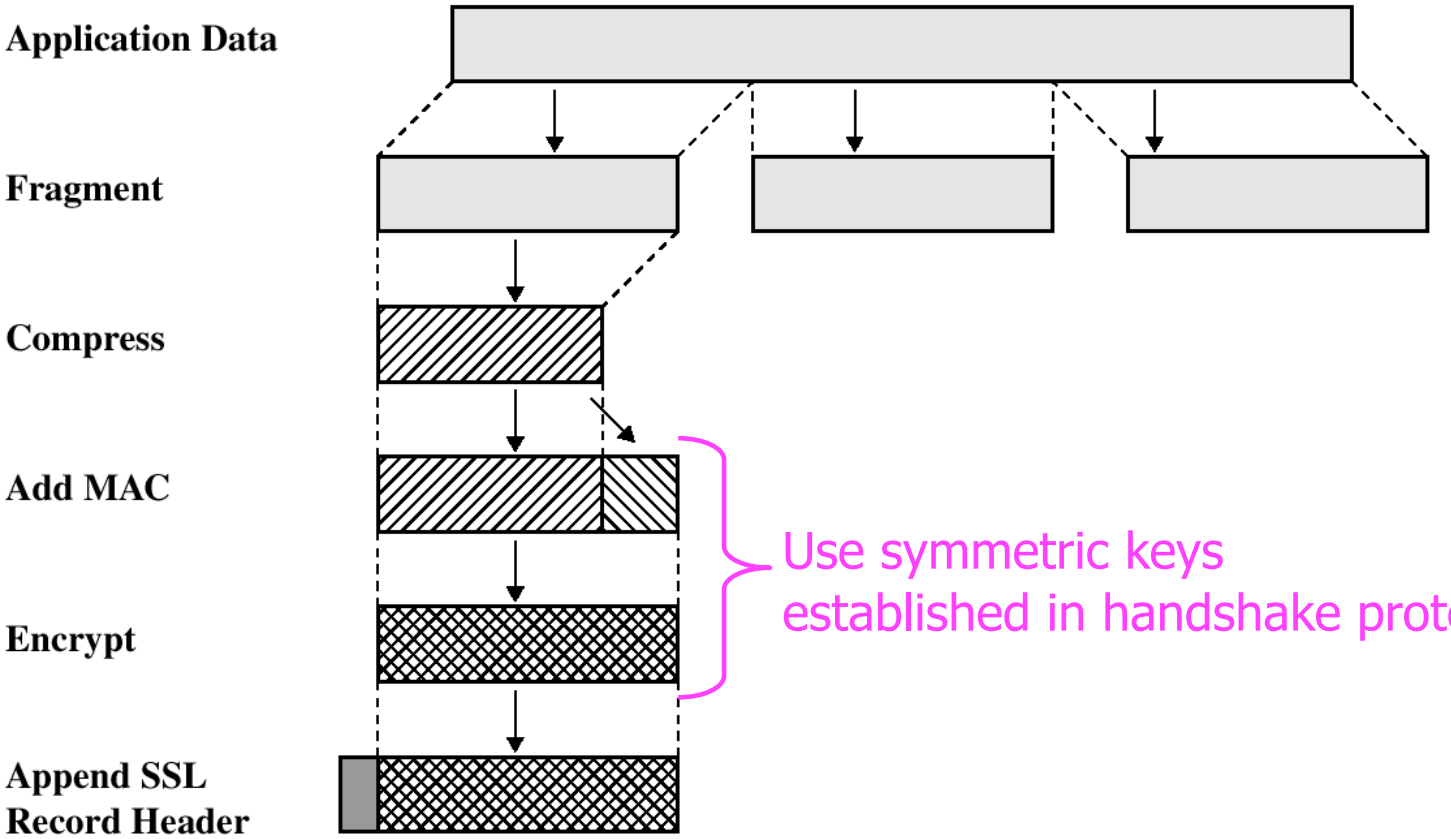
slide

Version Check in SSL 3.0



slide

SSL/TLS Record Protection



slide