CSE 484 / CSE M 584: Computer Security and Privacy

Crypto meets Web Security: Certificates and SSL/TLS

Spring 2017

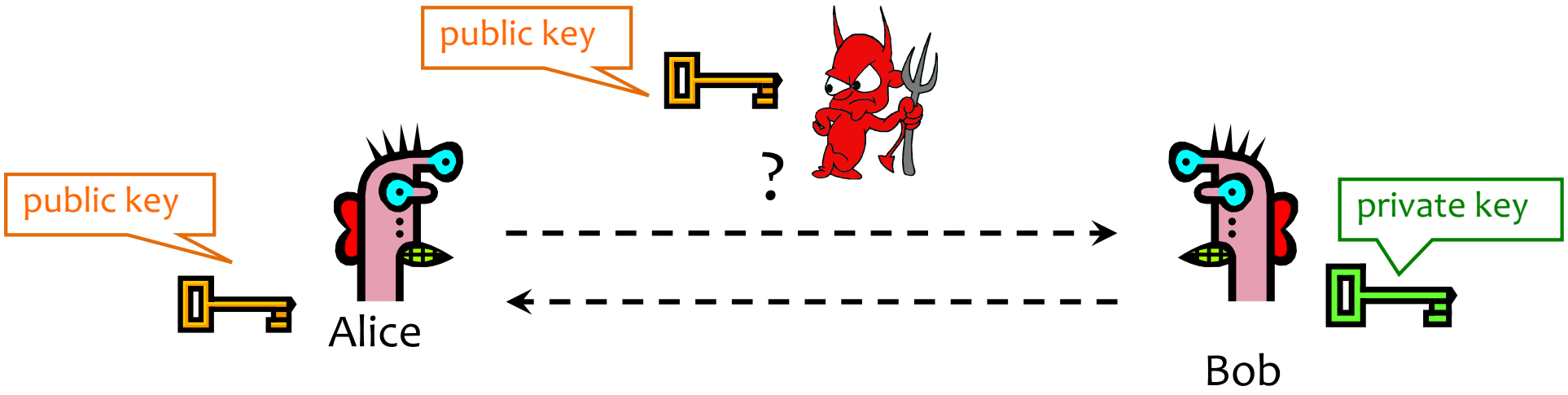
Franziska (Franzi) Roesner
franzi@cs.washington.edu

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 ...

Announcements

- Homework #2 (crypto) due Friday
  - Individual assignment
  - Please send your encrypted email early!
- Next part of course: web security
- Section this week: physical security

Public Key Crypto: Basic Problem



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

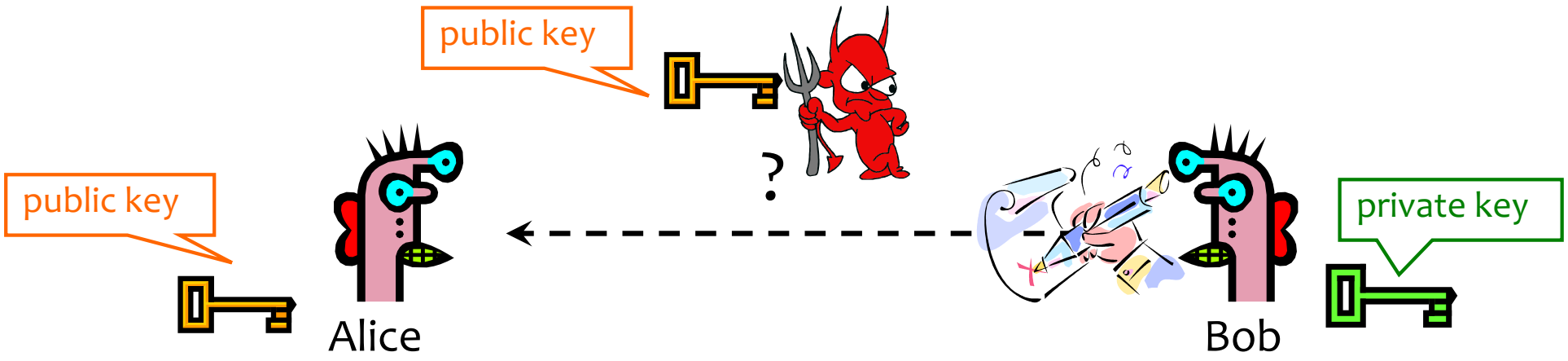
Goals: 1. Alice wants to send a secret message to Bob
2. Bob wants to authenticate himself

Last Week

- Public key crypto protocols
  - Based on underlying assumptions about hard problems
  - Diffie Hellman and RSA
  - Not in this course: elliptic curves

- Last time: confidentiality (no integrity or authentication)

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
  - Standard padding/hashing schemes exist for RSA signatures

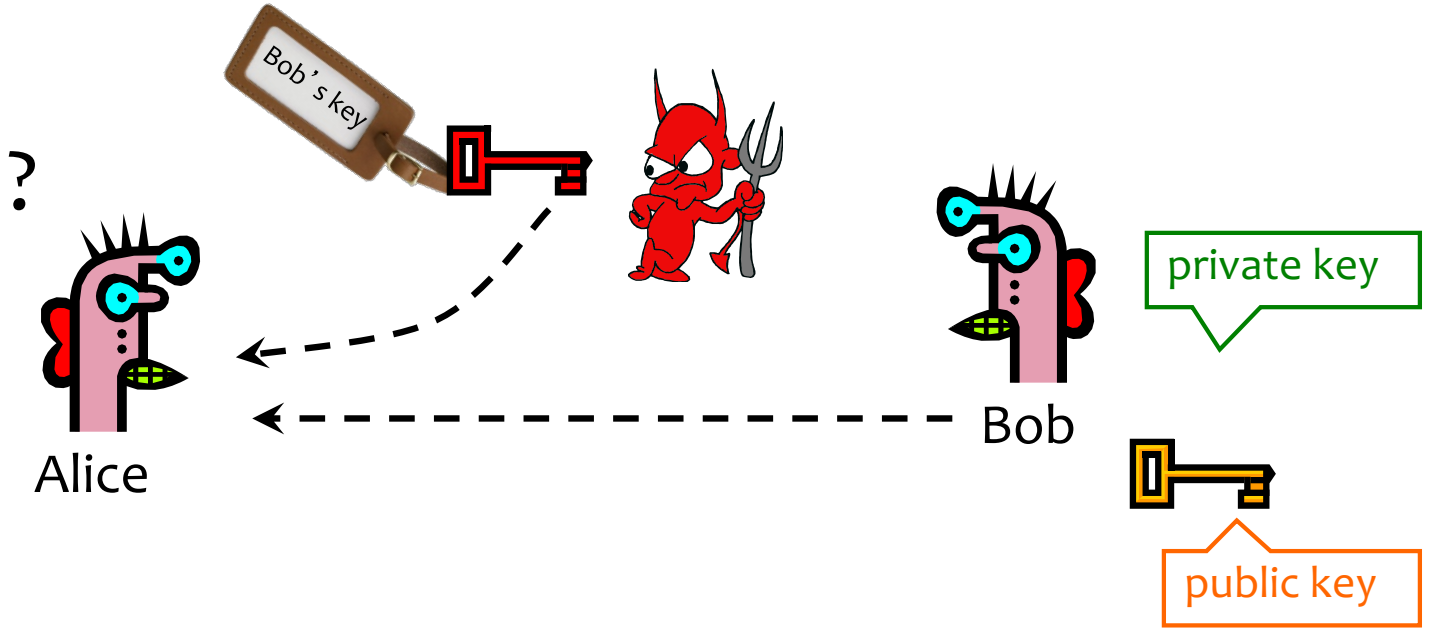
DSS Signatures

- 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
- 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)

Cryptography Summary

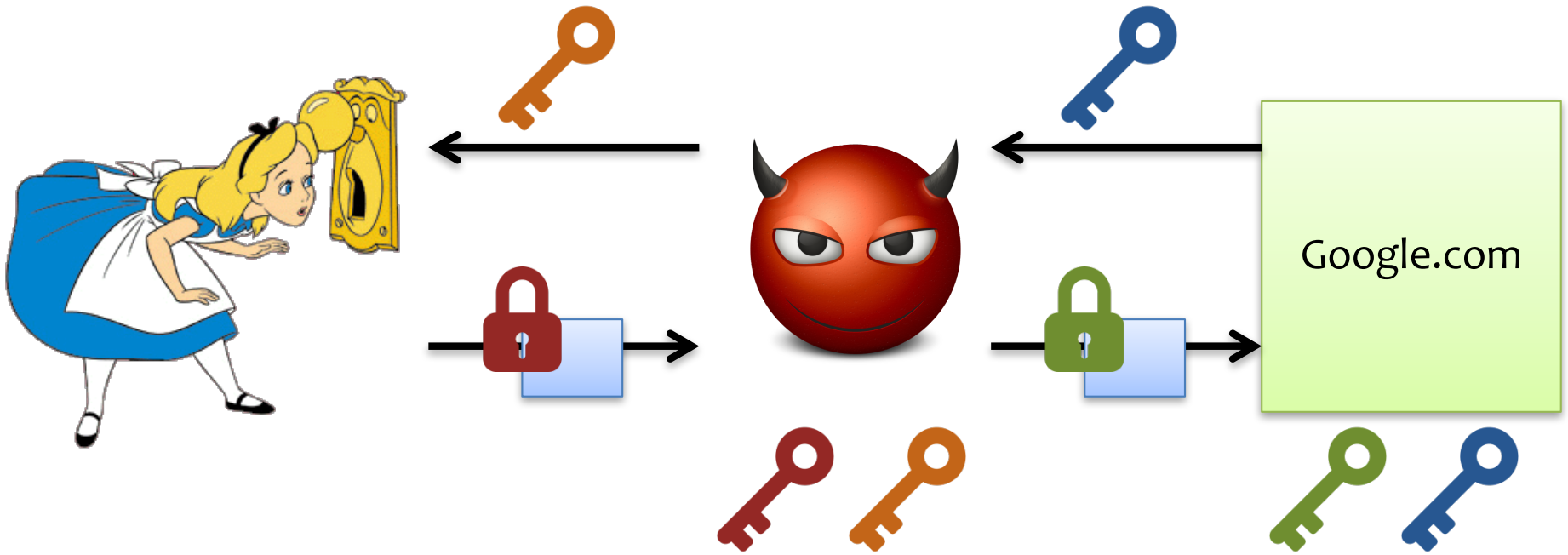
- Goal: Privacy
  - Symmetric keys:
    - One-time pad, Stream ciphers
    - Block ciphers (e.g., DES, AES) → modes: EBC, CBC, CTR
  - Public key crypto (e.g., Diffie-Hellman, RSA)
- Goal: Integrity
  - MACs, often using hash functions (e.g, MD5, SHA-256)
- Goal: Privacy and Integrity
  - Encrypt-then-MAC
- Goal: Authenticity (and Integrity)
  - Digital signatures (e.g., RSA, DSS)

Authenticity of Public Keys



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

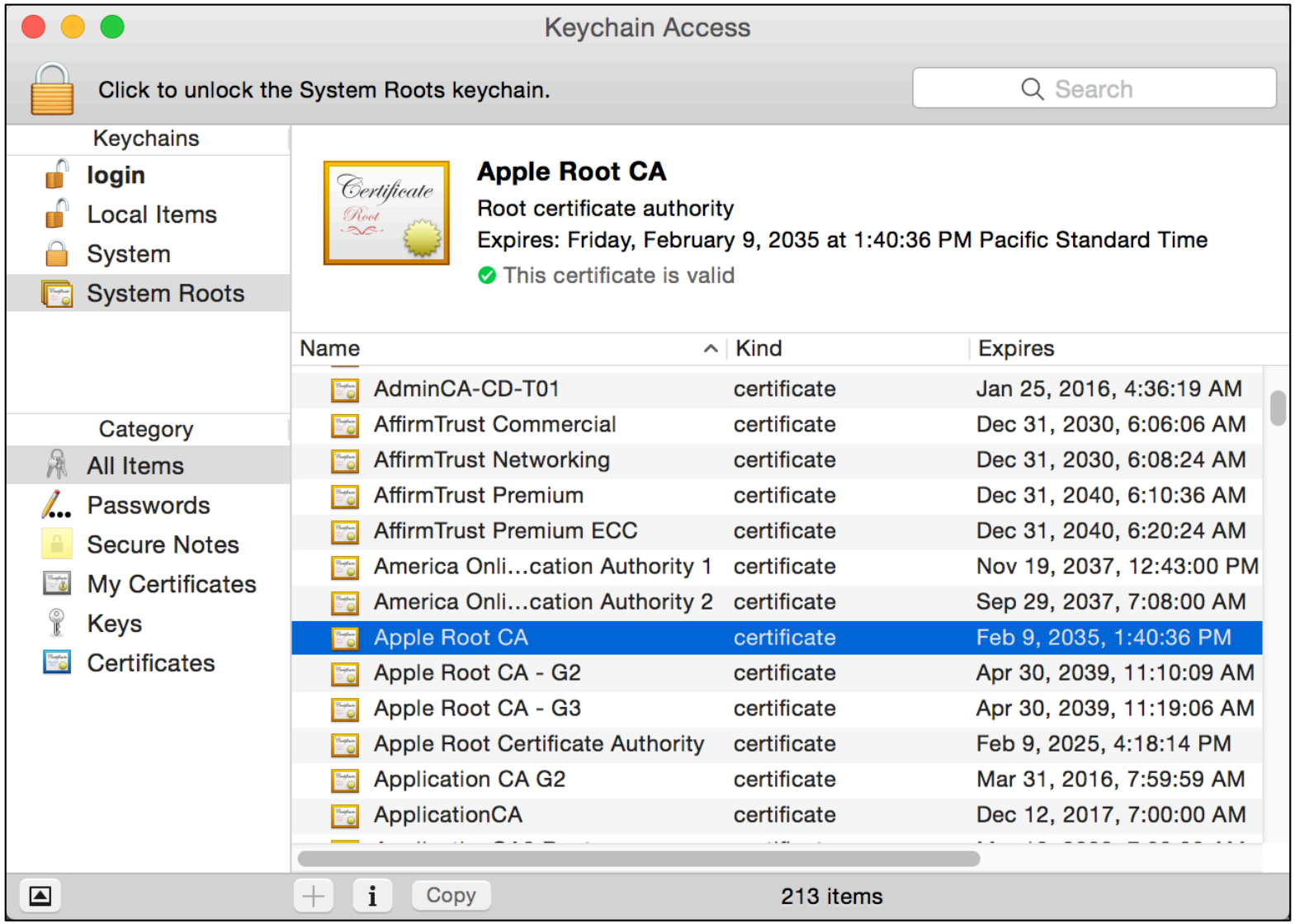
Threat: Man-In-The-Middle (MITM)



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

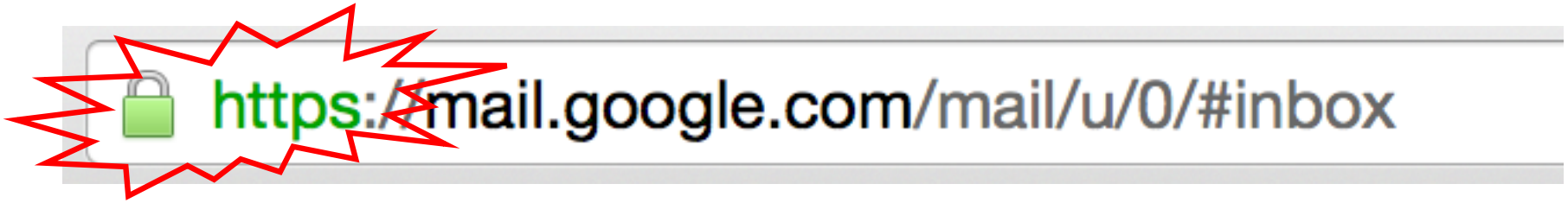
Trusted Certificate Authorities



Hierarchical Approach

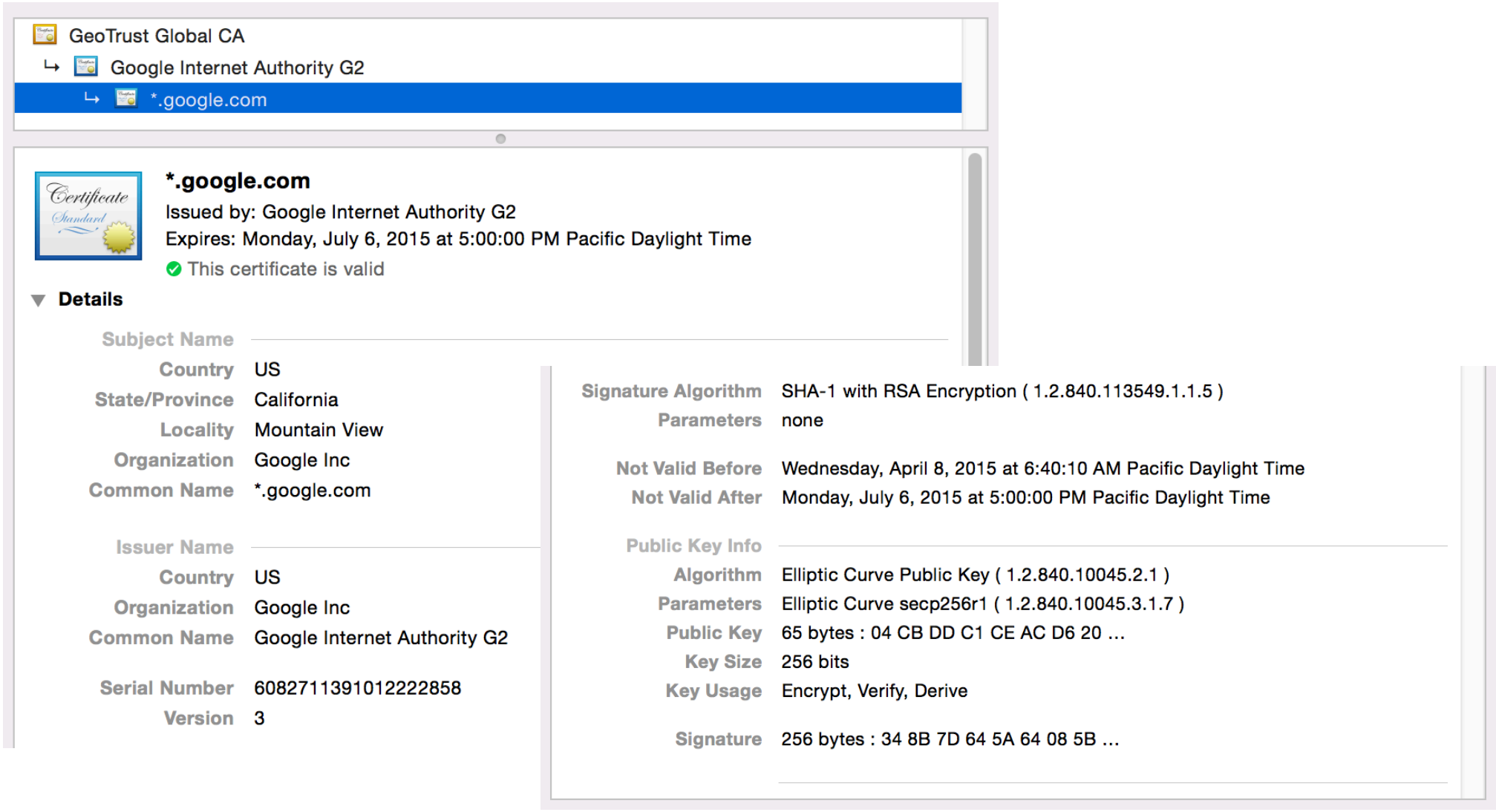
- Single CA certifying every public key is impractical
- Instead, use a trusted root authority
  - For example, Verisign
  - Everybody must know the public key for verifying root authority's signatures
- Root authority signs certificates for lower-level authorities, lower-level authorities sign certificates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

You encounter this every day…

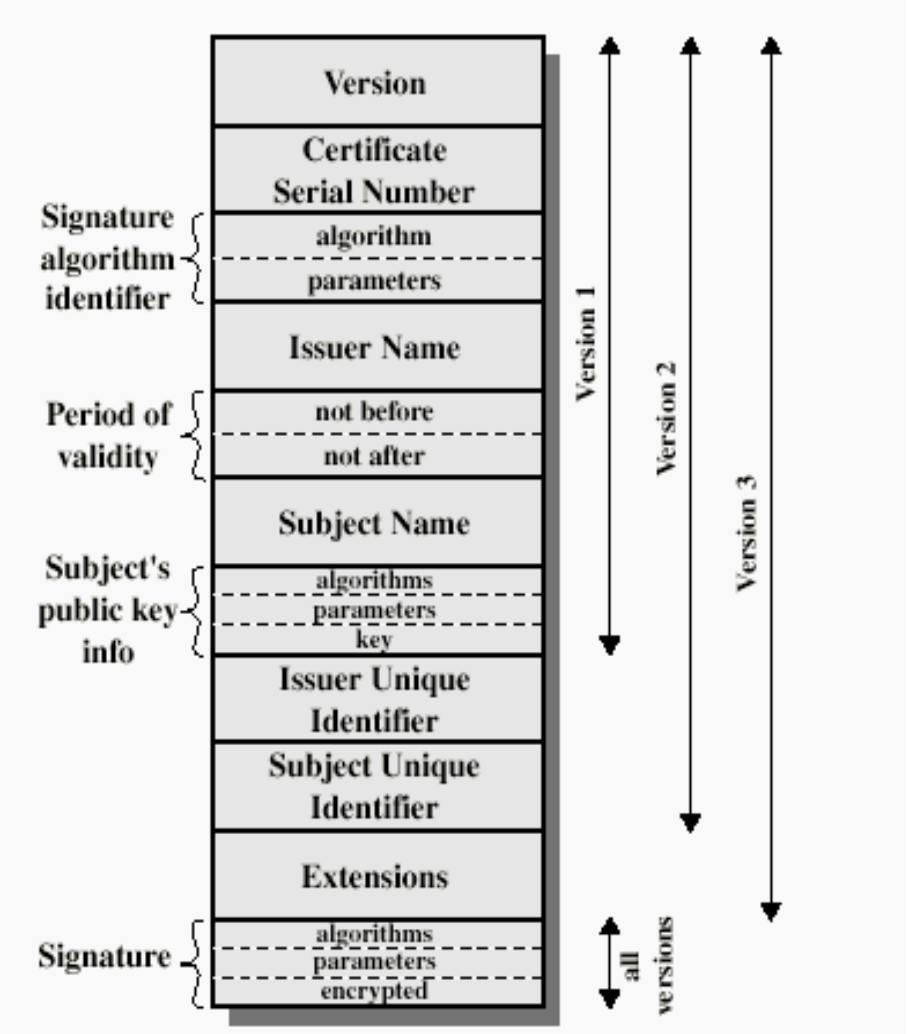


SSL/TLS: Encryption & authentication for connections

Example of a Certificate

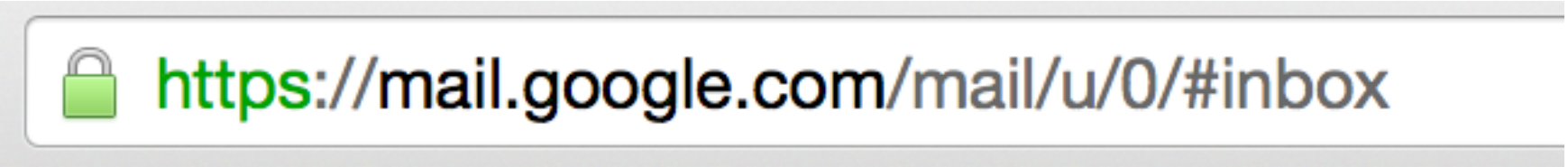


X.509 Certificate



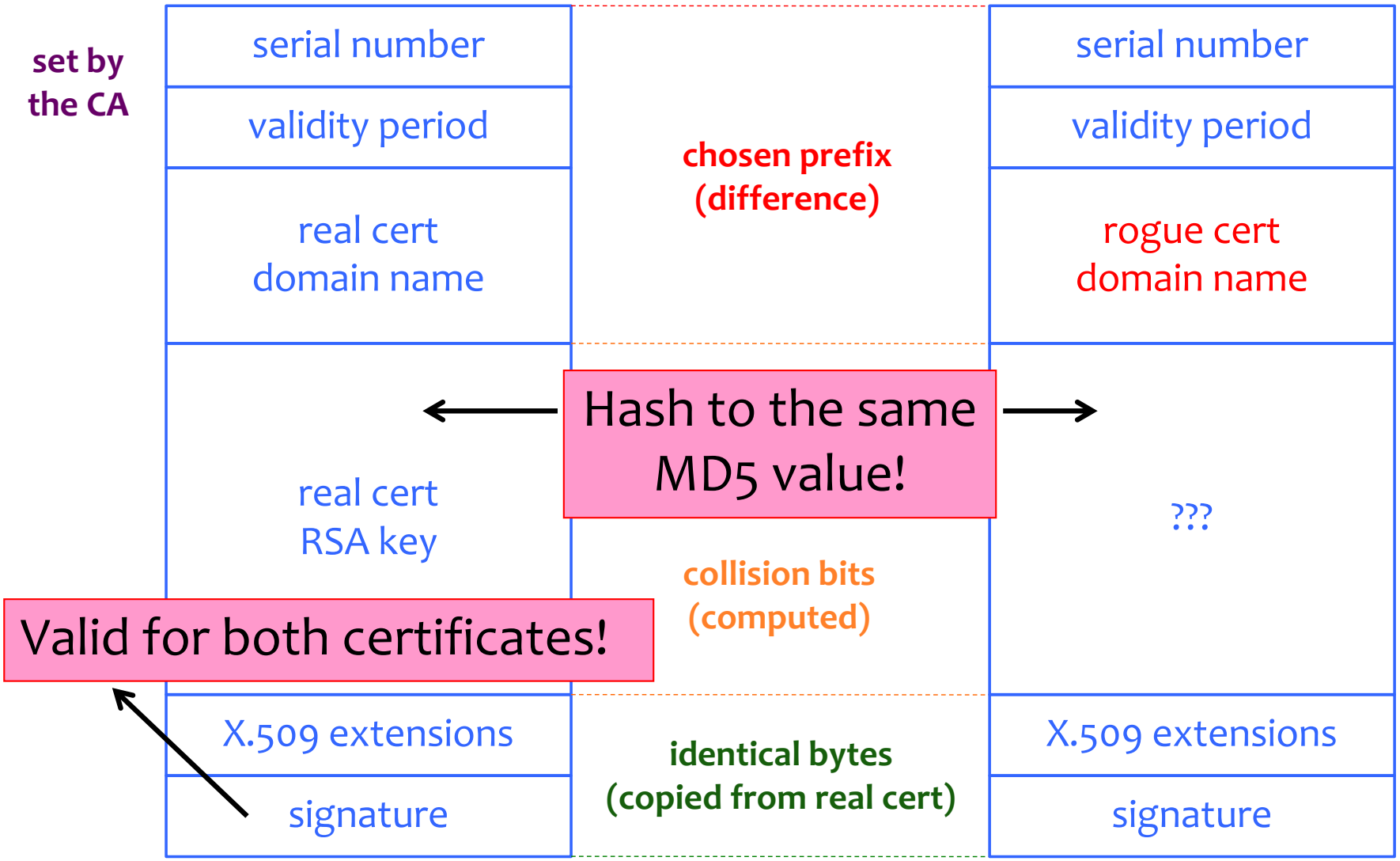
Many Challenges…

- 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
- Etc…



[Sotirov et al. "Rogue Certificates"]

Colliding Certificates



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

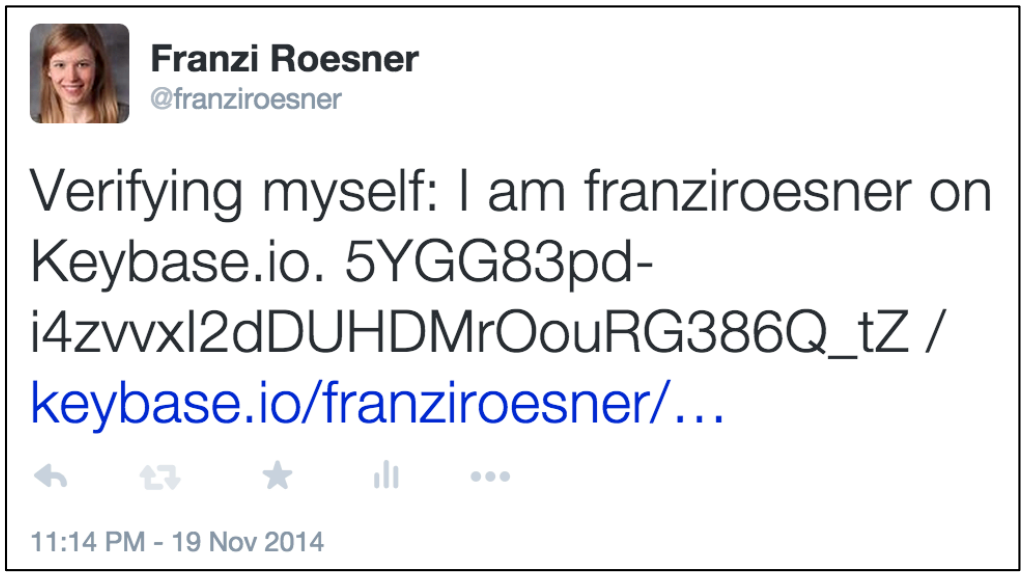
Attempt to Fix CA Problems: Convergence

- 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/

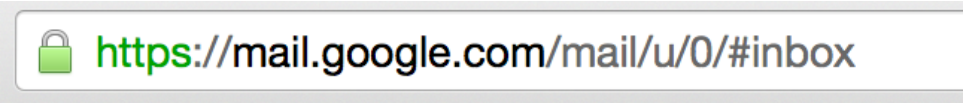
Keybase

- 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



https://keybase.io/

SSL/TLS

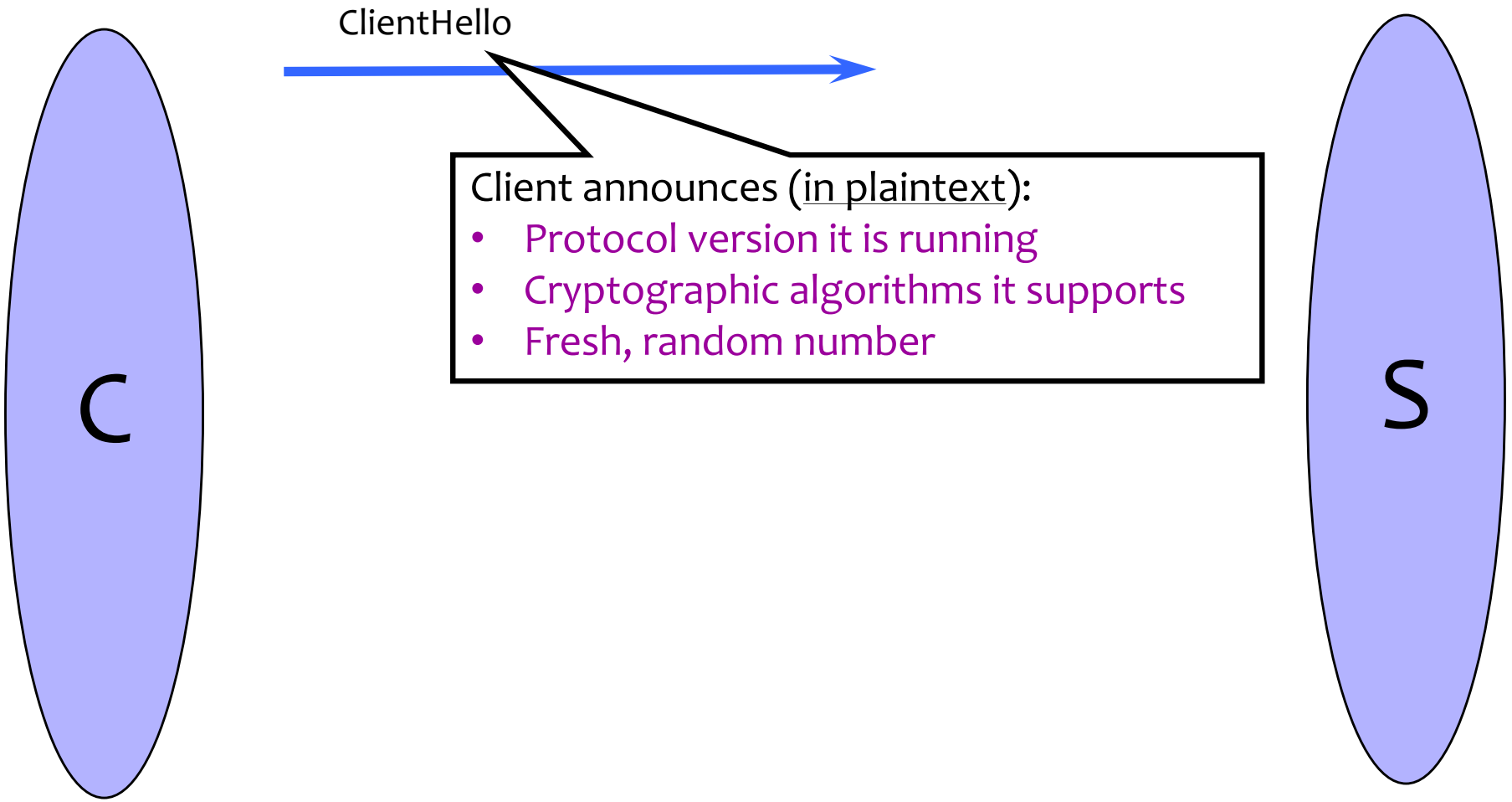


- Secure Sockets Layer and Transport Layer Security protocols
  - Same protocol design, different crypto algorithms
- De facto standard for Internet security
  - "The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications"
- Deployed in every Web browser; also VoIP, payment systems, distributed systems, etc.

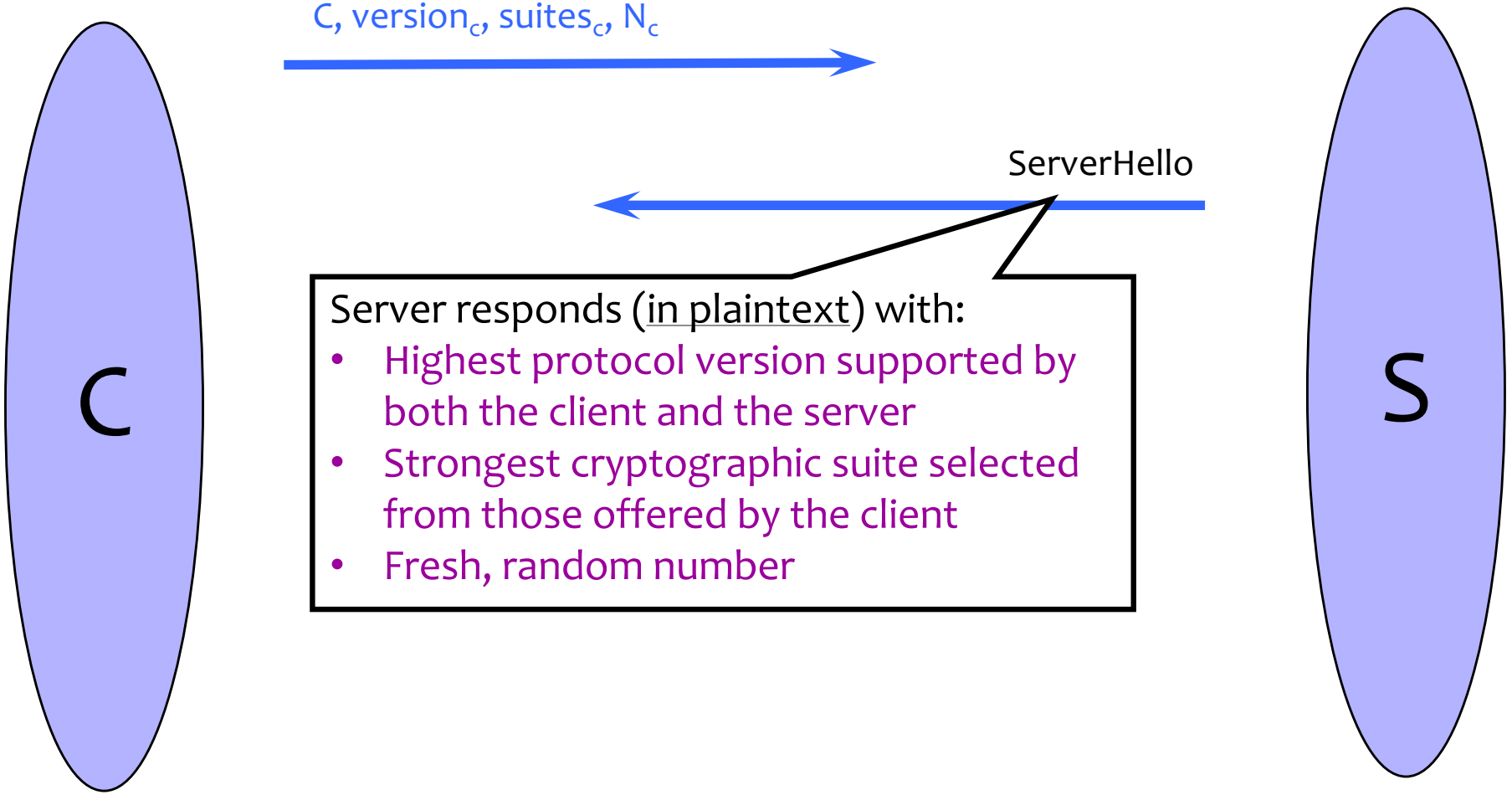
TLS Basics

- 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

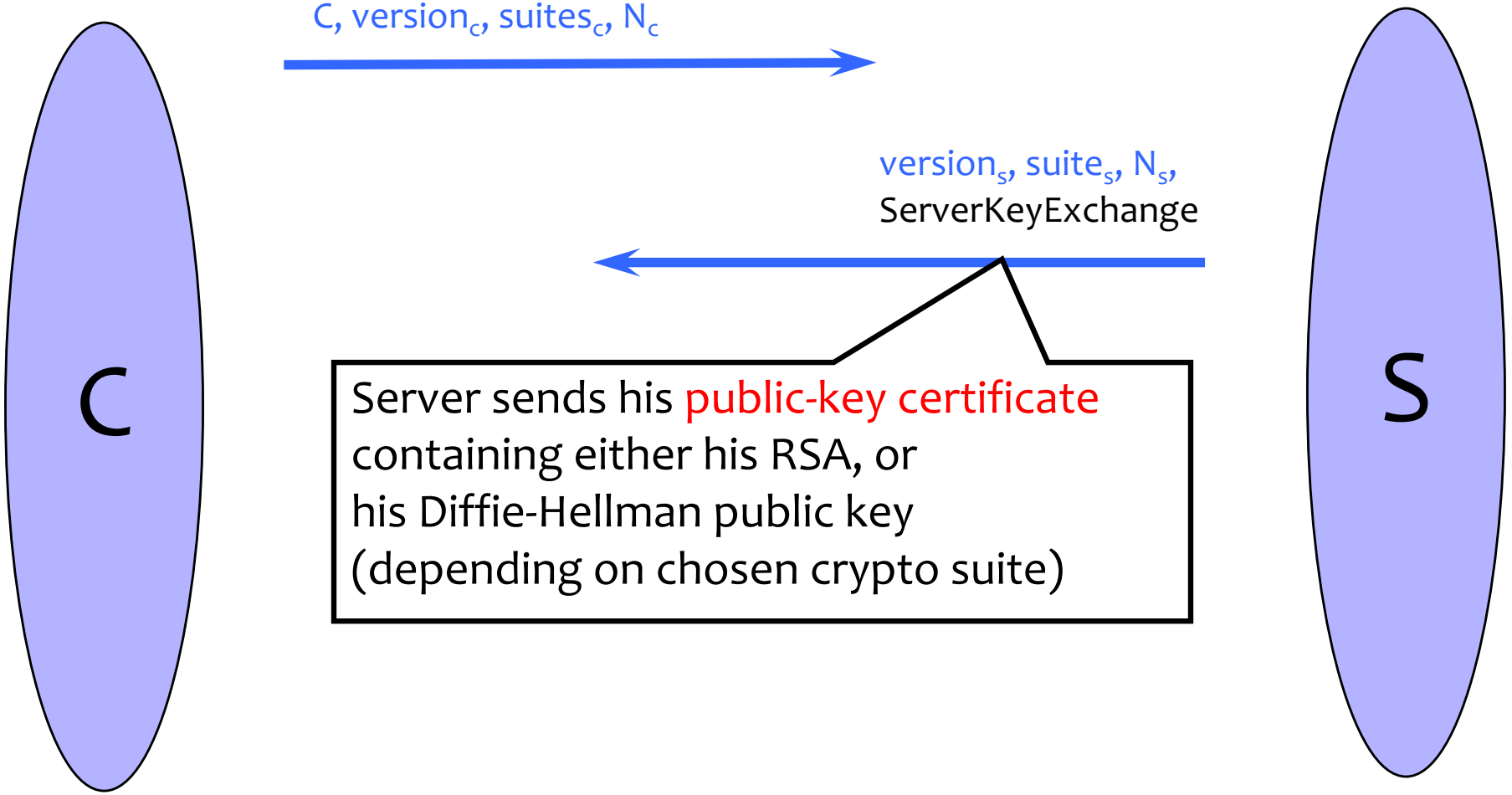
Basic Handshake Protocol



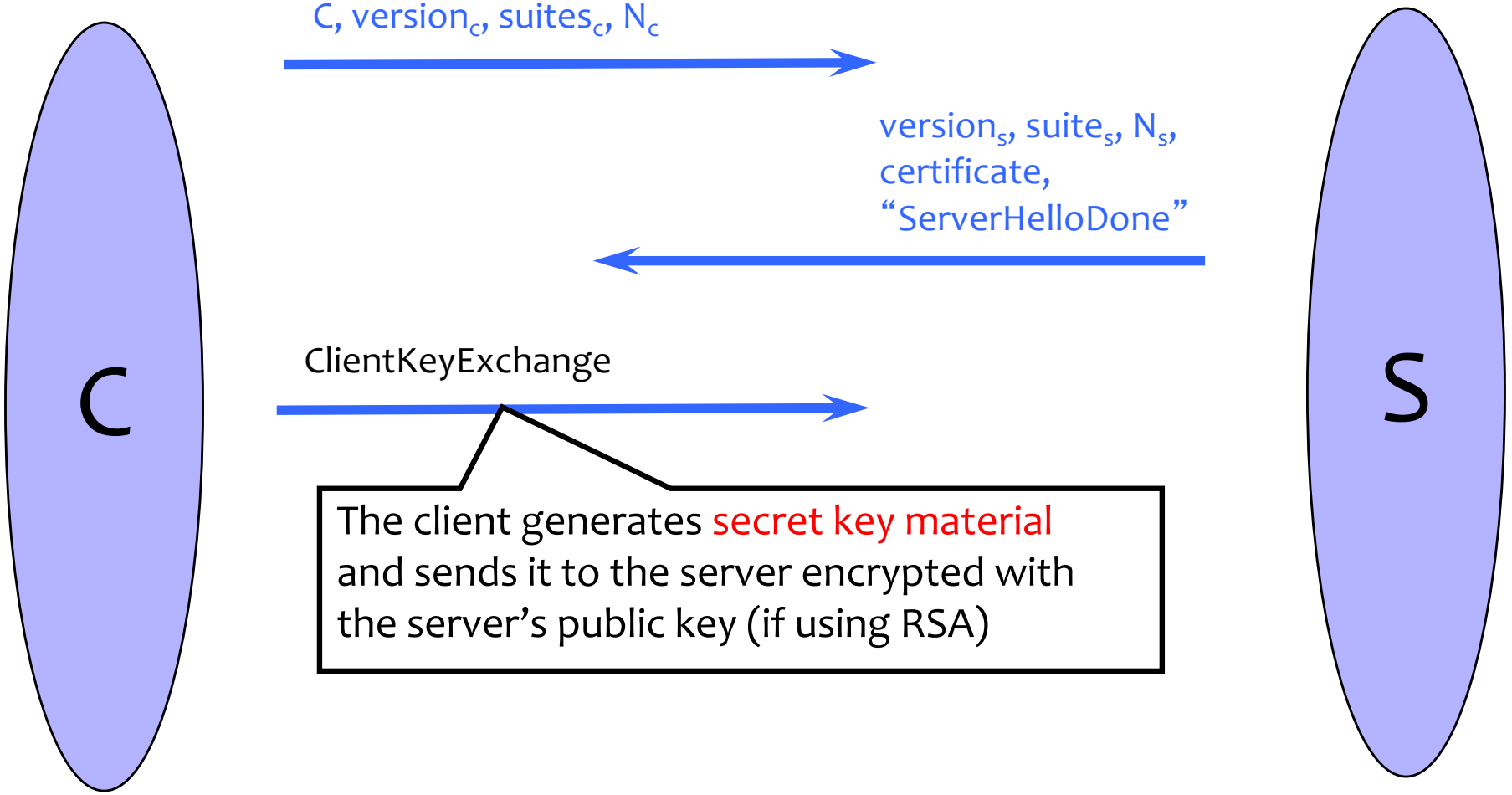
Basic Handshake Protocol



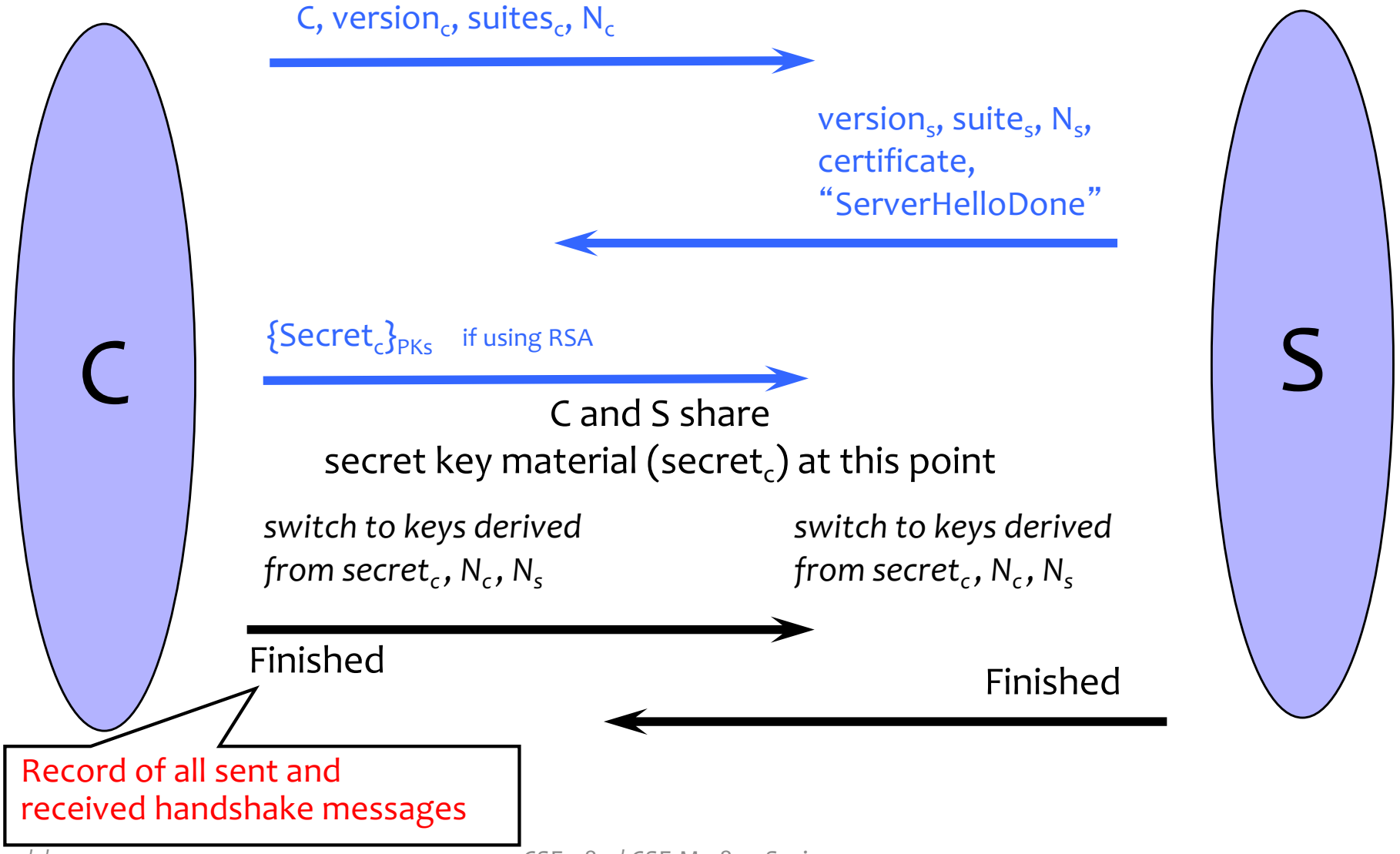
Basic Handshake Protocol



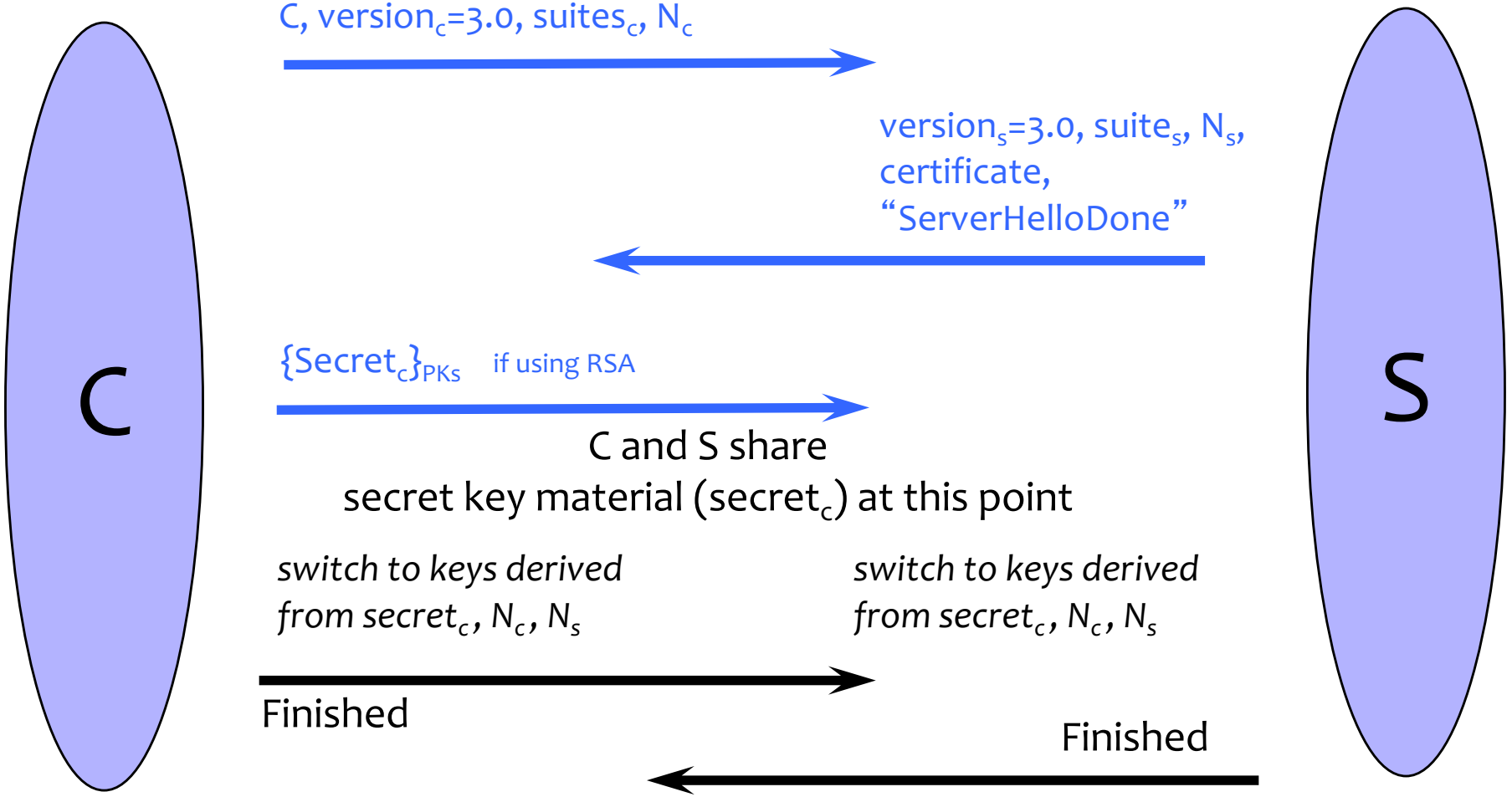
Basic Handshake Protocol



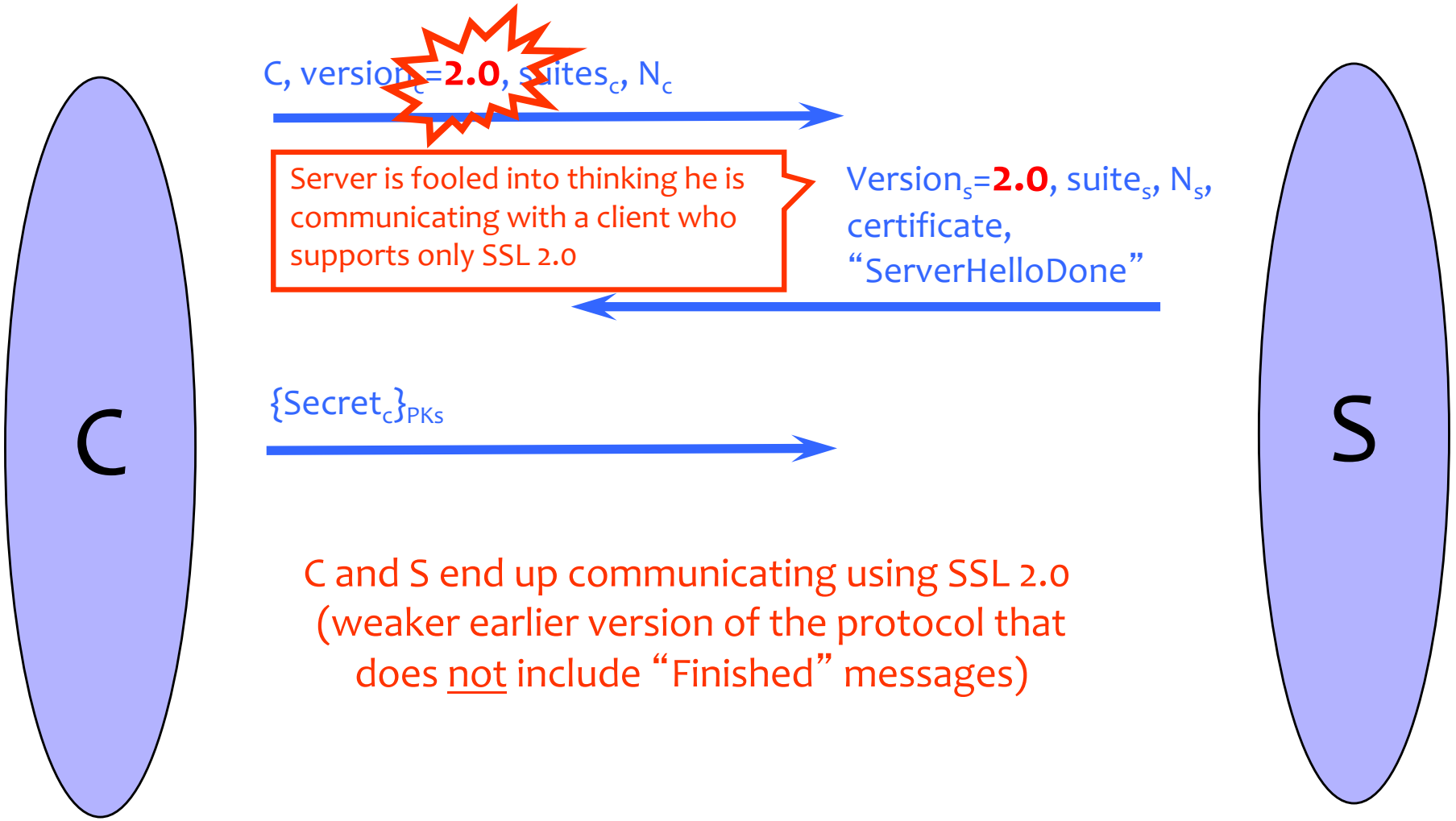
Basic Handshake Protocol



"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)

"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