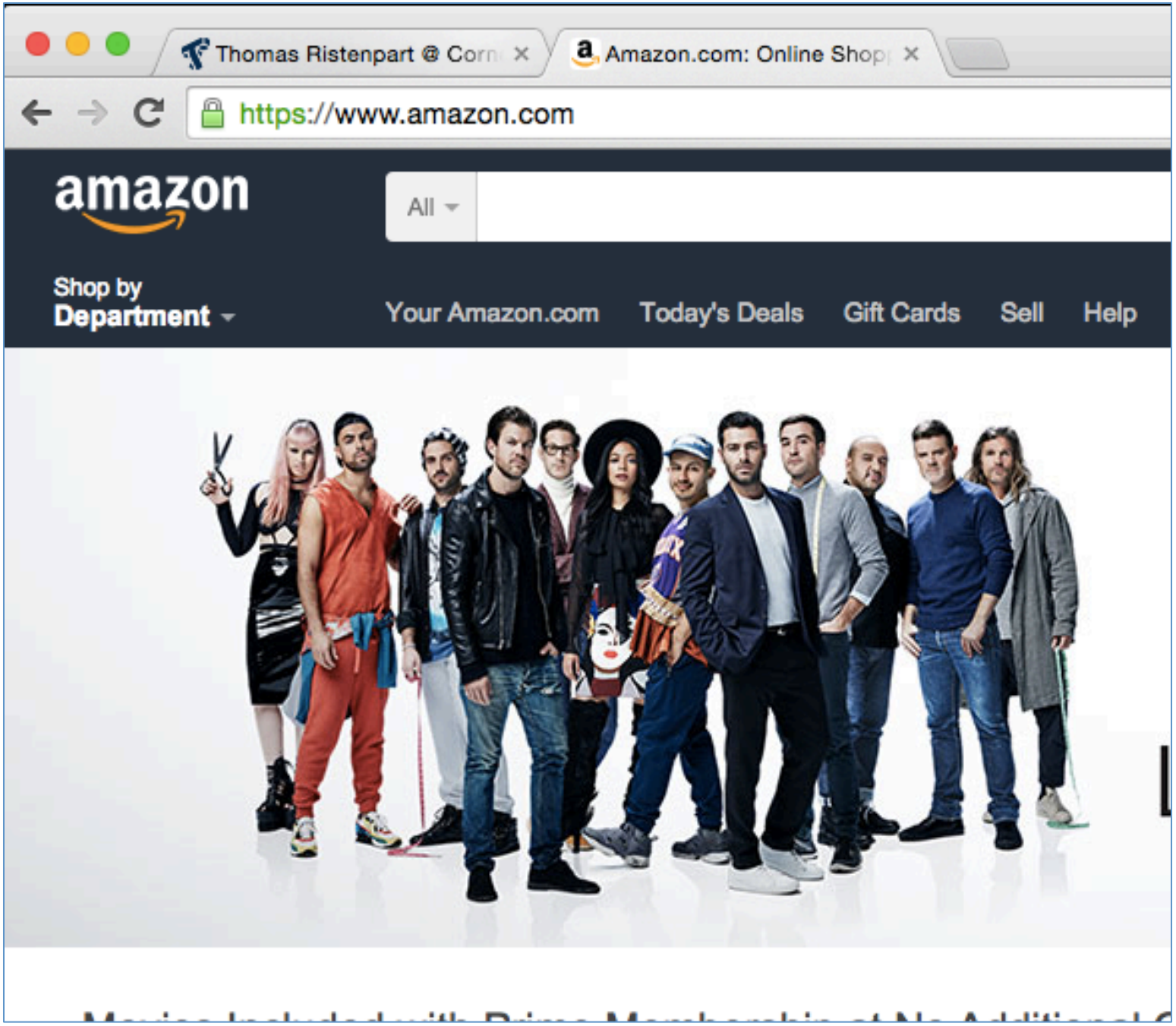
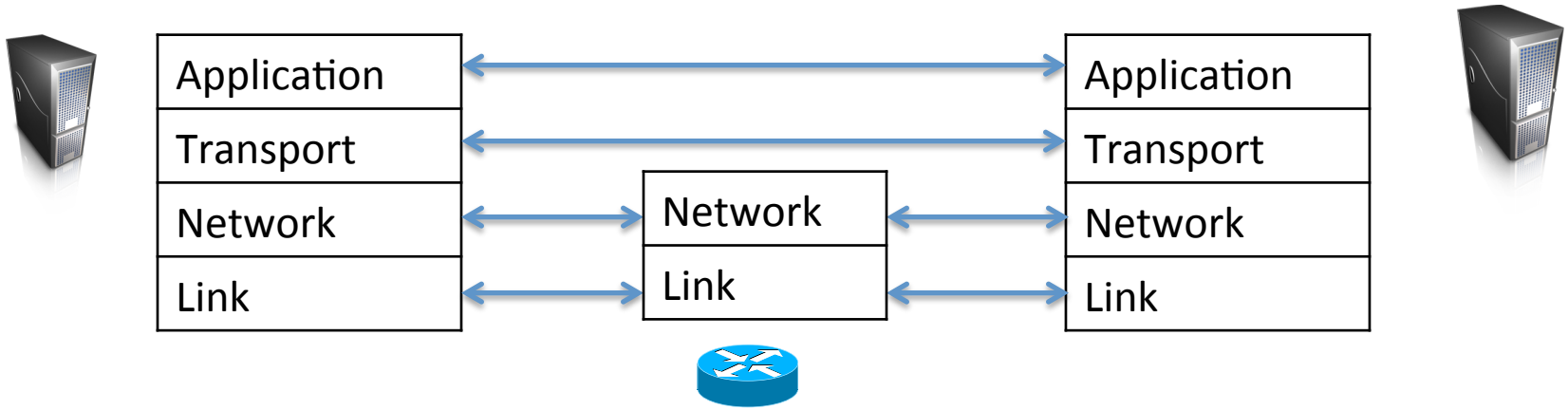
Today in Cryptography (5830)

TLS Overview
Public-key encryption
Key transport

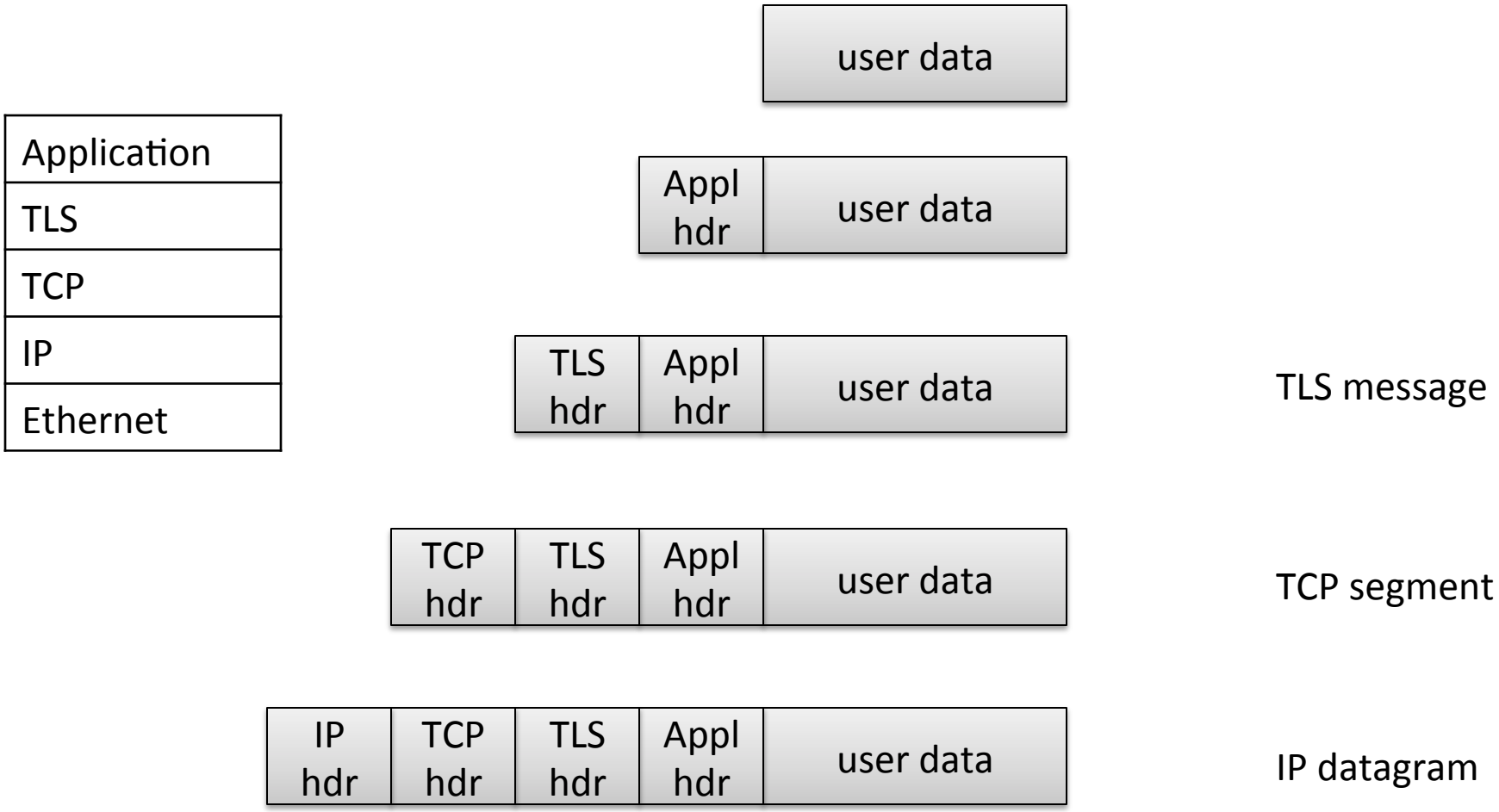

Thomas Ristenpart @ Corn
Amazon.com: Online Shop
https://www.amazon.com
amazon
All
Shop by
Department
Your Amazon.com
Today's Deals
Gift Cards
Sell
Help

Internet protocol stack

Application	HTTP, FTP, SMTP, SSH, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	802x (802.11, Ethernet)



TLS sits between application and TCP

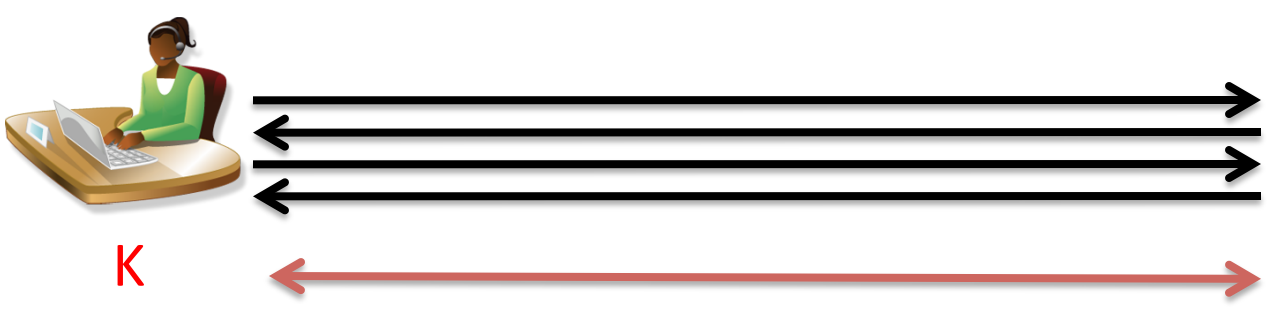


Places TLS is used

- HTTPS
  - HTTP messages but over TLS, not TCP
- Email connections
  - When getting information from your email server (not the email contents themselves)
- Virtual private networks (VPNs)
  - Tunnel other internet connections over a TLS connection

How does TLS work (high level)?

https://amazon.com



Step 1:
Key exchange protocol to share secret K

Step 2:
Send data via secure channel

The secure channel is implemented via our now familiar symmetric encryption primitives

Goals of handshake:

- Negotiate version
- Negotiate parameters (crypto to use)
- Authenticate server (Is server actually Amazon.com?)
  - Digital signatures and certificates
- Establish shared secret
  - Asymmetric encryption primitives

A short history of TLS

How many cryptographers involved?

SSL ver 2.0 designed by **Hickman** at Netscape

SSL ver 2

1994

Wagner, Goldberg break SSL ver 2

1995

Freier, Karlton, Kocher design SSL ver 3.0

SSL ver 3

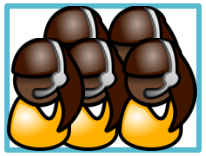
1998

Bleichenbacher breaks RSA PKCS #1 encryption, used in SSL ver 3

1999

TLS ver 1 released as IETF standard, based on SSL 3, **many cryptographers** involved

TLS ver 1.0



2001

Vaudenay, Klima et al. padding attacks

2002

Rogaway IV re-use insecurity

2003

Brumley, Boneh remote timing attacks

2006

TLS ver 1.1 released as standard

TLS ver 1.1

⋮

(more attacks and fixes)

TLS handshake for RSA transport

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

← CERT = (pk of bank, signature over it)

Pick random PMS
C <- E(pk,PMS)

C →

PMS <- D(sk,C)

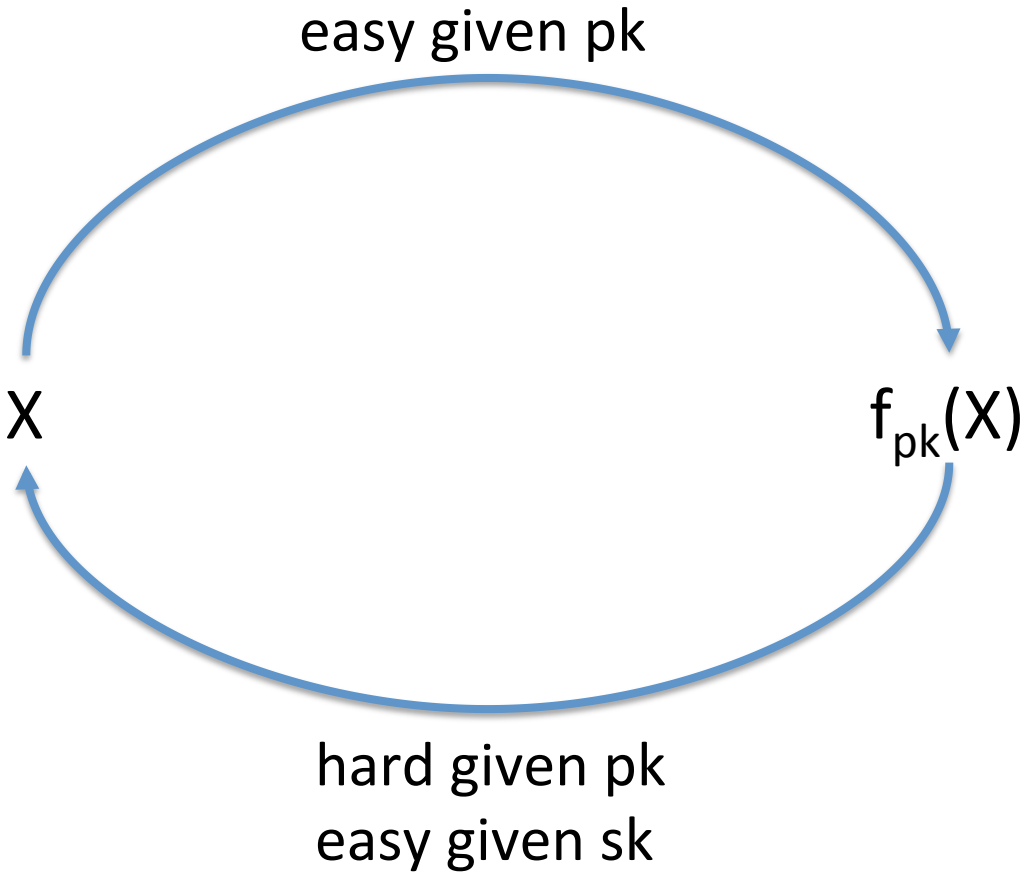
ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, "master secret" || Nc || Ns)

Trapdoor functions



TLS handshake for Diffie-Hellman Key Exchange

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods

Check CERT using CA public verification key

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Pick random Ns

Pick random x

$X = g^x$

CERT = (pk_s , signature over it)

p , g , X , σ = Sign(sk_s, p || g || X)

Y

Check σ

Pick random y

$Y = g^y$

$PMS = g^{xy}$

$PMS = g^{xy}$

ChangeCipherSpec,

{ Finished, PRF(MS, "Client finished" || H(transcript)) }

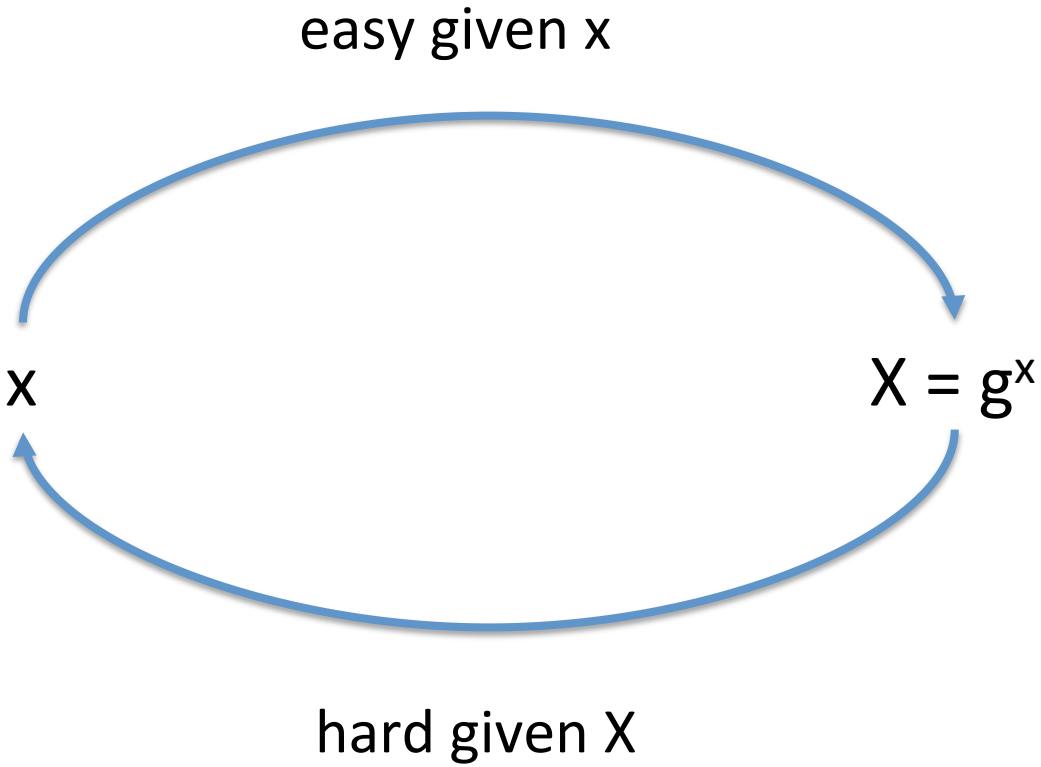
ChangeCipherSpec,

{ Finished, PRF(MS, "Server finished" || H(transcript')) }

Bracket notation means contents encrypted

MS <- PRF(PMS, "master secret" || Nc || Ns)

One-way functions



TLS Key derivation & use

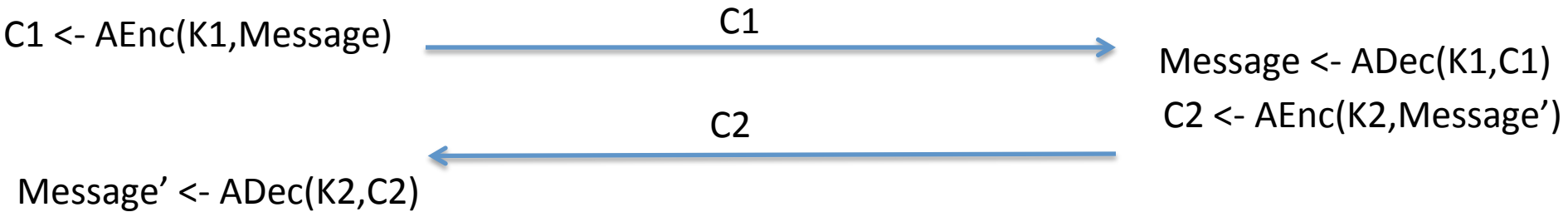
MS <- PRF(PMS, “master secret” || Nc || Ns)

K1,K2 <- PRF(MS, “key expansion” || Ns || Nc)

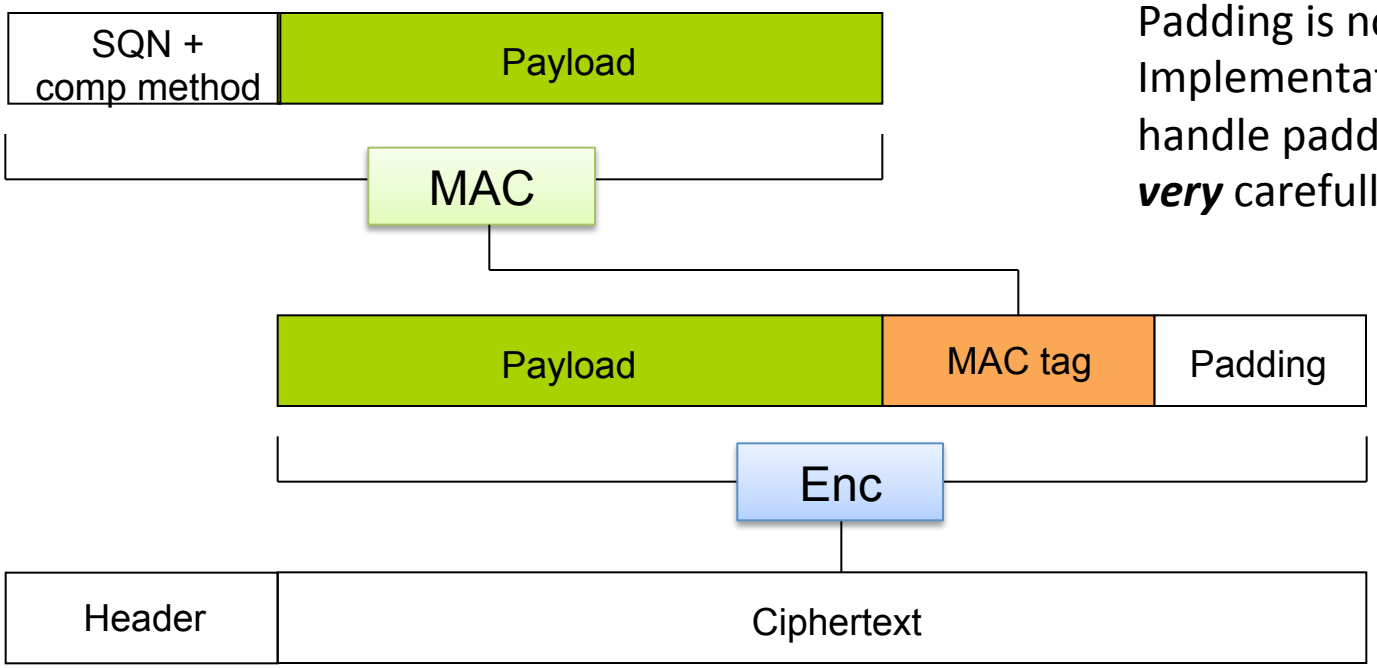
PRF(secret, message) = HMAC-HASH(secret, A(1) + seed) +
HMAC-HASH(secret, A(2) + seed) +
HMAC-HASH(secret, A(3) + seed) + ...

Where A(0) = seed and A(i) = HMAC_hash(secret, A(i-1))

This mess replaced with HKDF in 1.3



TLS 1.2 record protocol: MAC-Encode-Encrypt (MEE)



Padding is not MAC'd. Implementations must handle padding checks ***very*** carefully.

MAC	HMAC-MD5, HMAC-SHA1, HMAC-SHA256
Encrypt	CBC-AES128, CBC-AES256, CBC-3DES, RC4-128

Record layer details

- Fragmentation
  - Maximum TLS ciphertext handles 2^{14} bytes of message data
  - Split longer requested submission into multiple chunks
- Sequence numbers keep track of count of chunks sent in each direction
- Compression methods
  - Generally a bad idea to use (why?)