Topic 4.2 • Applications

TLS and SSH

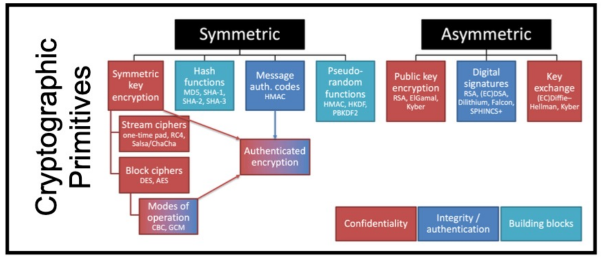
CO 487/687

Dr. Douglas Stebila

UNIVERSITY OF WATERLOO

Fall 2024

Applications

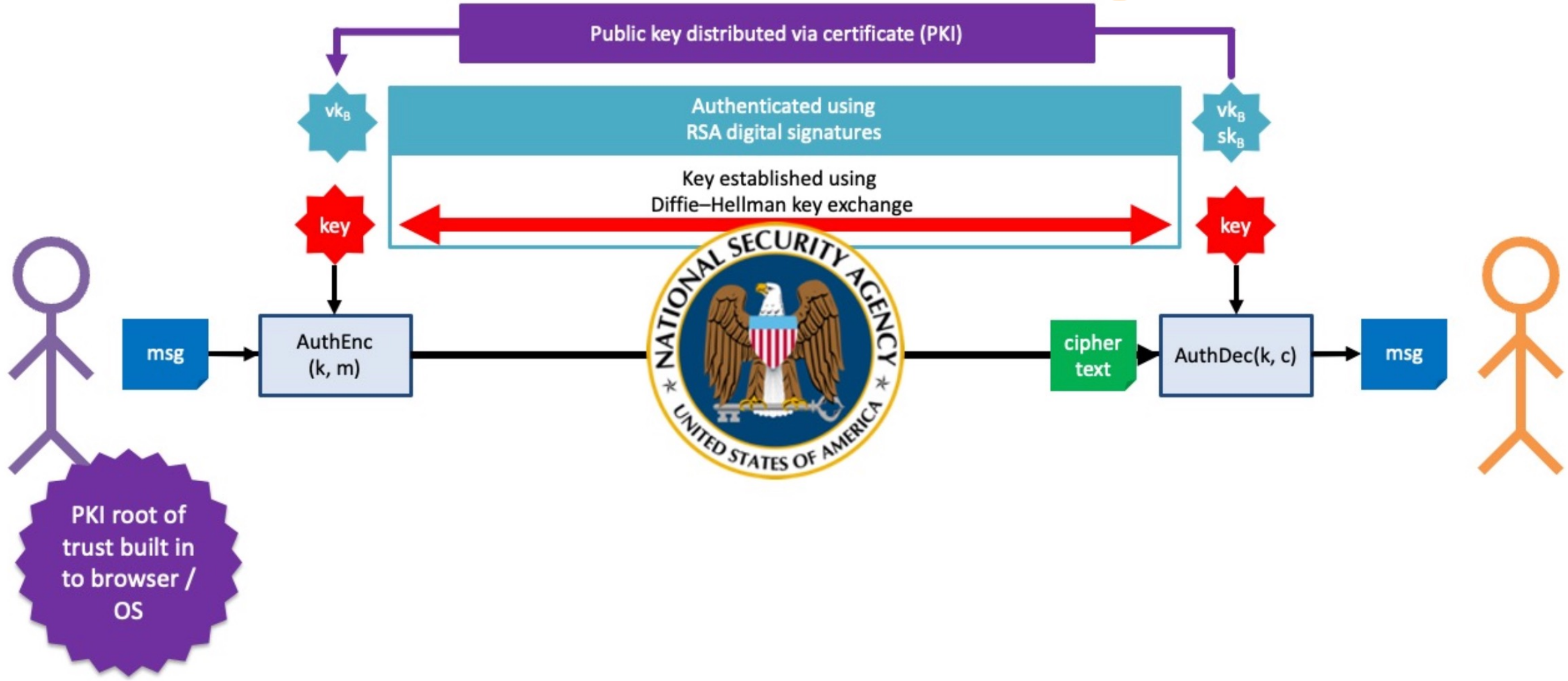


Key management
- PKI
- X.509 certificates

Secure channels
- TLS
- SSH
- Signal

Other applications
- Bitcoin
- Zero knowledge

Secure channels

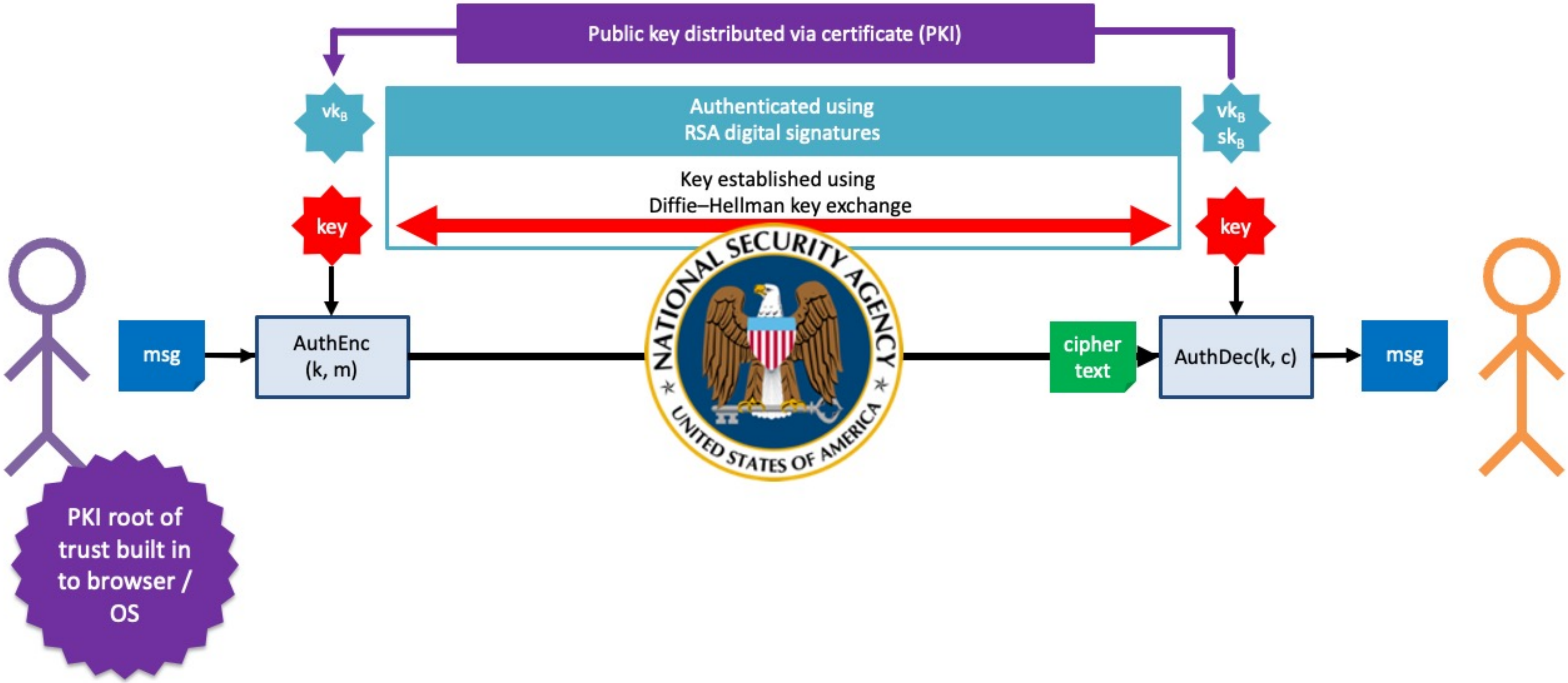


Authenticated encryption

Key exchange + authenticated encryption

Authenticated key exchange + authenticated encryption

Certified authenticated key exchange + authenticated encryption



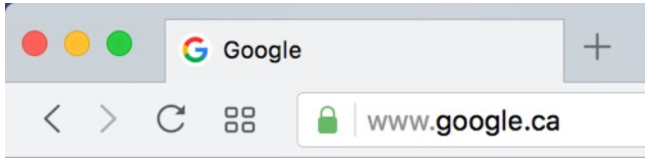
TLS

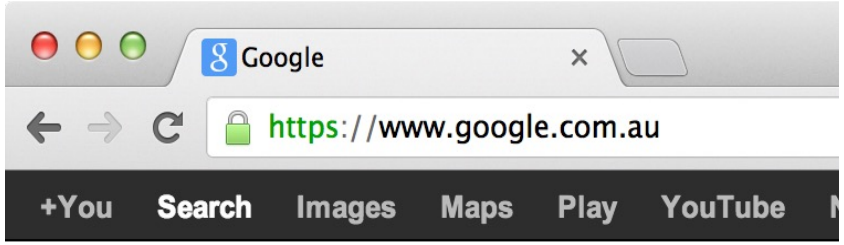
Terminology

- SSL: Secure Sockets Layer
- Proposed by Netscape
  - SSLv2: 1995
  - SSLv3: 1996

- TLS: Transport Layer Security
- IETF Standardization of SSL
  - TLSv1.0 = SSLv3: 1999
  - TLSv1.1: 2006
  - TLSv1.2: 2008
  - TLSv1.3: 2018

- HTTPS: HTTP (Hypertext Transport Protocol) over SSL





TLS

- Transport Layer Security (TLS) is a cryptographic protocol that operates above the transport layer to provide security services to applications
  - TLS runs over TCP
  - Datagram TLS (DTLS) runs over UDP

- Consists of a variety of modes and has many options
- Usually relies on a public key infrastructure

IETF Internet Protocol suite

TLS adds encryption to many application level protocols

Layer	Examples
Application	web (HTTP, HTTPS) email (SMTP, POP3, IMAP) login (SSH, Telnet)
Transport	connection-oriented (TCP) connectionless (UDP)
Internet	addressing and routing: • IPv4, IPv6 control (ICMP) security (IPsec)
Link	packet framing (Ethernet) physical connection • WLAN • ADSL • GSM/3G

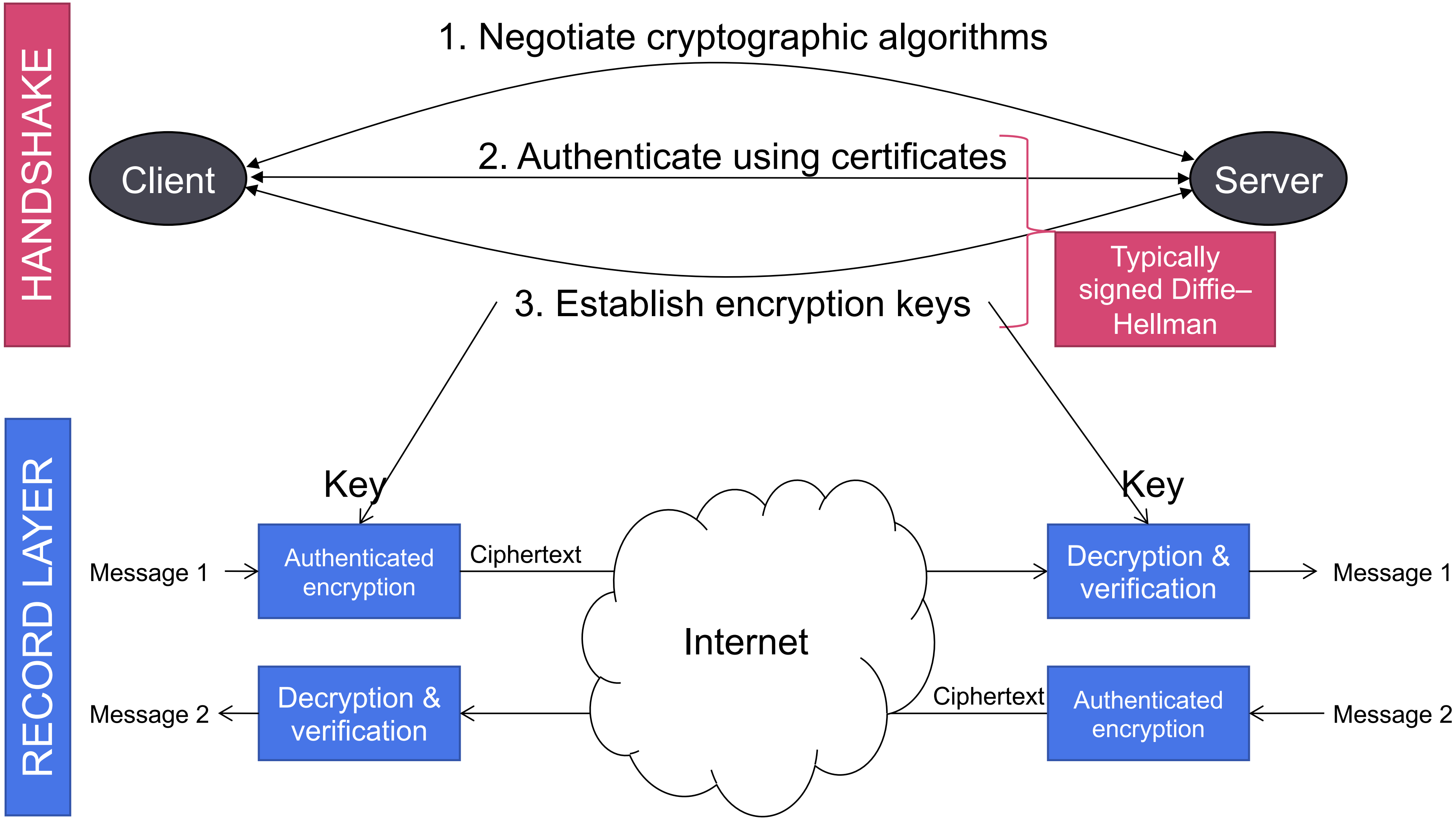
TLS

Security goals of TLS

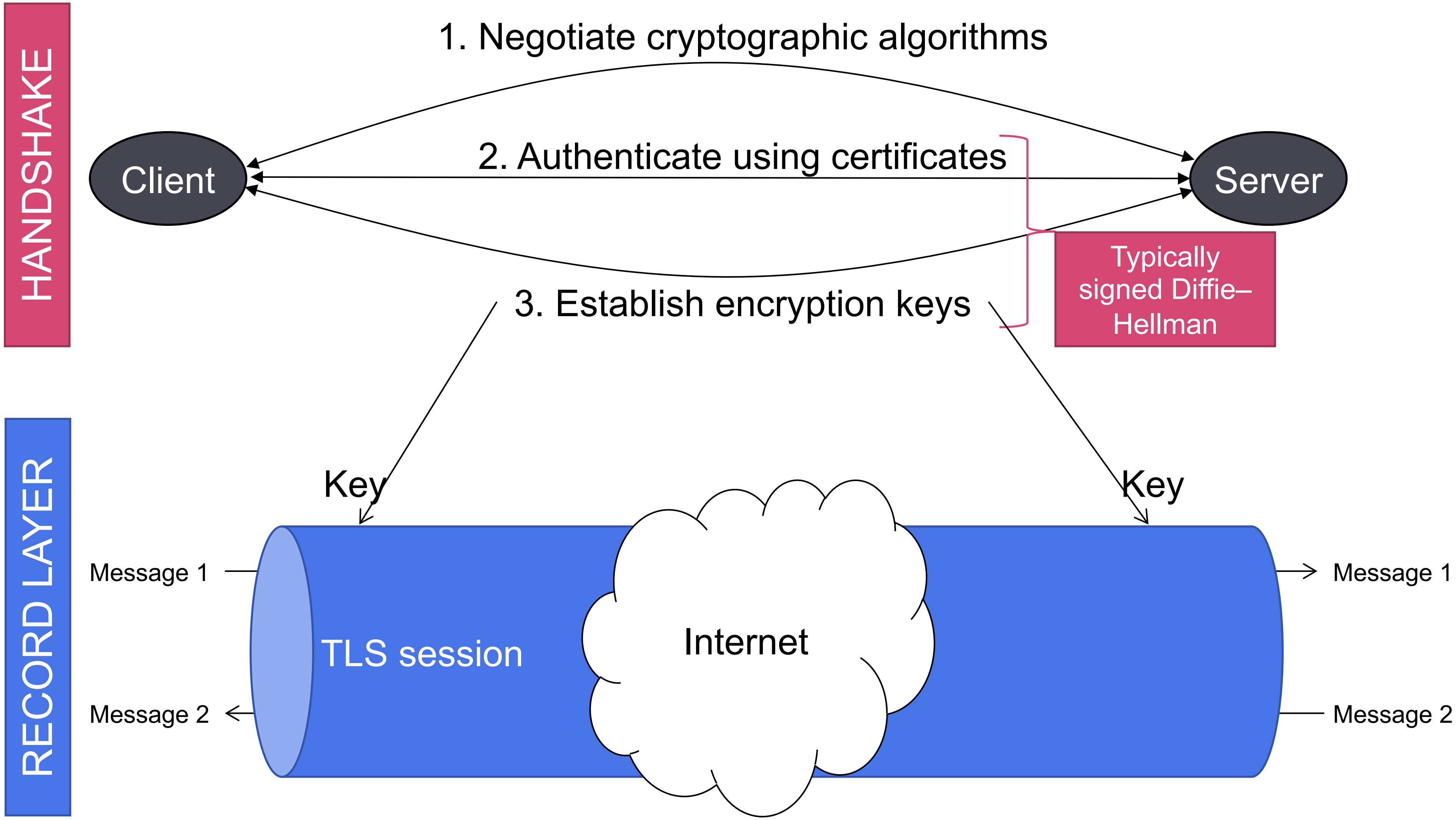
- Provides **authentication** based on public key certificates
  - server-to-client (always)
  - client-to-server (optional)
- Provides **confidentiality** and integrity of message transmission

- But only protects confidentiality if authentication is correct.

SSL/TLS Protocol



SSL/TLS Protocol



Structure of TLS

HANDSHAKE PROTOCOL

Negotiation of cryptographic parameters

Authentication (one-way or mutual) using public key certificates

Establishment of a master secret key

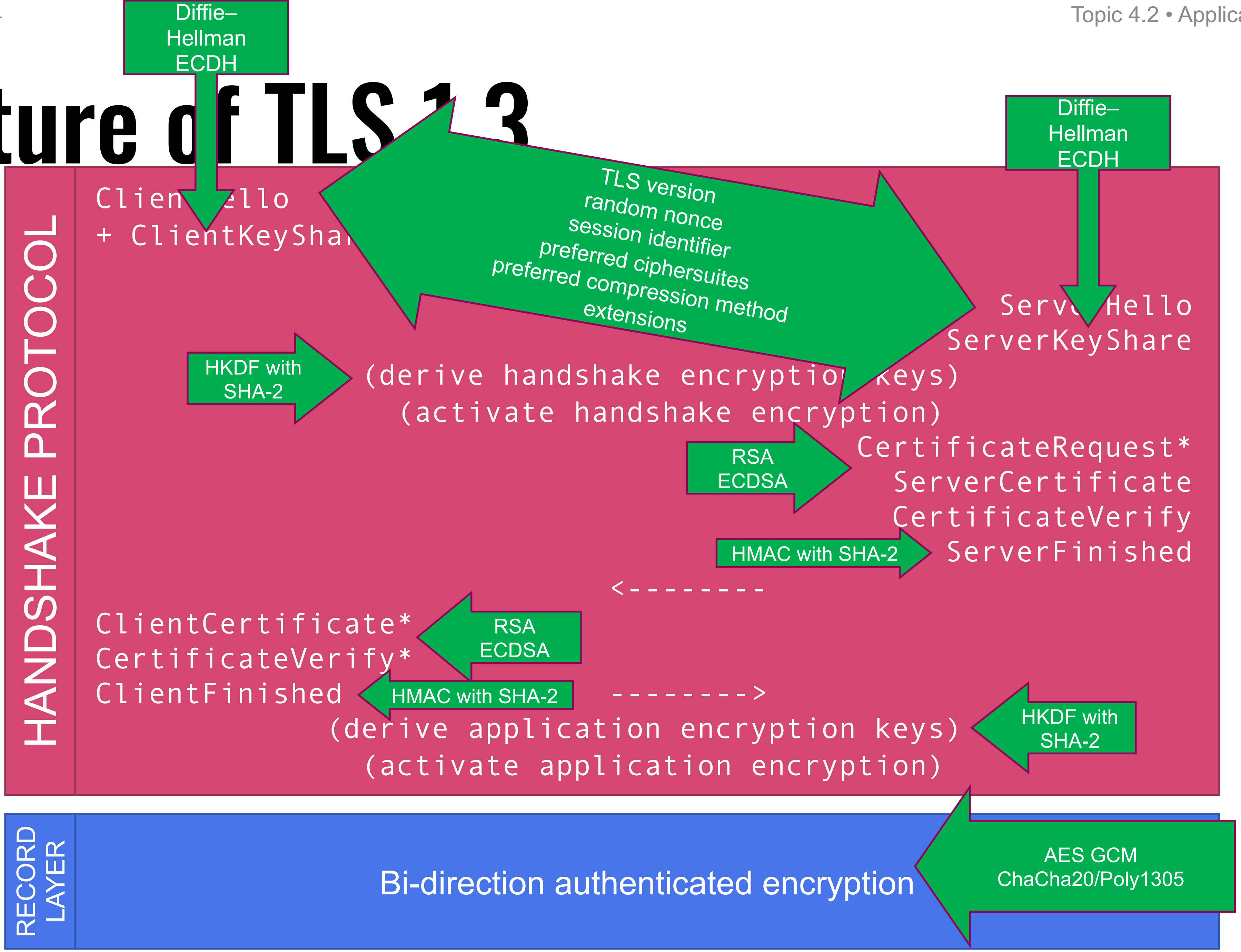
Derivation of encryption and authentication keys

Key confirmation

RECORD LAYER

Bi-direction authenticated encryption

Structure of TLS 1.3



TLS: Handshake Protocol

- **Authentication** (server-to-client)
  - Ensures that the connection really is with the server with the given domain name
  - Typically uses X.509 certificates
- **Authentication** (client-to-server): optional

- Handshake protocol also establishes keys that will be used in the record protocol for additional security services.

Key exchange in TLS 1.2

Option 1: RSA key transport

1. Server sends its RSA public key to the client inside certificate
2. Client picks a random "premaster" secret
3. Client sends premaster secret encrypted under server's RSA public key

- No forward secrecy
- Not permitted in TLS 1.3

Option 2: Ephemeral Diffie–Hellman

1. Server generates a temporary ("ephemeral") (EC)DH public key and sends to client, signs its using its signature key from certificate
2. Client generates a temporary (EC)DH public key and sends to server
3. Compute (EC)DH shared secret

- Has forward secrecy
- Only permitted method in TLS 1.3

TLS 1.3 – Setup in advance

- CA setup
  1. Certificate authority generates an RSA signature key pair (pk_{CA}, sk_{CA})
  2. Client has pk_{CA} installed in browser
- Certificate issuance
  1. Server generates an RSA signature key pair (pk_B, sk_B)
  2. Server gets CA to issue a certificate for its public key: $cert_B = \text{Sign}(sk_{CA}, \text{"Bob"} \parallel pk_B)$
  3. Bob gets $cert_B$

TLS 1.3 using signed Diffie–Hellman – Handshake (basic idea)

Client

Generate DH keypair (g^x, x)

$\xrightarrow{\text{hello, } g^x}$

Server

Generate DH keypair (g^y, y)

$s = \text{Sign}(sk_B, g^y)$

$k_{master} = H(g^{xy})$

$k_{AES}, k_{HMAC} = \text{KDF}(k_{master})$

$fin = \text{HMAC}(k_{HMAC}, \text{transcript})$

$\xleftarrow{\quad}$ g^y, $\text{AES}(k_{AES}, cert_B \mid s \mid fin)$

Client

$k_{master} = H(g^{xy})$

$k_{AES}, k_{HMAC} = \text{KDF}(k_{master})$

$\text{Verify}(pk_{CA}, \text{"Bob"} \mid\mid pk_B, cert_B)$

$\text{Verify}(pk_B, g^y, s)$

$fin \mathrel{?=} \text{HMAC}(k_{HMAC}, \text{transcript})$

TLS: Record Protocol Overview

- **Message Confidentiality**:
  - Ensure that the message contents cannot be read in transit.
  - The Handshake Protocol is used to establish a symmetric key to be used to encrypt SSL/TLS payloads in the record protocol.
- **Message Integrity**:
  - Ensure that the receiver can detect if a message is modified in transmission.
  - The Handshake Protocol establishes a shared secret key used to construct a Message Authentication Code.
- **Message Replay Protection**
  - The same data is not delivered multiple times
  - Achieved using counters and integrity protection
- Supplied by an **authenticated encryption** (with associated data) scheme (AEAD)

Is TLS secure?

What should TLS do?

- Server-to-client authentication
- Client-to-server authentication (optional)
- Confidential communication with integrity and replay protection

What doesn't TLS do?

- Hide source/destination
- Hide length information
- (Trusted creation of certificates)
- Password-based authentication
- Stop denial of service attacks
- Prevent web application vulnerabilities

TLS security considerations

Trust and digital certificates

- TLS uses public keys – provided in digital certificates
- Certificates should be verified – requires tracing certificate pathways
- Web browsers come with pre-configured lists of root certificates but users can add or remove root CAs

One-way or mutual authentication?

- Authentication is usually of server to client only, not mutual
- Users usually do not have client certificates
- Typically, authentication of users is not performed in handshake
- Instead, password authentication over server-authenticated HTTPS channel

(Perfect) Forward secrecy

- An adversary who later learns the server's long-term private key shouldn't be able to read previous transmissions
- Signed Diffie–Hellman key exchange provides forward secrecy
- TLS ≤1.2 supported RSA public key encryption for key exchange which does not provide forward secrecy

Components of TLS

Crypto primitives

- RSA, DSA, ECDSA
- Diffie–Hellman, ECDH
- HMAC
- MD5, SHA1, SHA-2
- DES, 3DES, RC4, AES
- Export grade

Ciphersuite details

- Data structures
- Key derivation
- Encryption modes, IVs
- Padding

Advanced functionality

- Alerts & errors
- Certification / revocation
- Negotiation
- Renegotiation
- Session resumption
- Key reuse
- Compression
- State machine

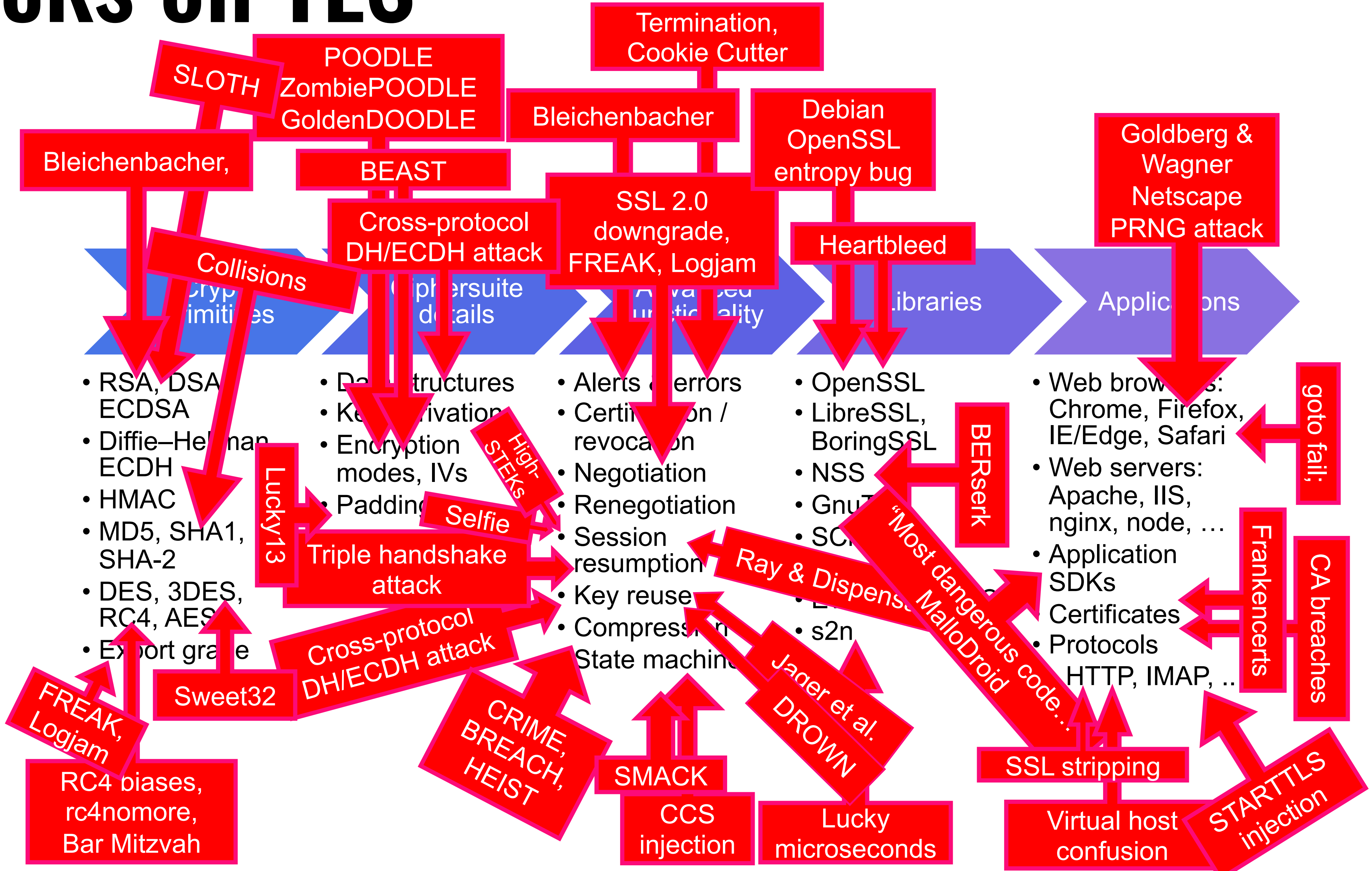
Libraries

- OpenSSL
- LibreSSL, BoringSSL
- NSS
- GnuTLS
- SChannel
- Java JSSE
- Everest / miTLS
- s2n

Applications

- Web browsers: Chrome, Firefox, IE/Edge, Safari
- Web servers: Apache, IIS, nginx, node, …
- Application SDKs
- Certificates
- Protocols
  - HTTP, IMAP, ..

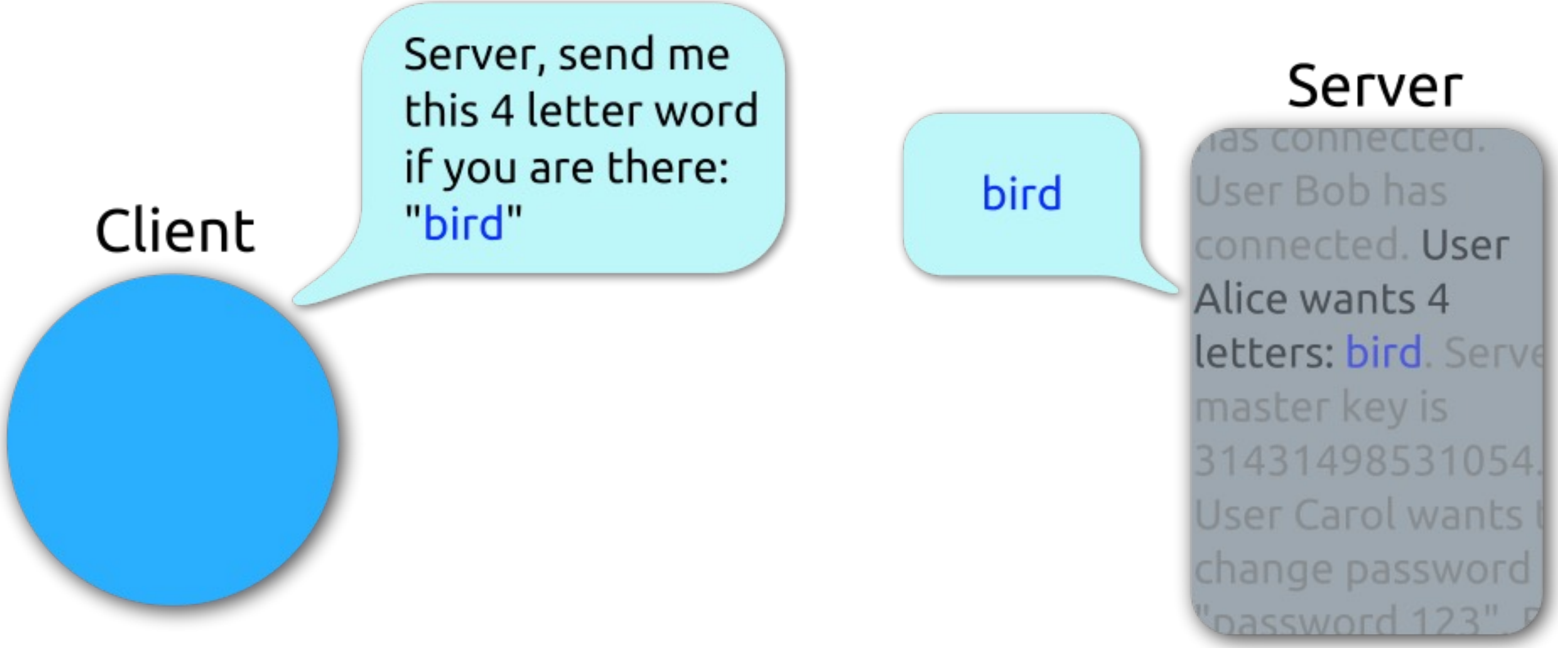
Attacks on TLS



The Heartbleed Bug (April 2014)

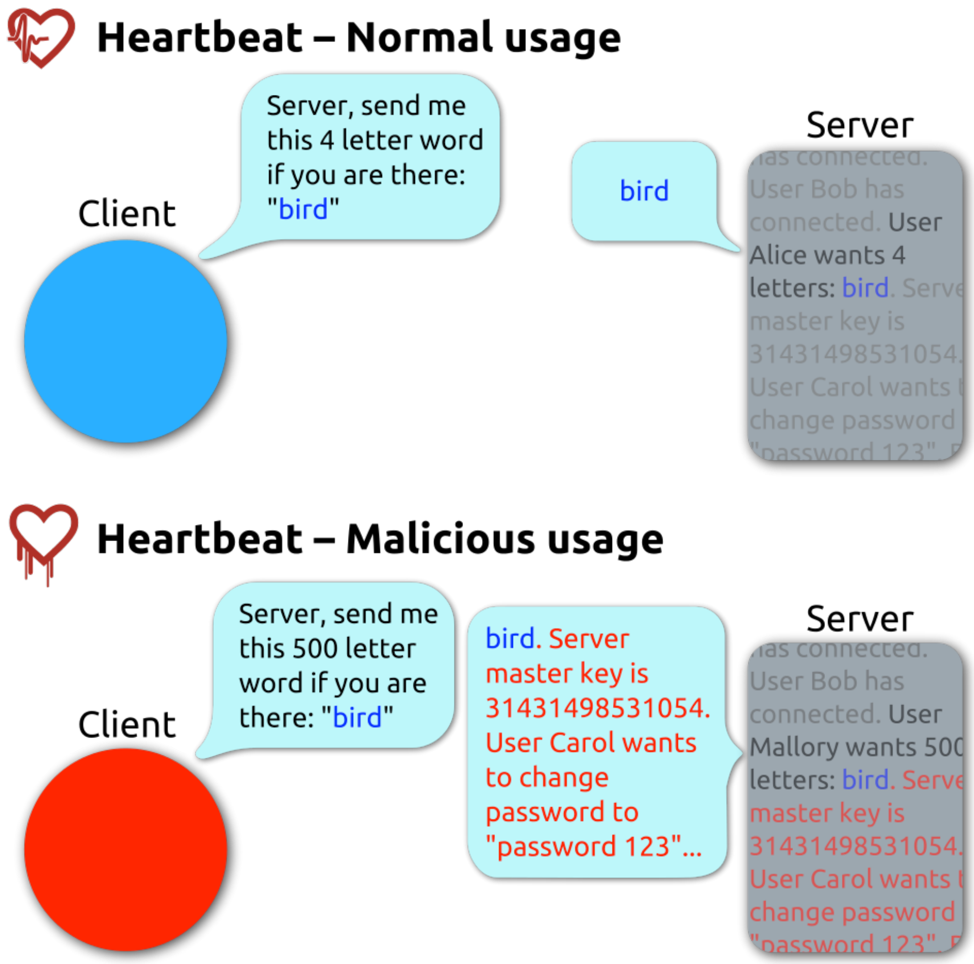
- The Heartbeat Extension for TLS and DTLS protocols
- Published and implemented in 2012
- OpenSSL's implementation contained a buffer overflow bug that allowed up to 64Kb of memory to be returned to malicious

TLS Heartbeat Message



http://en.wikipedia.org/wiki/File:Simplified_Heartbleed_explanation.svg

Malicious Heartbleed Message



http://en.wikipedia.org/wiki/File:Simplified_Heartbleed_explanation.svg

Impact of Heartbleed

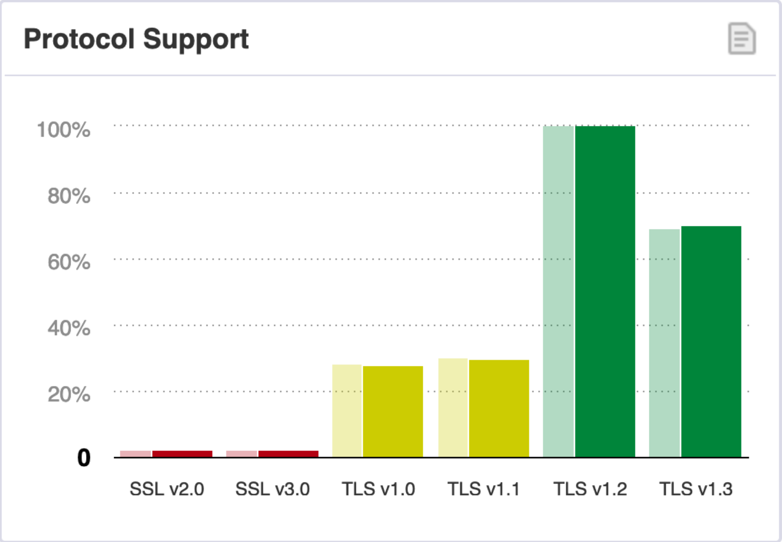
- Affected versions of OpenSSL are OpenSSL 1.0.1 through 1.0.1f (inclusive)
- Vulnerable systems could have had portions of memory exposed
  - Including server's RSA private keys or users' passwords, depending on the application
- After patching, private keys need to be renewed and passwords should be changed

- Really just a software bug – not at all related to cryptography or TLS

TLSv1.3: The Next Generation

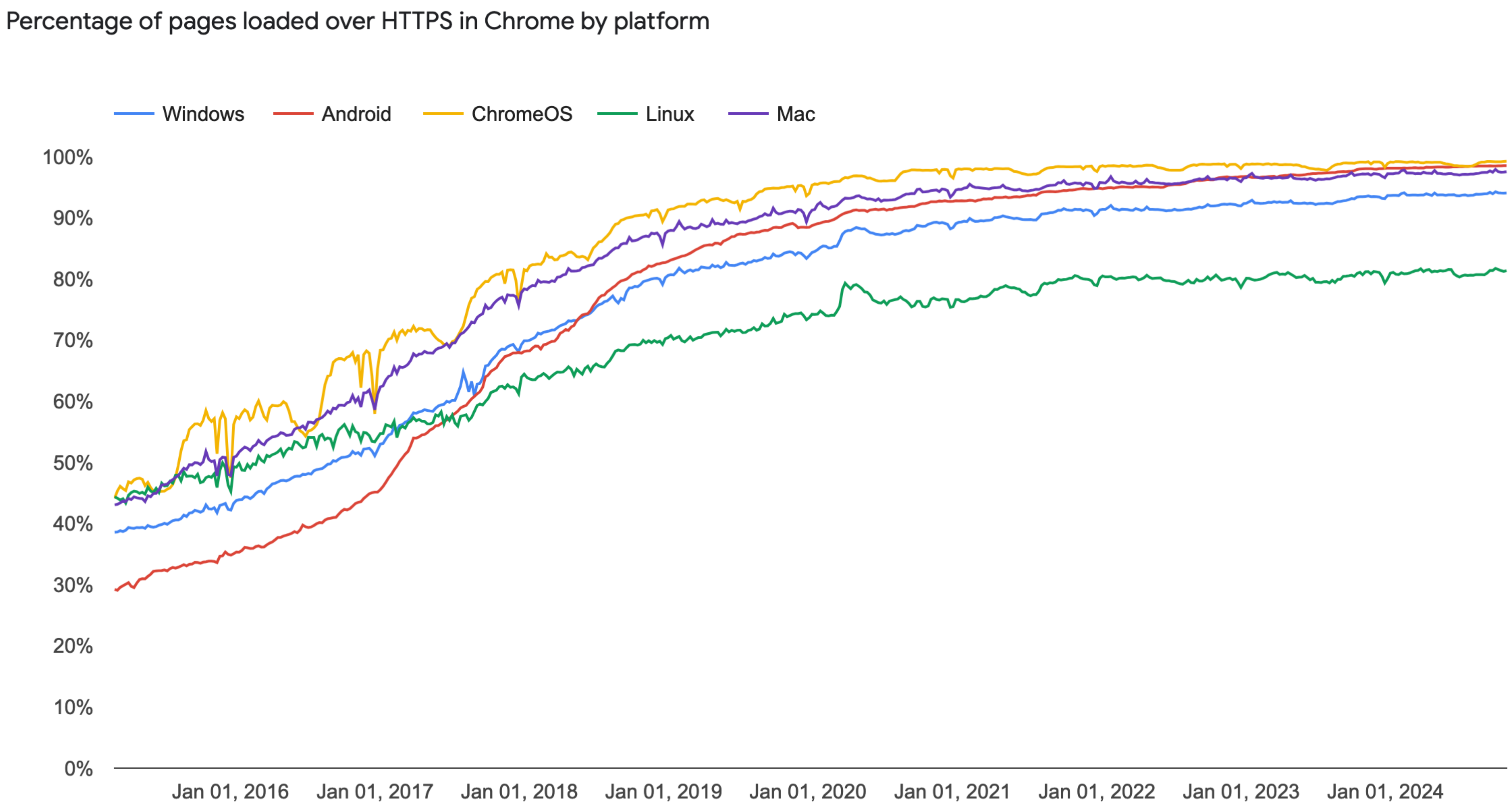
- Multi-year process involving good interaction between academics and industry
- Standardized in August 2018

- Primary goals:
  - remove ciphersuites without forward secrecy
  - remove obsolete / deprecated algorithms and modes
  - provide low-latency mode with fewer round trips
  - encrypt more of the handshake to improve privacy



https://www.ssllabs.com/ssl-pulse/ May, 2024

Adoption of HTTPS on the Internet



https://transparencyreport.google.com/https/overview?hl=en

Recent activity on TLS

Post-quantum:

- Hybrid post-quantum + classical (Kyber + elliptic curve) key exchange

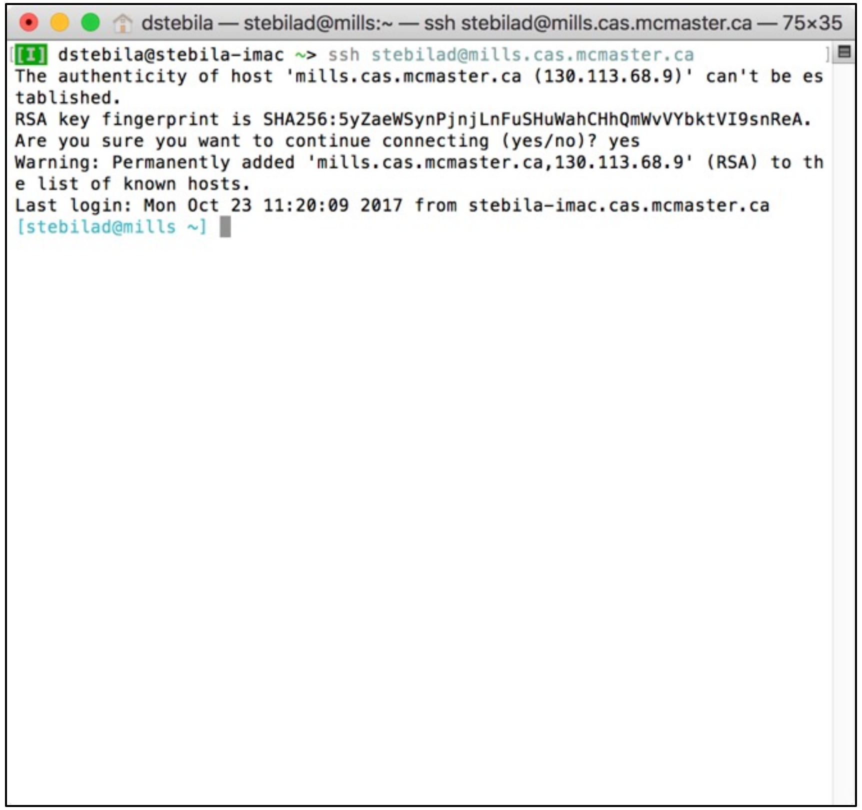
Encrypted ClientHello:

- Encrypting more metadata earlier in the handshake

https://datatracker.ietf.org/group/tls/documents/

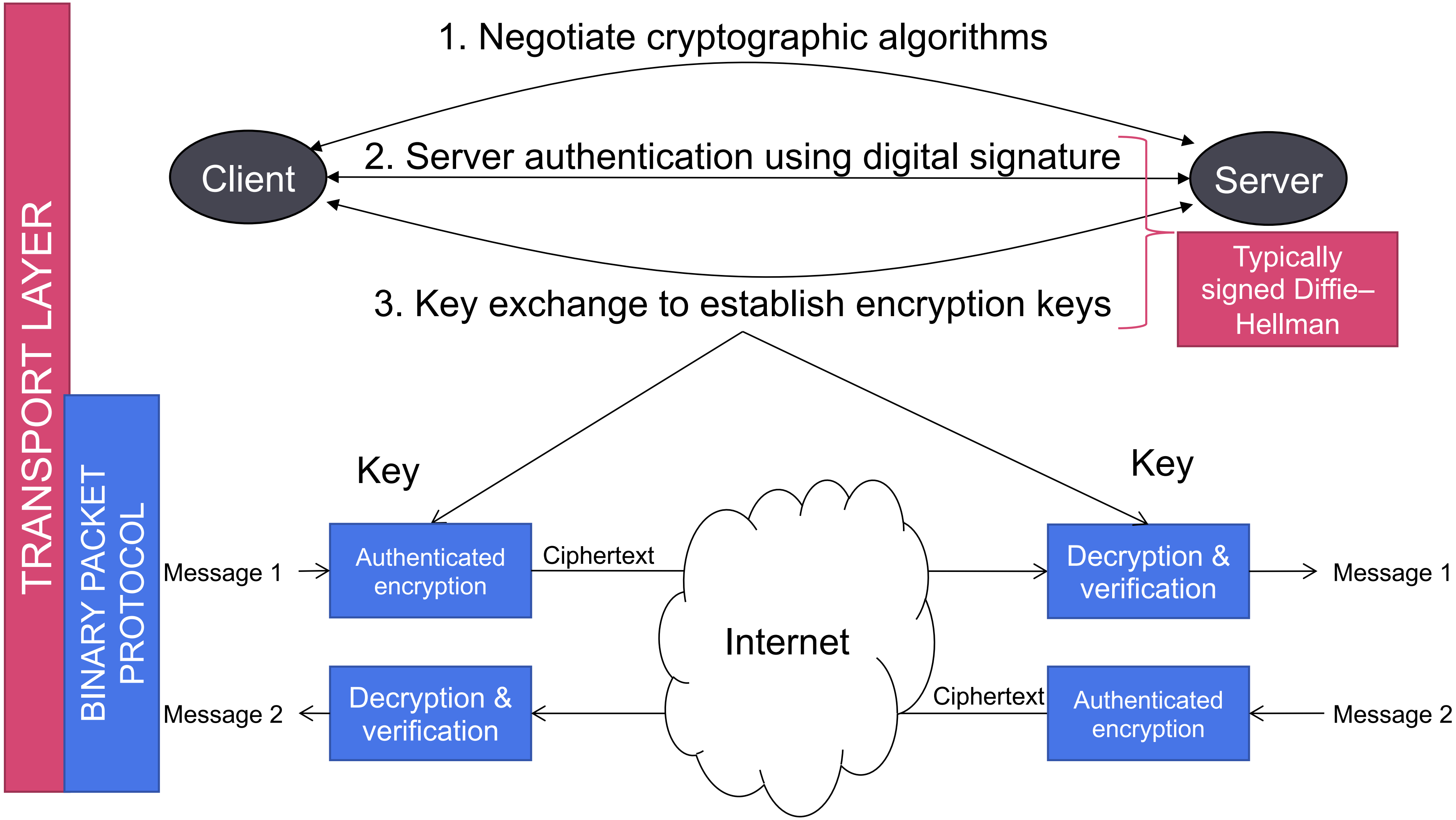
SSH

SSH (Secure Shell) protocol

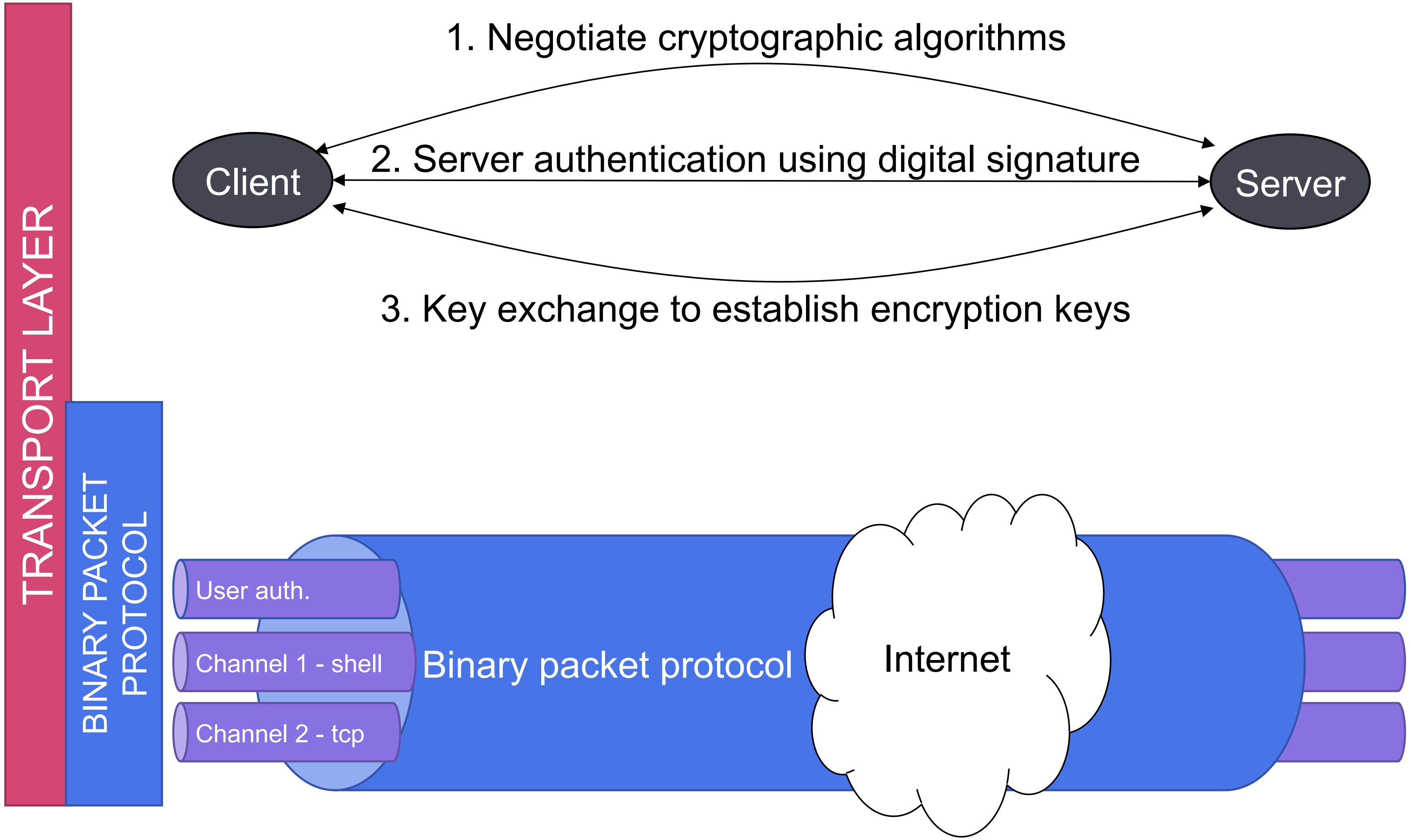


- SSH used for secure remote access (like telnet, but secure)
  - Occasionally used as a "poor man's VPN"
- Run over TCP, typically on port 22
- Provides public key authentication of servers and clients and encrypted communication
- Specified in RFCs by the IETF

SSH protocol



SSH protocol



Security goals of SSH

- **Message Confidentiality**
  - Protects against unauthorised data disclosure
  - Achieved using encryption
- **Message Integrity**
  - Protects against unauthorised changes to data during transmission (intentional or unintentional)
  - Achieved using message authentication code
- **Message Replay Protection**
  - The same data is not delivered multiple times
  - Achieved using counters and integrity protection
- **Peer Authentication**
  - Ensures that traffic is being sent from the expected party
  - Server-to-client auth:
    - based on public keys
  - Client-to-server auth:
    - based on passwords or public keys

Server authentication in SSH

- Based on public key digital signatures
- Unlike TLS, (typically) does not use X.509 certificates – just a raw public key

```
bash-5.0$ ssh dstebila@cpu141.math.private.uwaterloo.ca
The authenticity of host 'cpu141.math.private.uwaterloo.ca (172.27.7.113)' can't
 be established.
ECDSA key fingerprint is SHA256:Bfo4cNjTisSAQ0aboFQuAziStlVcZUvwzwHiSAqI3PI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'cpu141.math.private.uwaterloo.ca,172.27.7.113' (ECDS
A) to the list of known hosts.
```

```
bash-5.0$ ssh dstebila@cpu141.math.private.uwaterloo.ca
```

```
bash-5.0$ ssh dstebila@cpu141.math.private.uwaterloo.ca
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:Bfo4cNjTisSAQ0aboFQuAziStlVcZUvwzwHiSAqI3PI.
Please contact your system administrator.
```

- No systematic solution for authentic distribution of public keys
  - Console displays public key fingerprint (hash) on first login
  - User should check hash through some out-of-band method
    - E.g. phone call to sysadmin
  - SSH client saves hash for verifying future connections
  - and raises alert if changed or authentication fails

In the news

- Suppose a bit-flip fault happens during RSA signing
- Given a faulted signature and an unfaulted signature, can recover secret key
- In a dataset of 5.2 billion SSH records, observed 590,000 faults and used 4,900 of these to recover 189 RSA private keys

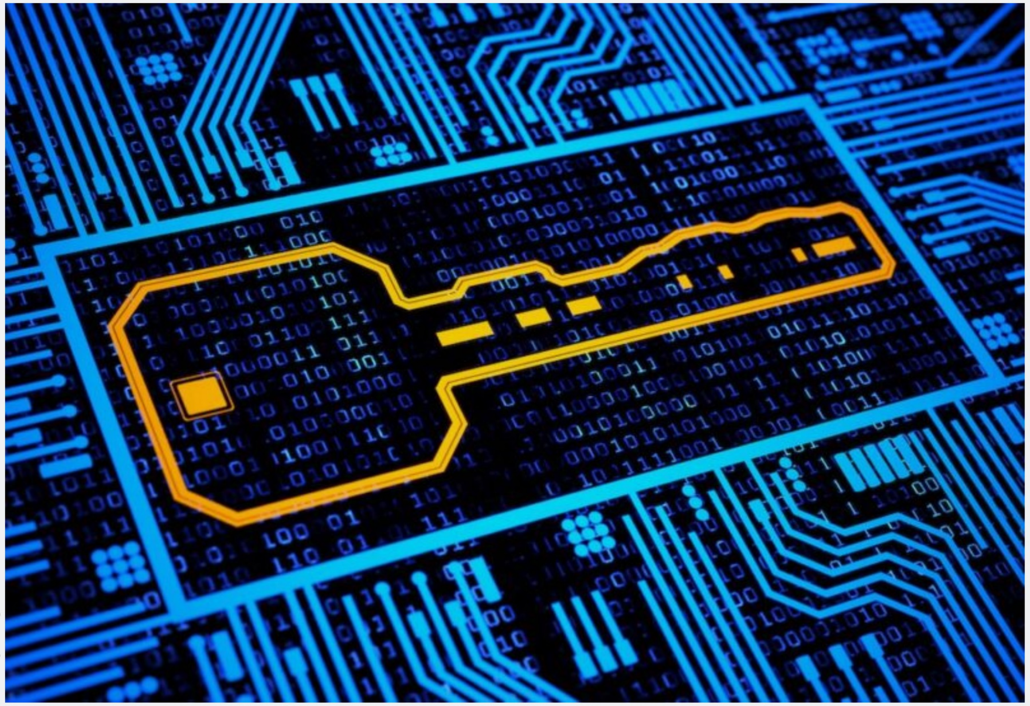
ars PRO++

MORE FUN WITH LATTICE-BASED CRYPTOGRAPHY —

In a first, cryptographic keys protecting SSH connections stolen in new attack

An error as small as a single flipped memory bit is all it takes to expose a private key.

DAN GOODIN - 11/13/2023, 7:30 AM



Getty Images

Enlarge

138

For the first time, researchers have demonstrated that a large portion of cryptographic keys used to protect data in computer-to-server SSH traffic are vulnerable to complete compromise when naturally occurring computational errors occur while the connection is being established.

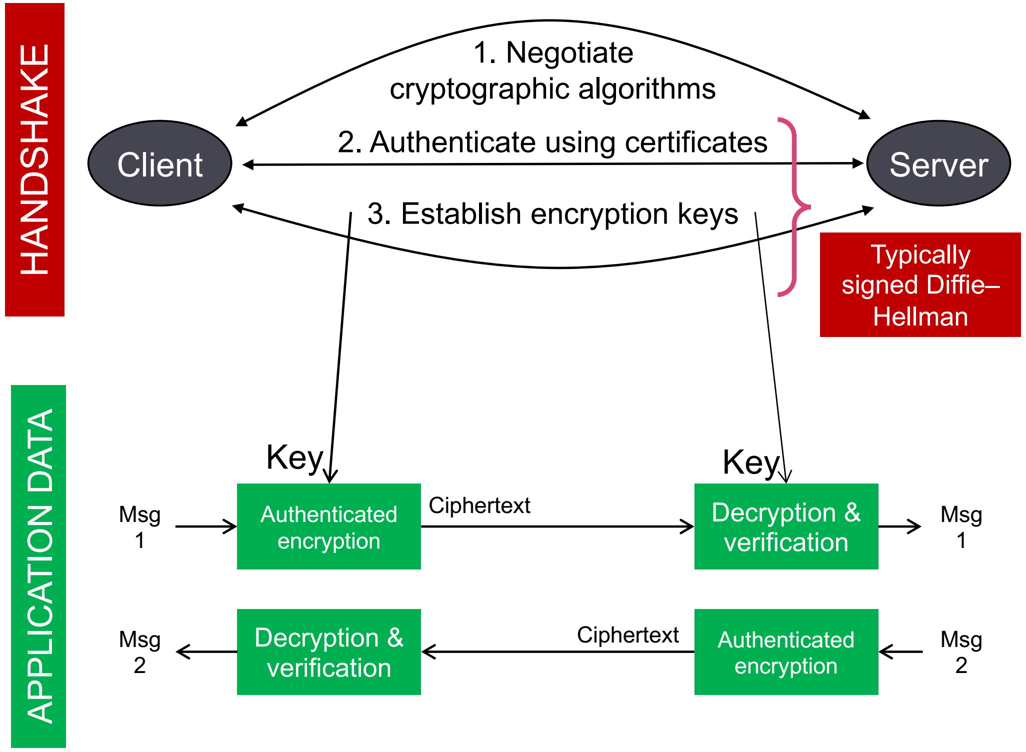
https://arstechnica.com/security/2023/11/hackers-can-steal-ssh-cryptographic-keys-in-new-cutting-edge-attack/
https://eprint.iacr.org/2023/1711

4+8+7 things to remember from CO 487

CO 487/687 • Fall 2023 Things to remember

Secure channels

e.g. Transport Layer Security (TLS) protocol



22