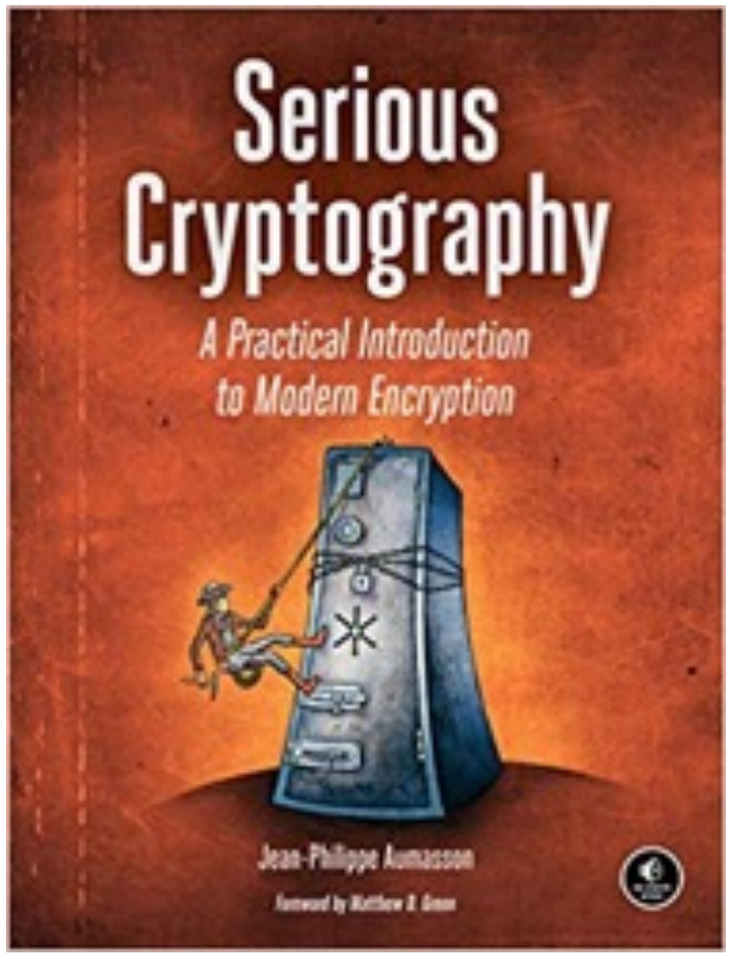
CNIT 141

Cryptography for Computer Networks



13. TLS

Updated 12-8-21

Topics

- Target Applications and Requirements
- The TLS Protocol Suite
- TLS 1.3 Improvements Over TLS 1.2
- The Strengths of TLS Security
- How Things Can Go Wrong

TLS

- Protects communications at layer 4
- Can carry any type of content
  - Email, Web traffic, mobile apps, ...
  - Machine-to-machine comms for IoT

TLS Vulnerabilities

- TLS grew too big and bloated
- Many attacks
  - Heartbleed
  - BEAST
  - CRIME
  - POODLE

TLS 1.3

- Overhaul of protocol
  - Removed unnecessary features
  - Replaced old algorithms
- Simpler, faster, and more secure

Target Applications and Requirements

Secure Channel

- TLS ensures that data is confidential, authenticated, and unmodified
- Prevents MiTM attacks
  - By authenticating servers with trusted ***Certificate Authorities***

Four Requirements

- Efficient
  - Minimizing CPU load compared to unencrypted comms
- Interoperable
  - Work on any hardware or OS
- Extensible
  - Support additional features & algorithms
- Versatile
  - Not bound to any specific application

The TLS Protocol Suite

Transport Layer

- TLS is not in the transport layer
  - It's above layer 4, adding security to it
- Can run over TCP or UDP
- UDP version is called DTLS
  - Datagram Transport Layer Security

7. **Application layer**
6. **Presentation layer**
5. **Session layer**
4. **Transport layer**
3. **Network layer**
2. **Data link layer**
1. **Physical layer**

History

- SSL: began in 1995, from Netscape
  - SSL 2.0 and SSL 3.0 had security flaws
    - Should no longer be used
- TLS
  - TLS 1.0 (1999): least secure
  - TLS 1.1 (2006): better but contains weak algorithms

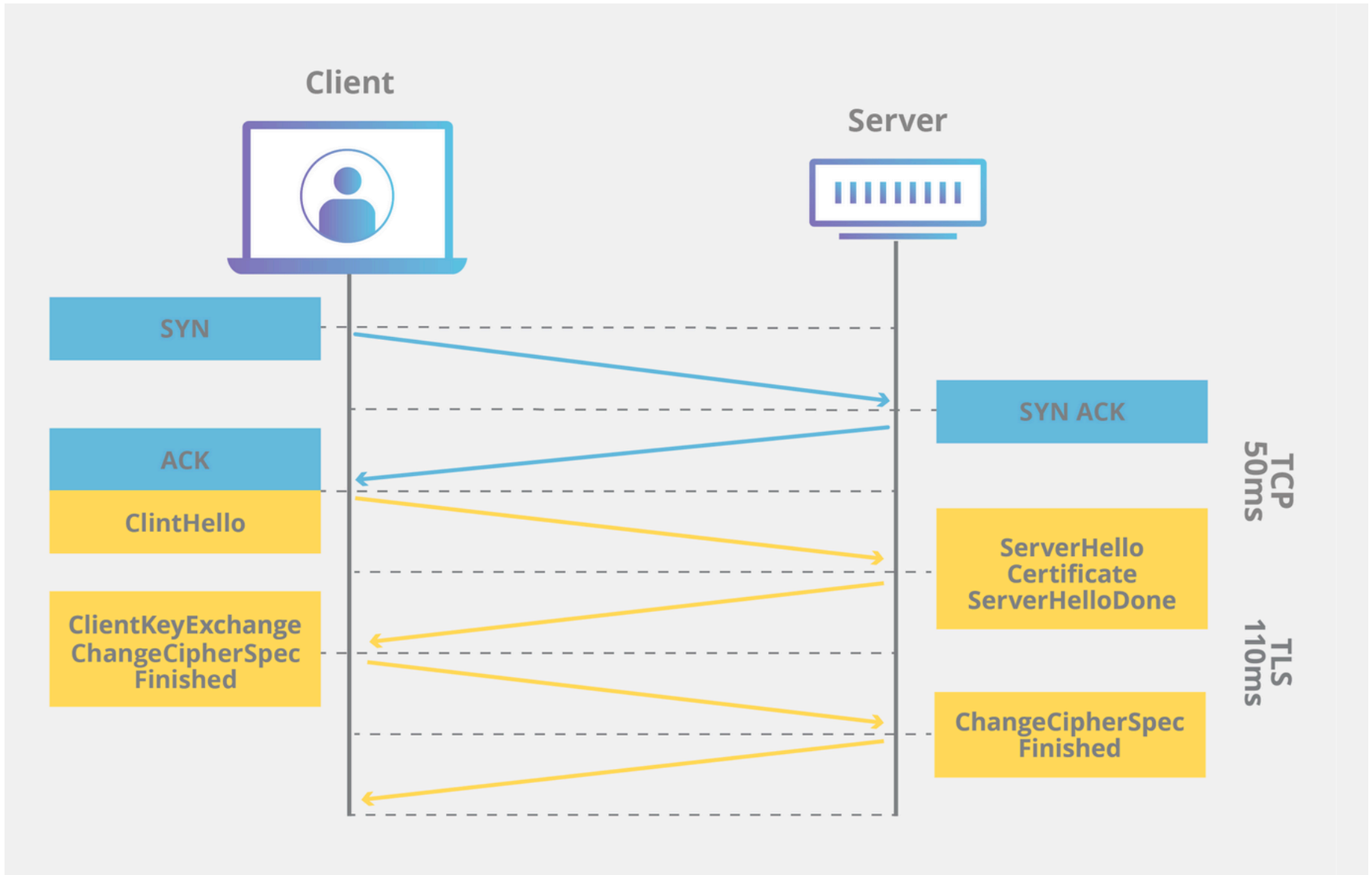
TLS 1.2

- From 2008
  - Better than previous versions
  - Complex and hard to configure
    - Supports AES-CBC, vulnerable to padding oracle attacks
    - Inherited dozens of features and design choices from earlier versions
- TLS 1.3 is a major overhaul and improvement

TLS in a Nutshell

- ***Record protocol***
  - Data encapsulation
- ***Handshake protocol***
  - Key agreement

TLS Handshake



- From Cloudflare (link Ch 13a)

Hello

- ClientHello
  - Lists ciphers available
- ServerHello
  - Selects a cipher to use

ClientHello

```
      1374 12.773720   10.203.14.99      74.6.160.107      TLSv1.3      583  Client Hello
      1375 12.774820   10.203.14.99      34.214.70.219     TLSv1.2      583  Client Hello
      1376 12.781006   192.229.211.36    10.203.14.99      TLSv1.3     1514  Continuation Data
▶ Transmission Control Protocol, Src Port: 61840, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
▼ Secure Sockets Layer
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 512
    ▼ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 508
        Version: TLS 1.2 (0x0303)
        Random: 0b9c16e5bd0d5c0b3fd07b83907ba413a4b9106d95f8c995...
        Session ID Length: 32
        Session ID: 650117fcb9c6222d6c3cb73aca45e9fca05969d66b632cc1...
        Cipher Suites Length: 36
      ▼ Cipher Suites (18 suites)
          Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
          Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
          Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
          Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
          Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
          Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
          Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
          Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
          Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
          Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
```

ServerHello

```
    1396 12.973387   74.6.160.107        10.203.14.99        TLSv1.3     1514  Server Hello, Change Cipher Spec, Application Data
▶ Frame 1396: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: CiscoMer_b4:b2:86 (e0:cb:bc:b4:b2:86), Dst: Apple_97:ee:3d (f0:18:98:97:ee:3d)
▶ Internet Protocol Version 4, Src: 74.6.160.107, Dst: 10.203.14.99
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 61840, Seq: 1, Ack: 518, Len: 1448
▼ Secure Sockets Layer
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 122
    ▼ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 118
        Version: TLS 1.2 (0x0303)
        Random: d3efb45cc0c8b9ad1b5af7447bbd670c6c32fe02eb7b2eb2...
        Session ID Length: 32
        Session ID: 650117fcb9c6222d6c3cb73aca45e9fca05969d66b632cc1...
        Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
        Compression Method: null (0)
        Extensions Length: 46
      ▶ Extension: supported_versions (len=2)
      ▶ Extension: key_share (len=36)
▼ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
▼ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 36
    Encrypted Application Data: 827c7faff7df9fe4a723a055f631b3902d6cc533f52ecaa5...
```

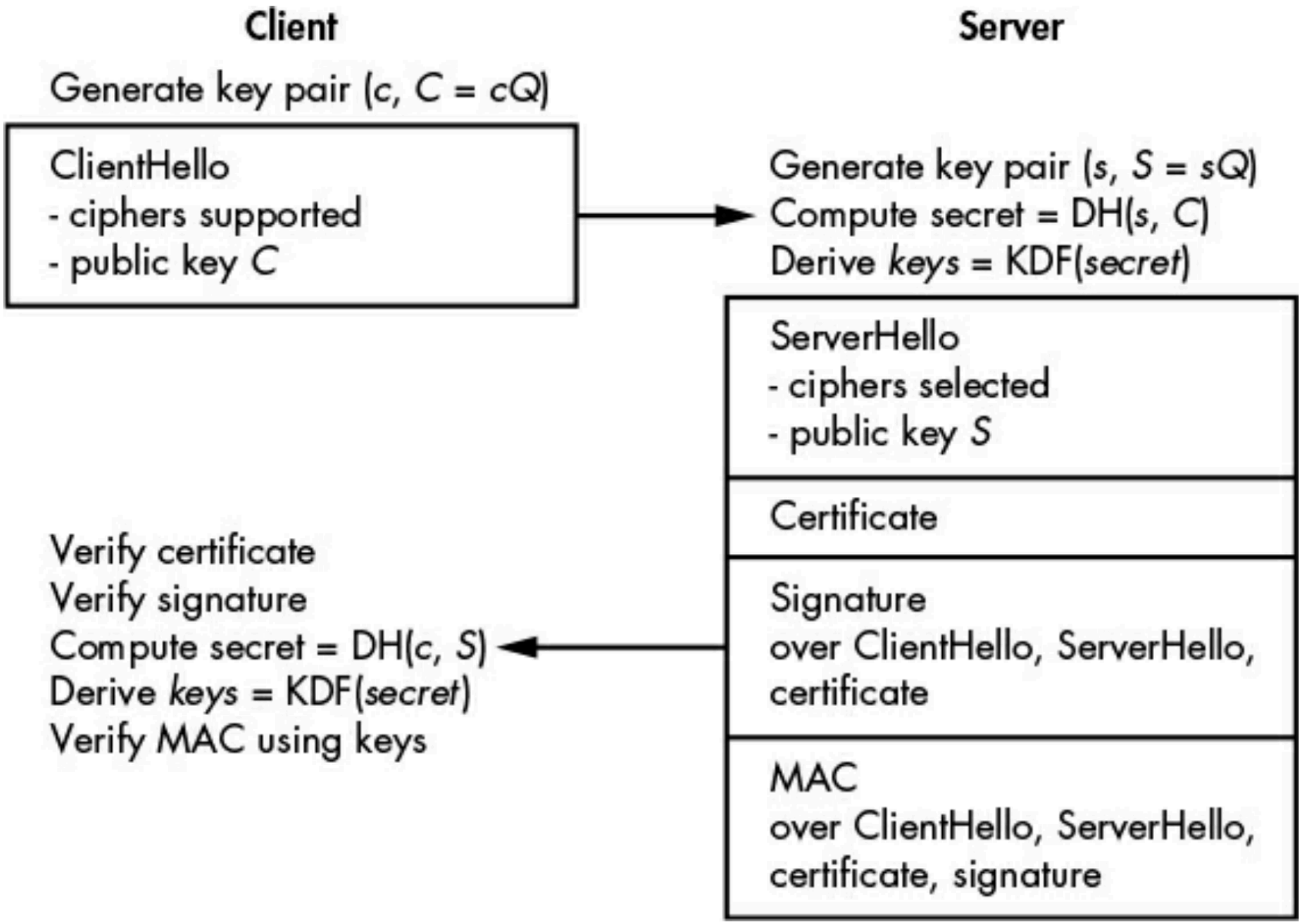


Figure 13-1: The TLS 1.3 handshake process when connecting to HTTPS websites

Certificates and Certificate Authorities (CA)

- Server uses a ***certificate*** to authenticate itself
  - Verified by a CA
- CA's public keys hard-coded into browsers
  - Trusted third party

Certificate

This certificate has been verified for the following uses:

SSL Client Certificate

SSL Server Certificate

Issued To

Common Name (CN)	cloudflare.com
Organization (O)	Cloudflare, Inc.
Organizational Unit (OU)	
Serial Number	0A:68:BB:98:4A:50:73:99:F4:71:6E:80:9A:44:A7:B0

Issued By

Common Name (CN)	DigiCert ECC Extended Validation Server CA
Organization (O)	DigiCert Inc
Organizational Unit (OU)	www.digicert.com

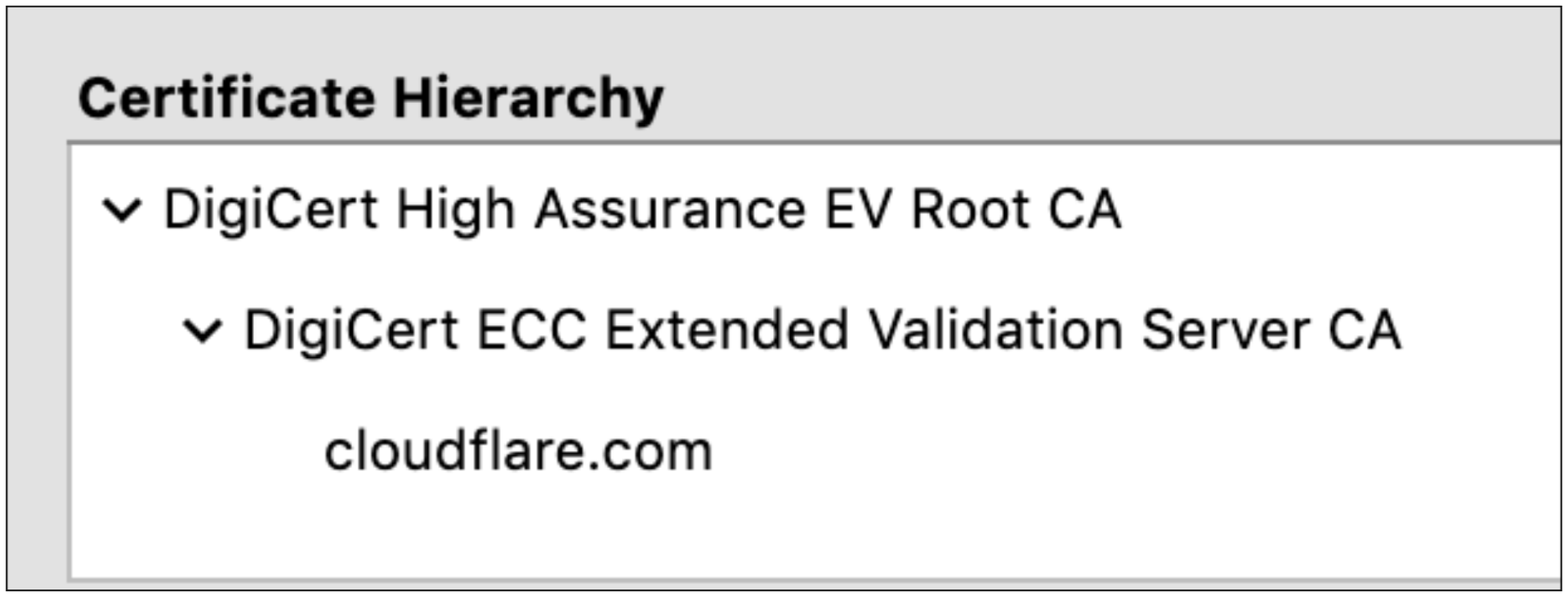
Period of Validity

Begins On	October 29, 2018
Expires On	November 3, 2020

Fingerprints

SHA-256 Fingerprint	84:69:46:45:CB:D1:83:BF:9B:E4:72:CE:70:2B:D1:89: 24:6E:1F:CD:15:44:FF:EA:D4:66:D9:56:A7:16:90:BF
SHA1 Fingerprint	6A:4C:B2:49:B7:1B:66:59:DA:BB:96:01:15:FD:7A:01:BF:C9:96:8C

Certificate Chain



Record Protocol

Structure of a TLS Record

A TLS record is a chunk of data of at most 16 kilobytes, structured as follows:

- **The first byte represents the type of data transmitted and is set to the value 22 for handshake data, 23 for encrypted data, and 21 for alerts. In the TLS 1.3 specifications, this value is called ContentType.**
- **The second and third byte are set to 3 and 1, respectively. These bytes are fixed for historical reasons and are not unique to TLS version 1.3. In the specifications, this 2-byte value is called ProtocolVersion.**
- **The fourth and fifth bytes encode the length of the data to transmit as a 16-bit integer, which can be no larger than 2^{14} bytes (16KB).**
- **The rest of the bytes are the data to transmit (also called the *payload*), of a length equal to the value encoded by the record's fourth and fifth bytes.**

Zero Padding

- Adds zeroes to plaintext for short messages
- Mitigates ***traffic analysis***
  - Deducing contents from message size

TLS 1.3 Cryptographic Algorithms

- Three types of algorithms are used
- **Authenticated encryption**
- **Key derivation function**
  - Hash function that derives secret keys from a shared secret
- A **Diffie-Hellman** operation

Authenticated Ciphers

- TLS 1.3 supports only three algorithms
  - AES-GCM
  - AES-CCM
    - Slightly less efficient than AES-GCM
  - ChaCha20 stream cipher
    - Combined with Poly1305 MAC
- Secret key can be 128 or 256 bits
  - Unsafe 64-bit or 80-bit keys not allowed

Key Derivation Function (KDF)

- HKDF, based on HMAC
  - Uses SHA-256 or SHA-384

Diffie-Hellman
Two Options

- Elliptic Curve
  - With three NIST curves, or
  - Curve25519, or Curve448
- Group of integers modulo a prime number
  - Traditional Diffie-Hellman
  - 2048 - 8192 bits
  - Security of 2048-bit group is weak link
    - Less than 100 bits

TLS 1.3 Improvements Over TLS 1.2

Removed Weak Algorithms

- MD5, SHA-1
- RC4, AES-CBC
- MAC-then-Encrypt alorithms
  - Like HMAC-SHA-1
  - Replaces with authenticated ciphers
    - More efficient and secure

Removed Insecure Feature

- Optional data compression
  - Enabled the CRIME attack
    - Length of the compressed message leaked information about contents

New Features

- That make TLS 1.3 more secure
  - Downgrade protection
  - Single round-trip handshake
  - Session resumption

Downgrade Attack

- MiTM attacker modifies ClientHello

I want
TLS 1.3

Send her
TLS 1.1
instead

Sending
ServerHello
for TLS 1.1

Downgrade Protection

- To prevent this, 8 bytes in the ServerHello denote the TLS version
  - They are cryptographically signed so the MiTM can't change them
  - The client can check them to see what TLS version is being provided

Single Round-Trip Handshake

- In TLS 1.2
  - Client sends some data, waits for response
  - Client sends more data, waits for response
- TLS 1.3 combines it all into one round-trip
  - Saves time

Session Resumption

- Leverages the pre-shared key exchanged in a previous session
  - To bootstrap a new session
- Two benefits
  - Client can start encrypting immediately
  - No need for certificates

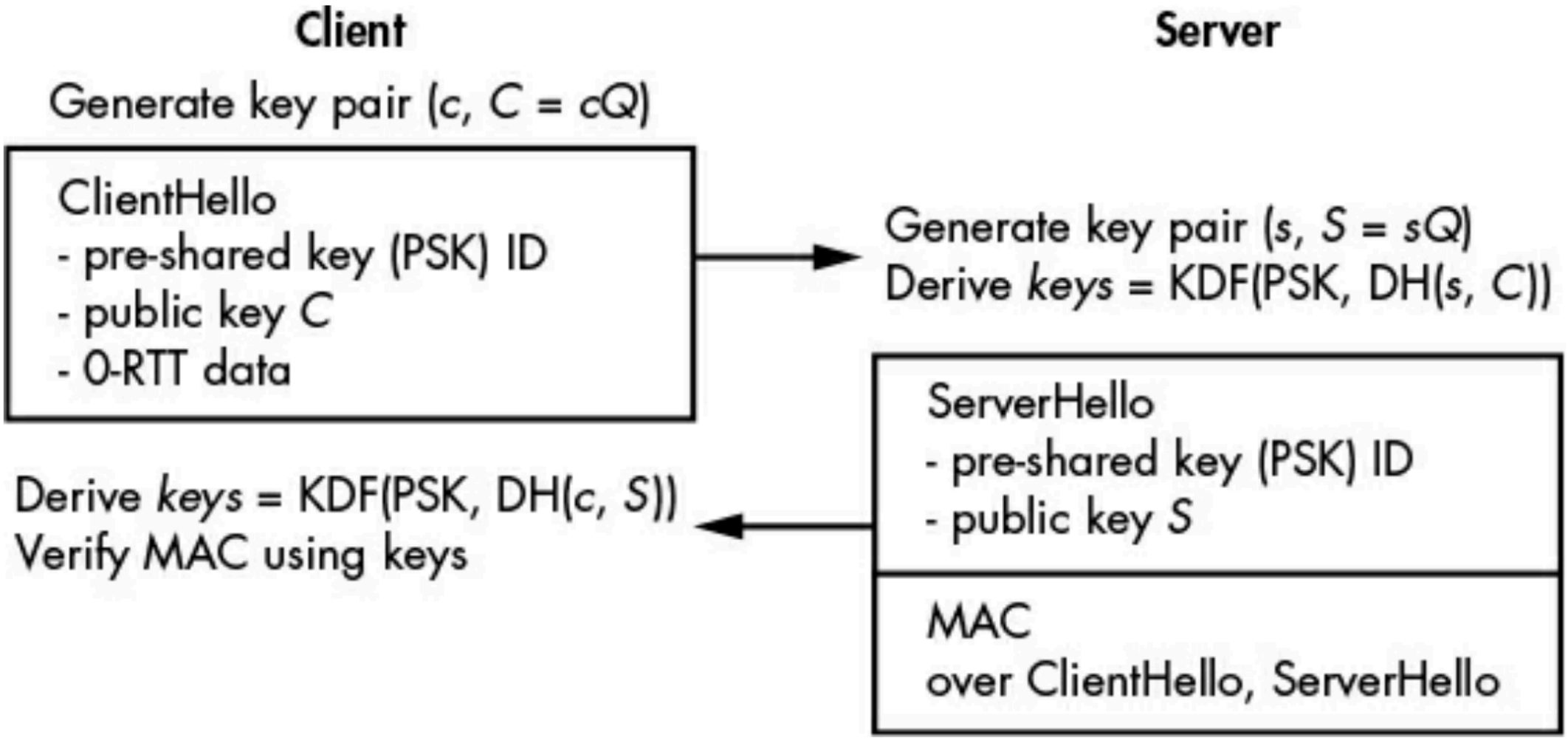


Figure 13-2: The TLS 1.3 session resumption handshake. The 0-RTT data is the session resumption data sent along with the ClientHello.

The Strengths of TLS Security

Authentication

- TLS 1.3 handshake authenticates the server with a certificate and CA
- Client is not authenticated, but can authenticate after the TLS handshake with:
  - Username & password in a TLS record
  - Secure cookie over TLS
  - A ***client certificate*** (rarely used)

Forward Secrecy

- TLS 1.3 provides forward secrecy in both a data leak and a breach
  - If an attacker can steal a client's RAM
  - Exposes keys and secrets for the current session
  - And any old sessions still stored in RAM
- Solution: use Secure Strings

SecureString Class

Namespace: System.Security

Assemblies: System.Security.SecureString.dll, mscorlib.dll, netstandard.dll, System.Runtime.InteropServices.dll

Represents text that should be kept confidential, such as by deleting it from computer memory when no longer needed. This class cannot be inherited.

- Provided by Microsoft
  - Since 2005 in .NET 2.0
  - Link Ch 13b

How Things Can Go Wrong

Compromised CA

- Happened in 2011 to DigiNotar
- CA's private key compromised
- Attacker created fake certificates for Google services

RYAN SINGEL SECURITY 03.24.10 01:55 PM

LAW ENFORCEMENT APPLIANCE SUBVERTS SSL

Trustwave sold root certificate for surveillance

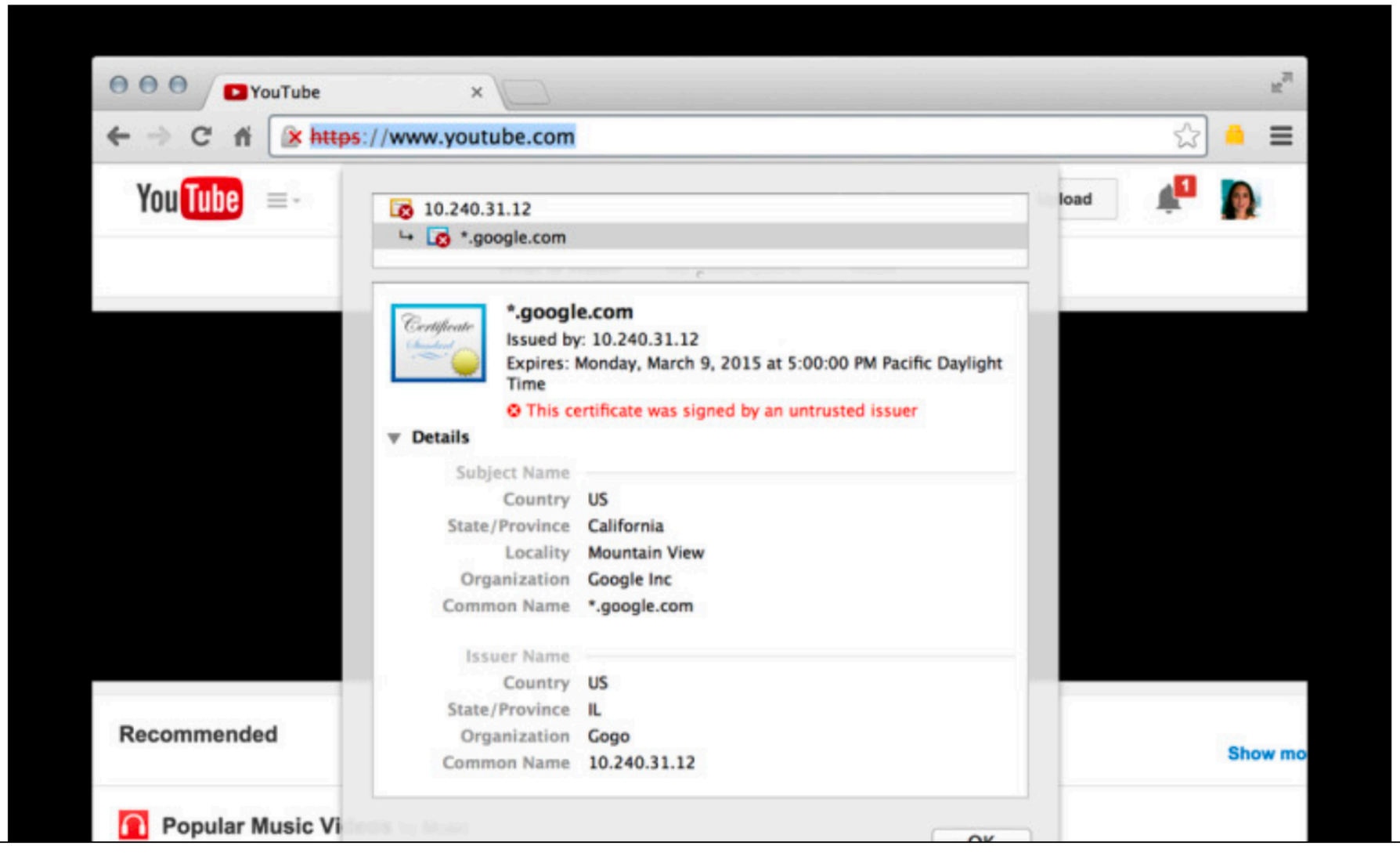
The certificate authority has admitted selling a subordinate root certificate to an organisation to allow it to eavesdrop on encrypted employee traffic, prompting Mozilla to consider revoking trust in Trustwave

By Tom Espiner | February 9, 2012 -- 15:55 GMT (07:55 PST) | Topic: Security

Gogo Inflight Internet is intentionally issuing fake SSL certificates

By Steven Johns · Jan 4, 2015 22:10 EST · **HOT!**

41



SSL broken! Hackers create rogue CA certificate using MD5 collisions

Using computing power from a cluster of 200 PS3 game consoles and about $700 in test digital certificates, a group of hackers in the U.S.

By Ryan Naraine for Zero Day | December 30, 2008 -- 06:00 GMT (22:00 PST) | Topic: Enterprise Software

Compromised Client

- Attacker who controls client can
  - Capture session keys
  - Read decrypted data
- Or install a rogue CA certificate

Bugs in Implementation

- Heartbleed
  - Leaked secrets from HTTPS servers in 2014

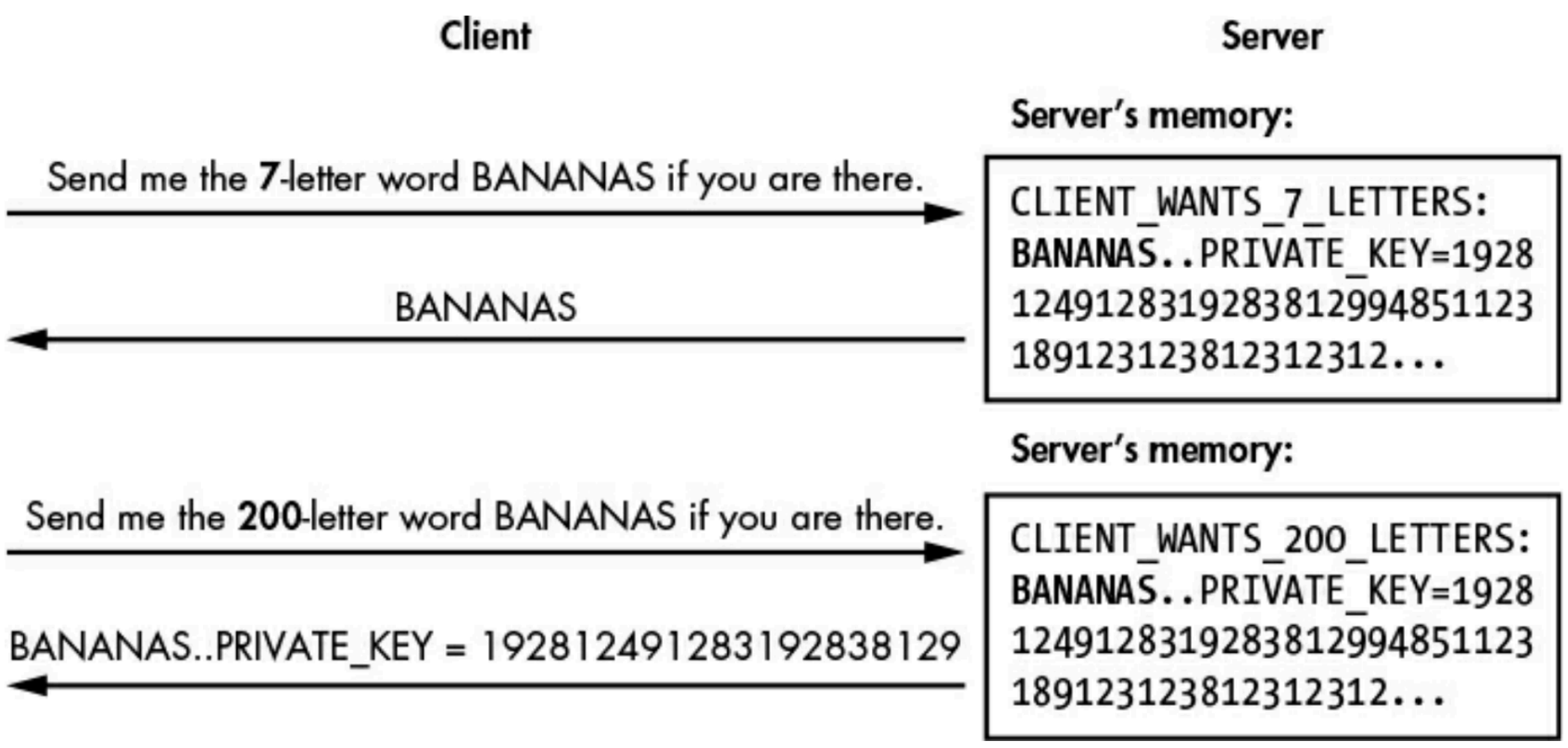


Figure 13-3: The Heartbleed bug in OpenSSL implementations of TLS

Improving TLS

- SSL Labs TLS test
  - Lets you test any site's certificate
  - Or a browser's TLS configuration
- Let's Encrypt
  - Free TLS certificates for everyone

HTTP Strict Transport Security

From Wikipedia, the free encyclopedia

HTTP Strict Transport Security (**HSTS**) is a web security policy mechanism that helps to protect websites against protocol downgrade attacks[1] and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should interact with it using only secure HTTPS connections,[2] and never via the insecure HTTP protocol. HSTS is an IETF standards track protocol and is specified in RFC 6797.

HTTP Public Key Pinning

From Wikipedia, the free encyclopedia

HTTP Public Key Pinning (**HPKP**) is an Internet security mechanism delivered via an HTTP header which allows HTTPS websites to resist impersonation by attackers using mis-issued or otherwise fraudulent digital certificates.[1] It does this by delivering a set of public keys to the client (e.g. web browser), which should be the only ones trusted for future connections to the same domain name.

For example, attackers might compromise a certificate authority, and then mis-issue certificates for a web origin. To combat this risk, the HTTPS web server serves a list of “pinned” public key hashes valid for a given time; on subsequent

The mechanism was deprecated by the Google Chrome team in late 2017 because of its complexity and dangerous side-effects. Google recommends using the Expect-CT as a safer alternative.[2][3]

Expect-CT

While HPKP has been deprecated, a new header stepped in to prevent fraudulent SSL certificates from being served to clients: `Expect-CT`.


Kahoot!