IoT Security and Privacy

Other IoT communication protocols - HTTP, HTTPS and Websockets

YIER JIN

UNIVERSITY OF FLORIDA

EMAIL: YIER.JIN@ECE.UFL.EDU

SLIDES ARE ADAPTED FROM PROF. XINWEN FU @ UCF/UMASS

Learning Outcomes

Upon completion of this unit:

- Students will be able to analyze the HTTP protocol so that student can analyze IoT system that use HTTP as the communication protocol
- Students will be able to analyze the HTTPS protocol so that student can analyze IoT system that use HTTP as the communication protocol
- Students will be able to explain webscokets so that student can analyze IoT system that use webscokets as the communication protocol

Prerequisites and Module Time

Prerequisites

- Students should have taken classes on operating system and computer architecture.
- Students must have taken crypto and know how public key crypto and symmetric key crypto work.
- Students should have mastered programming Raspberry Pi.
- Students should know basic concepts of networking.

Module time

- Two-hour lecture
- One-hour homework

Outline

http

https

Websockets

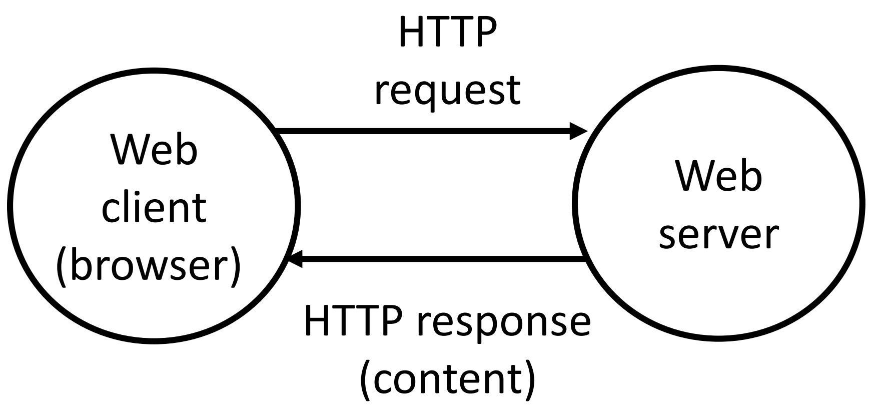
HyperText Transfer Protocol (HTTP)

HTTP, communication protocol between a client (browser) and a web server

- Client and server establish TCP connection
- Client requests content
- Server responds with requested content
- Client and server close connection

HTTP history

- Initiated by Tim Berners-Lee at CERN in 1989
- Standardized by IETF and W3C, the first definition of HTTP/1.1 in 1997, 1999 and 2014
- Standardized HTTP/2 in 2015



HTTP messages

HTTP is the language between web clients and web servers

Each message has three parts

A client request message has components in the following order

- Request line
- Header section
- Message body

A server response message has components in the following order

- Response line
- Header section
- Message body

Anatomy of an HTTP Transaction

```
unix> telnet www.aol.com 80                     Client: open connection to server
Trying 205.188.146.23...                        Telnet prints 3 lines to the terminal
Connected to aol.com.
Escape character is '^]'.
GET / HTTP/1.1                                  Client: request line
host: www.aol.com                               Client: required HTTP/1.1 HOST header
                                                Client: empty line terminates headers.
HTTP/1.0 200 OK                                 Server: response line
MIME-Version: 1.0                               Server: followed by five response headers
Date: Mon, 08 Jan 2001 04:59:42 GMT
Server: NaviServer/2.0 AOLserver/2.3.3
Content-Type: text/html                         Server: expect HTML in the response body
Content-Length: 42092                           Server: expect 42,092 bytes in the resp body
                                                Server: empty line ("\r\n") terminates headers
<html>                                          Server: first HTML line in response body
...                                             Server: lines of HTML not shown.
</html>                                         Server: last HTML line in response body
Connection closed by foreign host.
unix>                                           Client: closes connection and terminates
```

Client – Request line

Client sends a request message to server at a port, 80 by default

The first part of the message is the request line, containing:

- A method (HTTP command) such as GET or POST
  GET – request data from a specified resource
  POST – submit data for processing to a specified resource
- A document address, and
- An HTTP version number

Example:

- GET /index.html HTTP/1.0

Other methods

Other methods beside GET and POST are:

- HEAD: Like GET, but ask that *only* a header be returned
- PUT: Request to store the entity-body at the URI
- DELETE: Request removal of data at the URI
- LINK: Request header information to be associated with a document on the server
- UNLINK: Request to undo a LINK request
- OPTIONS: Request information about communications options on the server
- TRACE: Request that the entity-body be returned as received (used for debugging)

Client – Header information

The second part of a request is optional header information, notifying the server:

- Client software
- Acceptable data/file formats

All information is in the form of Name: Value

Example:

- User-Agent: Mozilla/2.02Gold (WinNT; I)
- Accept: image/gif, image/jpeg, */*

A blank line ends the header

Client request headers

Accept: type/subtype, type/subtype, ...

- Specifies media types that the client prefers to accept

Accept-Language: **en**, **fr**, **de**

- Preferred language (For example: English, French, German)

User-Agent: string

- The browser or other client program sending the request

From: **dave@acm.org**

- Email address of user of client program

Cookie: name=value

- Information about a cookie for that URL
- Multiple cookies can be separated by commas

Client – Entity body

The third part of a request (after the blank line) is the entity-body for optional data

- Used mostly by POST requests
- Always empty for a GET request

Server – Status line

The first part is the status line, including:

- The HTTP version
- A status code
- A short description of what the status code means

Example: HTTP/1.1 404 Not Found

Status codes are in groups:

100-199	Informational
200-299	The request was successful
300-399	The request was redirected
400-499	The request failed
500-599	A server error occurred

Common status codes

200 OK

- Everything worked, here's the data

301 Moved Permanently

- URI was moved, but here's the new address for your records

302 Moved temporarily

- URL temporarily out of service, keep the old one but use this one for now

400 Bad Request

- There is a syntax error in your request

403 Forbidden

- You can't do this, and we won't tell you why

404 Not Found

- No such document

408 Request Time-out, 504 Gateway Time-out

- Request took too long to fulfill for some reason

Server – Header information

The second part of the response is header information, ended by a blank line

Example:

- Server: Apache
- Last-Modified: Tue, 20 Mar 2018 15:36:52 GMT
- ETag: "1d6ef3d29e3b6654c7c8e7de310a062c"
- Access-Control-Allow-Origin: *
- Link: <https://www.ucf.edu/alert/wp-json/>; rel="https://api.w.org/"
- Vary: Accept-Encoding
- Content-Encoding: gzip
- X-Apache-Server: SMCAWEB1
- Content-Type: text/xml; charset=UTF-8
- Content-Length: 467
- Accept-Ranges: bytes
- Date: Fri, 06 Apr 2018 15:14:44 GMT
- X-Varnish: 1230425297 1230350034
- Age: 2035
- Via: 1.1 varnish
- Connection: keep-alive
- X-Cache: HIT
- X-Varnish-Server: SMCACHE2

Viewing the response

Live HTTP Headers for Firefox

An example

Status line	HTTP/1.1 200 OK
Response headers	see below

```
Date: Wed, 10 Sep 2003 00:26:53 GMT
Server: Apache/1.3.26 (Unix) PHP/4.2.2 mod_perl/1.27
                                                 mod_ssl/2.8.10 OpenSSL/0.9.6e
Last-Modified: Tue, 09 Sep 2003 19:24:50 GMT
ETag: "1c1ad5-1654-3f5e2902”
Accept-Ranges: bytes
Content-Length: 5716
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Server response headers

Server: NCSA/1.3

- Name and version of the server

Content-Type: *type / subtype*

- Should be of a type and subtype specified by the client's Accept header

Set-Cookie: *name=value*; *options*

- Requests the client to store a cookie with the given name and value

Server – Entity body

The third part of a server response is the entity body

This is often an HTML page

- But it can also be a jpeg, a gif, plain text, etc.-- anything the browser (or other client) is prepared to accept

HTTP Summary

Client (browser) requests documents/files from server using the HTTP protocol

- Browser then displays the documents (HTMLs/images/…)

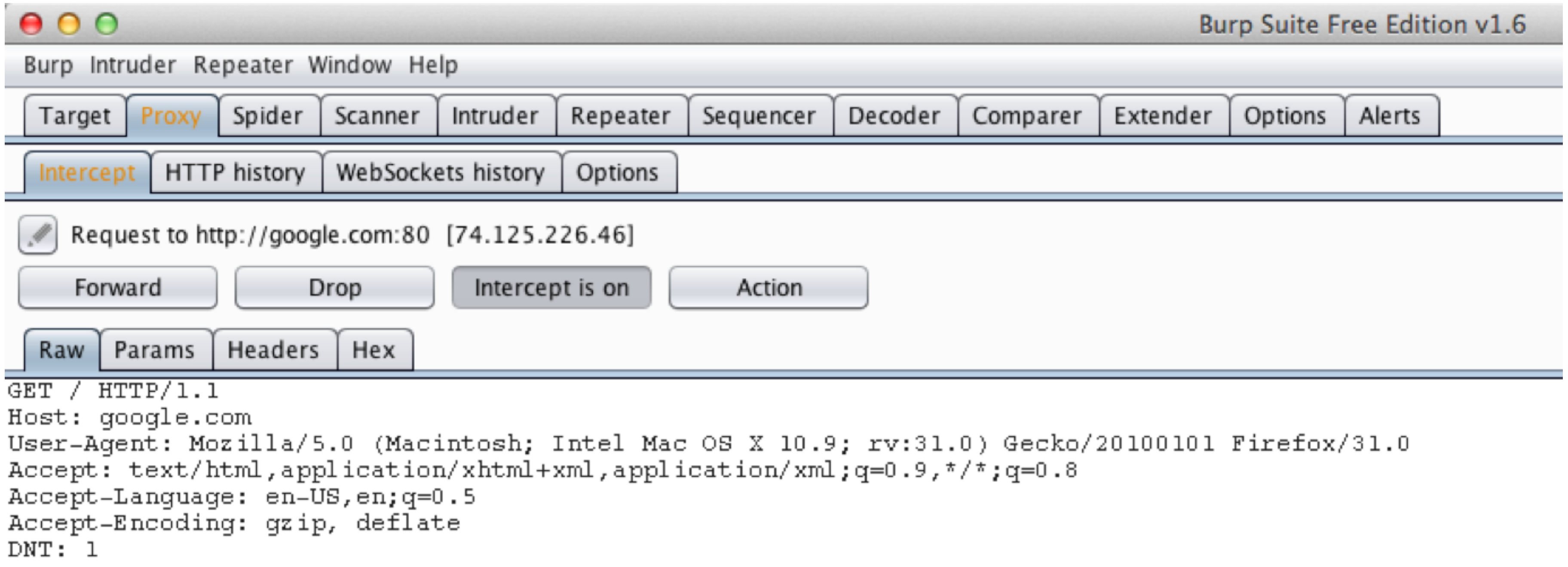
Users of the browser do not see the underlying HTTP message exchanges

- Only see the results

HTTP hacking tools

Burp - intercept, view, and modify HTTP requests and responses.

- First, download and run Burp locally
- Configure a few settings to ensure our browser uses Burp


Burp Suite Free Edition v1.6
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Params Headers Hex

Request to http://google.com:80 [74.125.226.46]

```
GET / HTTP/1.1
Host: google.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:31.0) Gecko/20100101 Firefox/31.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
```

mitmproxy

An open course interactive HTTPS proxy

Can manipulate the ongoing request and response

Supports the man-in-the-middle attack kind of analysis of https sessions

Outline

http

https

Websockets

https

https = http over SSL/TLS

TLS builds a secure tunnel between the client and server and http messages are exchanged through the tunnel

Certificates play a critical role in https

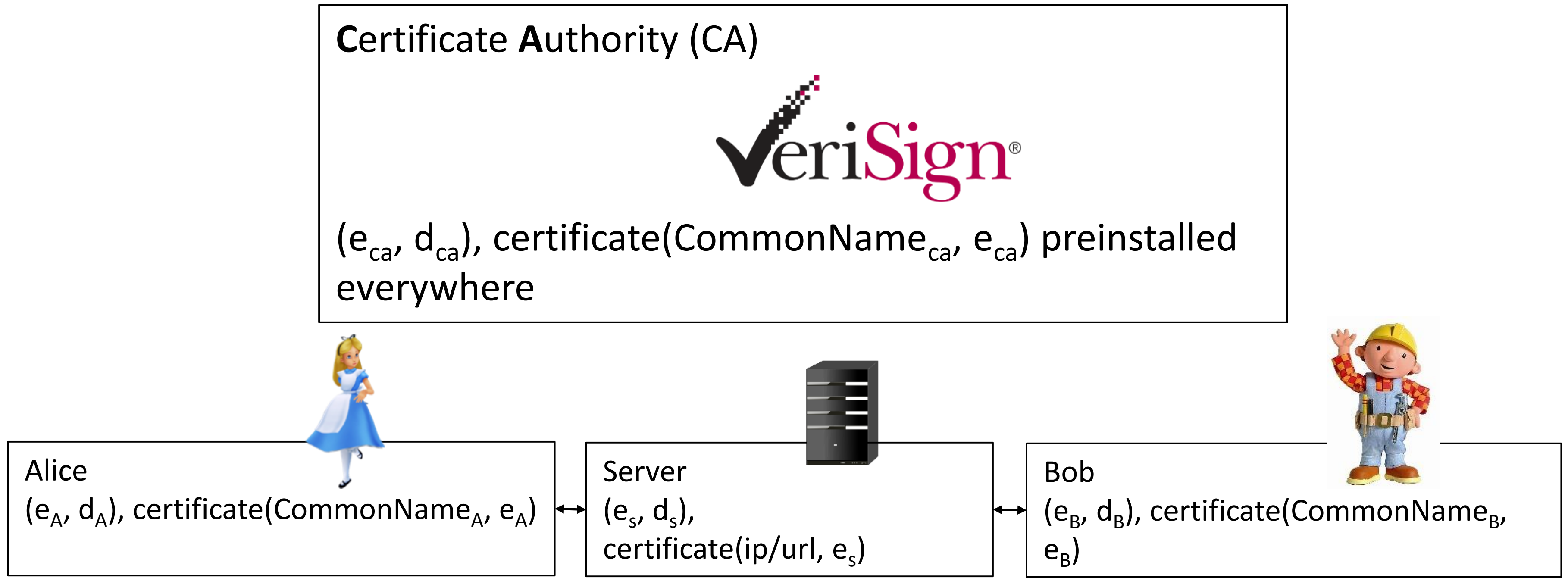
Certificate

X.509 - the most common format for public key certificates

- Very general

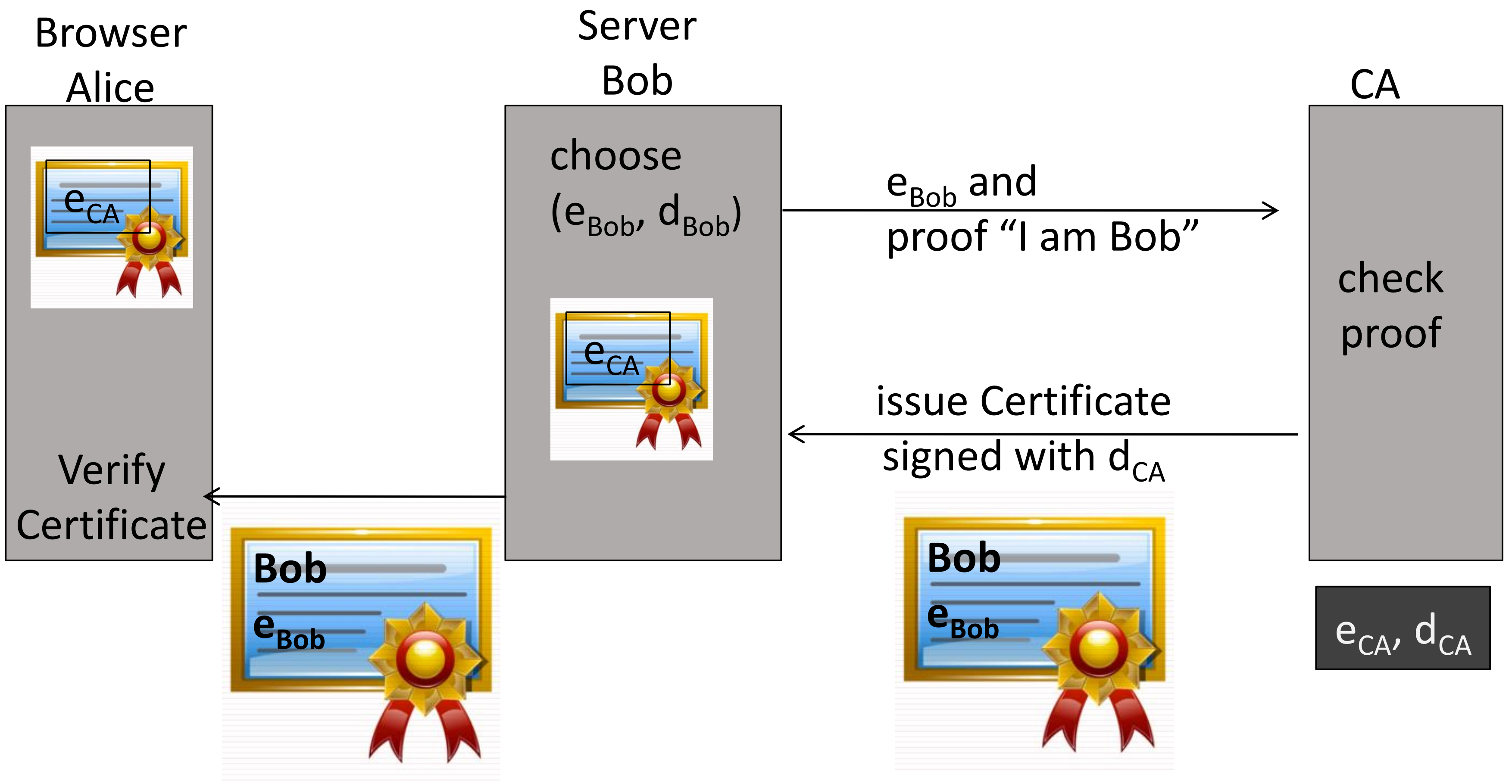
The format is use case oriented

- e.g. Public Key Infrastructure (PKI) X.509 in RFC 5280.

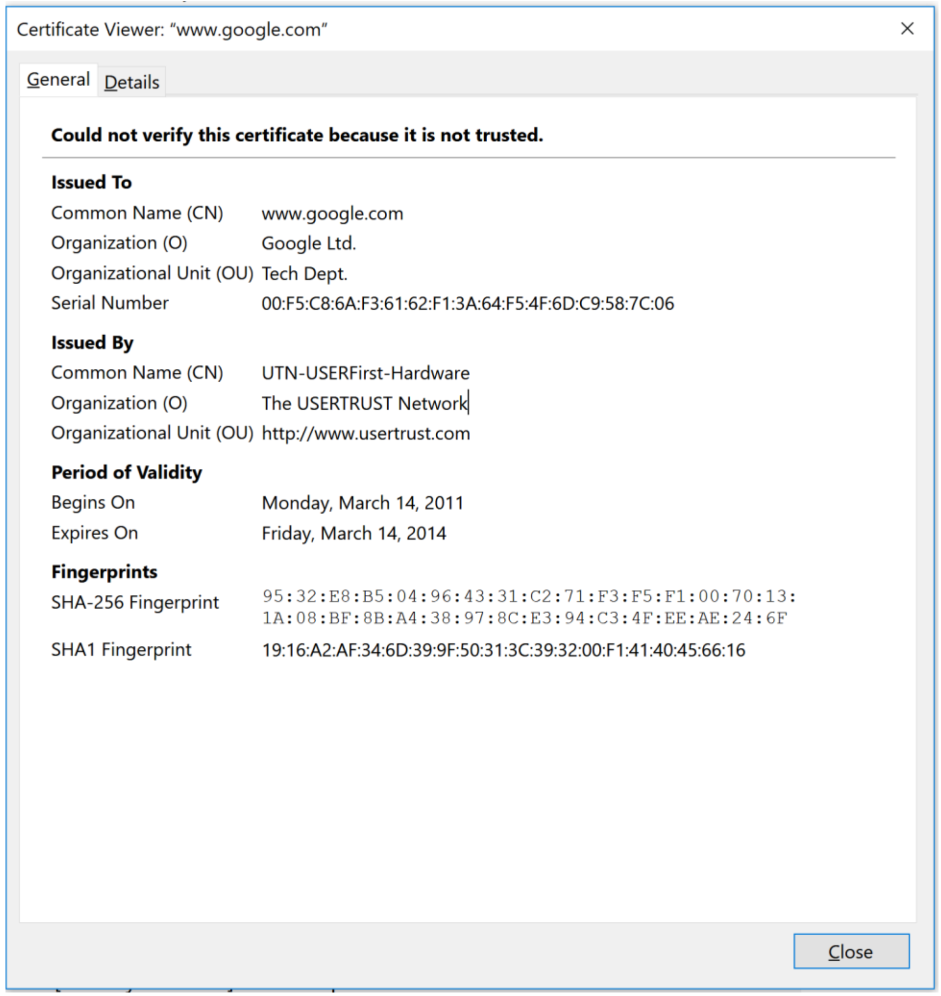

Certificate **A**uthority (CA)
VeriSign®
(e_{ca}, d_{ca}), certificate(CommonName$_{ca}$, e_{ca}) preinstalled everywhere
Alice
(e_A, d_A), certificate(CommonName$_A$, e_A)
Server
(e_s, d_s), certificate(ip/url, e_s)
Bob
(e_B, d_B), certificate(CommonName$_B$, e_B)

Verifying Certificates

How does Alice (browser) obtain e_{Bob} ?



Certificates: example – Firefox/Tools/Options/Privacy&Security/View Certificates



Certificate Viewer: "www.google.com"

General | Details

Could not verify this certificate because it is not trusted.

Issued To

Common Name (CN)	www.google.com
Organization (O)	Google Ltd.
Organizational Unit (OU)	Tech Dept.
Serial Number	00:F5:C8:6A:F3:61:62:F1:3A:64:F5:4F:6D:C9:58:7C:06

Issued By

Common Name (CN)	UTN-USERFirst-Hardware
Organization (O)	The USERTRUST Network
Organizational Unit (OU)	http://www.usertrust.com

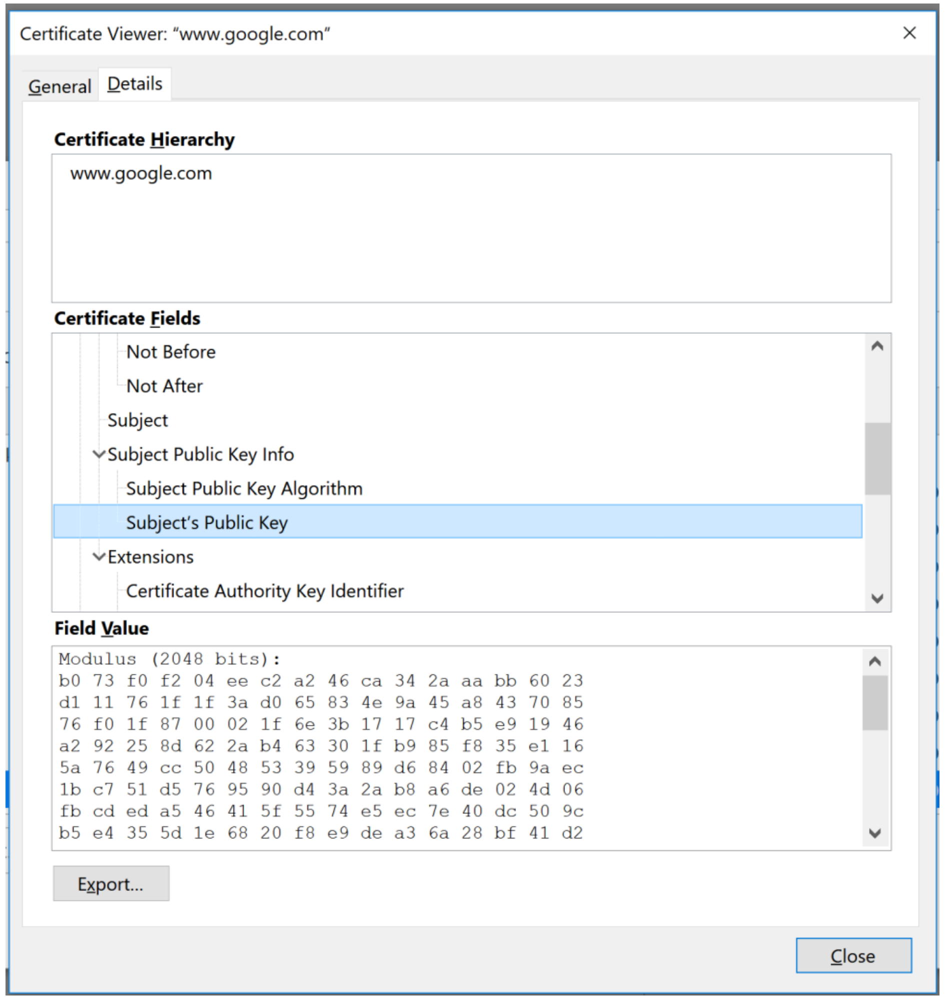
Period of Validity

Begins On	Monday, March 14, 2011
Expires On	Friday, March 14, 2014

Fingerprints

SHA-256 Fingerprint	95:32:E8:B5:04:96:43:31:C2:71:F3:F5:F1:00:70:13: 1A:08:BF:8B:A4:38:97:8C:E3:94:C3:4F:EE:AE:24:6F
SHA1 Fingerprint	19:16:A2:AF:34:6D:39:9F:50:31:3C:39:32:00:F1:41:40:45:66:16

Close



Certificate Viewer: "www.google.com"

General | Details

Certificate Hierarchy

www.google.com

Certificate Fields

- Not Before
- Not After
- Subject
- Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key
- Extensions
  - Certificate Authority Key Identifier

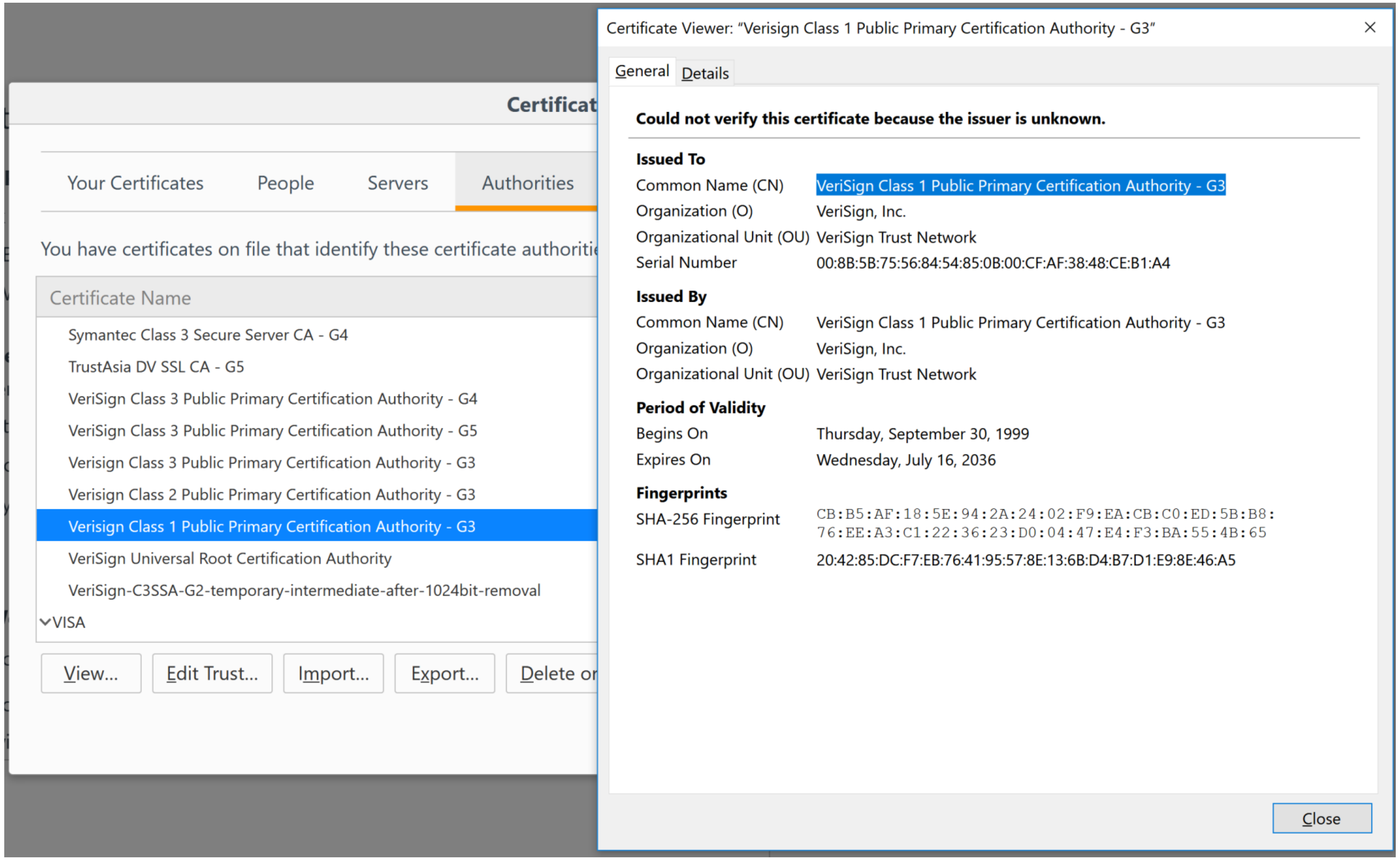
Field Value

```
Modulus (2048 bits):
b0 73 f0 f2 04 ee c2 a2 46 ca 34 2a aa bb 60 23
d1 11 76 1f 1f 3a d0 65 83 4e 9a 45 a8 43 70 85
76 f0 1f 87 00 02 1f 6e 3b 17 17 c4 b5 e9 19 46
a2 92 25 8d 62 2a b4 63 30 1f b9 85 f8 35 e1 16
5a 76 49 cc 50 48 53 39 59 89 d6 84 02 fb 9a ec
1b c7 51 d5 76 95 90 d4 3a 2a b8 a6 de 02 4d 06
fb cd ed a5 46 41 5f 55 74 e5 ec 7e 40 dc 50 9c
b5 e4 35 5d 1e 68 20 f8 e9 de a3 6a 28 bf 41 d2
```

Export...

Close

Certificate Authorities



Certificat

You have certificates on file that identify these certificate authoriti

Certificate Name

Symantec Class 3 Secure Server CA - G4
TrustAsia DV SSL CA - G5
VeriSign Class 3 Public Primary Certification Authority - G4
VeriSign Class 3 Public Primary Certification Authority - G5
Verisign Class 3 Public Primary Certification Authority - G3
Verisign Class 2 Public Primary Certification Authority - G3
Verisign Class 1 Public Primary Certification Authority - G3
VeriSign Universal Root Certification Authority
VeriSign-C3SSA-G2-temporary-intermediate-after-1024bit-removal
VISA

Certificate Viewer: "Verisign Class 1 Public Primary Certification Authority - G3"

Could not verify this certificate because the issuer is unknown.

Issued To

Common Name (CN)	VeriSign Class 1 Public Primary Certification Authority - G3
Organization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network
Serial Number	00:8B:5B:75:56:84:54:85:0B:00:CF:AF:38:48:CE:B1:A4

Issued By

Common Name (CN)	VeriSign Class 1 Public Primary Certification Authority - G3
Organization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network

Period of Validity

Begins On	Thursday, September 30, 1999
Expires On	Wednesday, July 16, 2036

Fingerprints

SHA-256 Fingerprint	CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:65
SHA1 Fingerprint	20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5

Certificates on the web

The owner of a certificate is called a subject

Common name is the identity of the owner

Subject's *CommonName* can be:

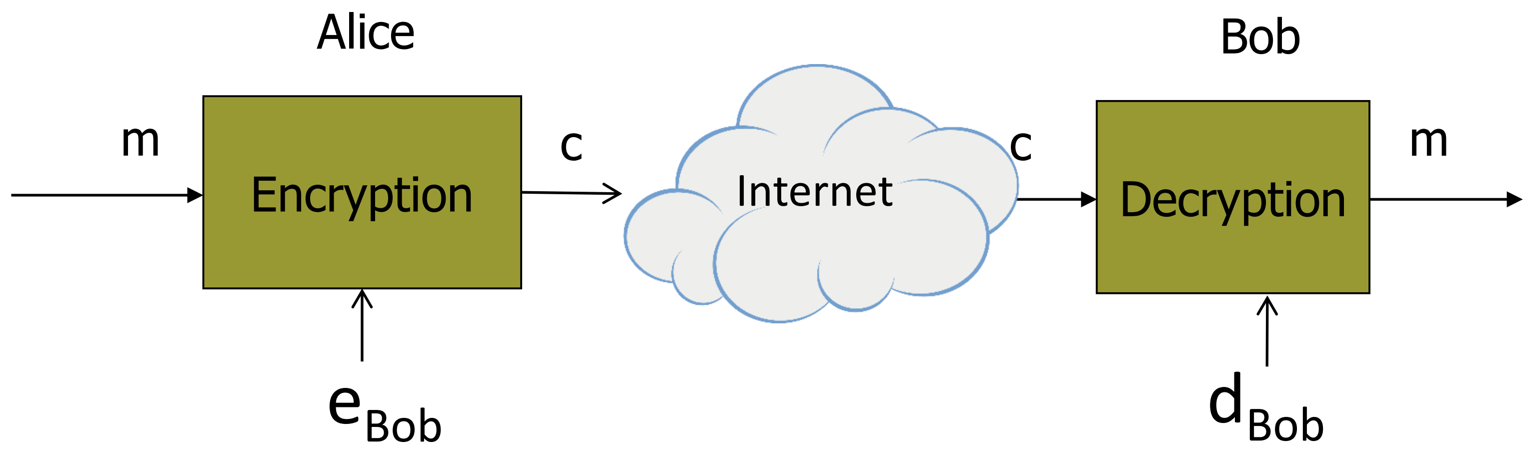
- An explicit name, e.g. ece.ufl, cs.ucf, or
- A name with a wildcard character, e.g. *.ufl.edu, *.ucf.edu   or   cs*.ucf.edu

SSL/TLS Review

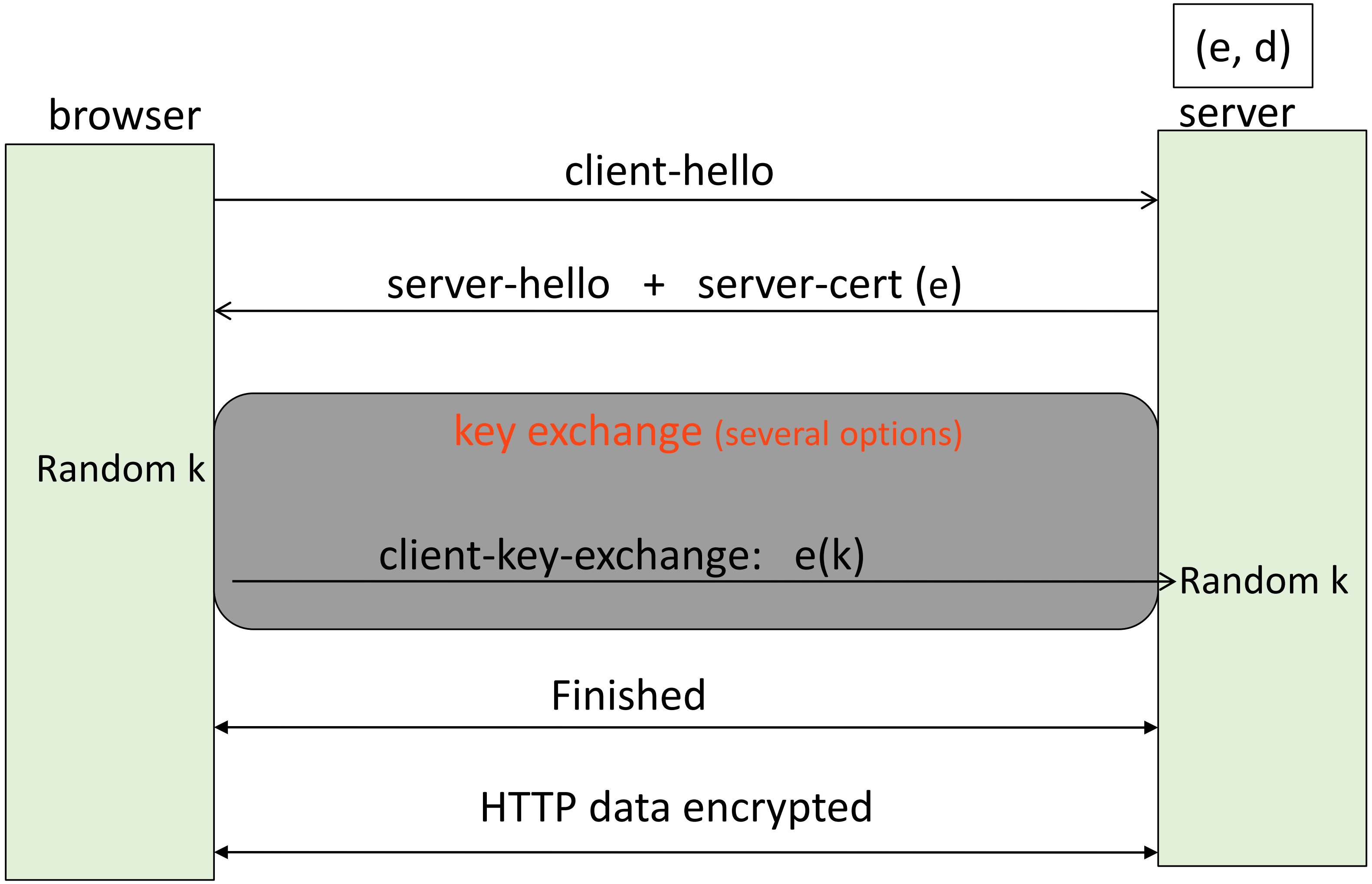
Bob generates (e_{Bob}, d_{Bob})

Alice uses e_{Bob} to encrypts *m* and only Bob can decrypt c to get *m*

Public-key encryption:



Overview of SSL/TLS and HTTPS


(e, d)
browser
server
client-hello
server-hello + server-cert (e)
key exchange (several options)
Random k
client-key-exchange: e(k)
Random k
Finished
HTTP data encrypted

Most common: server authentication only

TLS/SSL server certificate [1]

SSL client performs the certification path validation algorithm :

- The subject of the certificate matches the hostname to which the client is trying to connect.
- The certificate is valid.

The primary hostname (domain name of the website) is listed as the **Common Name** in the **Subject** field of the certificate.

- **Subject Alternative Name (SAN) certificates** or **Unified Communications Certificates (UCC certificates):** a certificate with multiple hostnames (multiple websites) in the field Subject Alternative Name, or in the **Subject Common Name** backward compatibility.
- **wildcard certificate -** hostnames with an asterisk (*)

TLS/SSL client certificate

Authenticate the client connecting to a TLS service

- For access control, for example

Contain an email address or personal name

- Rather than a hostname

Supported by many web browsers

- Most services use passwords and cookies to authenticate users

Can be used to authenticate devices to ensure that only authorized devices can connect to the server

Outline

http

https

Websockets

Issues with HTTP

Half-duplex: request and then response

- Half-duplex - each party can communicate with the other but not simultaneously
- Full-duplex – two parties can communicate with each other simultaneously

Too much overhead for real-time communication

- Request line, header

WebSockets [3]

WebSockets builds a full-duplex connection between a client and server

- Both parties can send data anytime.

How it works

- The client sends a regular HTTP request to the server with an “Upgrade” header
- If the servers agrees to the upgrade request, it responds with an Upgrade header
- Now a WebSocket connection replaces the initial HTTP connection

Where is WebSocket used? [2]

Social feeds

Multiplayer games

Collaborative editing/coding

Clickstream data

Financial tickers

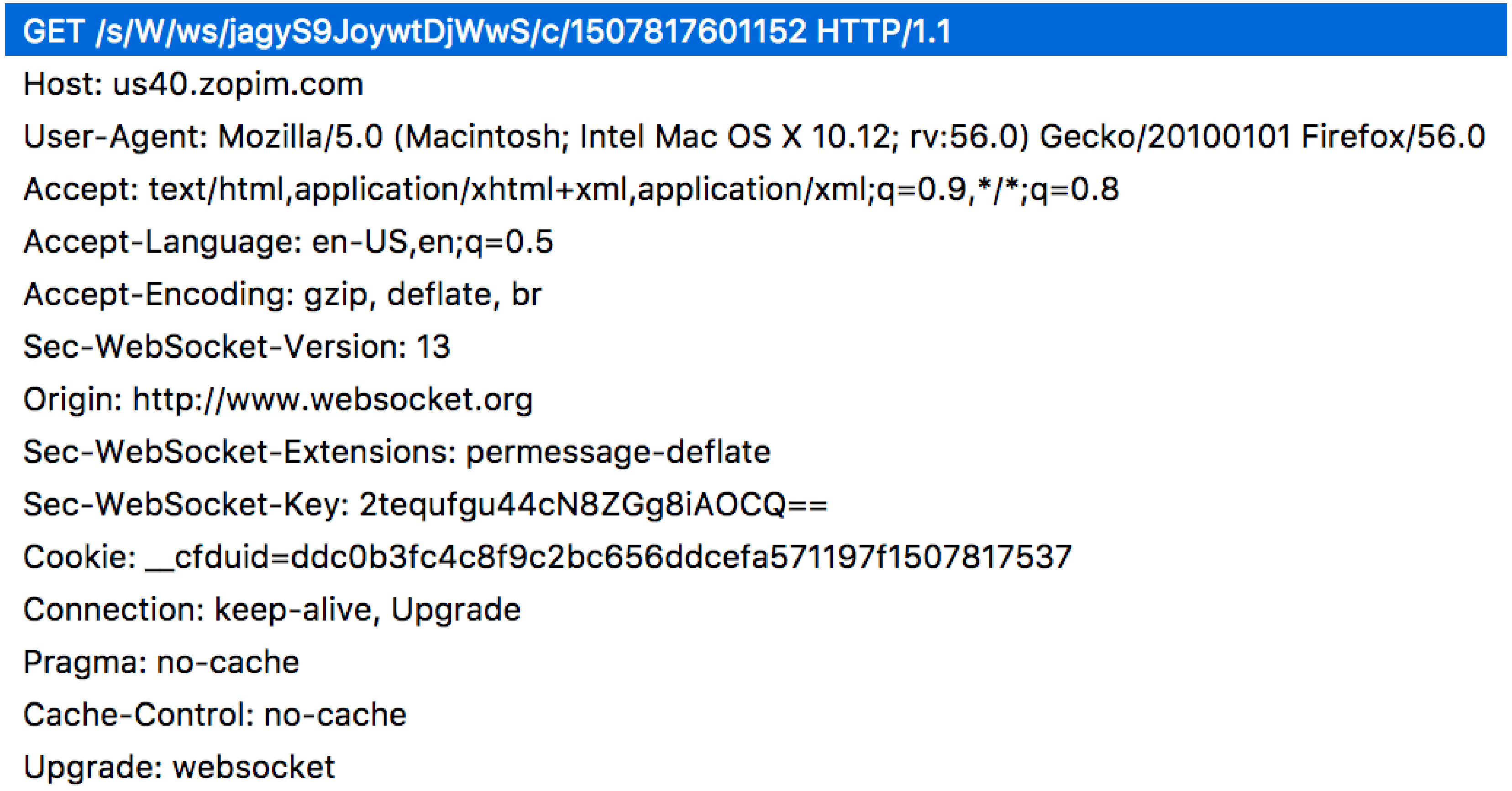
Sports updates

Multimedia chat

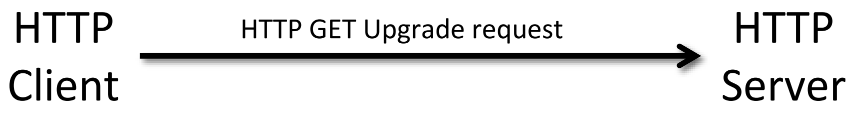
Location-based apps

Online education

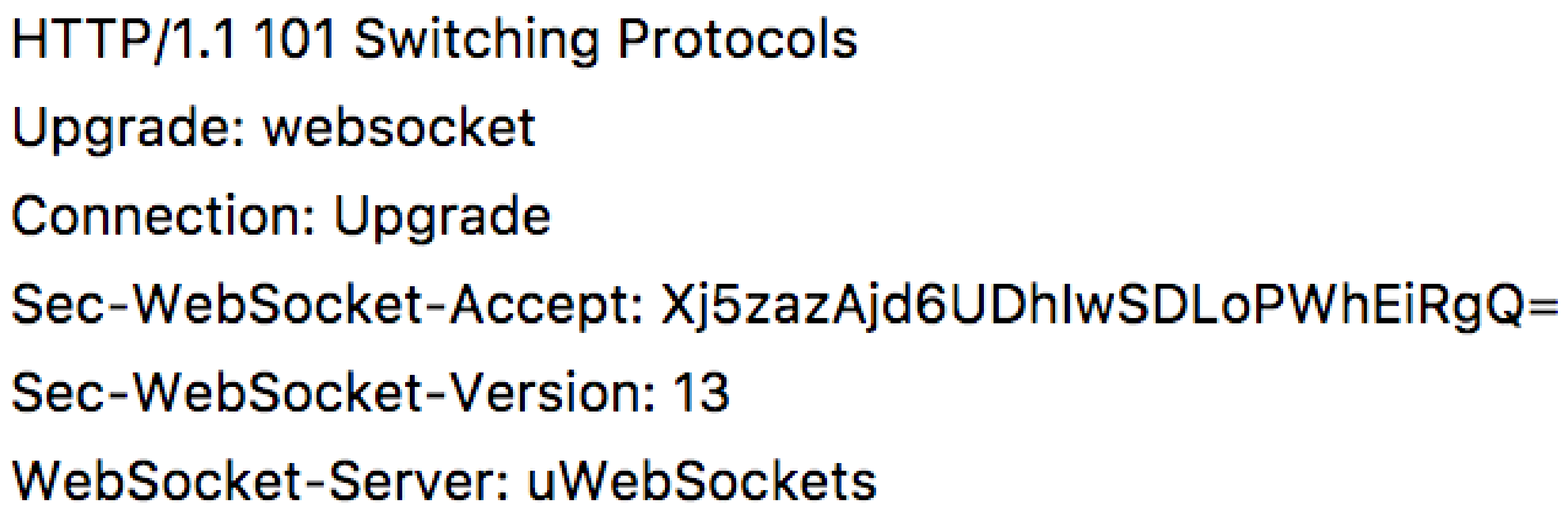
Example of a WebSocket Session



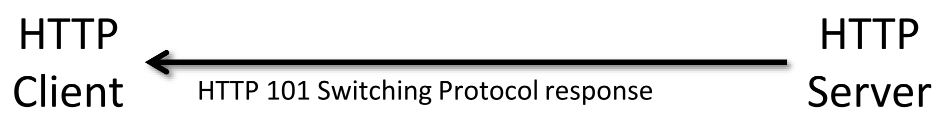
```
GET /s/W/ws/jagyS9JoywtDjWwS/c/1507817601152 HTTP/1.1
Host: us40.zopim.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Sec-WebSocket-Version: 13
Origin: http://www.websocket.org
Sec-WebSocket-Extensions: permessage-deflate
Sec-WebSocket-Key: 2tequfgu44cN8ZGg8iAOCQ==
Cookie: __cfduid=ddc0b3fc4c8f9c2bc656ddcefa571197f1507817537
Connection: keep-alive, Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket
```



WebSocket protocol



```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: Xj5zazAjd6UDhlwSDLoPWhEiRgQ=
Sec-WebSocket-Version: 13
WebSocket-Server: uWebSockets
```



References

[1] Public key certificate, Wikipedia, 2017

[2] Jonathan Freeman, 9 killer uses for WebSockets, InfoWorld, Nov 14, 2013

[3] Matt West, An Introduction to WebSockets, 2018