CIS 371 Web Application Programming

HTTP

GRAND VALLEY STATE UNIVERSITY®

Lecturer: Dr. Yong Zhuang

Based on the original version by Professor Hans Dulimarta.

HTTP

- HyperText Transfer Protocol
- Invented by Tim Berners Lee @ CERN
- A protocol for delivering resources over the web
- TCP/IP connections, default (server) port 80
- HTTP client & HTTP server

Other network Transfer Protocols

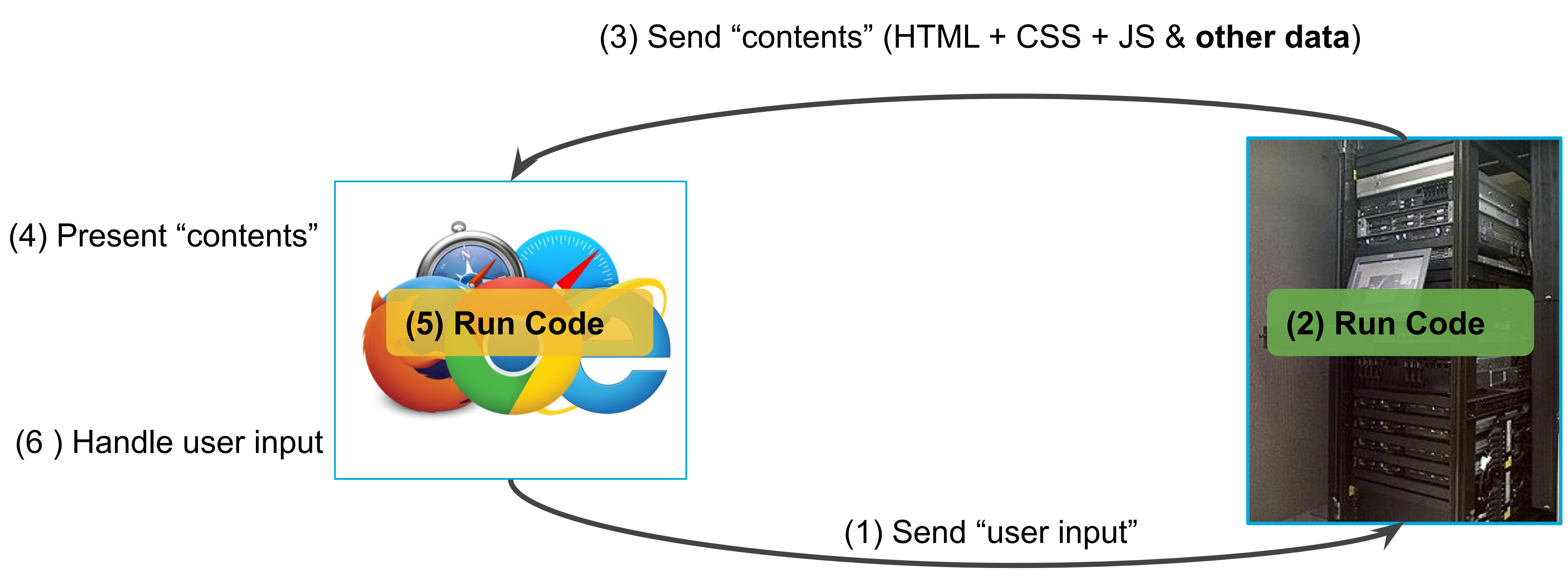
- FTP: File Transfer Protocol
- FTPS: Secure FTP
- SMTP: Simple Message Transfer Protocol
- NTP: Network Time Protocol

Why learn the details of HTTP?

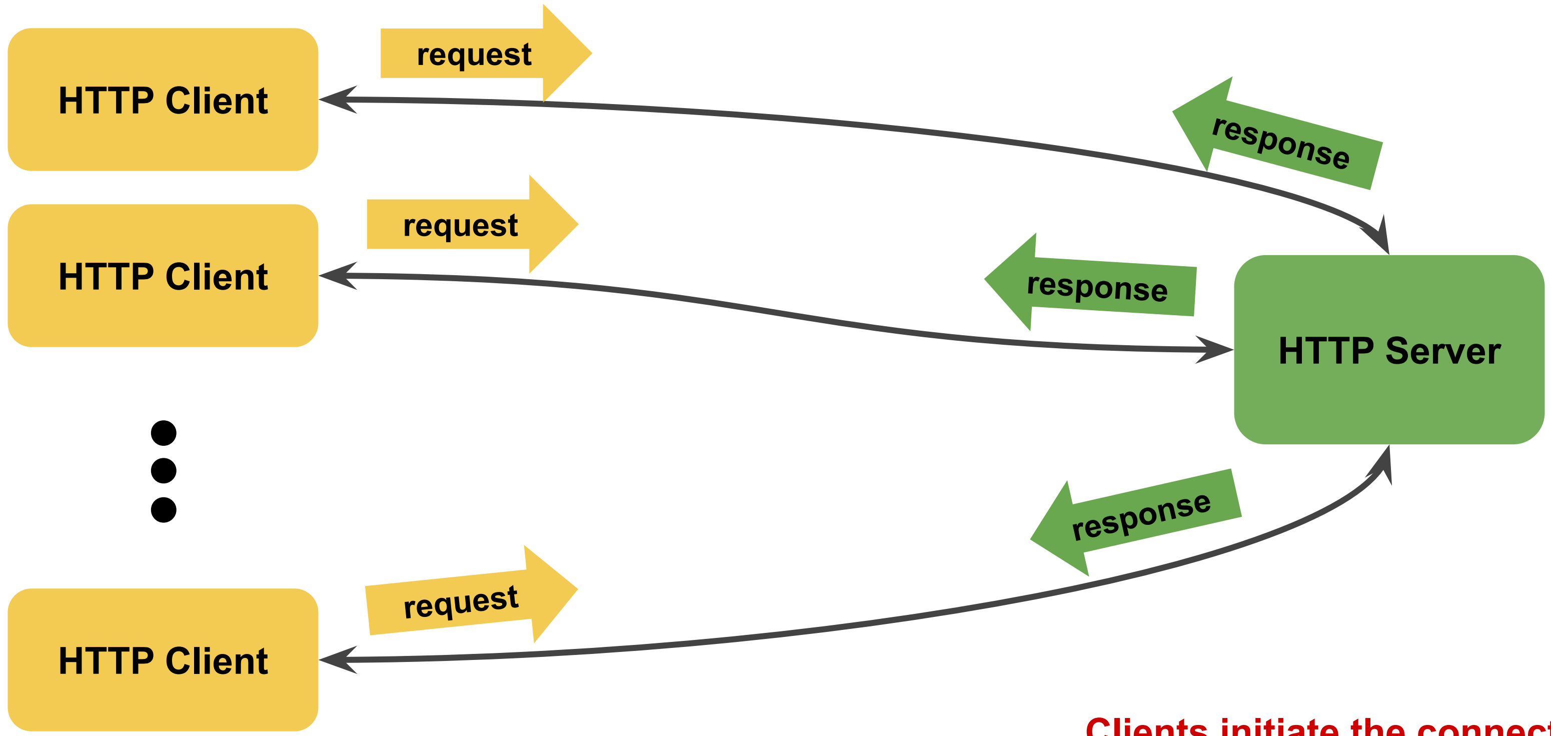
Why learn the details of HTTP?

HTTP requests from your program

Web Client/Server Architecture

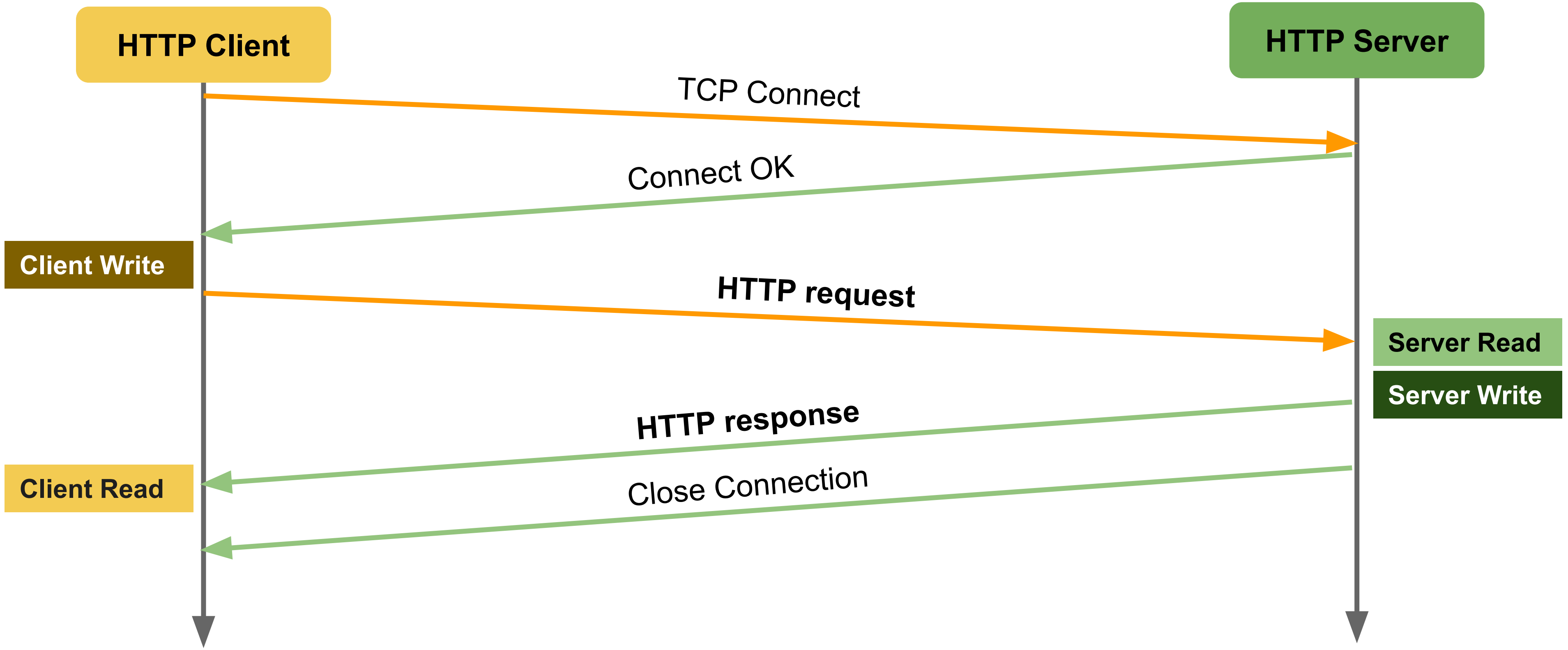


HTTP Communication Model

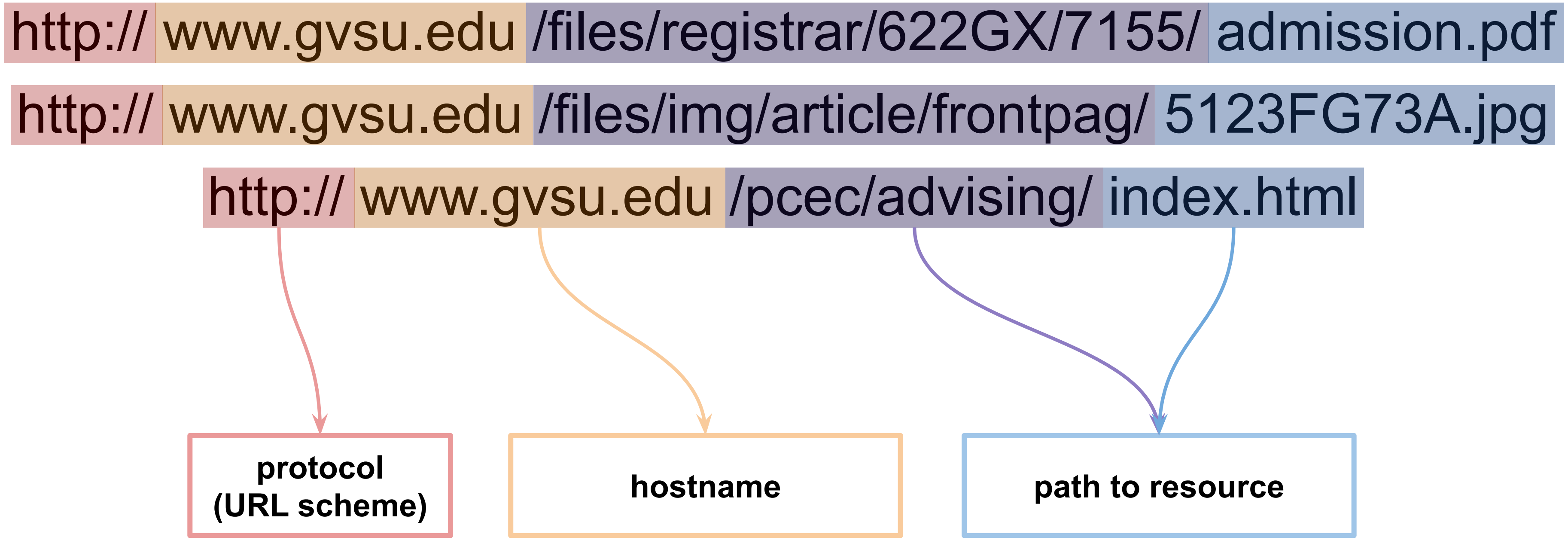


Clients initiate the connection!!!

Transaction Timeline (TCP Sockets)



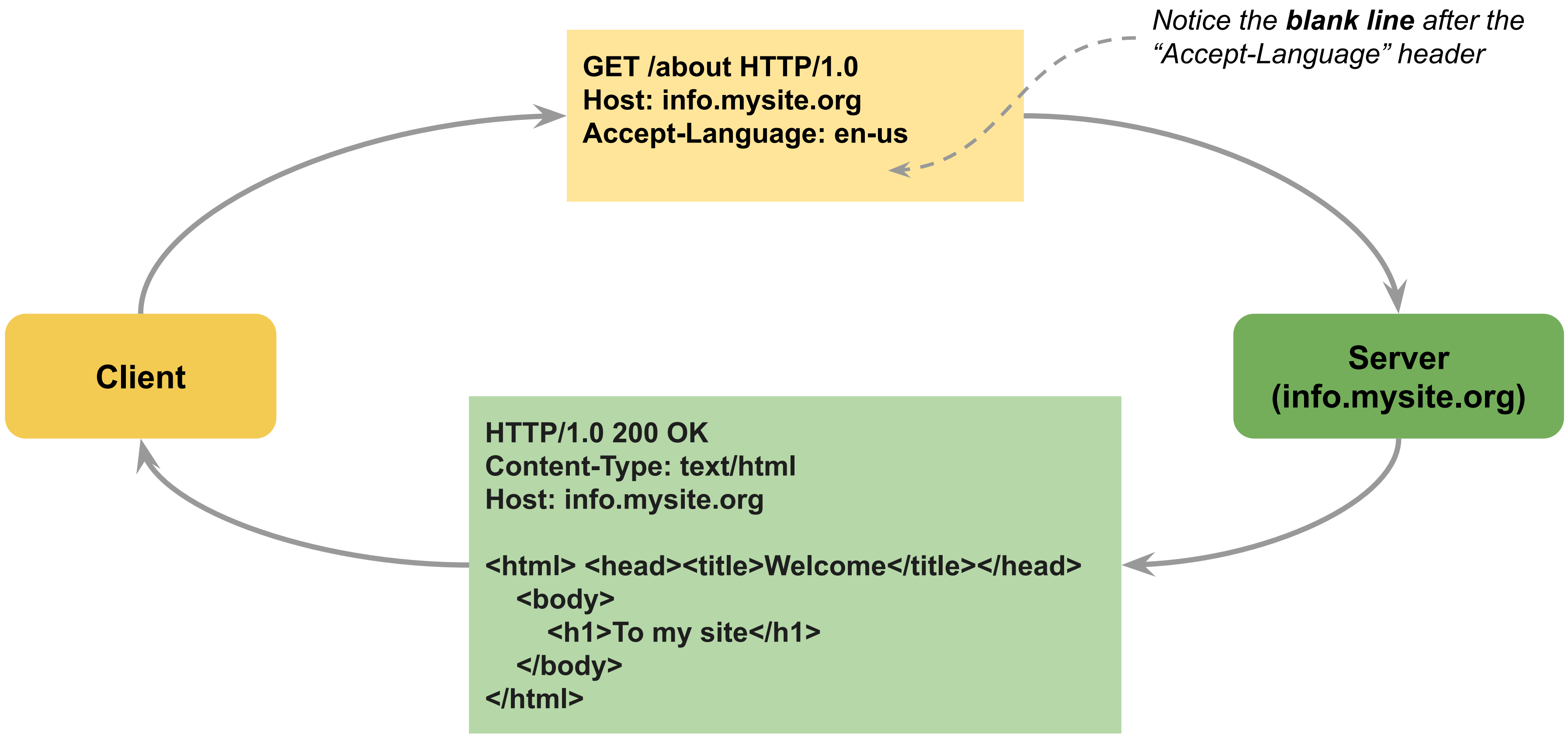
HTTP URL: Uniform Resource Locator


http:// www.gvsu.edu /files/registrar/622GX/7155/ admission.pdf
http:// www.gvsu.edu /files/img/article/frontpag/ 5123FG73A.jpg
http:// www.gvsu.edu /pcec/advising/ index.html
protocol (URL scheme)
hostname
path to resource

HTTP Messages: Request & Response

Demo: URL & Web Dev Tools

http://info.mysite.org/about/

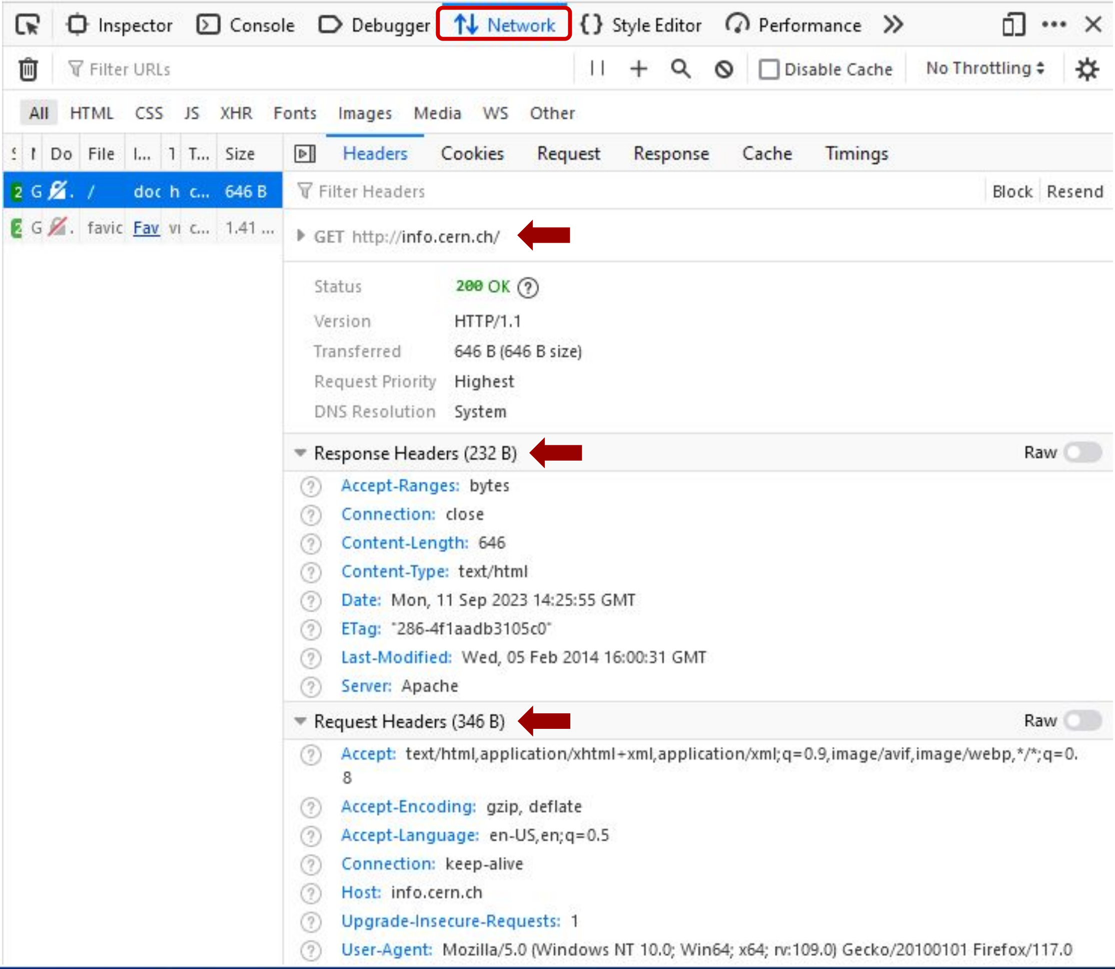

Client
Server
(info.mysite.org)
*Notice the **blank line** after the “Accept-Language” header*

```
GET /about HTTP/1.0
Host: info.mysite.org
Accept-Language: en-us

```

```
HTTP/1.0 200 OK
Content-Type: text/html
Host: info.mysite.org

<html> <head><title>Welcome</title></head>
    <body>
        <h1>To my site</h1>
    </body>
</html>
```

Web Browser DevTools (Network Tab)

http://info.cern.ch



Do	File	I...	T...	Size
/	doc	h	c...	646 B
favic	Fav	vi	c...	1.41 ...

GET http://info.cern.ch/

Status	200 OK
Version	HTTP/1.1
Transferred	646 B (646 B size)
Request Priority	Highest
DNS Resolution	System

Response Headers (232 B)

```
Accept-Ranges: bytes
Connection: close
Content-Length: 646
Content-Type: text/html
Date: Mon, 11 Sep 2023 14:25:55 GMT
ETag: "286-4f1aadb3105c0"
Last-Modified: Wed, 05 Feb 2014 16:00:31 GMT
Server: Apache
```

Request Headers (346 B)

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.
8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Host: info.cern.ch
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
```

Fetch the content via the command line

```
curl --verbose http://info.cern.ch
```

(On Linux/OSX/Windows 10 WSL)

Fetch the content via the command line

```
iwr http://info.cern.ch -UseBasicParsing
```

(On Windows PowerShell)

HTTP Request/Response

line		
1	Request/Response line	**required**
2 *3* *…* *N*	Header1: value1 Header2: value2 … more header lines here … HeaderN: valueN	Header lines (optional)
N+1	**One blank line**	**required**
N+2 *N+3* *N+4* *…*	message body (plain text or binary)	Message body (optional) ● Data for POST requests, examples ○ Encrypted userid/password ○ Encrypted credit card details ○ Content of uploaded file(s) ○ etc. ● Returned contents of server responses ○ HTML doc ○ Image data ○ etc.

HTTP headers of interest to web developers

Header	Description	Example
Accept	Inform server media-type to respond	Accept: image/jpg
Accept-Language	Inform the server which languages the client is able to understand	Accept-Language: en-US; en-UK
Content-Type	Media type of the returned content	Content-Type: plain/text
Content-Language	The languages of the content	Content-Language: en-US
Date	Date and time of the message	Date: Mon, 21 Aug 2017 18:14:36 GMT
ETag	Identifier used by caching algorithms	ETag: “"8a9-291e721905000”
Host	Specify the domain name of the intended server (mainly for Virtual Hosting)	Host: www.personal.me:5555

HTTP 1.0 Commands (Request Methods)

- GET
- POST
- HEAD

More-frequently used

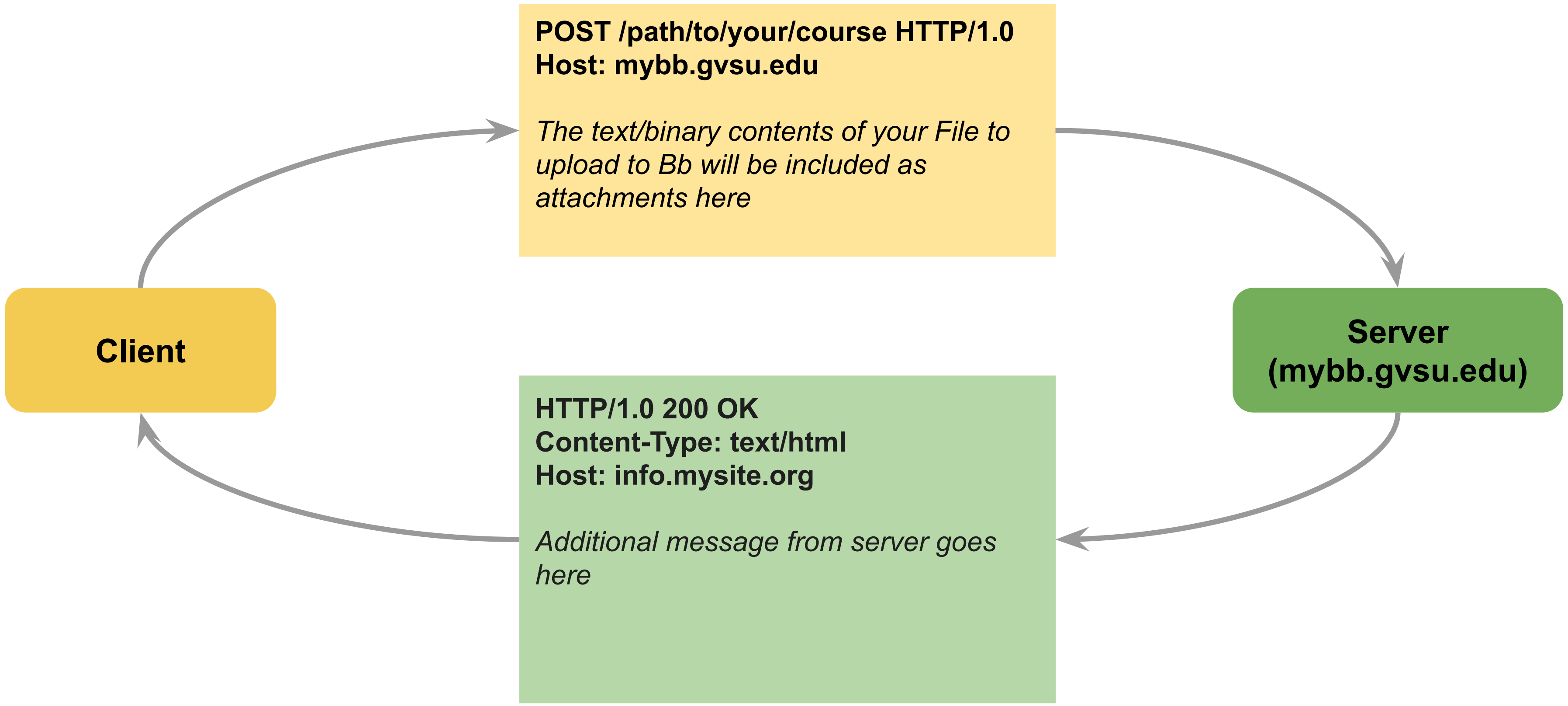
(like GET but the server responds only with header, no data)

- PUT
- DELETE
- OPTIONS

Less-frequently used

Operation	HTTP Request
Create	POST
Read	GET
Update	PUT
Delete	DELETE

POST: upload file to Bb



POST /path/to/your/course HTTP/1.0
Host: mybb.gvsu.edu

The text/binary contents of your File to upload to Bb will be included as attachments here

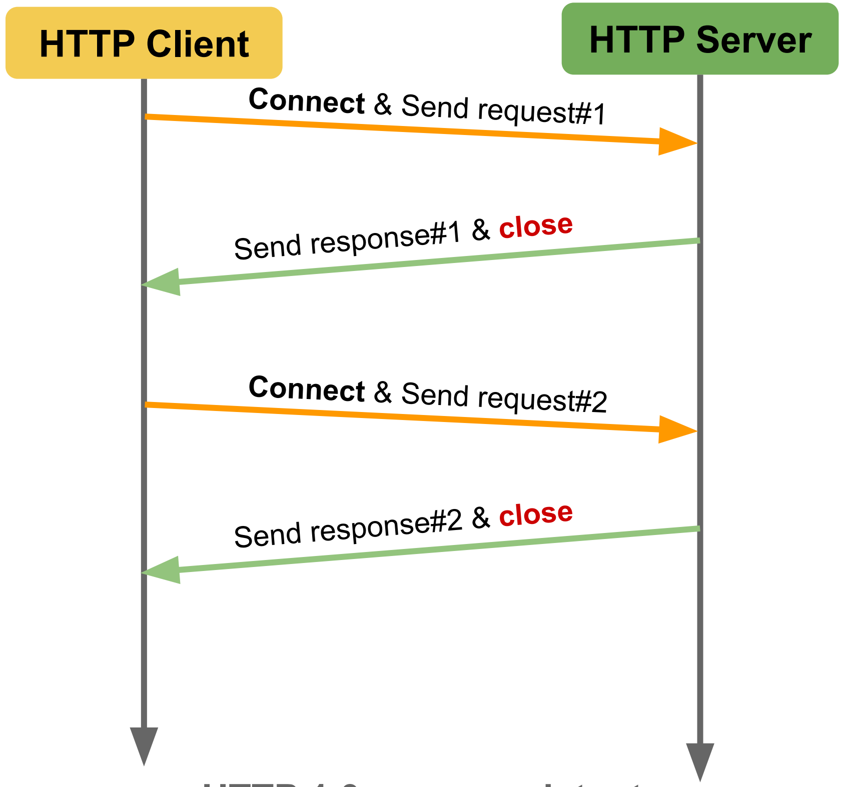
HTTP/1.0 200 OK
Content-Type: text/html
Host: info.mysite.org

Additional message from server goes here

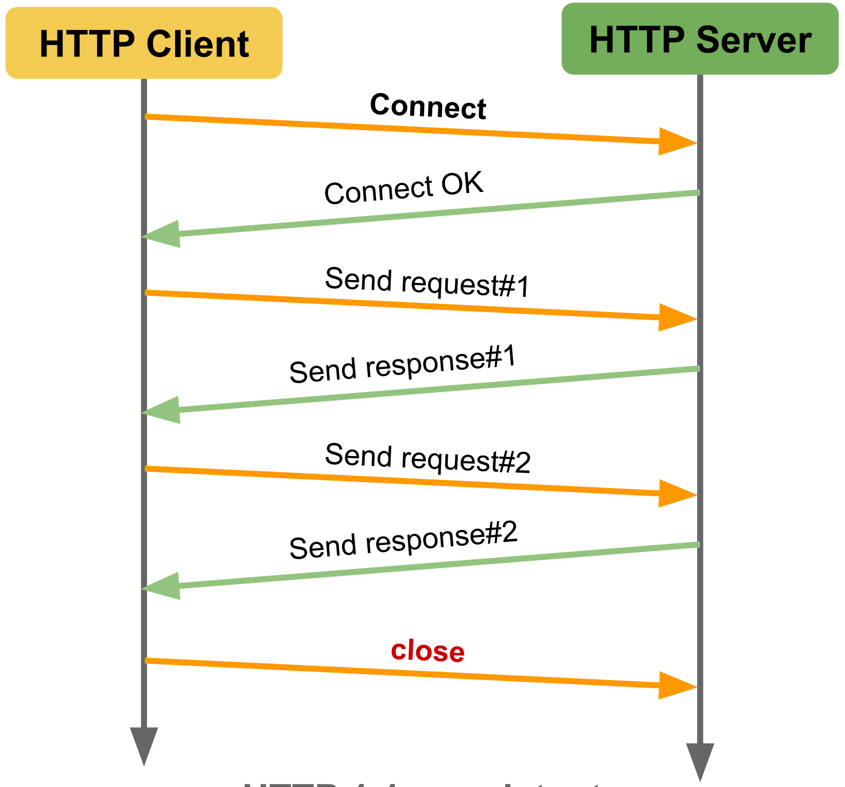
HTTP Status Code

Status Code	Description
1xx	Informational messages
2xx	Success messages
3xx	Redirect message
4xx	Error on the client’s behalf
5xx	Error on the server’s behalf

HTTP Connections: Persistence



HTTP 1.0: non-persistent



HTTP 1.1: persistent

HTTP 1.0 vs. HTTP 1.1

HTTP 1.0

- One request per connection (non-persistent)
- Cache control is **timestamp based** with one-second resolution (inaccurate)
- Client cannot request a portion of a resource
- Responses are delivered in one big chunk

HTTP 1.1

- N requests per connection (persistent)
- Response can be delivered in chunk
- Cache control is **content based**, responses include entity tag (Etag), similar to hash value
- Clients can request **partial content**
    - “Range:” header line in HTTP request
- Responses may be delivered in many small chunks

HTTP 1.1 vs. HTTP 2

HTTP 1.1

- HTTP messages encoded in text format
- Require multiple connections to achieve concurrency
- Uncompressed response headers
- No resource prioritization

HTTP 2

- HTTP messages encoded in binary format
  - Message = request or response
- Multiple concurrent channels on a single connection
- Compressed response headers
- Resource prioritization (important requests complete more quickly)

Secure HTTP ➡ HTTPS

HTTPS

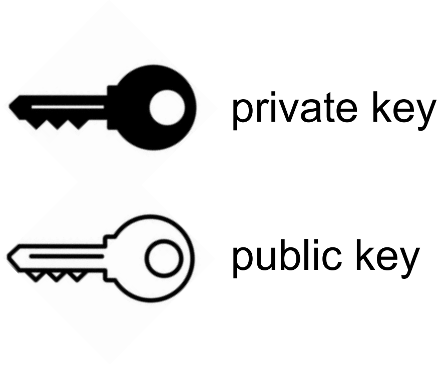
- HTTP Secure
  - HTTP over TLS (Transport Layer Security)
  - HTTP over SSL (Secure Socket Layer)

- PKI (Public Key Infrastructure)

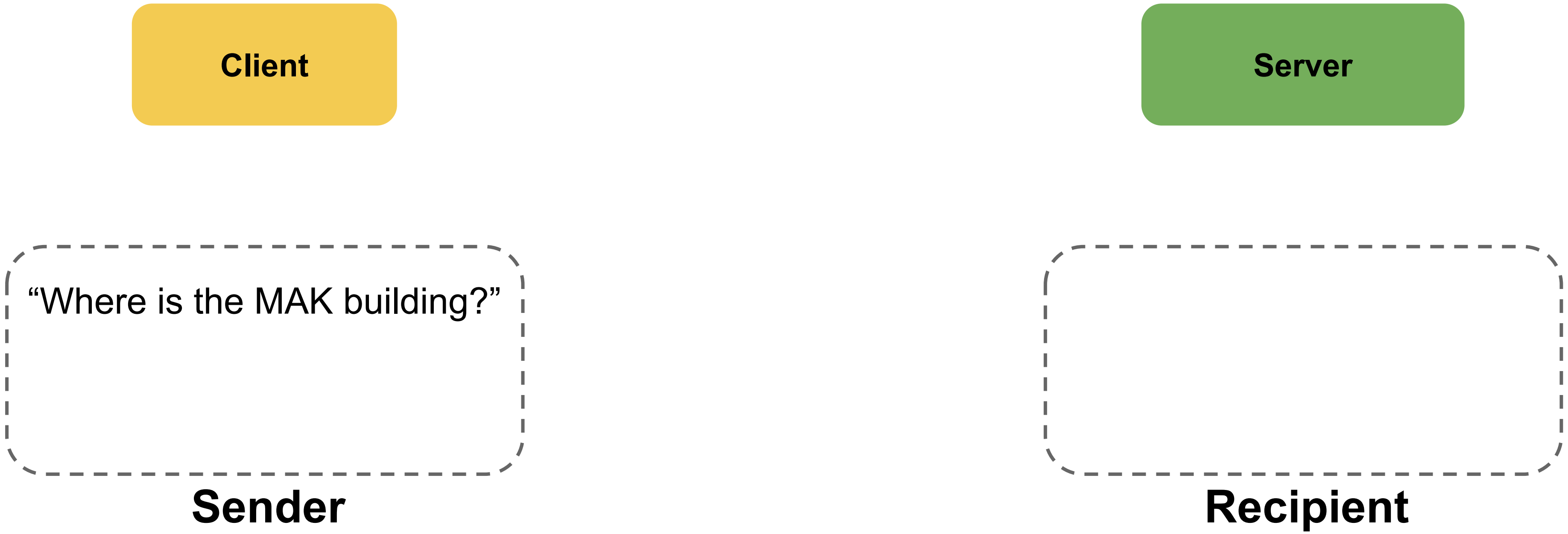
HTTPS

- HTTP Secure
  - HTTP over TLS (Transport Layer Security)
  - HTTP over SSL (Secure Socket Layer)

- PKI (Public Key Infrastructure)



Encrypted Message (with public+private key pair)



Encrypted Message (with public+private key pair)

Client

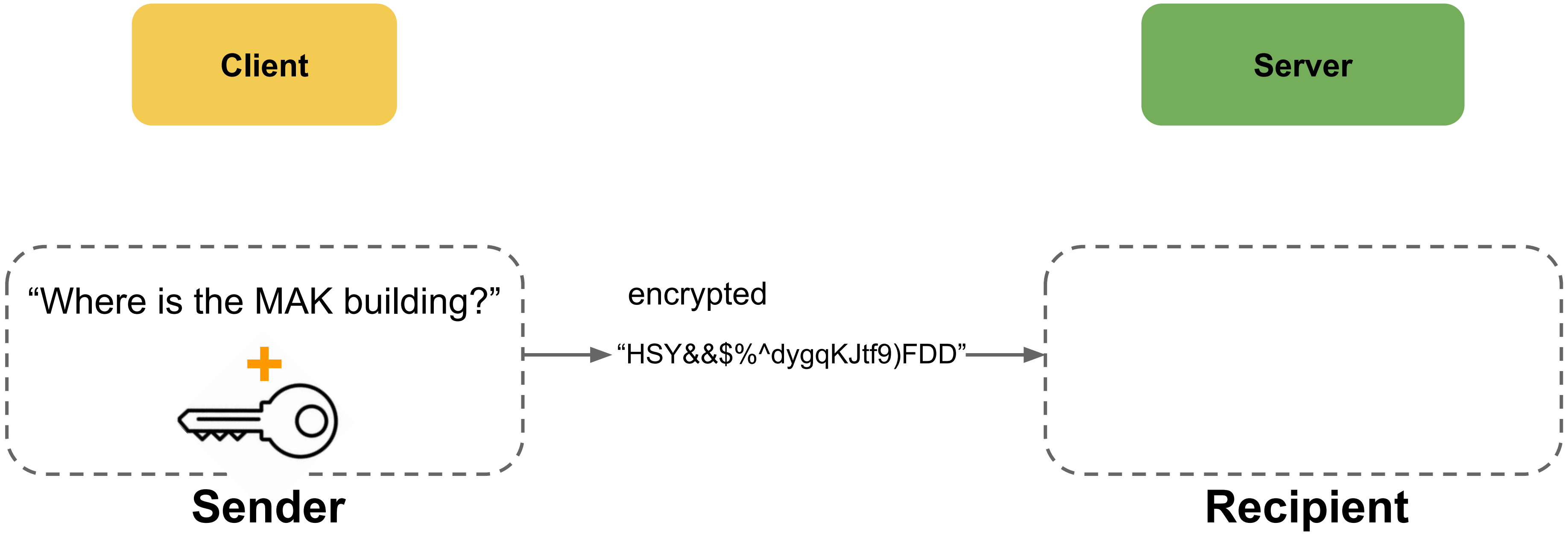
Server

“Where is the MAK building?”

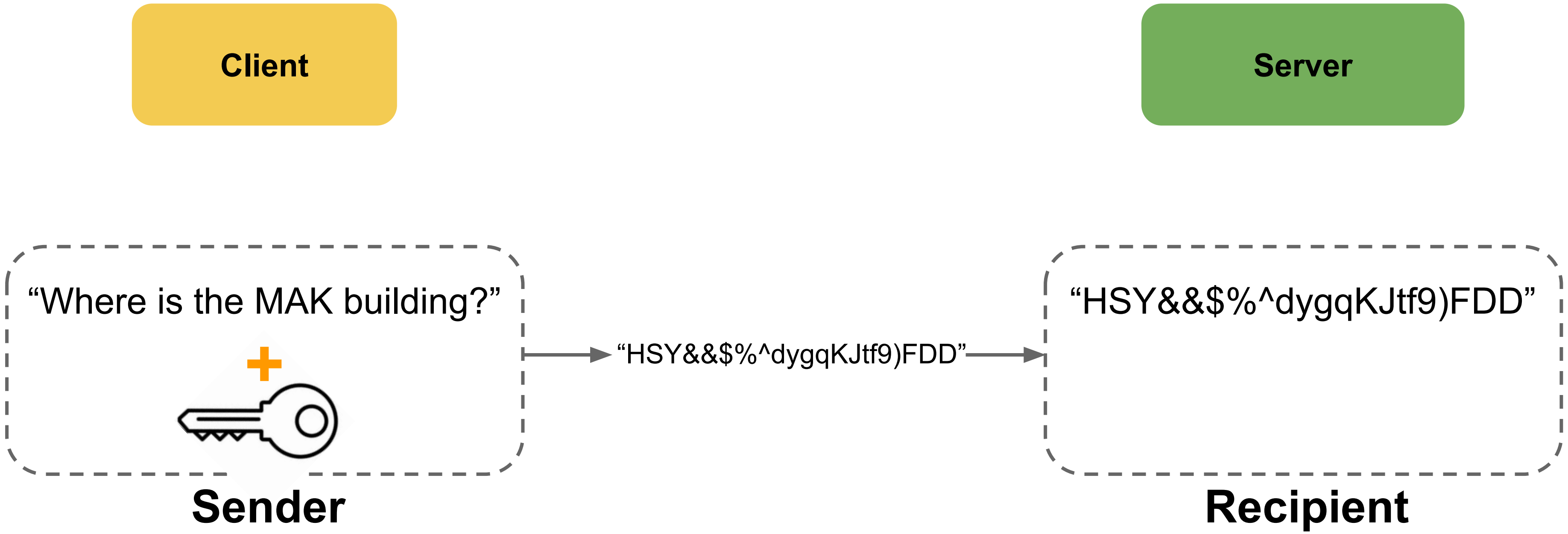
Sender

Recipient

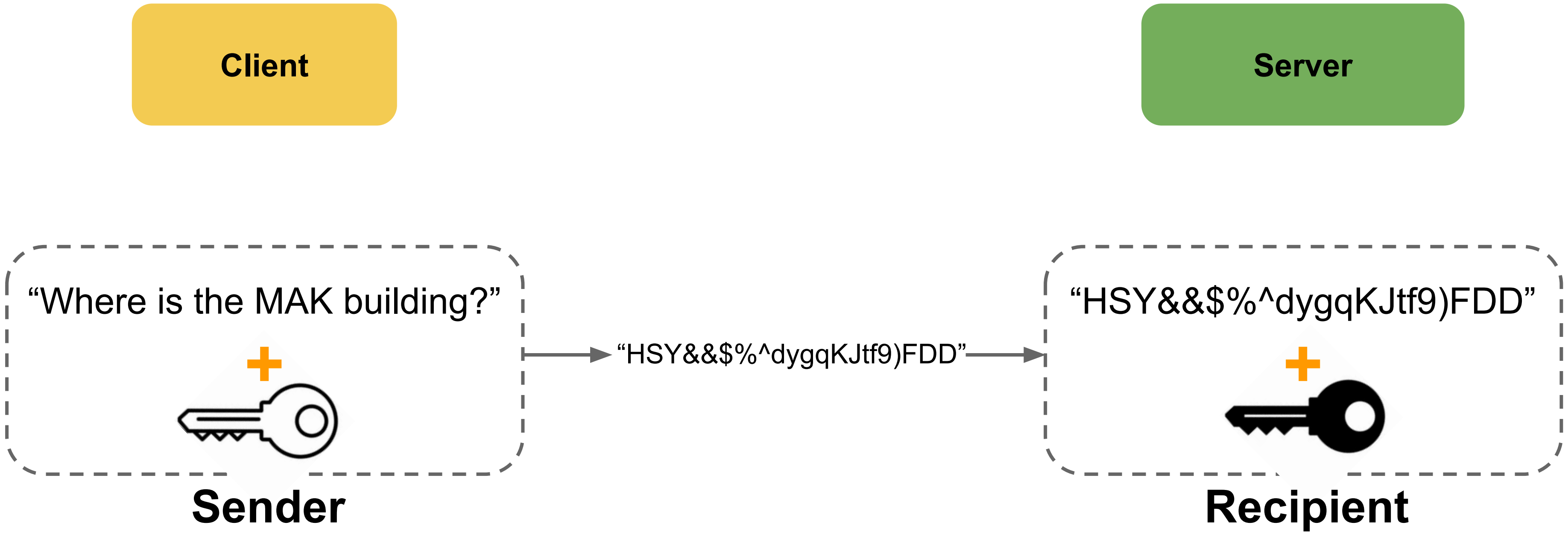
Encrypted Message (with public+private key pair)



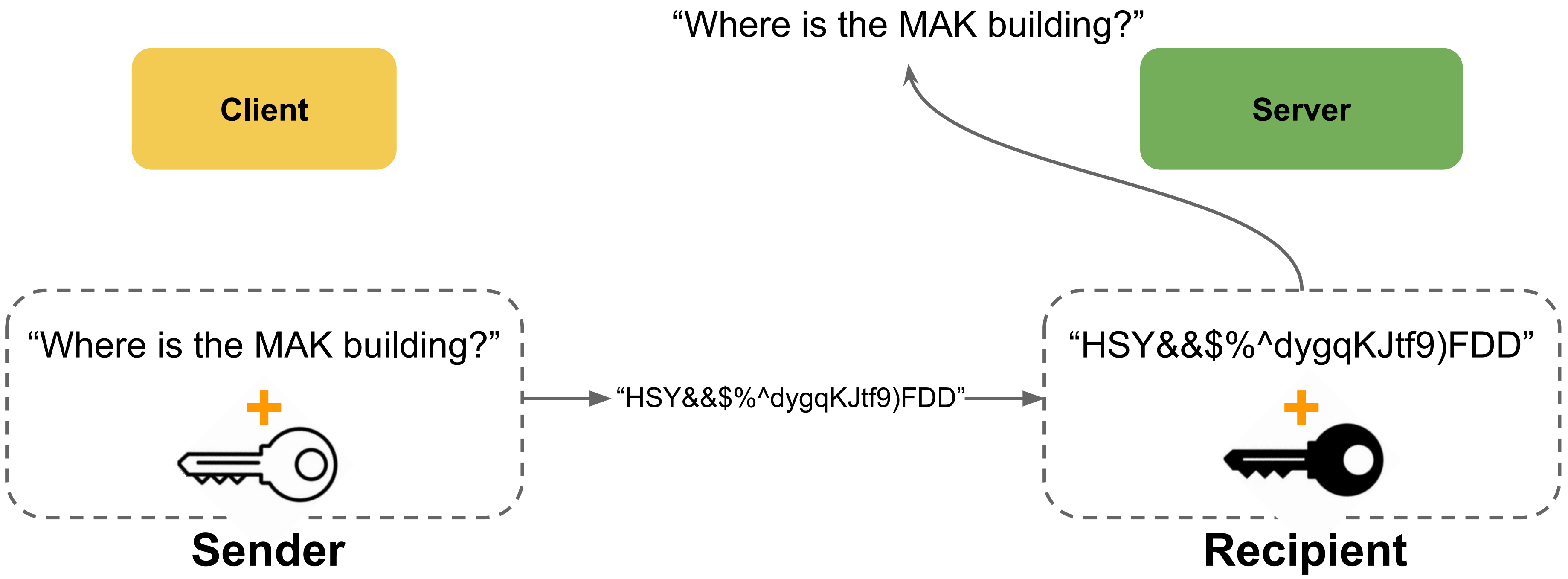
Encrypted Message (with public+private key pair)



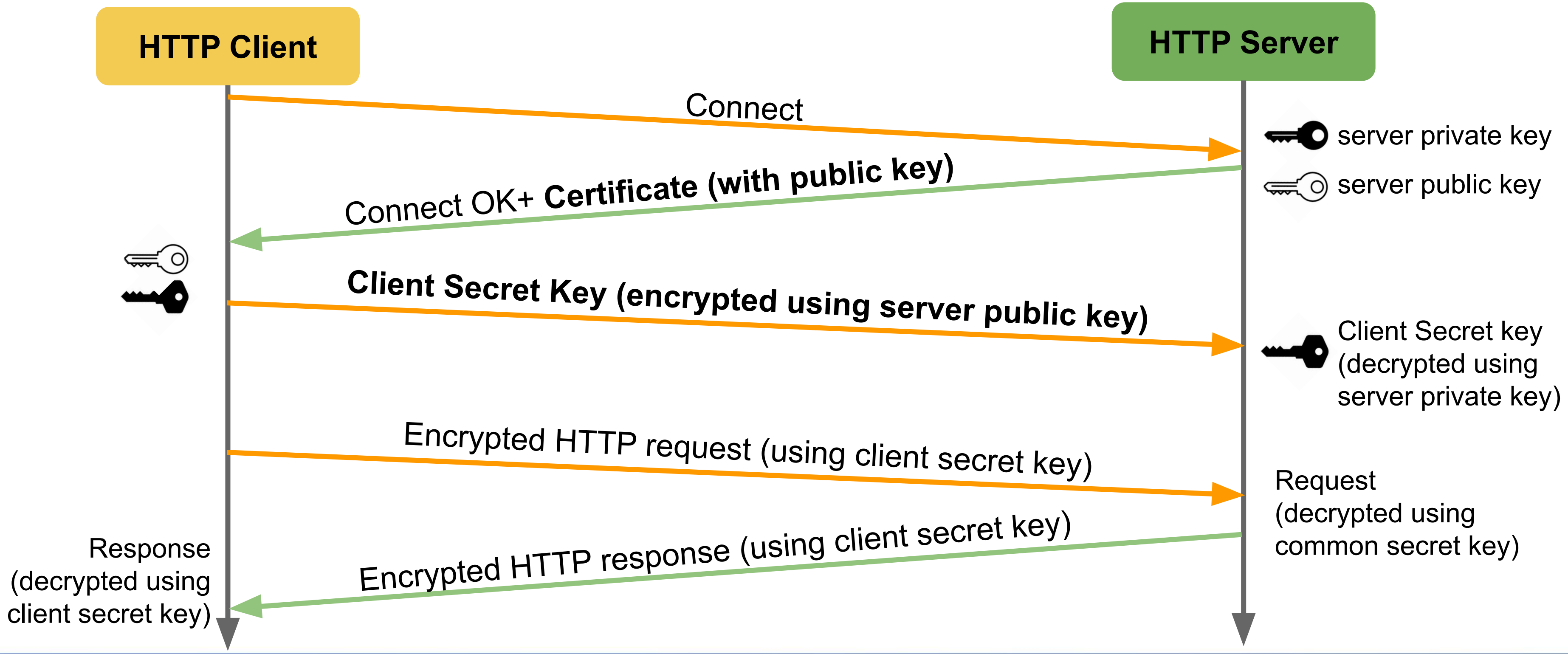
Encrypted Message (with public+private key pair)



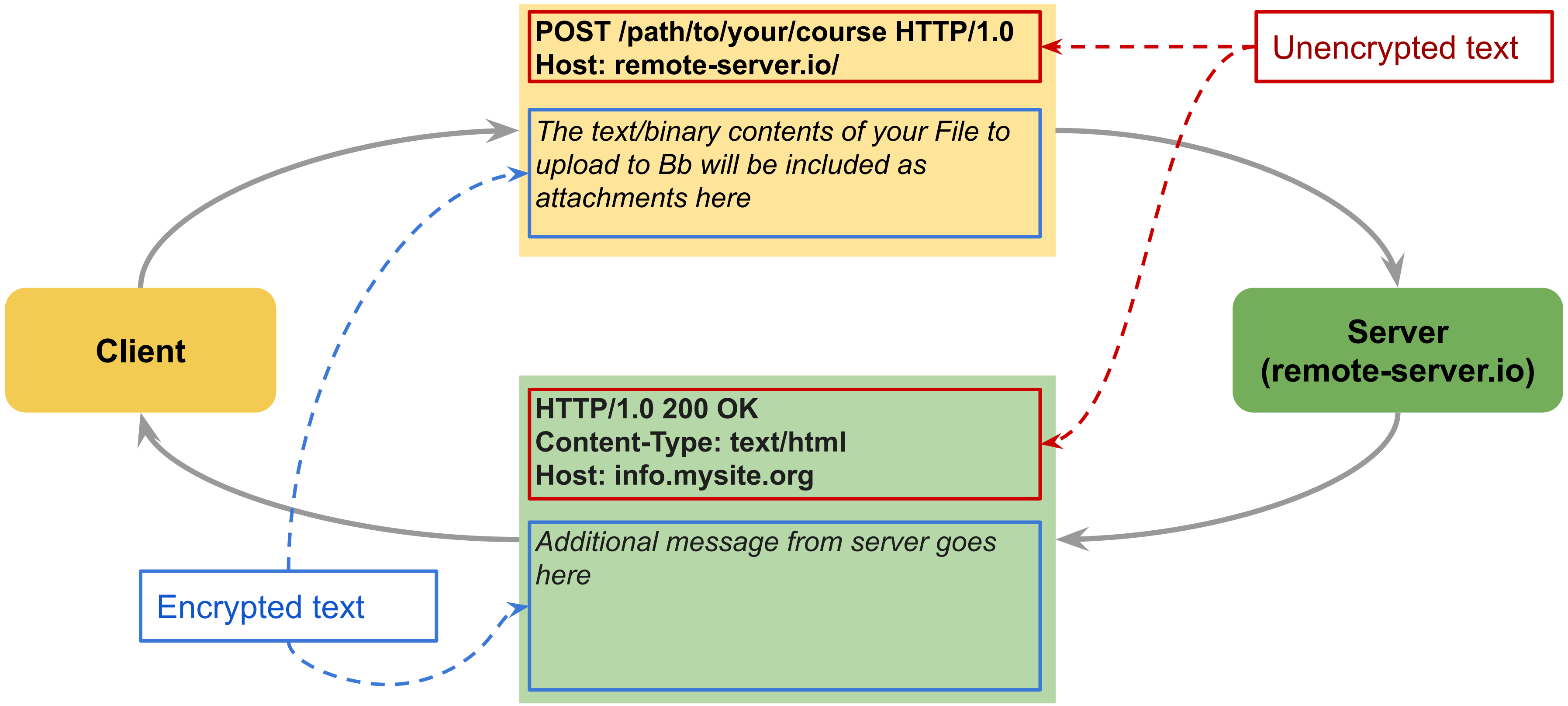
Encrypted Message (with public+private key pair)



Secure Message Exchange (over Persistent Connection)


HTTP Client
HTTP Server
Connect
server private key
server public key
Connect OK+ **Certificate (with public key)**
Client Secret Key (encrypted using server public key)
Client Secret key (decrypted using server private key)
Encrypted HTTP request (using client secret key)
Request (decrypted using common secret key)
Encrypted HTTP response (using client secret key)
Response (decrypted using client secret key)

GET or POST over secure connections



POST /path/to/your/course HTTP/1.0
Host: remote-server.io/
The text/binary contents of your File to upload to Bb will be included as attachments here
Unencrypted text
Client
Server (remote-server.io)
HTTP/1.0 200 OK
Content-Type: text/html
Host: info.mysite.org
Additional message from server goes here
Encrypted text

Uploading Sensitive Data over Encrypted Channel

- Embed the sensitive data in a GET request query string

```
GET /place/my/order/?creditcard=xxxxyyyyzzzzuuuu&zip=12345 HTTP/1.0
Host: www.amazon.co.uk
```

- Embed the sensitive data in a POST message payload

```
POST /place/my/order HTTP/1.0
Host: www.amazon.co.uk

creditcard=xxxxyyyyzzzzuuuu
zip=12345
```

Uploading Sensitive Data over Encrypted Channel

- Embed the sensitive data in a GET request query string

```
GET /place/my/order/?creditcard=xxxxyyyyzzzzuuuu&zip=12345 HTTP/1.0
Host: www.amazon.co.uk
```

Unencrypted

✗

- Embed the sensitive data in a POST message payload

```
POST /place/my/order HTTP/1.0
Host: www.amazon.co.uk
```

Unencrypted

```
creditcard=xxxxyyyyzzzzuuuu
zip=12345
```

Encrypted

✓

Certificate and Certificate Authority (CA)

Certificate: Proof of Your Identity

Certificate Authority:
Trusted Organizations who issue certificates

Michigan IDs vs. Browser Certificates

Michigan IDs	(Browser) Certificates
A formal proof of your identity	A formal proof of the web server identity
Issued and signed by Secretary of State	Issued and signed by Certificate Authority
Provide other proof of identity (birth certificate, passport) to apply for Michigan ID to the SoS	Certificate Signing Request: server request a CA to sign the server’s identity (public key) using the CA key
The SoS is a trusted government body	Trusted CAs

Obtaining Web Certificates ("Web ID Cards")

Proof of identity (passport, GOV ids, birth certificates)

Certificate (signed by CA)

Private+Public Key Pair → Certificate Authority (CA) **Known to Browsers** → Web Certificate (signed by CA's private key)

Watch:
http://www.youtube.com/watch?v=iQsKdtjwtYI