转载自 github：Interview-Notebook，有删减和改动 参考：《图解 HTTP》

一、基础概念

Web 基础

- HTTP（HyperText Transfer Protocol，超文本传输协议）。
- WWW（World Wide Web）的三种技术：HTML、HTTP、URL。
- RFC（Request for Comments，征求修正意见书），互联网的设计文档。

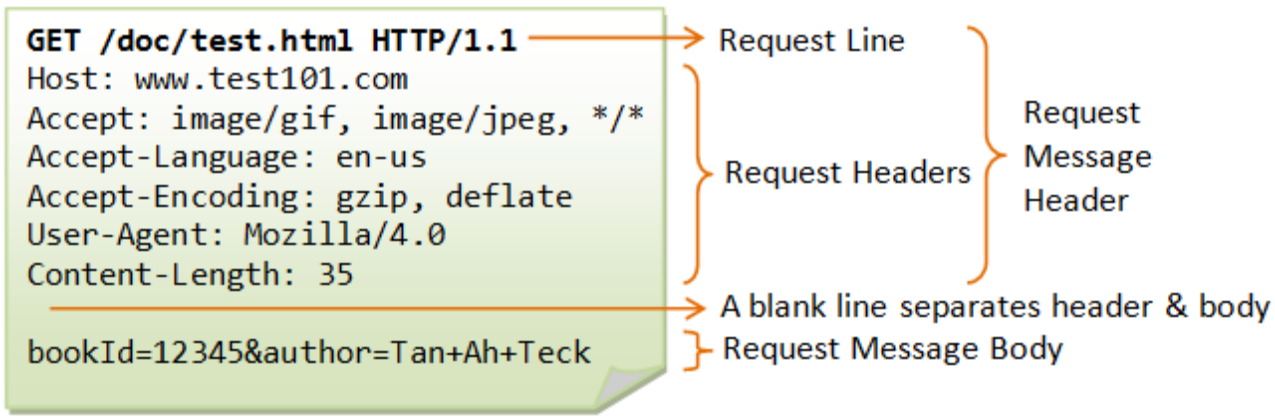
URL

- URI（Uniform Resource Indentifier，统一资源标识符）
- URL（Uniform Resource Locator，统一资源定位符）
- URN（Uniform Resource Name，统一资源名称），例如 urn:isbn:0-486-27557-4 。

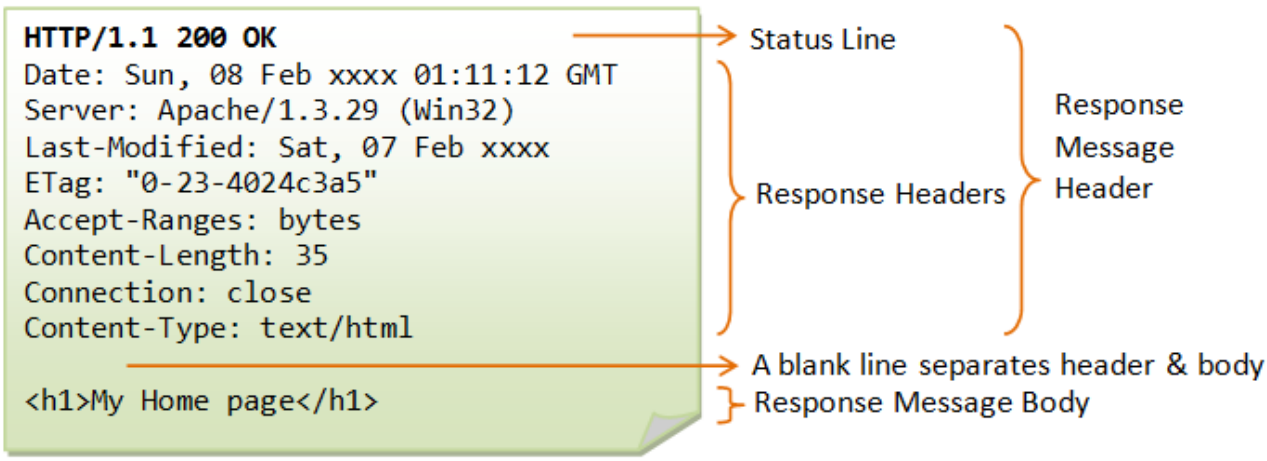
URI 包含 URL 和 URN，目前 WEB 只有 URL 比较流行，所以见到的基本都是 URL。

请求和响应报文

1. 请求报文



2. 响应报文



二、HTTP 方法

客户端发送的 **请求报文** 第一行为请求行，包含了方法字段。

GET

> 获取资源

当前网络请求中，绝大部分使用的是 GET 方法。

HEAD

> 获取报文首部

和 GET 方法一样，但是不返回报文实体主体部分。

主要用于确认 URL 的有效性以及资源更新的日期时间等。

POST

> 传输实体主体

POST 主要用来传输数据，而 GET 主要用来获取资源。

更多 POST 与 GET 的比较请见第八章。

PUT

> 上传文件

由于自身不带验证机制，任何人都可以上传文件，因此存在安全性问题，一般不使用该方法。

```
PUT /new.html HTTP/1.1
Host: example.com
Content-type: text/html
Content-length: 16

<p>New File</p>
```

PATCH

> 对资源进行部分修改

PUT 也可以用于修改资源，但是只能完全替代原始资源，PATCH 允许部分修改。

```
PATCH /file.txt HTTP/1.1
Host: www.example.com
Content-Type: application/example
If-Match: "e0023aa4e"
Content-Length: 100

[description of changes]
```

DELETE

> 删除文件

与 PUT 功能相反，并且同样不带验证机制。

```
DELETE /file.html HTTP/1.1
```

OPTIONS

> 查询支持的方法

查询指定的 URL 能够支持的方法。

会返回 Allow: GET, POST, HEAD, OPTIONS 这样的内容。

CONNECT

> 要求用隧道协议连接代理

要求在与代理服务器通信时建立隧道，使用 SSL（Secure Sockets Layer，安全套接层）和 TLS（Transport Layer Security，传输层安全）协议把通信内容加密后经网络隧道传输。

```
CONNECT www.example.com:443 HTTP/1.1
```

TRACE

> 追踪路径

服务器会将通信路径返回给客户端。

发送请求时，在 Max-Forwards 首部字段中填入数值，每经过一个服务器就会减 1，当数值为 0 时就停止传输。

通常不会使用 TRACE，并且它容易受到 XST 攻击（Cross-Site Tracing，跨站追踪），因此更不会去使用它。

三、HTTP 状态码

服务器返回的 **响应报文** 中第一行为状态行，包含了状态码以及原因短语，用来告知客户端请求的结果。

状态码	类别	原因短语
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器处理请求出错

1XX 信息

- **100 Continue** ：表明到目前为止都很正常，客户端可以继续发送请求或者忽略这个响应。

2XX 成功

- **200 OK**
- **204 No Content** ：请求已经成功处理，但是返回的响应报文不包含实体的主体部分。一般在只需要从客户端往服务器发送信息，而不需要返回数据时使用。
- **206 Partial Content** ：表示客户端进行了范围请求。响应报文包含由 Content-Range 指定范围的实体内容。

3XX 重定向

- **301 Moved Permanently** ：永久性重定向
- **302 Found** ：临时性重定向
- **303 See Other** ：和 302 有着相同的功能，但是 303 明确要求客户端应该采用 GET 方法获取资源。
- 注：虽然 HTTP 协议规定 301、302 状态下重定向时不允许把 POST 方法改成 GET 方法，但是大多数浏览器都会在 301、302 和 303 状态下的重定向把 POST 方法改成 GET 方法。
- **304 Not Modified** ：如果请求报文首部包含一些条件，例如：If-Match，If-ModifiedSince，If-None-Match，If-Range，If-Unmodified-Since，如果不满足条件，则服务器会返回 304 状态码。
- **307 Temporary Redirect** ：临时重定向，与 302 的含义类似，但是 307 要求浏览器不会把重定向请求的 POST 方法改成 GET 方法。

4XX 客户端错误

- **400 Bad Request** ：请求报文中存在语法错误。

- **401 Unauthorized** ：该状态码表示发送的请求需要有认证信息（BASIC 认证、DIGEST 认证）。如果之前已进行过一次请求，则表示用户认证失败。
- **403 Forbidden** ：请求被拒绝，服务器端没有必要给出拒绝的详细理由。
- **404 Not Found**

5XX 服务器错误

- **500 Internal Server Error** ：服务器正在执行请求时发生错误。
- **503 Service Unavilable** ：服务器暂时处于超负载或正在进行停机维护，现在无法处理请求。

四、HTTP 首部

有 4 种类型的首部字段：通用首部字段、请求首部字段、响应首部字段和实体首部字段。

各种首部字段及其含义如下（不需要全记，仅供查阅）：

通用首部字段

首部字段名	说明
Cache-Control	控制缓存的行为
Connection	控制不再转发给代理的首部字段、管理持久连接
Date	创建报文的日期时间
Pragma	报文指令
Trailer	报文末端的首部一览
Transfer-Encoding	指定报文主体的传输编码方式
Upgrade	升级为其他协议
Via	代理服务器的相关信息
Warning	错误通知

请求首部字段

首部字段名	说明
Accept	用户代理可处理的媒体类型
Accept-Charset	优先的字符集
Accept-Encoding	优先的内容编码
Accept-Language	优先的语言（自然语言）
Authorization	Web 认证信息
Expect	期待服务器的特定行为
From	用户的电子邮箱地址
Host	请求资源所在服务器
If-Match	比较实体标记（ETag）
If-Modified-Since	比较资源的更新时间
If-None-Match	比较实体标记（与 If-Match 相反）
If-Range	资源未更新时发送实体 Byte 的范围请求
If-Unmodified-Since	比较资源的更新时间（与 If-Modified-Since 相反）
Max-Forwards	最大传输逐跳数
Proxy-Authorization	代理服务器要求客户端的认证信息
Range	实体的字节范围请求
Referer	对请求中 URI 的原始获取方
TE	传输编码的优先级
User-Agent	HTTP 客户端程序的信息

响应首部字段

首部字段名	说明
Accept-Ranges	是否接受字节范围请求
Age	推算资源创建经过时间
ETag	资源的匹配信息
Location	令客户端重定向至指定 URI
Proxy-Authenticate	代理服务器对客户端的认证信息
Retry-After	对再次发起请求的时机要求
Server	HTTP 服务器的安装信息
Vary	代理服务器缓存的管理信息
WWW-Authenticate	服务器对客户端的认证信息

实体首部字段

首部字段名	说明
Allow	资源可支持的 HTTP 方法
Content-Encoding	实体主体适用的编码方式
Content-Language	实体主体的自然语言
Content-Length	实体主体的大小
Content-Location	替代对应资源的 URI
Content-MD5	实体主体的报文摘要
Content-Range	实体主体的位置范围
Content-Type	实体主体的媒体类型
Expires	实体主体过期的日期时间
Last-Modified	资源的最后修改日期时间

五、具体应用

Cookie

HTTP 协议是无状态的，主要是为了让 HTTP 协议尽可能简单，使得它能够处理大量事务。HTTP/1.1 引入 Cookie 来保存状态信息。

Cookie 是服务器发送给客户端的数据，该数据会被保存在浏览器中，并且客户端的下一次请求报文会包含该数据。通过 Cookie 可以让服务器知道两个请求是否来自于同一个客户端，从而实现保持登录状态等功能。

1. 创建过程

服务器发送的响应报文包含 Set-Cookie 字段，客户端得到响应报文后把 Cookie 内容保存到浏览器中。

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry

[page content]
```

客户端之后发送请求时，会从浏览器中读出 Cookie 值，在请求报文中包含 Cookie 字段。

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

2. 分类

- 会话期 Cookie：浏览器关闭之后它会被自动删除，也就是说它仅在会话期内有效。
- 持久性 Cookie：指定一个特定的过期时间（Expires）或有效期（Max-Age）之后就成为了持久性的 Cookie。

```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
```

3. Set-Cookie

属性	说明
NAME=VALUE	赋予 Cookie 的名称和其值（必需项）
expires=DATE	Cookie 的有效期（若不明确指定则默认为浏览器关闭前为止）
path=PATH	将服务器上的文件目录作为 Cookie 的适用对象（若不指定则默认为文档所在的文件目录）
domain=域名	作为 Cookie 适用对象的域名（若不指定则默认为创建 Cookie 的服务器的域名）
Secure	仅在 HTTPs 安全通信时才会发送 Cookie
HttpOnly	加以限制，使 Cookie 不能被 JavaScript 脚本访问

4. Session 和 Cookie 区别

Session 是服务器用来跟踪用户的一种手段，每个 Session 都有一个唯一标识：Session ID。当服务器创建了一个 Session 时，给客户端发送的响应报文包含了 Set-Cookie 字段，其中有一个名为 sid 的键值对，这个键值对就是 Session ID。客户端收到后就把 Cookie 保存在浏览器中，并且之后发送的请求报文都包含 Session ID。HTTP 就是通过 Session 和 Cookie 这两种方式一起合作来实现跟踪用户状态的，Session 用于服务器端，Cookie 用于客户端。

5. 浏览器禁用 Cookie 的情况

会使用 URL 重写技术，在 URL 后面加上 sid=xxx 。

6. 使用 Cookie 实现用户名和密码的自动填写

网站脚本会自动从保存在浏览器中的 Cookie 读取用户名和密码，从而实现自动填写。

但是如果 Set-Cookie 指定了 HttpOnly 属性，就无法通过 Javascript 脚本获取 Cookie 信息，这是出于安全性考虑。

缓存

1. 优点

1. 降低服务器的负担；
2. 提高响应速度（缓存资源比服务器上的资源离客户端更近）。

2. 实现方法

1. 让代理服务器进行缓存；
2. 让客户端浏览器进行缓存。

3. Cache-Control 字段

HTTP 通过 Cache-Control 首部字段来控制缓存。

```
Cache-Control: private, max-age=0, no-cache
```

4. no-cache 指令

该指令出现在请求报文的 Cache-Control 字段中，表示缓存服务器需要先向原服务器验证缓存资源是否过期；

该指令出现在响应报文的 Cache-Control 字段中，表示缓存服务器在进行缓存之前需要先验证缓存资源的有效性。

5. no-store 指令

该指令表示缓存服务器不能对请求或响应的任何一部分进行缓存。

no-cache 不表示不缓存，而是缓存之前需要先进行验证，no-store 才是不进行缓存。

6. max-age 指令

该指令出现在请求报文的 Cache-Control 字段中，如果缓存资源的缓存时间小于该指令指定的时间，那么就能接受该缓存。

该指令出现在响应报文的 Cache-Control 字段中，表示缓存资源在缓存服务器中保存的时间。

Expires 字段也可以用于告知缓存服务器该资源什么时候会过期。在 HTTP/1.1 中，会优先处理 Cache-Control : max-age 指令；而在 HTTP/1.0 中，Cache-Control : max-age 指令会被忽略掉。

持久连接

当浏览器访问一个包含多张图片的 HTML 页面时，除了请求访问 HTML 页面资源，还会请求图片资源，如果每进行一次 HTTP 通信就要断开一次 TCP 连接，连接建立和断开的开销会很大。持久连接只需要建立一次 TCP 连接就能进行多次 HTTP 通信。

持久连接需要使用 Connection 首部字段进行管理。HTTP/1.1 开始 HTTP 默认是持久化连接的，如果要断开 TCP 连接，需要由客户端或者服务器端提出断开，使用 Connection : close；而在 HTTP/1.1 之前默认是非持久化连接的，如果要维持持续连接，需要使用 Connection : Keep-Alive。

管线化处理

HTTP/1.1 支持管线化处理，可以同时发送多个请求和响应，而不需要发送一个请求然后等待响应之后再发下一个请求。

编码

编码（Encoding）主要是为了对实体进行压缩。常用的编码有：gzip、compress、deflate、identity，其中 identity 表示不执行压缩的编码格式。

分块传输编码

Chunked Transfer Coding，可以把数据分割成多块，让浏览器逐步显示页面。

多部分对象集合

一份报文主体内可含有多种类型的实体同时发送，每个部分之间用 boundary 字段定义的分隔符进行分隔，每个部分都可以有首部字段。

例如，上传多个表单时可以使用如下方式：

```
Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x
Content-Disposition: form-data; name="submit-name"

Larry
--AaB03x
Content-Disposition: form-data; name="files"; filename="file1.txt"
Content-Type: text/plain

... contents of file1.txt ...
--AaB03x--
```

范围请求

如果网络出现中断，服务器只发送了一部分数据，范围请求使得客户端能够只请求未发送的那部分数据，从而避免服务器端重新发送所有数据。

在请求报文首部中添加 Range 字段指定请求的范围，请求成功的话服务器发送 206 Partial Content 状态。

```
GET /z4d4kWk.jpg HTTP/1.1
Host: i.imgur.com
Range: bytes=0-1023
```

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 0-1023/146515
Content-Length: 1024
...
(binary content)
```

内容协商

通过内容协商返回最合适的内容，例如根据浏览器的默认语言选择返回中文界面还是英文界面。

涉及以下首部字段：Accept、Accept-Charset、Accept-Encoding、Accept-Language、Content-Language。

虚拟主机

HTTP/1.1 使用虚拟主机技术，使得一台服务器拥有多个域名，并且在逻辑上可以看成多个服务器。

使用 Host 首部字段进行处理。

通信数据转发

1. 代理

代理服务器接受客户端的请求，并且转发给其它服务器。

使用代理的主要目的是：缓存、网络访问控制以及访问日志记录。

代理服务器分为正向代理和反向代理两种，用户察觉得到正向代理的存在，而反向代理一般位于内部网络中，用户察觉不到。

2. 网关

与代理服务器不同的是，网关服务器会将 HTTP 转化为其它协议进行通信，从而请求其它非 HTTP 服务器的服务。

3. 隧道

使用 SSL 等加密手段，为客户端和服务器之间建立一条安全的通信线路。隧道本身不去解析 HTTP 请求。

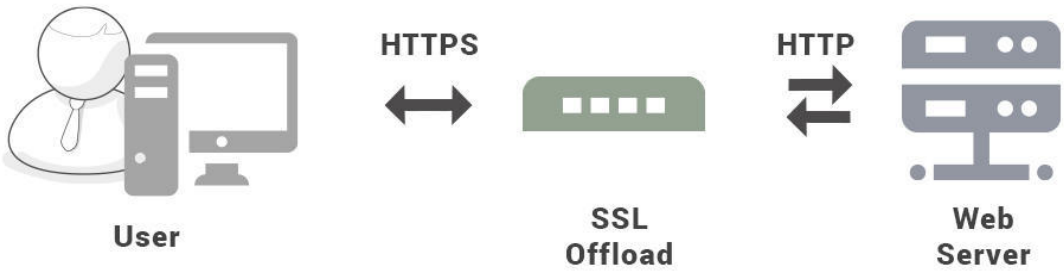
六、HTTPs

HTTP 有以下安全性问题：

1. 使用明文进行通信，内容可能会被窃听；
2. 不验证通信方的身份，通信方的身份有可能遭遇伪装；
3. 无法证明报文的完整性，报文有可能遭篡改。

HTTPs 并不是新协议，而是 HTTP 先和 SSL（Secure Sockets Layer）通信，再由 SSL 和 TCP 通信。也就是说 HTTPs 使用了隧道进行通信。

通过使用 SSL，HTTPs 具有了加密、认证和完整性保护。

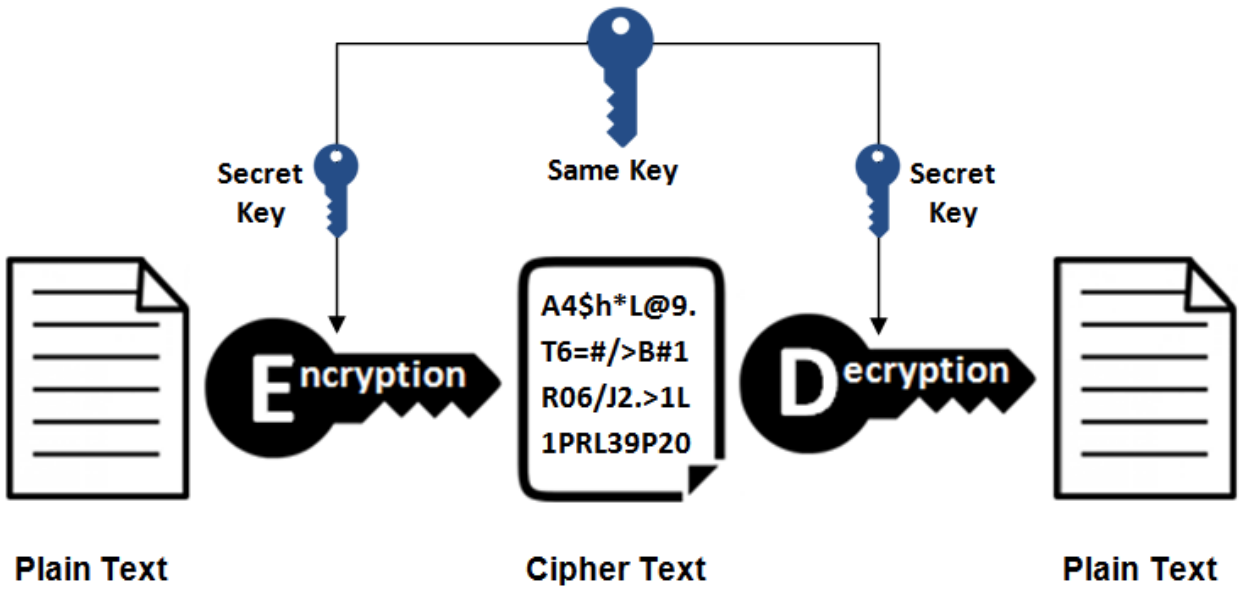


加密

1. 对称密钥加密

对称密钥加密（Symmetric-Key Encryption），加密的加密和解密使用同一密钥。

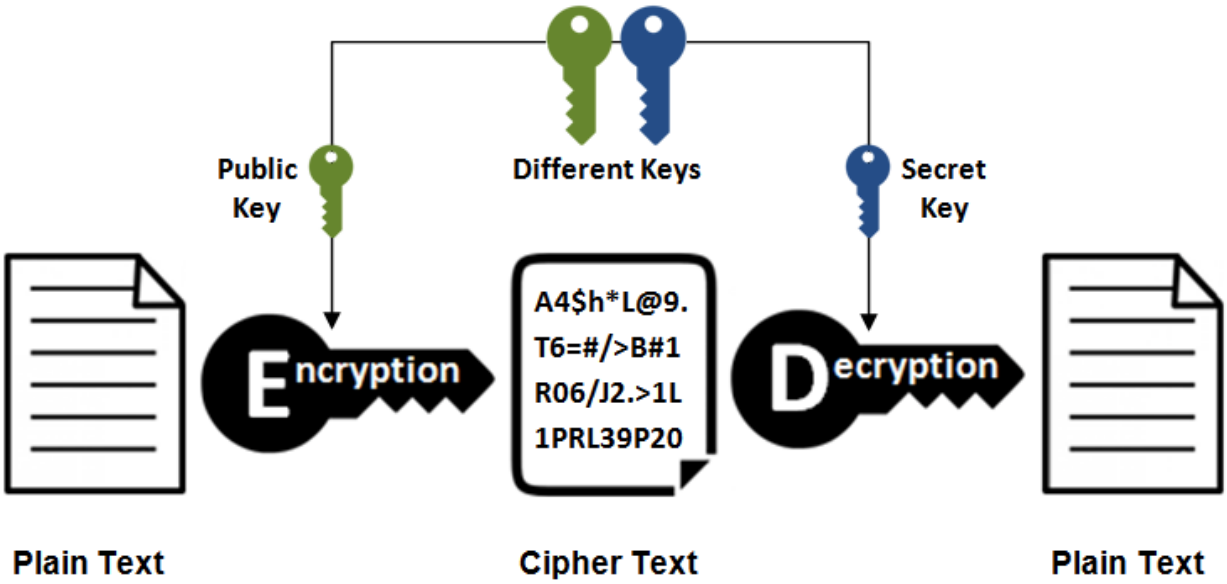
- 优点：运算速度快；
- 缺点：密钥容易被获取。



2. 公开密钥加密

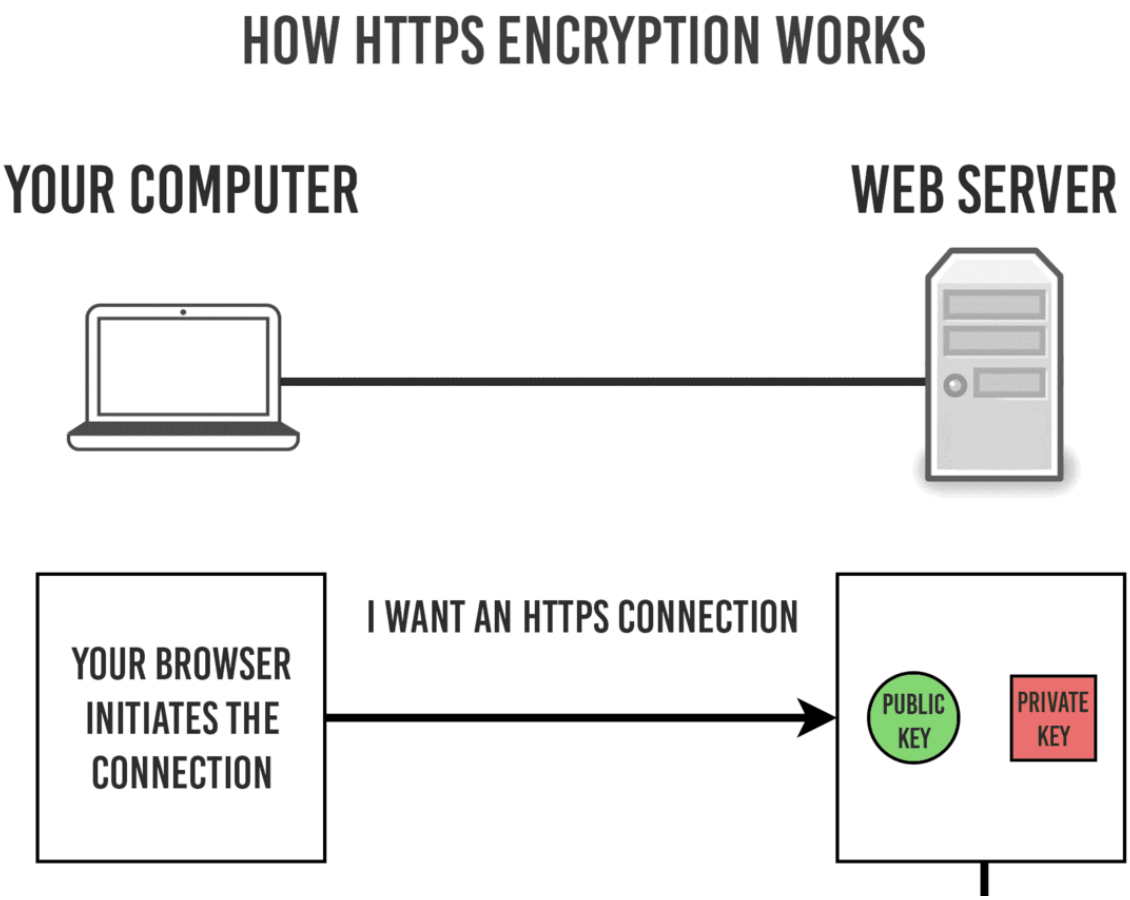
公开密钥加密（Public-Key Encryption），也称为非对称密钥加密，使用一对密钥用于加密和解密，分别为公开密钥和私有密钥。公开密钥所有人都可以获得，通信发送方获得接收方的公开密钥之后，就可以使用公开密钥进行加密，接收方收到通信内容后使用私有密钥解密。

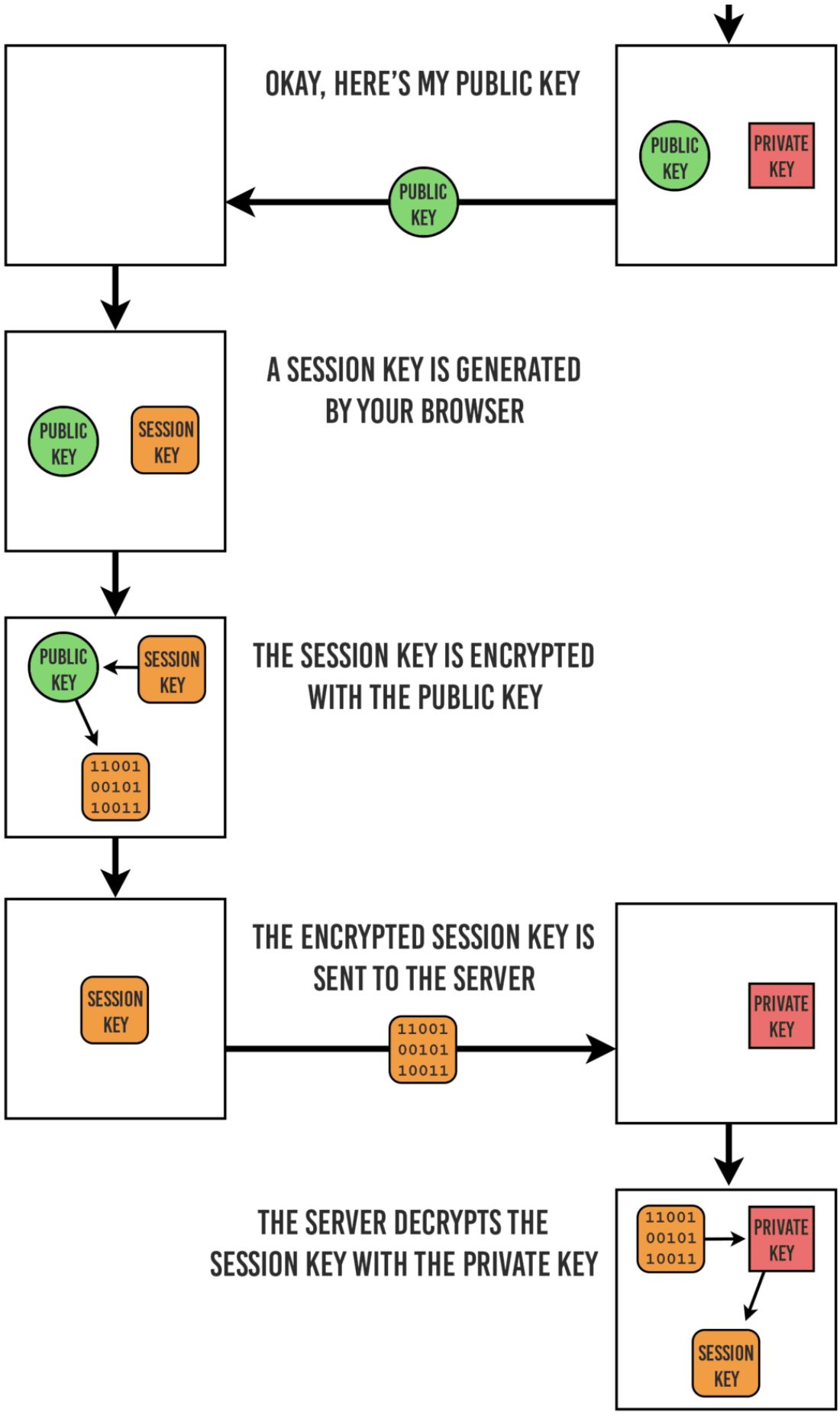
- 优点：更为安全；
- 缺点：运算速度慢；

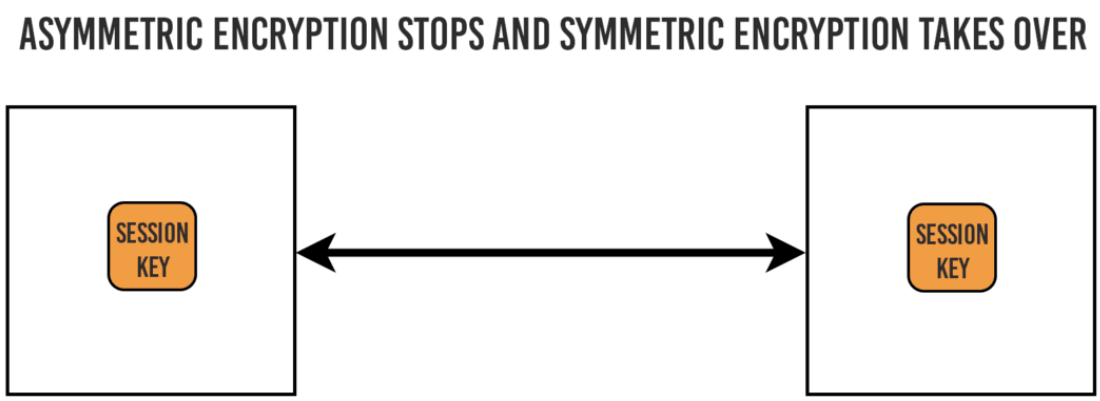


3. HTTPs 采用的加密方式

HTTPs 采用混合的加密机制，使用公开密钥加密用于传输对称密钥，之后使用对称密钥加密进行通信。（下图中的 Session Key 就是对称密钥）




OKAY, HERE'S MY PUBLIC KEY
PUBLIC KEY
PUBLIC KEY
PRIVATE KEY
A SESSION KEY IS GENERATED BY YOUR BROWSER
PUBLIC KEY
SESSION KEY
THE SESSION KEY IS ENCRYPTED WITH THE PUBLIC KEY
PUBLIC KEY
SESSION KEY
11001 00101 10011
THE ENCRYPTED SESSION KEY IS SENT TO THE SERVER
SESSION KEY
11001 00101 10011
PRIVATE KEY
THE SERVER DECRYPTS THE SESSION KEY WITH THE PRIVATE KEY
11001 00101 10011
PRIVATE KEY
SESSION KEY



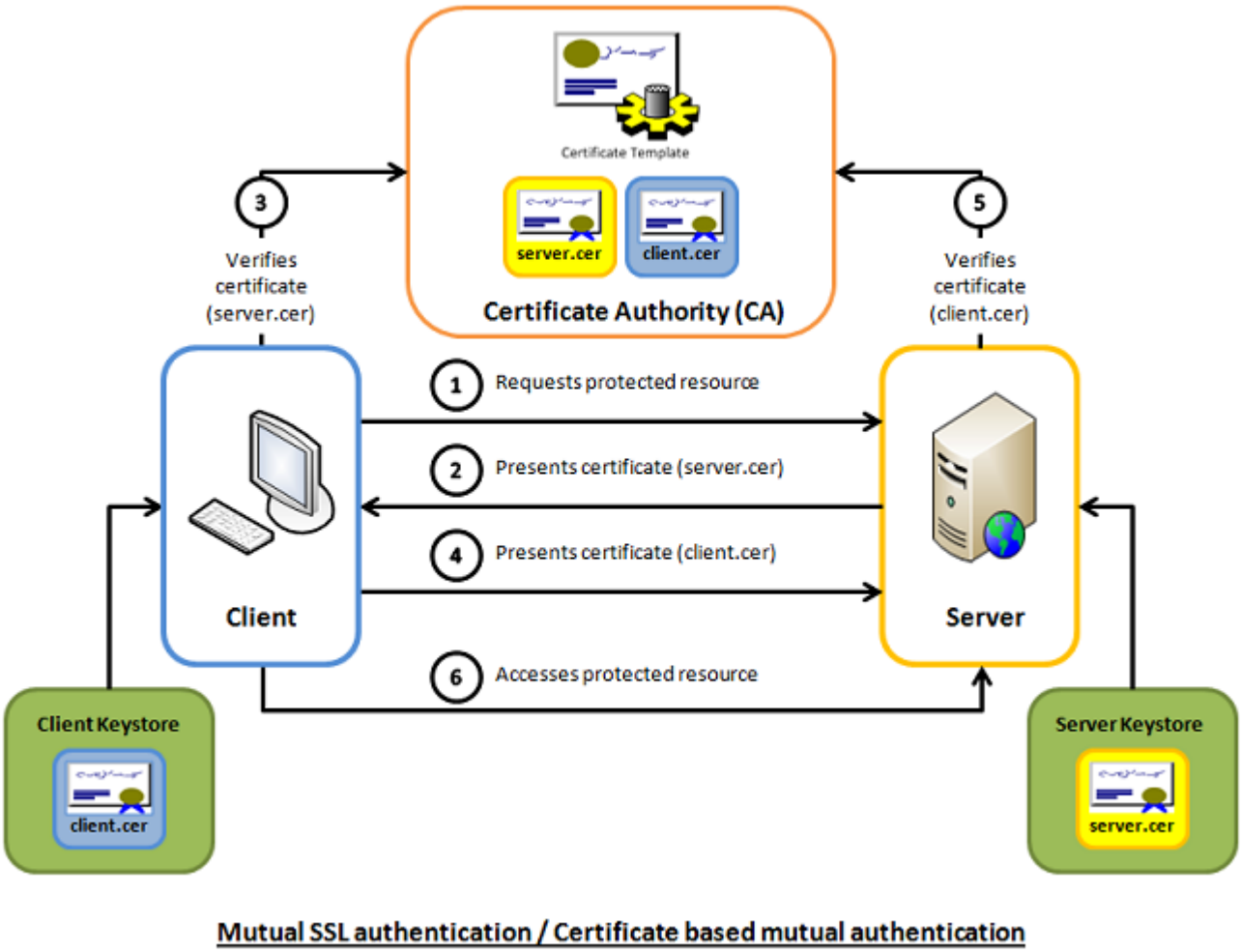
认证

通过使用 **证书** 来对通信方进行认证。

数字证书认证机构（CA，Certificate Authority）是客户端与服务器双方都可信赖的第三方机构。服务器的运营人员向 CA 提出公开密钥的申请，CA 在判明提出申请者的身份之后，会对已申请的公开密钥做数字签名，然后分配这个已签名的公开密钥，并将该公开密钥放入公开密钥证书后绑定在一起。

进行 HTTPs 通信时，服务器会把证书发送给客户端，客户端取得其中的公开密钥之后，先进行验证，如果验证通过，就可以开始通信。

使用 OpenSSL 这套开源程序，每个人都可以构建一套属于自己的认证机构，从而自己给自己颁发服务器证书。浏览器在访问该服务器时，会显示“无法确认连接安全性”或“该网站的安全证书存在问题”等警告消息。



Mutual SSL authentication / Certificate based mutual authentication

完整性

SSL 提供报文摘要功能来验证完整性。

七、Q and A

Get 请求和 Post 请求的区别

- 参数传递：GET 的参数只能带在URL后面，URL有长度限制，一般为2048个字符。POST 请求参数在 body 中，长度无明确规定的限制。GET 因为参数在URL中可见，安全性较差。
- 幂等性：GET 是幂等的，POST 不是幂等的。引入幂等主要是为了处理同一个请求重复发送的情况，比如在请求响应前失去连接，如果方法是幂等的，就可以放心地重发一次请求。GET后退按钮/刷新无害，POST数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
- 可缓存性：GET 是可被缓存的，POST是不可被缓存的。
- GET 不可指定编码类型（只允许ASCII字符），POST 可以指定编码类型，也允许二进制数据。
- GET产生一个TCP数据包；POST产生两个TCP数据包。对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）； 而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。