

## http协议VS https 协议

2019年5月10日 星期五 上午9:45

### 引用

- 1 [http://www.ruanyifeng.com/blog/2013/06/rsa\\_algorithm\\_part\\_one.html](http://www.ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html) 加密算法
- 2 <https://mp.weixin.qq.com/s/wVOhrL-N5YRRbHezdg9jZw>

### 首部

首部是key - value的格式。通过冒号分隔。里面保存着非常重要的字段。

1 Accept-Charset 表示客户端可以接受的字符集。

防止传过来时另外的字符集导致出现乱码

2 content-type 正文的格式

3 缓存-

1 区别

2 https 协议

3 原理是什么

4 和其他协议的区别，网络，有没有三次握手

5 怎么掉接口。

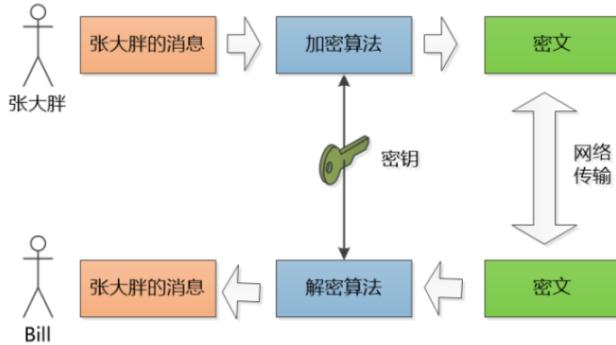
6 后段掉的方法是什么。

7 http能传什么东西。

### Https 安全传输原理

对称加密（加密和解密用的是同一个密钥）

加密和解密的算法是公开的但是密钥是保密的。



密钥发送其实也是需要加密的呀。（网络是不安全的密钥如何做到安全的发送）

后来想想我们，每个人都有不同的聊天对象。不能满世界的找别人交换密钥吧。

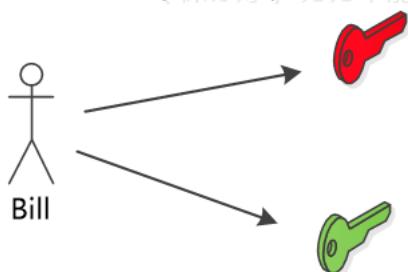
### 非对称加密

Rsa 的。 非对称加密

通信的双方都需要有一对密钥（一个是保密的私钥 另一个是公开的公钥）

用私钥加密的数据只有用公钥才能解密。用公钥加密的数据，只有对应的私钥才能解密

**Bill的私钥**  
(保存好，万万不能泄露)



**Bill的公钥**  
(地球人都可以知道)

有了这两个漂亮的特性，当张大胖给Bill发消息的时候，就可以先用Bill的公钥去加密（反正Bill的公钥是公开的，地球人都知道），等到消息被Bill收到后，他反过来也是如此，当Bill想给张大胖发消息的时候，就用张大胖的公钥加密，张大胖收到后，就用自己的私钥解密。

### 非对称加密 + 对称加密算法

(1) 我生成一个对称加密算法的密钥，用RSA的方式安全发给你，(2) 我们随后就不用RSA了，只用这个密钥，利用对称加密算法来通信，如何？”既解决了密

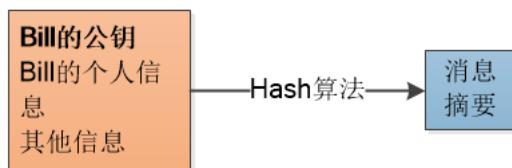
## 中间人攻击



1. 张大胖用中间人的公钥加密，发送出去
2. 中间人用自己的私钥解密，读取了消息内容
3. 中间人用Bill的公钥加密，发送出去
4. Bill用自己的私钥解密，读取了消息的内容  
但是没有意识到他们的通信已经被偷窥

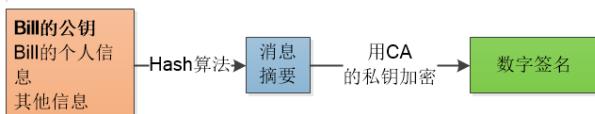
## 数字签名

1



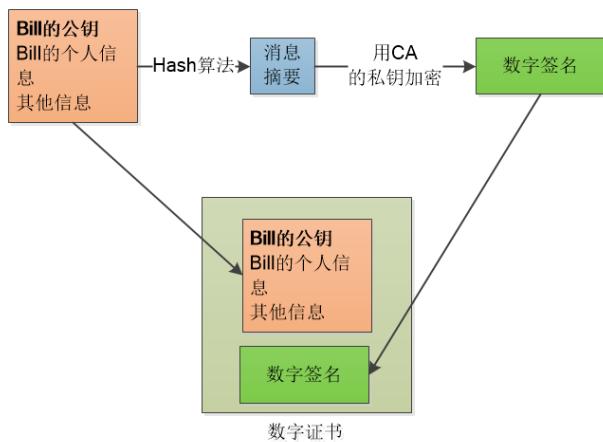
中间人如果能够，把个人信息和原始的信息全部替换掉，生成新的消息摘要

2



找到权威的认证中心。（简称 CA 形成数字签名）

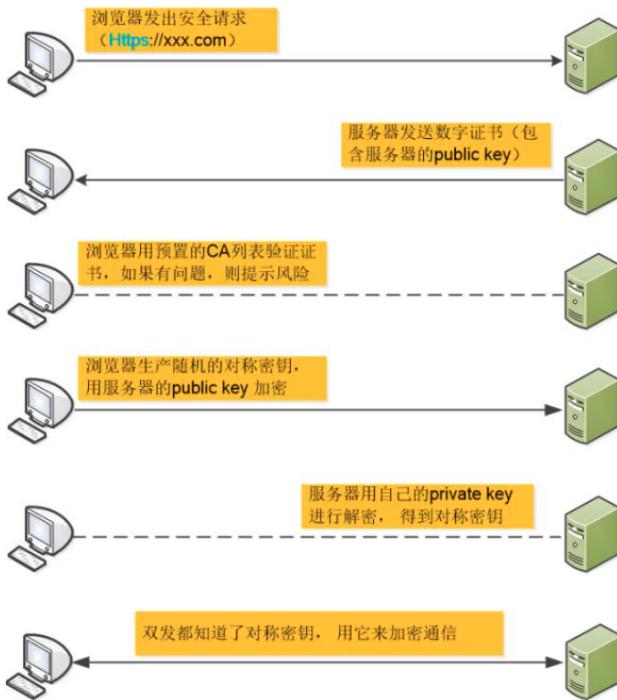
这还不算，还把原始信息和数据签名合并，形成一个全新的东西，叫做“数字证书”





## Https 的原理

一个简化的（例如下图没有包含Pre-Master Secret）https流程图是这样的，如果你理解了前面的原理，这张图就变得非常简单：



## Http 协议

<https://www.cnblogs.com/wangning528/p/6388464.html>

http有很多的标签

- 1 是无状态的协议
- 2 应用层协议
- 3 由请求和响应构成的协议
- 4 标准的客户端和服务器模型

Http 协议请求的过程

1 准备

- 如果想要访问 [www.163.com](http://www.163.com) 这个域名发送给DNS 服务器。DNS 服务器解析成ip地址。
- Http 协议是基于TCP协议的。当然是先建立TCP连接。（http 1.1 是默认开启了keep-alive 的这样建立的连接可以在多次请求中重复使用的。）

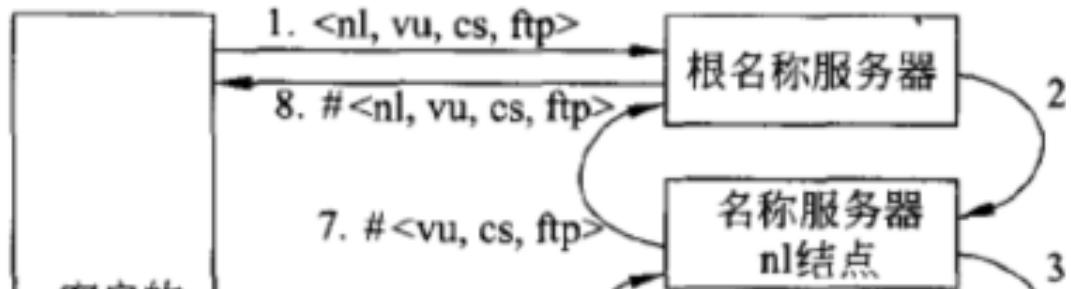
首先让我们从一个问题入手，当我们在浏览器中输入 <http://www.baidu.com/> 访问百度的时候浏览器做了哪些事情。（这里以 Chrome 浏览器为例）

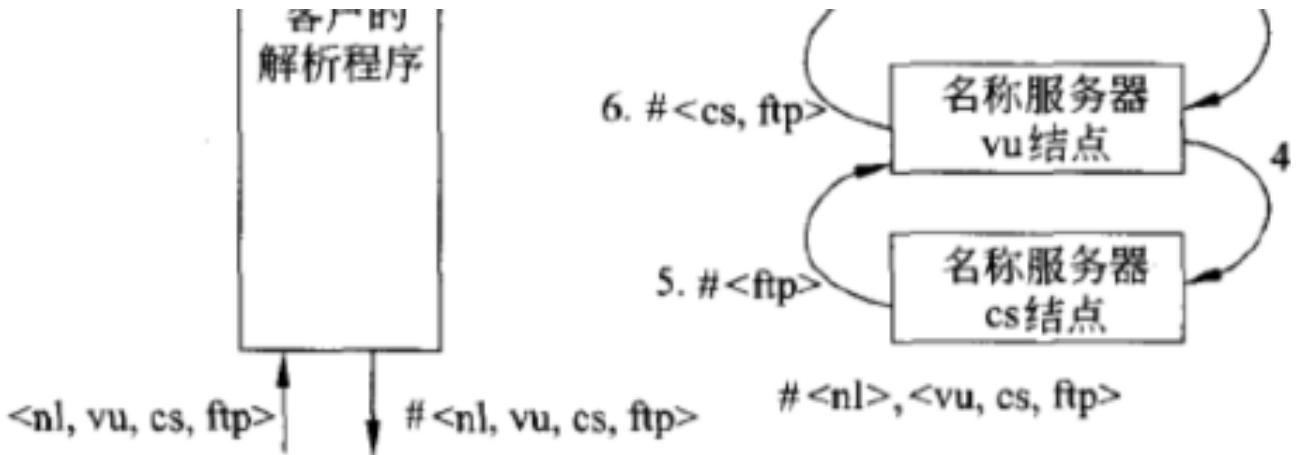
1. 首先 Chrome 搜索自身的 DNS 缓存。（如果 DNS 缓存中找到百度的 IP 地址，就跳过了接下来查找 IP 地址步骤，直接访问该 IP 地址。）
2. 搜索操作系统自身的 DNS 缓存。（浏览器没有找到缓存或者缓存已经失效）
3. 读取硬盘中的 host 文件，里面记录着域名到 IP 地址的映射关系，Mac 电脑中位于 /etc/hosts。（如果前1.2步骤都没有找到）
4. 浏览器向宽带运营商服务器或者域名服务器发起一个 DNS 解析请求，**这里服务器有两种方式解析请求**，这在稍后会讲到，之后浏览器获得了百度首页的 IP
5. 拿到 IP 地址后，浏览器就向该 IP 所在的服务器建立 TCP 连接(即三次握手)。
6. 连接建立起来之后，浏览器就可以向服务器发起 HTTP 请求了。（这里比如访问百度首页，就向服务器发起 HTTP 中的 GET 请求）
7. 服务器接受到这个请求后，根据路径参数，经过后台一些处理之后，把处理后的结果返回给浏览器，如果是百度首页，就可以把完整的 HTML 页面代码返回
8. 浏览器拿到了百度首页的完整 HTML 页面代码，内核和 JS 引擎就会解析和渲染这个页面，里面的 JS，CSS，图片等静态资源也通过一个个 HTTP 请求进行
9. 浏览器根据拿到的资源对页面进行渲染，最终把完整的页面呈现给用户。
10. 如果浏览器没有后续的请求，那么就会跟服务器端发起 TCP 断开(即四次挥手)。

至此，整个访问过程就结束了，可见浏览器帮我们做了许多的事。这里只是简单的概括，实际情况远比这些复杂。

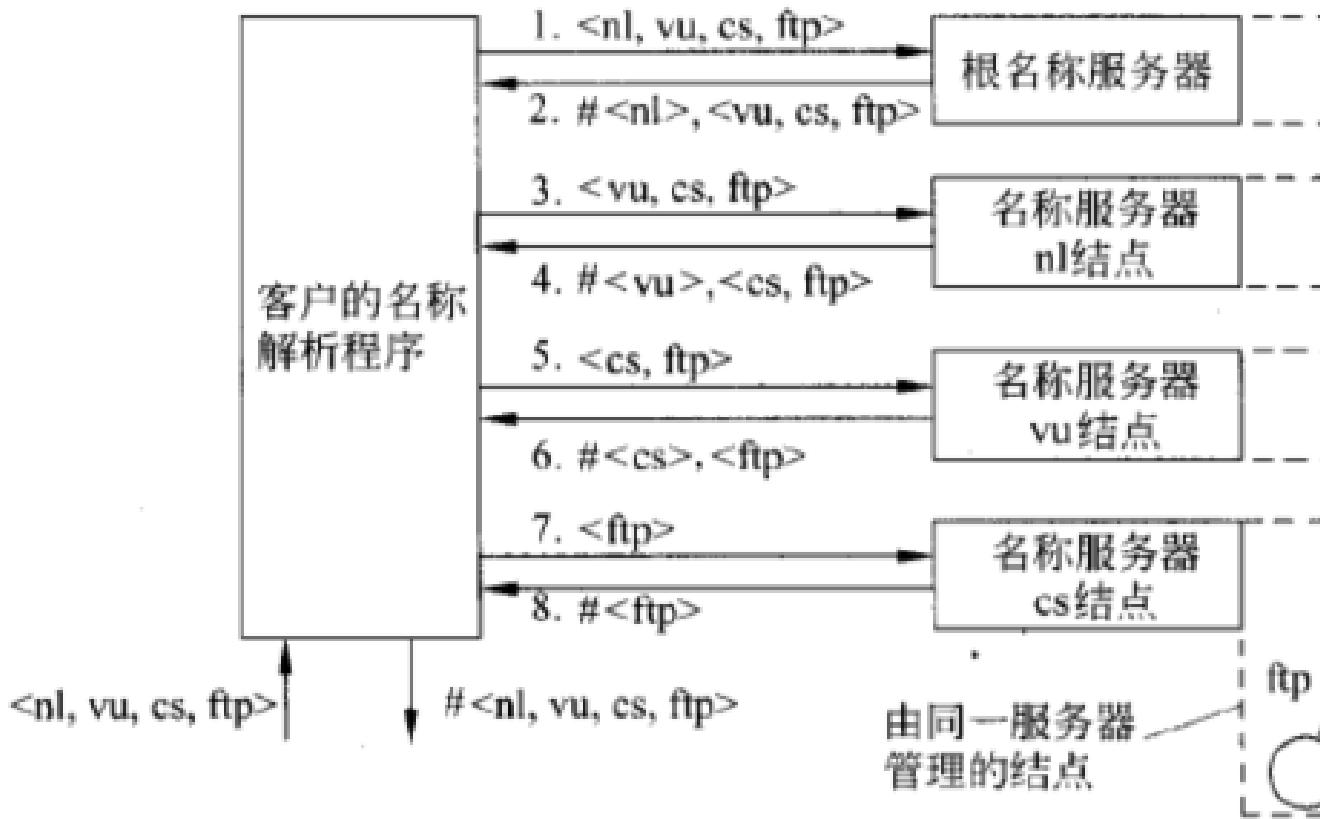
上面提到，服务器在接受 DNS 解析请求的时候一般会有两种处理方式，它们分别是**递归名称解析**和**迭代名称解析**。

递归名称解析：





用户在向根名称服务器发送请求如图中为访问网址为 <http://ftp.cs.vu.nl> 之后就不用管后续的请求了，该服务器知道 nl 服务器地址，并向其询问其子域 <http://ftp.cs.vu> 的地址迭代名称解析：



客户端向根名称服务器发送查询 <http://ftp.cs.vu.nl> 的地址时候，根名称服务器只知道 nl 地址，它并不管后续的请求，而是将该地址直接返回给用户，而用户在获得地址后完成，最后拿到 <http://ftp.cs.vu.nl> 的 IP 地址。

## HTTP 基本概念

HTTP，全称为 HyperText Transfer Protocol，即为超文本传输协议。是互联网应用最为广泛的一种网络协议，所有的 www 文件都必须遵守这个标准。  
HTTP 特性：

- HTTP 是无连接无状态的
- HTTP 一般构建于 TCP/IP 协议之上，默认端口号是 80

HTTP 可以分为两个部分，即请求和响应。

### HTTP 请求：

HTTP 定义了在与服务器交互的不同方式，最常用的方法有 4 种，分别是 GET, POST, PUT, DELETE。URL 全称为资源描述符，可以这么认为：一个 URL 地址就对应着对这个资源的查询，修改，增添，删除 4 个操作。

HTTP 请求由 3 个部分构成，分别是：状态行，请求头(Request Header)，请求正文。

GET 请求报文实例：

```
GET /books/?sex=man&name=Professional HTTP/1.1
Host: www.wrox.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

- 状态行由请求方式、路径、协议等构成，各元素之间以空格分隔。对应到图中即为 GET、/books/?sex=man&name=Professional、HTTP/1.1
  - 请求头提供一些参数比如：Cookie，用户代理信息，主机名等等。(图中即从第二行到最后一行)
  - 请求正文就放一些发送的数据，一般 GET 请求会将参数放在 URL 中，也就是在请求头中而请求正文一般为空，而 POST 请求将参数放在请求正文中。请求 GET 一般用于信息获取，比如刚才我们浏览百度首页，其使用的就是GET方法。
- GET 请求一般不会产生副作用，它仅仅只是获取资源信息，就像数据库查询一样，不会修改、增加数据，不会影响资源的状态，并且对同一个 URL 的多次GET请求而 POST 请求表示可能会修改服务器上的资源。

**POST / HTTP/1.1**

**Host: www.wrox.com**

**User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8 Gecko/20050225 Firefox/1.0.1**

**Content-Type: application/x-www-form-urlencoded**

**Content-Length: 40**

**Connection: Keep-Alive**

**sex=man&name=Professional**

#### GET 请求和 POST 请求的区别：

1. GET 和 POST 请求参数位置不同，从上面两个请求报文可以看出，GET 请求对应的参数放在 URL 中，而 POST 请求对应的参数放在 HTTP 请求主体中。(从图中可以看出)
2. 虽然 HTTP 协议的 [RFC规范](#) 并没有详细规定 URL 的最大字符长度限制，但实际上，在浏览器或者服务器中总会存在限制的，这就导致了 GET 请求中参数过多时容易出现乱码。
3. 处于安全考虑，在一些涉及安全的请求比如：登录请求需要用 POST 提交表单，而GET 请求一般用来获取静态资源。
4. GET 请求可以被缓存，可以被收藏为书签，但 POST 可以被缓存，但不能被收藏为书签。
5. GET 请求的参数在 URL 中，因此绝不能用 GET 请求传输敏感数据。POST 请求数据则写在 HTTP 的请求头中，安全性略高于 GET 请求。

#### HTTP 响应：

HTTP 响应是服务器在客户端发送 HTTP 请求后经过一些处理而做出的响应，HTTP 响应和 HTTP 请求相似，也是由三个部分构成。分别是：状态行，响应头(Response Headers)和响应体(Response Body)。下面是一个 HTTP 响应的例子：

**HTTP/1.1 200 OK**

**Server:Apache Tomcat/5.0.12**

**Date:Mon, 6 Oct 2003 13:23:42 GMT**

**Content-Length:112**

**<html>...**

HTTP 响应中包含一个状态码，用来表示服务器对客户端响应的结果。

状态码一般由3位构成：

- 1xx : 表示请求已经接受了，继续处理。
- 2xx : 表示请求已经处理掉了。
- 3xx : 重定向。
- 4xx : 一般表示客户端有错误，请求无法实现。
- 5xx : 一般为服务器端的错误。

比如常见的状态码：

- 200 OK 客户端请求成功。
- 301 Moved Permanently 请求永久重定向。
- 302 Moved Temporarily 请求临时重定向。
- 304 Not Modified 文件未修改，可以直接使用缓存的文件。
- 400 Bad Request 由于客户端请求有语法错误，不能被服务器所理解。

- 401 Unauthorized 请求未经授权，无法访问。
- 403 Forbidden 服务器收到请求，但是拒绝提供服务。服务器通常会在响应正文中给出不提供服务的原因。
- 404 Not Found 请求的资源不存在，比如输入了错误的URL。
- 500 Internal Server Error 服务器发生不可预期的错误，导致无法完成客户端的请求。
- 503 Service Unavailable 服务器当前不能够处理客户端的请求，在一段时间之后，服务器可能会恢复正常。

知道了 HTTP 请求和响应后，一个完整的流程一般是这样的：

通常，由 HTTP 客户端发起一个请求，建立一个到服务器指定端口（默认是 80 端口）的 TCP 连接。HTTP 服务器则在那个端口监听客户端发送过来的请求。一般地，客户端会发送一个“OK”，和（响应的）消息，消息的消息体可能是请求的文件、错误消息、或者其它一些信息。

## HTTP 头信息：

介绍完 HTTP 基本概念之后，下面介绍一些常见的 HTTP 请求头中字段的含义。

### HTTP 请求头：

比如以请求百度首页为例：

#### ▼ Request Headers [view source](#)

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6,zh-TW;q=0.4,pt;q=0.2
Cache-Control: no-cache
Connection: keep-alive
Cookie: BAIDUID=E590E33802E84E95368D04F8A57A847A;FG=1; BIDUPSID=E590E33802E84E95368D04F8A57A847A; PSTM=1472615194; BDUSS=k3Wk9XaWZ6MUxYbzdibkNJM0lJLXV4S1AmVXhEdk9uR3RZSVFBQUFBJCQAAAAAAAAAAEAAAAbm29Jc3ZzYWZndjQxNQAAAAAAAFAAAAAAAAAM4PRFjOD0RYeE; MCITY=-896073548_32501]=1484577616; BCLID=10043999916687650506; BDSFRCVID=2Q5t-hAfKeKBK2TTH6aI_uC1inL0QbKtSncgEG0PJU8g0Ku-XpmtoCKK3m0TH45P; H_EKPBjCvhDRTvhCTjh-FSMgTBKI62aKDsoJbg-hcqEIL4QPjJDjjWmqQItR0L2g6hab5c&fr=XWURIKe3JbbbrJ5h5nhMJ_3j7JDMP0qJ7j2R3y523iob6vQpPMMftu-n5jHj53eH803PSINO=7; H_PS_645EC=1d62UvJNC%2BMjhnlgeqRRaBfQy4eIgAbyAn8wQ3EDNL%2EuhYYPC; BD_HOME=1; H_PS_PSSID=1428_21086_17001_20239_20930; __bsi=1779_00_26_R_N_77_0303_C02F_N_I_I_0; BD_UPN=123253
Host: www.baidu.com
Pragma: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36
```

- Accept：指定客户端能够接收的内容类型，如常见的 text/html 等，最后返回的百度首页也是个 HTML 文件。

- Accept-Encoding：表示浏览器有能力解码的编码类型。

gzip 是 GNU zip 的缩写，它是一个 GNU 自由软件的文件压缩程序，也经常用来表示 gzip 这种文件格式。

deflate 是同时使用了 LZ77 算法与哈夫曼编码（Huffman Coding）的一个无损数据压缩算法。

- Accept-Language：表示浏览器所支持的语言类型。（这里指中文、简体中文和英文）
  - Cache-Control：指定请求和响应遵循的缓存机制。（这里表示不需要缓存）
  - Connection：表示是否需要持久连接。（HTTP 1.1 默认进行持久连接即为 keep-alive，HTTP 1.0 则默认为 close）
  - Cookie：用于会话追踪，在本文后面就继续介绍。
  - Host：表示请求的服务器网址
  - User-Agent：用户代理，简称 UA，它是一个特殊字符串头，使得服务器能够识别客户端使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎等。
- 还有另外还有一些常见的请求头：
- Content-Length：请求的内容长度
  - Referer：先前访问的网页的地址，当前请求网页紧随其后，说明你是先前是从哪个网址点击访问到该页面的，如果没有则不填。
  - Content-Type：内容的类型，GET 请求无该字段，POST 请求中常见的有 application/x-www-form-urlencoded 为普通的表单提交，还有文件上传为 multipart/form-data。

### HTTP 响应头：

还是以百度举例：

#### ▼ Response Headers [view source](#)

```
BDPAGETYPE: 1
BDQID: 0xfe071224000147ff
BDUSERID: 0
Cache-Control: private
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Gxy_all: baidu+eb3c63800288d3269320ab7d78cc21c9
Date: Tue, 17 Jan 2017 02:27:05 GMT
```

```

Expires: Tue, 17 Jan 2017 02:26:37 GMT
P3P: CP=" OTI DSP COR IVA OUR IND COM "
Server: bfe/1.0.8.18
Set-Cookie: BAIDUID=715D34C93D658B58415522AA77815FAC:FG=1; expires=Thu 5:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: H_PS_PSSID=1443_21118_21943_21616_20719; path=/; domain=.b
Set-Cookie: BD_HOME=0; path=/
Set-Cookie: BD5VRTM=0; path=/
Set-Cookie: __bsi=9012946489958564054_00_30_N_N_2_0303_C02F_N_N_Y_0; e
an-17 02:27:10 GMT; domain=www.baidu.com; path=/
Set-Cookie: BIDUPSID=715D34C93D658B58415522AA77815FAC; expires=Thu, 31
GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: PSTM=1484620025; expires=Thu, 31-Dec-37 23:55:55 GMT; max-
path=/; domain=.baidu.com
Strict-Transport-Security: max-age=172800
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Powered-By: PHP
X-UA-Compatible: IE=Edge,chrome=1

```

前一个是自定义字段，HTTP 请求头中的字段是可以自定义的。Connection, Content-Encoding, Content-Type 和请求头的内容差不多，不再赘述。

- Date: 原始服务器消息发出的时间。
- Last-Modified: 请求资源的最后修改时间。
- Expires: 响应过期的日期和时间，如果下次访问在时间允许的范围内，可以不用重新请求，直接访问缓存。
- Set-Cookie: 设置Http Cookie，下次浏览器再次访问的时候会带上这个Cookie值。
- Server: 服务器软件名称，常见的有Apache和Nginx。

上述列举的是比较常见的请求响应头及用法，如果想要全面了解更多参数，可以查阅 [HTTP请求头大全 - 常用参考表对照表 - 脚本之家在线工具](#)

## 其它：

了解以上那些概念后，已经对HTTP协议有了大致的了解了。下面介绍一些HTTP实现中其它内容。

### 会话追踪：

- 会话：客户端向服务器端发起请求到服务端响应客户端请求的全过程。
- 会话跟踪：会话跟踪指的是服务器对用户响应的监视。

为什么要会话跟踪：

浏览器与服务器之间的通信是通过HTTP协议进行通信的，而HTTP协议是“无状态”的协议，它不能保存客户的信息，即一次响应完成之后连接就断开了，下跟踪技术来实现这种要求。

比如你在访问淘宝登录之后会持续追踪你的会话，记录你的购物车记录等等。

### 会话跟踪常用方法：

- URL重写：URL重写技术就是在URL结尾添加一个附加数据以标识该会话，把会话ID通过URL的信息传递过去，以便在服务器进行识别不同的用户。
  - 隐藏表单域：将会话ID添加到HTML表单元素中提交到服务器，此表单元素并不在客户端显示。
  - Cookie：Cookie是Web服务器发送给客户端的一小段信息，客户端请求时可以读取该信息发送给服务器端，进而进行用户的识别，对于客户端的每次请用。
- 客户端可以采用两种方式来保存这个Cookie对象，一种方式是保存在客户端内存中，称为临时Cookie，浏览器关闭后这个Cookie对象将消失。

另外一种方式是保存在客户机的磁盘上，称为永久Cookie。以后客户端只要访问该网站，就会将这个Cookie再次发送到服务器上，前提是这个Cookie在

Cookie是可以被禁止的，当你打开Chrome，在设置里面关闭Cookie，那么你将再也无法登录淘宝页面。

- Session：在服务器端会创建一个session对象，产生一个sessionID来标识这个session对象，然后将这个sessionID放入到Cookie中发送到客户端，户。

每一个用户都有一个不同的session，各个用户之间是不能共享的，是每个用户所独享的，在session中可以存放信息。

Session的实现依赖于Cookie，如果Cookie被禁用，那么session也将失效。

### 持久连接：

我们知道HTTP协议采用“请求-应答”模式，当使用普通模式，即非Keep-Alive模式时，每个请求/应答客户和服务器都要新建一个连接，完成之后立即断开连接、连接重用)时，Keep-Alive功能使客户端到服务器端的连接持续有效，当出现对服务器的后继请求时，Keep-Alive功能避免了建立或者重新建立连接。

在HTTP 1.0版本中，并没有官方的标准来规定Keep-Alive如何工作，因此实际上它是被附加到HTTP 1.0协议上，如果客户端浏览器支持Keep-Alive，那么就带有Connection: Keep-Alive的请求时，它也会在响应头中添加一个同样的字段来使用Keep-Alive。这样一来，客户端和服务器之间的HTTP连接就会被保持，端发送另外一个请求时，就使用这条已经建立的连接。

在HTTP 1.1版本中，默认情况下所有连接都被保持，如果加入“Connection: close”才关闭。目前大部分浏览器都使用HTTP 1.1协议，也就是说默认都会发起看服务器设置情况。

由于HTTP 1.0没有官方的Keep-Alive规范，并且也已经基本被淘汰。

**HTTP Keep-Alive简单说就是保持当前的TCP连接，避免了重新建立连接。HTTP是一个无状态无连接的协议，那么这是不是与Keep-Alive冲突？**

- Keep-Alive与无连接的特性冲突，而对于无状态的特性两者并无矛盾，HTTP无状态无连接是在1.0版本中就规定的，而Keep-Alive则是在1.1版本中才
- 无连接的意思是限制每个连接只有一个请求的意思，在服务器处理完客户的请求，并收到客户的反应，即断开。通过这种方式可以节省传输时间。
- Keep-Alive确实破坏了这一特性，而无状态协议则意味着每个请求都是独立的，互不干扰的，互相没有记忆的。所以才需要有会话跟踪这种机制来识别用户

### 缓存机制：

HTTP条件GET是HTTP协议为了减少不必要的带宽浪费，提出的一种方案：

1. HTTP条件GET使用时机：客户端之前已经访问过某网站，并打算再次访问该站点。
2. HTTP条件GET使用的方法：客户端向服务器发送一个包询问是否在上一次访问网站的时间后是否更改了页面，如果服务器没有更新，显然不需要把整个网的时间已经更新了客户端请求的网页，则发送这个更新了的网页给用户。

下面是一个具体的发送接受报文示例：

```
GET / HTTP/1.1  
Host: www.sina.com.cn:80  
If-Modified-Since:Thu, 4 Feb 2010 20:39:13 GMT  
Connection: Close
```

第一次请求时，服务器端返回请求数据，之后的请求，服务器根据请求中的 If-Modified-Since 字段判断响应文件没有更新，如果没有更新，服务器返回一个 304 用已缓存的上次获取的文件。

```
HTTP/1.0 304 Not Modified  
Date: Thu, 04 Feb 2010 12:38:41 GMT  
Content-Type: text/html  
Expires: Thu, 04 Feb 2010 12:39:41 GMT  
Last-Modified: Thu, 04 Feb 2010 12:29:04 GMT  
Age: 28  
X-Cache: HIT from sy32-21.sina.com.cn  
Connection: close
```

如果服务器端资源已经更新的话，就返回正常的响应。

## 公钥加密算法

公开密钥的原理

### 非对称加密算法

1977年，三位数学家Rivest、Shamir 和 Adleman 设计了一种算法，可以实现非对称加密。  
只要有计算机网络的地方就会有RSA算法。

hash算法的特性：只要输入的数据有一点点的变化，生成的消息摘要就会发生巨变。

## 什么是协议

1 一式双份的。

2 双方都遵守的规范 双方达成共识。

Eg. ftp http pop tcp/ip。租房协议，离婚协议。

（你到楼下给我响一下手机我就下去了）你来问我来答，你怎么问，我怎么答。

只有通过协议，计算机才会知道我们想让他做什么。

只有通过网络协议，才能使一大片的机器相互协作，共同完成一件事情。

## 什么是公钥什么是私钥

密钥是完成加解密所必需的参数。

最初的加密算法是不需要密钥的。知道解密算法就能进行解密，这样一旦敌人知道了解密算法，敌人就也能解密了。

所以后来有个叫柯克霍夫的人提出使用密钥配合加解密算法进行保密。密钥其实是个参数，它可以影响加密算法的执行。所以同样的信息使用不同的密钥，会得到不同的加密结果。因此，在解密的时候，如果不知道正确的参数，就不能解密出正确的内容，保密的效果当然比不使用密钥更好。至于怎么建立密钥这个不好说，要看算法，不同的算法参数形式是不一样的。

作者：wanlan zhou

链接：<https://www.zhihu.com/question/277144227/answer/396956508>

来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

你只要想：既然是加密，那肯定是不希望别人知道我的消息，所以只有我才能解密，所以可得出公钥负责加密，私钥负责解密；同理，既然是签名，那肯定是不希望有人冒充我发消息，只有我才能发布这个签名，所以可得出私钥负责签名，公钥负责验证。

### ip地址和mac 地址的作用

1. mac地址是唯一的，为什么可以修改？想想身份证，身份证号是唯一的，不能改变的，但是可以造假。  
mac地址全球唯一，它是固化在网卡里的。网卡毕竟是个硬件，需要软件支持，既操作系统识别。  
重点来了，操作系统识别出来的mac地址是可以更改的，它只不过是一个字符串。我们常说的修改mac指的是修改电脑中记录的既注册表中的记录。
2. 有了mac地址为什么还要有ip地址。举个例子，身份证号是你的唯一标识，不会重复，一落户就有（网卡一出厂就有mac）。现在我要和你通信（写信给你），地址用你的姓名+身份证，信能送到你手上吗？明显不能！身份证号前六位能定位你出生的县。mac地址前几位也可以定位生产厂家。但是你出生后会离开这个县（哪怕在这个县，也不能具体找到你）。所以一般写个人信息就要有出生地和现居地址了。

### Http 和 https 的区别

- 1 url的开头不一样
- 2 http是不安全的 https 是安全的
- 3 http 的标准端口是 80 https 的标准端口是443
- 4 http 是应用层的协议。Https 的安全传输机制工作在传输层
- 5 http 是无法加密的。Https对传输的数据进行加密。
- 6 https需要ca 机构颁发的SSL证书。

### 什么是无状态协议？怎么解决http协议无状态

- 1 无状态指的是对事物没有记忆的能力。缺少状态就意味着如果后续的处理需要前面的信息。  
(也就是说，当前客户端。一次http请求完成以后，客户端再发送一次http请求，http缤不知道当前客户端是一个老用户)
- 2 可以使用cookie 来解决无状态的问题，cookie相当于一个通行证，第一次访问的时候给客户端发送一个Cookie，当客户端再次来的时候，拿着Cookie(通行证)，那么服务器就知道这个是“老用户”。

### CURL 命令详解

Command line url viewer

- 1 查看网页源码  
Curl [www.baidu.com](http://www.baidu.com)
- Curl -L [www.sina.com](http://www.sina.com). 自动跳转到。www.sina.com.cn
- Curl - I [www.sina.com](http://www.sina.com). 显示http response 请求的头部信息
- Curl -v [www.sina.com](http://www.sina.com) 显示通信过程

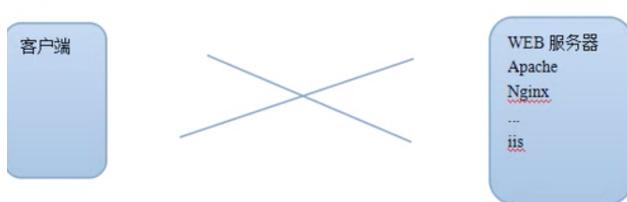
### HTTP 请求响应的过程

- 1 客户端连接到web服务器
- 2 发送http 请求
- 3 服务器接收并返回HTTP响应
- 4 释放连接 tcp 连接
- 5 客户端没事html 的内容。

### HTTP 协议的应用场景

1

### http协议的工作流程



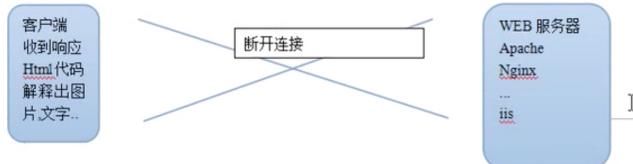
0: 原始状态 客户端和服务器之间没有任何的关系。



1 什么叫链接？就是网络上的虚拟电路



## 2 接受返回的消息



## 3 断开链接

### 详解

"field": "杭州市",

Console Sources Network Performance Memory Application Security Audits

View: Group by frame Preserve log Disable cache Offline No throttling

Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

200 ms 300 ms 400 ms 500 ms 600 ms 700 ms 800 n

x Headers Preview Response Timing

General

Request URL: http://localhost:9000/api/v1/district/1234/domestic/tourists/coming/xiaocheng  
Request Method: GET  
Status Code: 200 OK (from disk cache)  
Remote Address: [::1]:9000  
Referrer Policy: no-referrer-when-downgrade

Response Headers

Access-Control-Allow-Origin: \*  
Content-Encoding: gzip  
Content-Length: 159  
Content-Type: application/json; charset=UTF-8  
Date: Sun, 12 May 2019 04:45:39 GMT  
Vary: Accept-Encoding

Request Headers

Http 请求信息和响应信息的格式

### 请求

#### 1 请求行

请求方法, 请求路径, 所用协议

请求方法: GET POST PUT DELETE TRACE OPTIONS

#### 2 请求头信息

#### 3 请求主体信息

```

C:\Windows\system32\cmd.exe
GET /0606/01.php HTTP/1.1      请求行
Host: localhost                请求头部信息
HTTP/1.1 200 OK
Date: Thu, 06 Jun 2013 12:39:02 GMT
Server: Apache/2.2.21 (Win32) PHP/5.3.8
X-Powered-By: PHP/5.3.8
Content-Length: 5
Content-Type: text/html

hello
  
```

1: GET 就是请求方法 method

2: /0606/01.php 请求的资源

3: HTTP/1.1 请求所用的协议版本(1.0,0.9基本没人用了)

**注意:头信息结束后,有一个空行.**

头信息和主体信息(如果有),需要这个  
空行做区分.

即使没有主体信息,空行也不能少.

```
"": "2.2"}], "zone": "330102000000", "time": "2019-05-12T12:45:09.545+0800"}]}  
wangrui@192 ~ ➜ curl -v http://localhost:9000/api/v1/district/1234/domestic/tourists/coming/xiacheng  
* Trying ::1...  
* TCP_NODELAY set  
* Connected to localhost (::1) port 9000 (#0)  
> GET /api/v1/district/1234/domestic/tourists/coming/xiacheng HTTP/1.1  
> Host: localhost:9000  
> User-Agent: curl/7.54.0  
> Accept: */*  
>  
< HTTP/1.1 200 OK  
< Access-Control-Allow-Origin: *  
< Content-Length: 252  
< Content-Type: application/json; charset=UTF-8  
< Date: Sun, 12 May 2019 05:04:45 GMT  
<  
* Connection #0 to host localhost left intact  
[{"data": [{"field": "杭州市", "value": "5.8"}, {"field": "温州市", "value": "2.6"}, {"field": "上海市", "value": "2.6"}, {"field": "金华市", "value": "2.3"}, {"field": "": "2.2"}], "zone": "330102000000", "time": "2019-05-12T12:45:09.545+0800"}]}  
wangrui@192 ~ ➜
```

请求所用的一般是