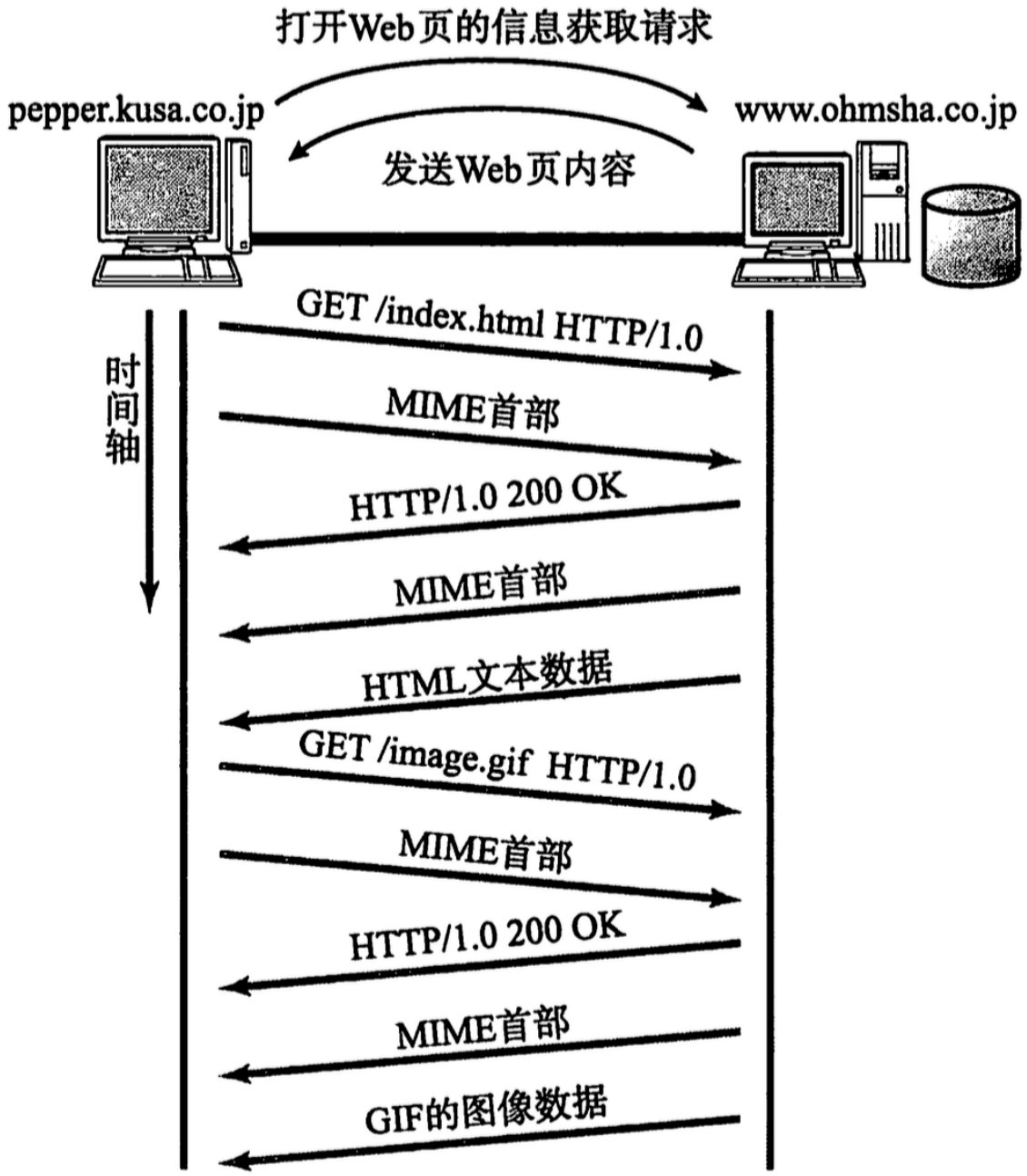
[面试·网络] TCP/IP（六）：HTTP 与 HTTPS 简介

本文是准备面试过程中网络部分总结整理的最后一篇文章，主要介绍以下知识：

- HTTP 协议概述
- POST 请求和 GET 请求
- Cookie 和 Session
- 数据传输时的加密
- HTTPS 简介

HTTP 协议

在 OSI 七层模型中，HTTP 协议位于最顶层的应用层中。通过浏览器访问网页就直接使用了 HTTP 协议。使用 HTTP 协议时，客户端首先与服务端的 80 端口建立一个 TCP 连接，然后在这个连接的基础上进行请求和应答，以及数据的交换。



HTTP 工作原理

HTTP 有两个常用版本，分别是 1.0 和 1.1。主要区别在于 HTTP 1.0 中每次请求和应答都会使用一个新的 TCP 连接，而从 HTTP 1.1 开始，运行在一个 TCP 连接上发送多个命令和应答。因此大幅度减少了 TCP 连接的建立和断开，提高了效率。

由 HTTP 协议加载出来的网页，通常使用 HTML 语言来描述，因此 HTML 也可以理解为网页的一种数据格式。HTML 是一段纯文本，可以指定网页中的文字、图像、音频视

频图片、链接，以及它们的颜色、位置等。无论计算机的底层结构如何，也无论网络底层使用了哪些协议，使用 HTML 展示出来的效果基本上是一致的。从这个角度来说 HTML 位于 OSI 七层模型的表现层。

POST 请求和 GET 请求

HTTP 有八种请求（也称方法），其中最常见的是 GET 请求和 POST 请求。

GET 请求通常用于查询、获取数据，而 POST 请求则用于发送数据，除了用途上的区别，它们还有以下这些不同：

1. GET 请求可以被缓存，可以被收藏为书签，但 POST 不行。
2. GET 请求会保留在浏览器的历史记录中，POST 不会。
3. GET 请求的长度有限制（不同的浏览器不一样，大约在几 Kb 左右），URL 的数据类型只能是 ASCII 字符，POST 请求没有限制。
4. 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。

注意：

> POST 请求仅比 GET 请求略安全一点，它的数据不在 URL 中，但依然以明文的形式存放于 HTTP 的请求头中。

Cookie 和 Session

HTTP 是一种无状态的连接，客户端每次读取 web 页面时，服务器都会认为这是一次新的会话。但有时候我们又需要持久保持某些信息，比如登录时的用户名、密码，用户上一次连接时的信息等。这些信息就由 Cookie 和 Session 保存。

这两者的根本性区别在于，cookie 保存在客户端上，而 session 则保存在服务器中。由此我们还可以拓展出以下结论：

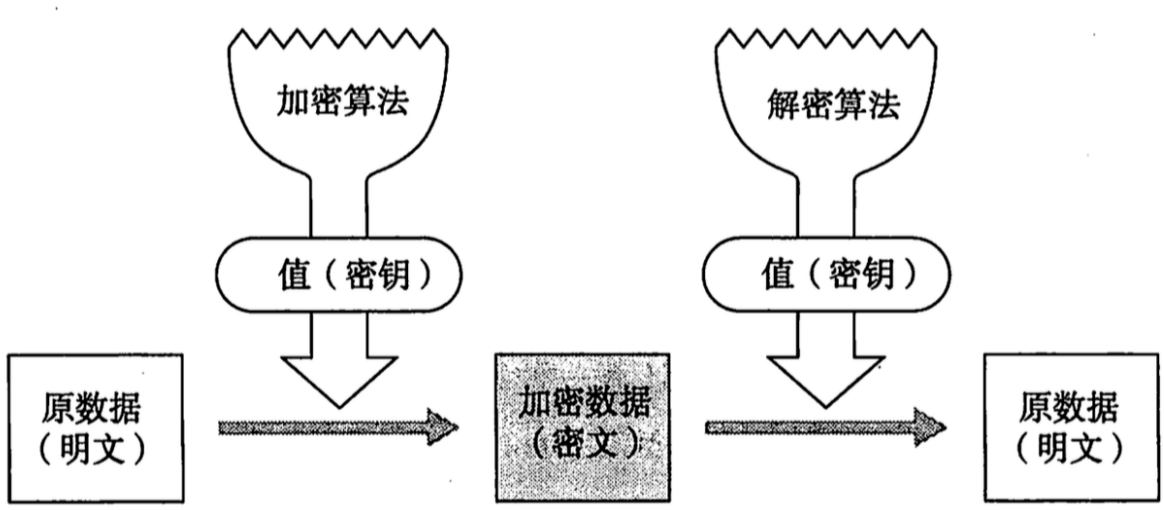
1. cookie 相对来说不安全，浏览器可以分析本地的 cookie 进行 cookie 欺骗。
2. session 可以设置超时时间，超过这个时间后就失效，以免长期占用服务端内存。

3. 单个 cookie 的大小有限制（4 Kb），每个站点的 cookie 数量一般也有限制（20 个）。
4. 客户端每次都会把 cookie 发送到服务端，因此服务端可以知道 cookie，但是客户端不知道 session。

当服务器接收到 cookie 后，会根据 cookie 中的 SessionID 来找到这个客户的 session。如果没有，则会生成一个新的 SessionID 发送给客户端。

加密

加密分为两种，对称加密和非对称加密。在解释这两者的含义前，先来看一下简单的加密、解密过程：



加密和解密过程

所谓的对称，就是指加密秘钥和解密秘钥相同，而非对称自然就是指两者不同。

举一个对称加密的例子。假设这里的加密算法是加法，解密算法是减法。如果明文数据是 10，秘钥是 1，那么加密数据就是 `10 + 1 = 11`，如果接收方不知道秘钥，就不知道密文 11 应该减去几。反之，如果接收方知道秘钥是 1，就可以通过 `11 - 1 = 10` 计算出明文数据。

常见的一个非对称加密算法是 RSA 算法，它主要利用了“两个素数求乘积容易，但是将乘积分解为两个素数很难”这一思想。它的具体原理不在本文讨论范围，有兴趣的读者可

以查看文章末尾的参考文章。

在非对称加密中，利用公钥加密的数据能且只能通过私钥解密，通过私钥加密的数据能且只能通过公钥解密。

对称加密的优点在于速度快，但是假设秘钥由服务器保存，如何安全的让客户端得到秘钥是需要解决的问题。因此实际的网络传输中，通常使用对称加密与非对称加密结合的方式，服务端通过非对称加密将对称秘钥发给客户端。此后双方使用这个对称密钥进行通信。

HTTPS

我们知道 HTTP 协议直接使用了 TCP 协议进行数据传输。由于数据没有加密，都是直接明文传输，所以存在以下三个风险：

1．窃听风险：第三方节点可以获知通信内容。

2．篡改风险：第三方节点可以修改通信内容。

3．冒充风险：第三方节点可以冒充他人身份参与通信。

比如你在手机上打开应用内的网页时，有时会看到网页底部弹出了广告，这实际上就说明你的 HTTP 内容被窃听、并篡改了。

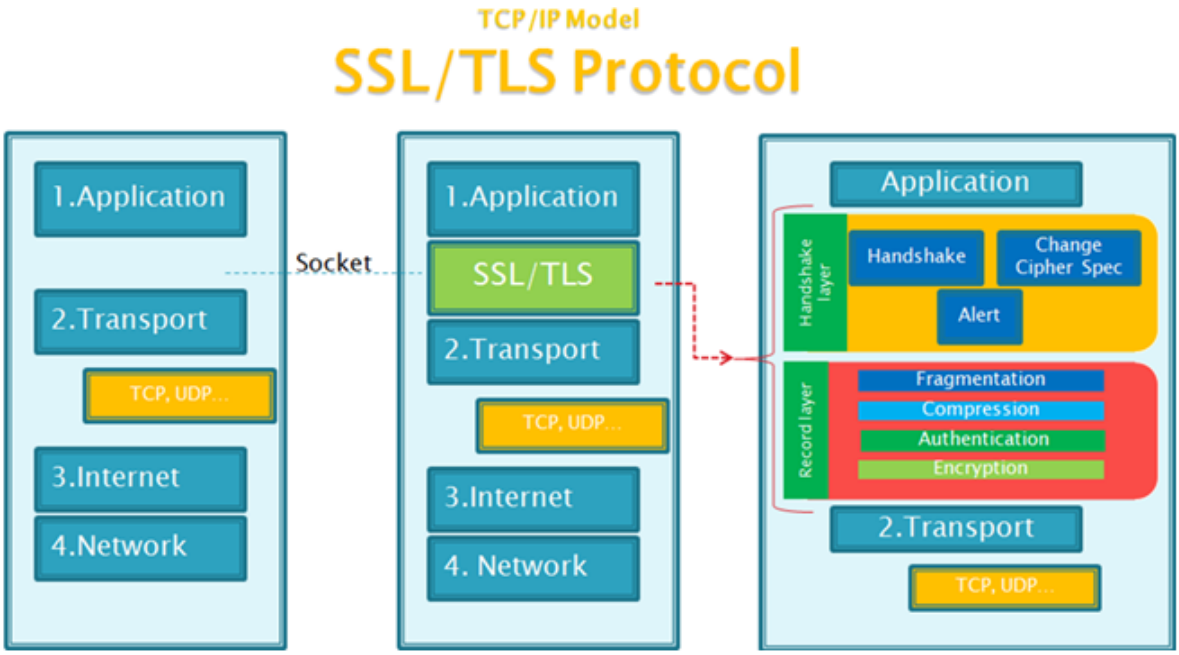
HTTPS 协议旨在解决以上三个风险，因此它可以：

1．保证所有信息加密传输，无法被第三方窃取。

2．为信息添加校验机制，如果被第三方恶意破坏，可以检测出来。

3．配备身份证书，防止第三方伪装参与通信。

HTTPS 的结构如图所示：



HTTPS 协议

可见它仅仅是在 HTTP 和 TCP 之间新增了一个 TLS/SSL 加密层，这也印证了一句名言：“一切计算机问题都可以通过添加中间层解决”。

使用 HTTPS 时，服务端会将自己的证书发送给客户端，其中包含了服务端的公钥。基于非对称加密的传输过程如下：

1. 客户端使用公钥将信息加密，密文发送给服务端
2. 服务端用自己的私钥解密，再将返回数据用私钥加密发回客户端
3. 客户端用公钥解密

这里的证书是服务器证明自己身份的工具，它由权威的证书颁发机构（CA）发给申请者。如果证书是虚假的，或者是自己给自己颁发的证书，服务器就会不认可这个证书并发出警告：

12306 的自签名证书

总结一下 HTTPS 协议是如何避免前文所说的三大风险的：

1. 先用非对称加密传输密码，然后用这个密码对称加密数据，使得第三方无法获得通信内容
2. 发送方将数据的哈希结果写到数据中，接收方解密后对比数据的哈希结果，如果不一致则说明被修改。由于传输数据加密，第三方无法修改哈希结果。
3. 由权威机构颁发证书，再加上证书校验机制，避免第三方伪装参与通信。

参考文章

1. HTTPS科普扫盲帖
2. SSL/TLS协议运行机制的概述
3. RSA 加密
4. HTTP 方法：GET 对比 POST