Linux网络

1 网络各层的基础

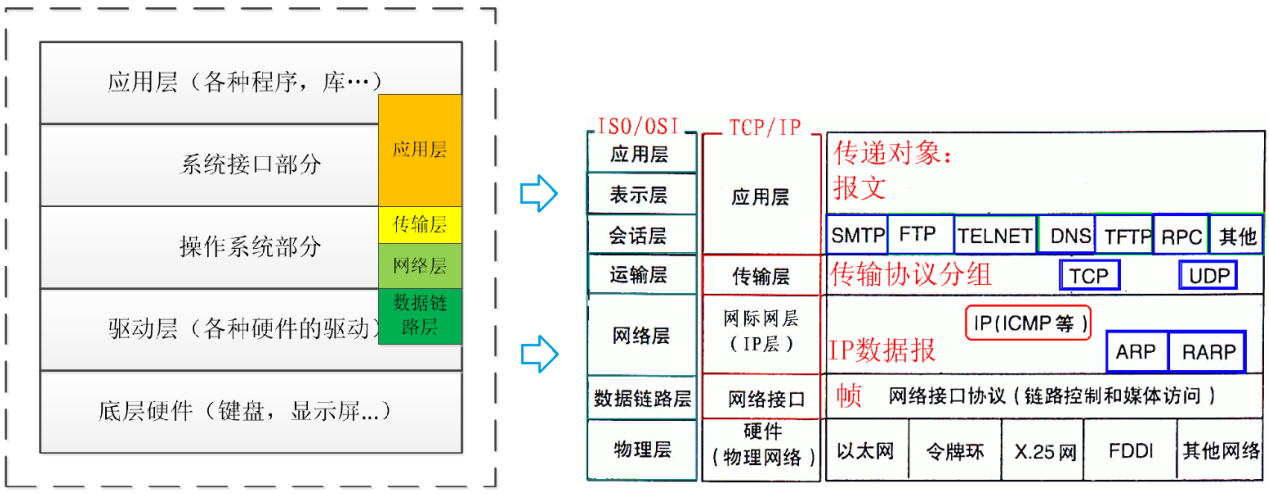
1.1 网络通信的5层传输模型

1. **物理层**：主要用来决定最大的传输速率，传输距离，抗干扰性
2. **数据链路层**：负责设备之间的数据帧传送与识别
3. **网络层**：负责地址管理与路由的选择**(IP协议)**
4. **传输层**：负责两台主机之间的数据传输**(TCP/UDP协议)**
5. **应用层**：负责应用程序间的沟通

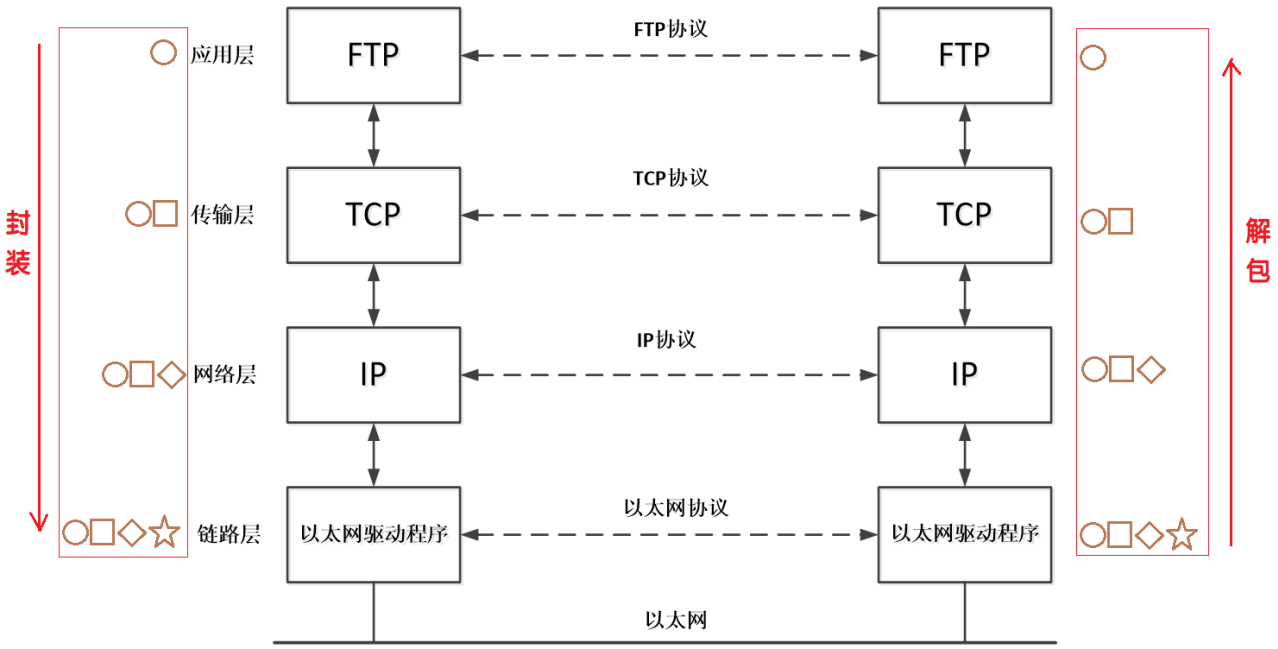
1.2 数据包在不同层之间的称号

数据链路层 (数据帧)	网络层 (数据报)	传输层 (数据段)

1.3 网络各层在OS里面的相对位置



1.4 网络各层的封装与解包



1.每一层都有其独特的报头，作为相同层之间进行联系

2.如何理解报头：需要报头当中的数据来指导当前层进行某种协议决策

3.每一层的报头都包含有如下的两种字段

3.1当前报文的有效载荷要交付给上层的哪一个协议

3.2几乎每一个报头都明确了报头和有效载荷的边界

1.5 数据包在网络里面传输，难道是传输到目的地就可以了吗?

如何解析 VS 如何使用

1.6 IP地址与MAC地址

- IP地址：数据在网络当中进行传输的过程当中，起始地址与终点地址(唐僧西行当中的 东土大唐->西天)
- MAC地址：数据在网络当中进行传输的过程当中，行驶途中的具体某个地址(唐僧西行过程当中 女儿国->车迟国)

1.7 IP地址与端口号

- IP地址：可以唯一标识公网当中的一台主机
- Port端口号：可以唯一标识一台主机当中的唯一个进程
- 网络通信：将互联网看成是一个大的OS，网络通信其实就是两个进程之间的通信，这两个进程之间看到的同一份资源就是网络

2 网络层http/https协议

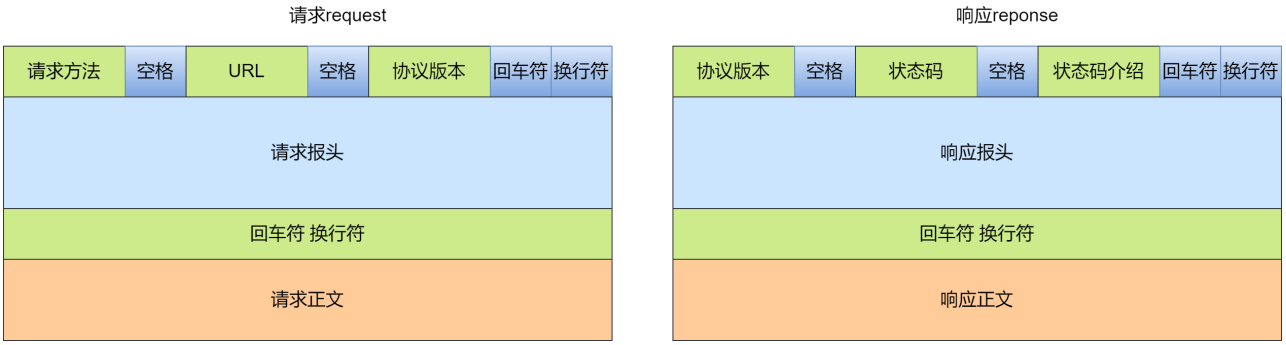
2.1 前言

- 数据由多变一的过程叫做序列化。数据由一变多的过程叫反序列化。
- 短连接：客户端发起请求，服务器处理后，发送响应，服务器将连接断开。一个服务周期结束。
- 长连接：客户端和服务器长时间连接

2.2 http协议的基本特征

- 无连接：http协议在客户端和服务器之间没有连接。http协议是无连接的，但是它是建立在TCP协议上的，TCP协议是有连接的。
- 无状态：并不会记录用户的任何信息。
- 简单快速：短连接，文本传输。

2.3 http的请求和响应报头



1. 请求方法

Get方法和Post方法：Get和Post请求方法都可以传参(用户的账号密码信息)，Get不带正文只能通过URL进行传参，Post有正文，可以通过正文进行传参。

2. 版本协议

http1.0：短链接 http1.1：长连接

3. 状态码

- 1XX informational(信息状态码) 接受的请求正在处理
- 2XX success(成功状态码) 请求正常处理完毕
- 3XX redirection(重定向状态码) 需要进行附加操作已完成请求(比如:跳转到其它的网页)
- 4XX client error(客户错误状态码) 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(比如:创建进程或者线程失败)

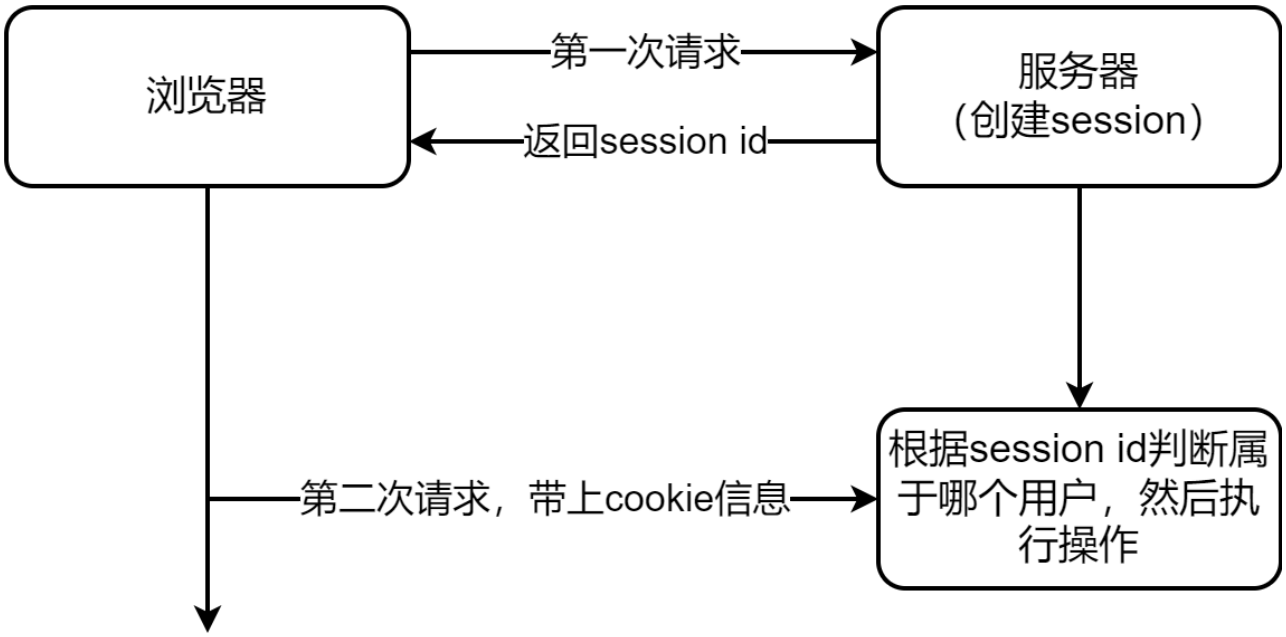
常见的状态码

200(OK)，404(NOT FOUND)，403(Forbidden)，301(永久重定向)，302(临时重定向)，307(临时重定向)，504(Bad Gateway)。

4. 请求/响应报头

- 无论是请求还是响应，基本.上http都是按照行(\n)为单位进行构建请求或者响应的
- Content-Type：数据类型(text/html等)。
- Content-Length：正文长度(字节)
- Host：客户端告知服务器，请求资源的主机号和端口号。
- User-Agent：声名用户的操作系统和浏览器版本信息。
- referer：当前页面从哪个页面跳转过来的。
- location：搭配3XX状态码使用，告诉客户端接下来要跳转到哪里访问。
- Cookie：同于再客户端存储少了信息，用于实现会话的功能。

Cookie和Session之间的区别



说起来为什么需要 Cookie ，浏览器是没有状态的(HTTP 协议无状态)，这意味着浏览器并不知道是谁在和服务端打交道。这个时候就需要有一个机制来告诉服务端，本次操作用户是否登录，是哪个用户在执行的操作，那这套机制的实现就需要 Cookie 和 Session 的配合：用户第一次请求服务器的时候，服务器根据用户提交的相关信息，创建创建对应的 Session ，请求返回时将此 Session 的唯一标识信息SessionID 返回给浏览器，浏览器接收到服务器返回的 SessionID 信息后，会将此信息存入到 Cookie 中，同时 Cookie 记录此 SessionID 属于哪个域名。

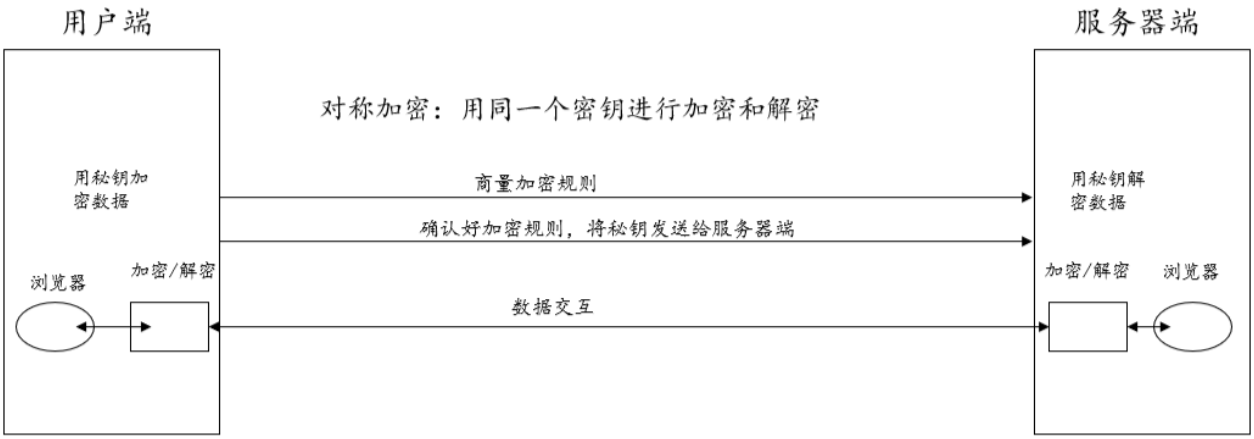
当用户第二次访问服务器的时候，请求会自动判断此域名下是否存在 Cookie 信息，如果存在自动将 Cookie 信息也发送给服务端，服务端会从 Cookie 中获取 SessionID，再根据 SessionID 查找对应的 Session 信息，如果没有找到说明用户没有登录或者登录失效，如果找到 Session 证明用户已经登录可执行后面操作。

SessionID 是连接 Cookie 和 Session 的一道桥梁，大部分系统也是根据此原理来验证用户登录状态。

2.5 http与https协议

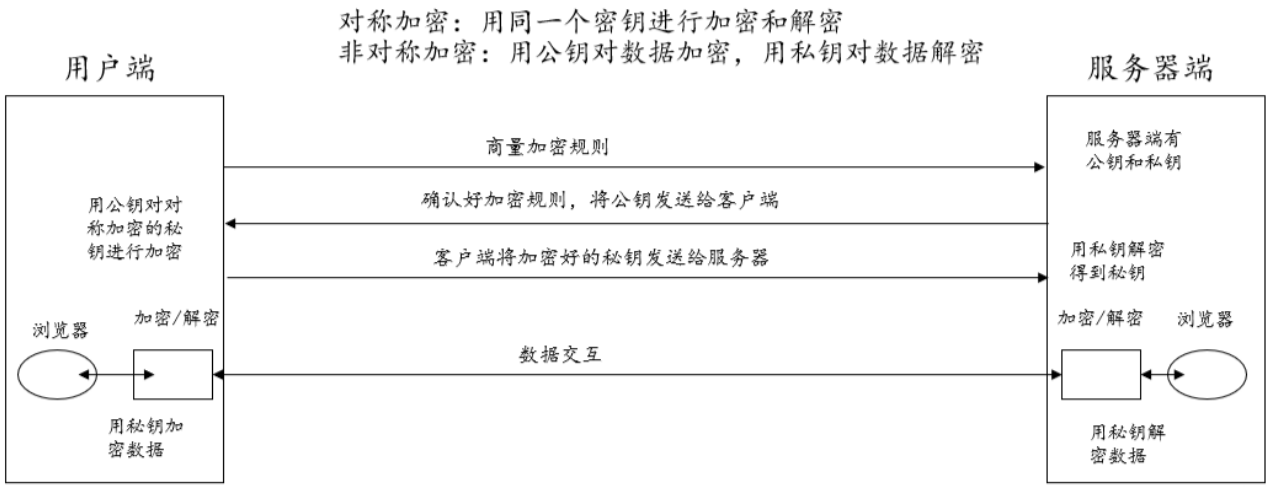
1. http的端口号是80 https的端口号是443

2.对称加密(公钥加密，公钥解密)

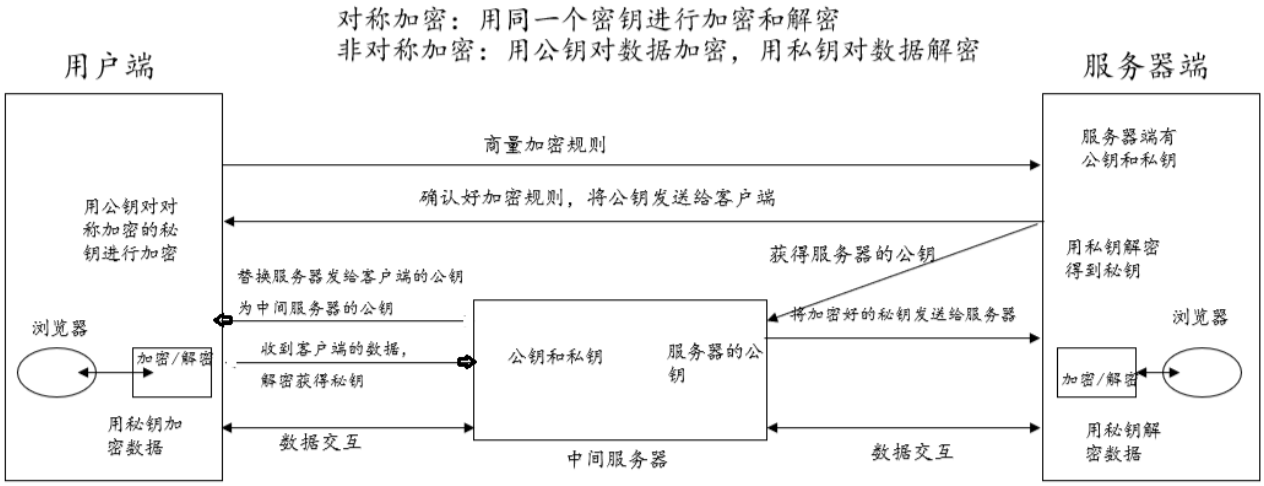


用户端和服务器端确定好加密规则后，客户端和服务器端都得知道秘钥，秘钥也会需要传输，秘钥也是数据，也要保证秘钥的安全。

3.非对称加密(公钥加密，私钥解密)



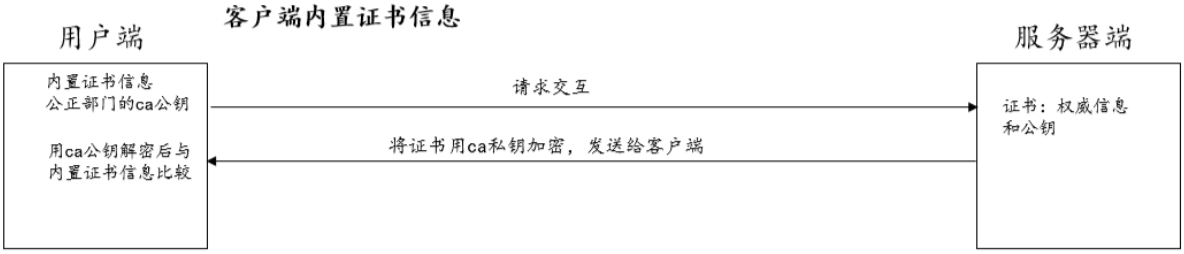
4.非对称加密的安全问题



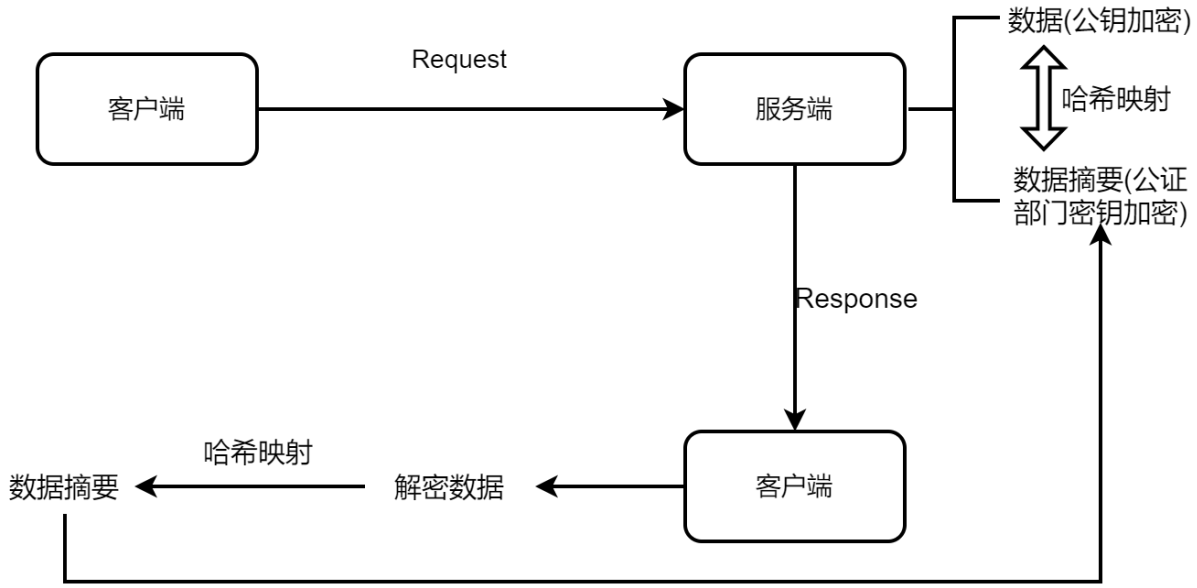
这样中间服务器也能获得客户端的秘钥，数据和服务器的响应。并且还能修改服务器的响应，再响应给客户端，导致客户端收到一个错误的响应。

5.非对称加密安全版

远端服务器认证问题 **服务器在使用前要向公正部门申请证书，证书包括权威信息和公钥**
客户端内置证书信息



证书的权威信息发送时用哈希算法映射成了数据摘要。这个权威信息中间服务器是没有的。
数据摘要进行加密了无法获取。



解析出来的数据摘要与服务端给的数据摘要是否一致，如果一致就表明数据没有被修改，否则就被修改