1、Http请求的8种方法



2、Http状态码



3、Https加密原理

http缺点：

1、通信使用明文（不加密），内容可能会被窃听

2、不验证通信方的身份，因此有可能遭遇伪装

3、无法证明报文的完整性，所以有可能已遭篡改

http优点：

1、传输速度快

**https**:http通信接口部分用**SSL(安全套接字层)和TLS(安全传输层协议)**代替，即添加了加密及认证机制的http

https=http+加密+认证+完整性保护

过程;

1、客户端发起HTTPS请求

2、服务端的配置，采用https协议的服务器必须要有一套数字证书，可以自己制作也可以向组织申请，区别就是自己颁发的证书需要客户端验证通过，才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面（如startssl）。这套证书说白了就是一对公钥和私钥。

3、传送证书，包括证书的颁发机构，过期时间等等。

4、客户端解析证书，这部分工作是由客户端的TLS来完成的，首先会验证公钥是否有效，如颁发机构，过期时间等等，如果发现异常，则会弹出一个警告，提示证书有问题，如果证书没问题，那么就生成一个随机值，然后利用证书对该随机值加密。

5、传送加密信息，这部分传送的是用证书加密后的随机值，目的就是让服务端得到这个随机值，以后客户端和服务器端通信就可以通过这个随机值来进行加密解密了。

6、服务端解密信息，服务端用私钥解密后，得到了客户端传送过来的随机值（私钥），然后把内容通过该值进行对称加密。

7、传输加密后的信息，这部分信息是服务器端用私钥加密后的信息，可以在客户端被还原。

8、客户端解密信息，客户端用自己的私钥解密服务器端传递过来的信息，获取解密后的内容。

**SSL的位置:** 安全套接字层

SSL介于应用层和TCP层之间，应用层数据不在直接传递给传输层，而是传递给SSL层，SSL层对从应用层收到的数据进行加密，并增加自己的SSL头。

用到的非对称加密：RSA 性能比较低,因为在寻找大素数、大数计算、数据分割需要耗费很多CPU周期，所以一般的HTTPS连接只在第一次握手时使用非对称加密，通过握手交换对称加密密钥，之后的数据通信走对称加密。

TLS/SSL中使用了非对称加密，对称加密以及HASH算法。

HTTPS一般使用的加密与HASH算法如下：

非对称加密算法：RSA，DSA/DSS

对称加密算法：AES，RC4，3DES

HASH算法：MD5，SHA1，SHA256

**使用https（自有数字证书的生成）**

用openssl（命令行）工具生成自签发的数字证书, HTTPS通信所用的数字证书格式为X.509。

步骤如下：

Step1、生成自己的CA根证书

1、生成CA私钥文件ca.key

**2、**生成x.509证书签名请求文件ca.csr，在这个过程中，会让输入一些组织信息。

3、生成x.509格式的CA根证书ca\_public.crt（公钥证书）

**Step2 生成服务端证书**

**1、**先生成服务器私钥文件server\_private.key：

2、根据服务器私钥生成服务器公钥文件server\_public.pem：

3、服务器端需要向CA机构申请签名证书

对于https的csr证书签名请求文件，common name必须和网站域名一致，以便之后进行Host Name校验，服务器端用server.csr文件向CA申请公钥证书，最终颁发一个带有CA签名的服务器端证书server.crt

如果服务器端还想校验客户端证书，可以按生成服务器端证书的形式生成客户端证书。

**Android 端使用HttpsURLConnection进行HTTPS通信：**

**直接使用HttpsURLConnection会有以下问题：**

1、由Android系统校验服务端数字证书的合法性，用可信CA签发的数字证书的网站才可以正常访问，私有CA签发的数字证书的网站无法访问。

2、不能抵御在用户设备上安装证书（将中间人服务器的证书放到设备的信任列表中）进行中间人攻击

如果要使用私有CA签发的证书，必须重写校验证书链TrustManager中的方法，否则会出现

javax.net.ssl.SSLHandshakeException: java.security.cert.CertPathValidatorException:

在重写TrustManger中的checkServerTrusted()过程中，很多人什么也么有做，会导致证书弱校验（其实就是没有真正校验证书），因为默认是信任所有主机，未真正实现校验服务器证书域名是否符合。正确的写法，对服务器证书域名进行强校验或者真正实现HostnameVerifier的verify()方法

**使用OKHttp3.0进行HTTPS通信**

还可以使用第三方库,以OKhttp3.0为例，同理也是要先校验服务器端证书链，并校验服务端证书域名

**Webview的HTTPS安全**

目前很多应用都用webview加载H5页面，如果服务端采用的是可信的CA颁发的证书，在webView.setWebViewClient（webviewClient）时重载WebViewClient的onReceivedSslError（）,如果出现证书错误，直接调用handler.proceed()会忽略错误继续加载证书有问题的页面，如果调用handler.cancel()可终止加载证书有问题的页面。

当然，如果webview加载https需要强校验服务端证书，可以在onPageStarted()中用HttpsURLConnection强校验证书的方式来校验服务端证书，如果校验不通过则停止加载网页。但是这样做会拖慢网页的加载速度。

在TSL握手和数据传输的不同阶段会采用相应的算法：

服务端身份验证：数字签名（RSA，ECDSA）

密钥交换：RSA、密钥交换算法（ECDH）

加密/解密：流加密(RC4)和分组加密(3DES,AES)

生成消息认证码:SHA

4、http的结构

请求报文和响应报文都是由以下4个部分组成

1、请求行2、请求头 3、空行 4、消息主体

**请求行：**

格式为：Method Request-URI HTTP-Version，结尾符一般用\r\n

**请求头:**

**通用报头:**既可以出现在请求报头，也可以出现在响应报头中。

Date：表示消息产生的日期和时间

Connection：允许发送指定连接选项

Cache-Control：用于缓存指令。

**请求报头：**

Host：请求的主机名

User-Agent：发送请求的浏览器类型、操作系统等信息

Accept：指定客户端接收哪些类型的信息

Accept-Encoding：客户端可识别的数据编码

Accept-Language：表示浏览器所支持的语言类型

Connection：允许客户端和服务器指定与请求/响应连接有关的选项

Transfer-Encoding：告知接收端为了保证报文的可靠传输，对报文采用了什么编码方式。

**响应报头：**

Location：用于重定向接受者到一个新的位置，常用在更换域名的时候

Server：包含可服务器用来处理请求的系统信息，与User-Agent请求报头是相对应的

**实体报头：**

用来定义被传送资源的信息，既可以用于请求也可以用于响应，请求和响应都可以传送一个实体。

Content-Type：发送给接收者的实体正文的媒体类型

Content-Lenght：实体正文的长度

Content-Language：描述资源所用的自然语言

Last-Modified：实体报头用于指示资源的最后修改日期和时间

**空行：**

Http协议规定的格式，一般用\r\n

**消息主体：**

multipart/form-data

1、该格式是post常见的提交方式，由post方法来组合实现

2、使用该提交方法需要规定一个内容分隔符用于分割请求体中的多个post的内容，分割符般情况用一长串不会和业务数据重复的字符串表示

5、一次完整的HTTP请求过程

域名解析->发起TCP的三次握手->建立TCP连接后发起http请求->服务器响应http请求，浏览器获取得到html代码->浏览器解析html代码，并请求html代码中的资源->浏览器渲染页面。

域名解析->浏览器DNS缓存->操作系统DNS缓存->查找hosts文件->DNS系统调用（根据配置的DNS地址查询）

6、三次握手四次挥手

三次握手的目的同步连接双方的序列号和确认号，并交换TCP窗口信息

1、为什么连接的时候是三次握手，关闭的时候却是四次握手？

主要原因是因为TCP是半关闭的，而TCP连接是全双工的，当关闭连接时，不是说客户端发出关闭请求就关闭的，Server端接收到FIN报文，Socket不会立马关闭，因为Server端的报文很可能没发送完，只能先ACK,然后等发送结束，再主动发送FIN

2、为什么不能用两次握手进行连接？

三次握手的作用就是，既要让双发都做好发送数据的准备工作，也要允许双方就序列号进行协商，这个序列号在握手过程中被发送和确认。两次握手可能发生死锁，server的ACK丢失，这时客户端一直等待，形成死锁。