HTTPS

@M了个J
李明杰

https://github.com/CoderMJLee

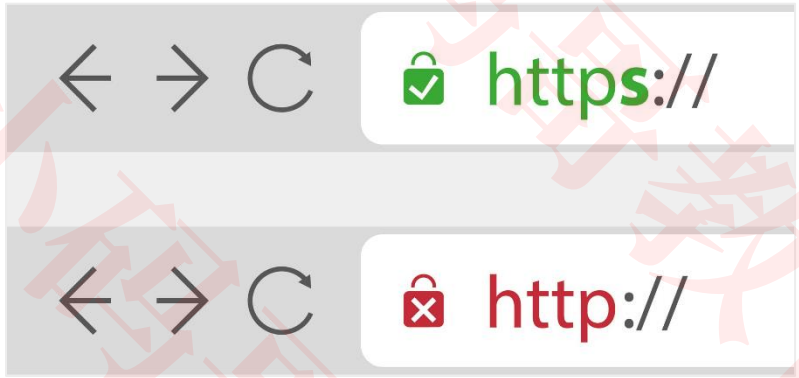
https://space.bilibili.com/325538782

小码哥教育 SEEMYGO

实力IT教育 www.520it.com

HTTPS

- HTTPS（**H**yper**T**ext **T**ransfer **P**rotocol **S**ecure），译为：超文本传输安全协议
  - 常称为HTTP over TLS、HTTP over SSL、HTTP Secure
  - 由网景公司于1994年首次提出

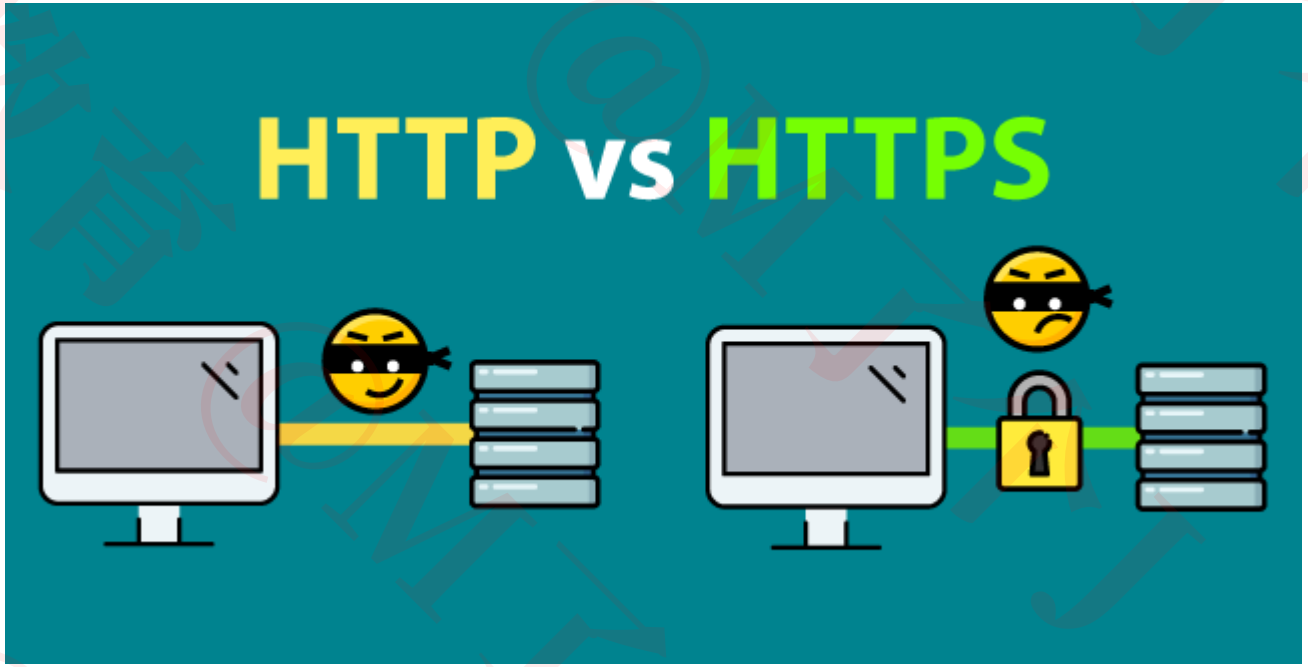


- HTTPS的默认端口号是443（HTTP是80）

- 现在在浏览器上输入http://www.baidu.com
  - 会自动重定向到https://www.baidu.com

SSL/TLS

- HTTPS是在HTTP的基础上使用SSL/TLS来加密报文，对窃听和中间人攻击提供合理的防护



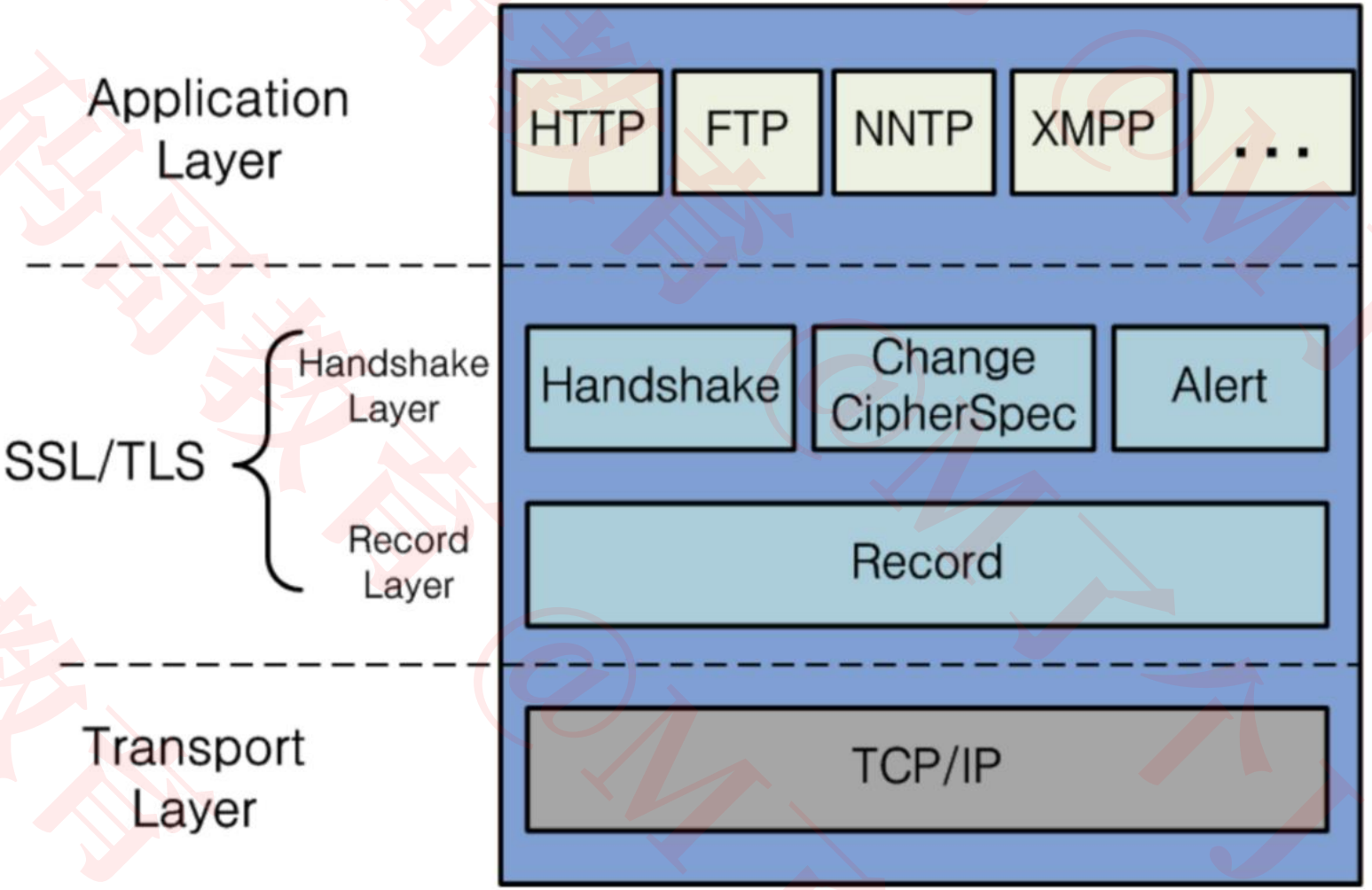
- SSL/TLS也可以用在其他协议上，比如
  - FTP → FTPS
  - SMTP → SMTPS

SSL/TLS

- TLS（**T**ransport **L**ayer **S**ecurity），译为：传输层安全性协议
  - 前身是SSL（**S**ecure **S**ockets **L**ayer），译为：安全套接层

- 历史版本信息
  - SSL 1.0：因存在严重的安全漏洞，从未公开过
  - SSL 2.0：1995年，已于2011年弃用（RFC 6176）
  - SSL 3.0：1996年，已于2015年弃用（RFC 7568）
  - TLS 1.0：1999年（RFC 2246）
  - TLS 1.1：2006年（RFC 4346）
  - TLS 1.2：2008年（RFC 5246）
  - TLS 1.3：2018年（RFC 8446）
  - ✓有没有发现：TLS的RFC文档编号都是以46结尾

SSL/TLS – 工作在哪一层



OpenSSL

- OpenSSL是SSL/TLS协议的开源实现，始于1998年，支持Windows、Mac、Linux等平台
  - Linux、Mac一般自带OpenSSL
  - Windows下载安装OpenSSL：https://slproweb.com/products/Win32OpenSSL.html

- 常用命令
  - 生成私钥：openssl genrsa -out mj.key
  - 生成公钥：openssl rsa -in mj.key -pubout -out mj.pem

- 可以使用OpenSSL构建一套属于自己的CA，自己给自己颁发证书，称为“自签名证书”

HTTPS的成本

- 证书的费用
- 加解密计算
- 降低了访问速度
- 有些企业的做法是：包含敏感数据的请求才使用HTTPS，其他保持使用HTTP
  - http://www.icbc.com.cn/
  - https://mybank.icbc.com.cn/

HTTPS的通信过程

- 总的可以分为3大阶段

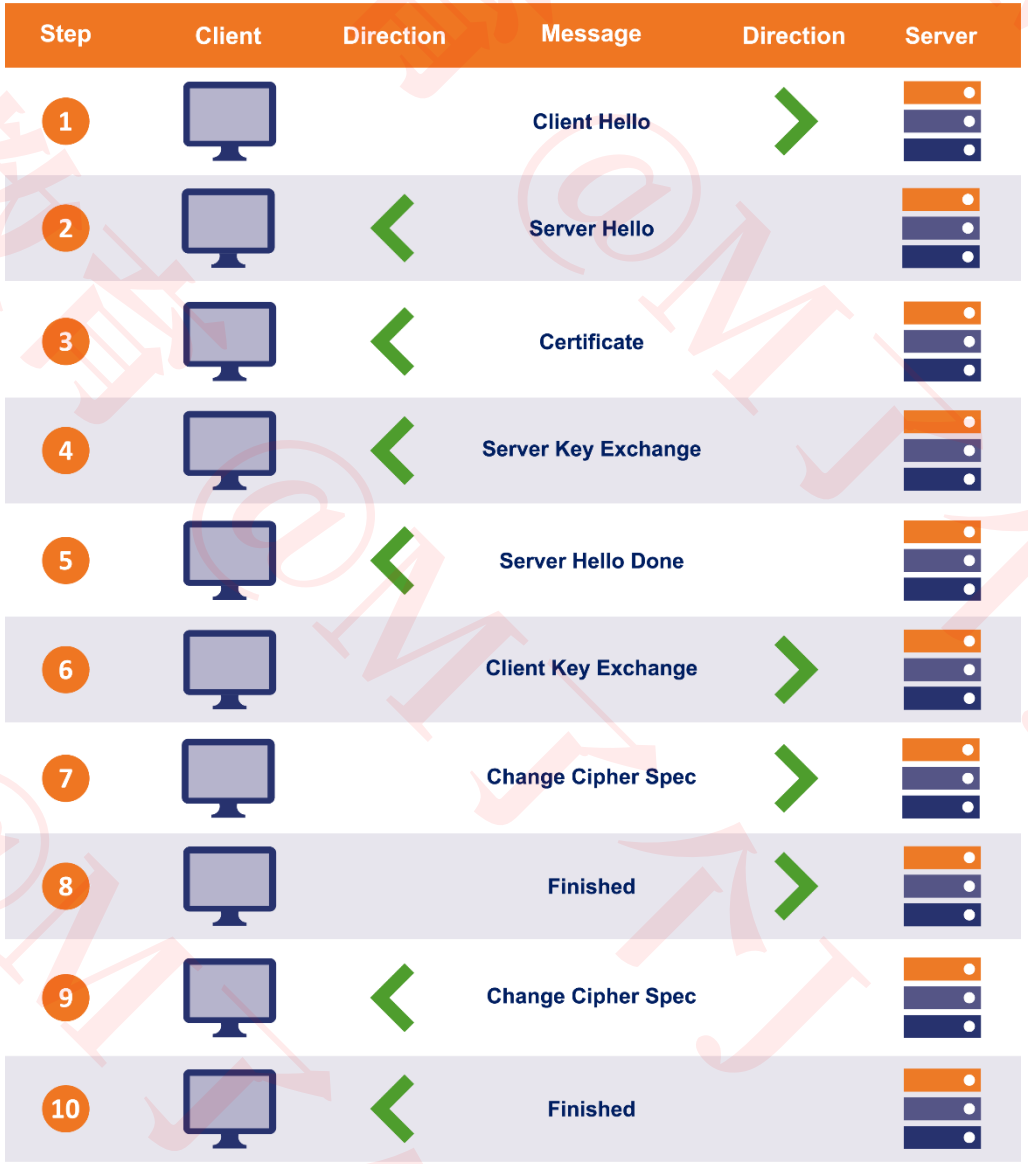
① TCP的3次握手

② TLS的连接

③ HTTP请求和响应

TLS 1.2的连接

- 大概是有10大步骤
- 图片中省略了中间产生的一些ACK确认



TLS 1.2的连接 – ①

① Client Hello

- TLS的版本号
- 支持的加密组件（Cipher Suite）列表
  - ✓加密组件是指所使用的加密算法及密钥长度等
- 一个随机数（Client Random）

```
Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 184
    Version: TLS 1.2 (0x0303)
  > Random: 5feaf4e531379dd15436b0251fe90cbd0c9fb9cfe9f32bb3e118673355757f8e
    Session ID Length: 0
    Cipher Suites Length: 42
  v Cipher Suites (21 suites)
      Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
      Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
      Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
      Cipher Suite: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
```

TLS 1.2的连接 – ②

② Server Hello

- TLS的版本号
- 选择的加密组件

✓是从接收到的客户端加密组件列表中挑选出来的

- 一个随机数（Server Random）

```
Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 59
    Version: TLS 1.2 (0x0303)
  > Random: 5feaf4e6ad10a031ac930f6a7ab480b02681a5e78e4980706ab6d491790d7aa1
    Session ID Length: 0
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
```

TLS 1.2的连接 – ③

③ Certificate

- 服务器的公钥证书（被CA签名过的）

```
Handshake Protocol: Certificate
    Handshake Type: Certificate (11)
    Length: 4711
    Certificates Length: 4708
  Certificates (4708 bytes)
      Certificate Length: 2399
    > Certificate: 3082095b30820843a003020102021008 34b97cec2a5babdfb2dc582a160f3a300d06092a…
      Certificate Length: 1176
    > Certificate: 308204943082037ca003020102021001fda3eb6eca75c888438b724bcfbc91300d06092a…
      Certificate Length: 1124
    > Certificate: 3082046030820348a003020102021 00f5bc3a176cb789e2020c7893c8167b4300d06092a…
```

TLS 1.2的连接 – ④

④ Server Key Exchange

- 用以实现ECDHE算法的其中一个参数（Server Params）
  - ✓ ECDHE是一种密钥交换算法
  - ✓ 为了防止伪造，Server Params经过了服务器私钥签名

```
▼ Handshake Protocol: Server Key Exchange
    Handshake Type: Server Key Exchange (12)
    Length: 329
  ▼ EC Diffie-Hellman Server Params
      Curve Type: named_curve (0x03)
      Named Curve: secp256r1 (0x0017)
      Pubkey Length: 65
      Pubkey: 04bbddd608c2d4b6bdbb09ddf17f40769574a2626a20387aa52db164dcbe8397fa59fca2…
    ▶ Signature Algorithm: rsa_pkcs1_sha256 (0x0401)
      Signature Length: 256
      Signature: 2c5659580b5aa5f055c4e7c146ed78318ef9d9d5944c6196cfa61fb08b393d62b1cc30a2…
```

TLS 1.2的连接 – ⑤

⑤ Server Hello Done

- 告知客户端：协商部分结束

```
▾ Handshake Protocol: Server Hello Done
    Handshake Type: Server Hello Done (14)
    Length: 0
```

- 目前为止，客户端和服务器之间通过明文共享了
- Client Random、Server Random、Server Params

- 而且，客户端也已经拿到了服务器的公钥证书，接下来，客户端会验证证书的真实有效性

TLS 1.2的连接 – ⑥

⑥ Client Key Exchange

- 用以实现ECDHE算法的另一个参数（Client Params）

```
Handshake Protocol: Client Key Exchange
    Handshake Type: Client Key Exchange (16)
    Length: 66
  EC Diffie-Hellman Client Params
      Pubkey Length: 65
      Pubkey: 045009ee8fbf9c321412e43f71bf6de7fade98fa7917f4a09d871c0a641d86f7d8044118…
```

- 目前为止，客户端和服务器都拥有了ECDHE算法需要的2个参数：Server Params、Client Params
- 客户端、服务器都可以
- 使用ECDHE算法根据Server Params、Client Params计算出一个新的随机密钥串：Pre-master secret
- 然后结合Client Random、Server Random、Pre-master secret生成一个主密钥
- 最后利用主密钥衍生出其他密钥：客户端发送用的会话密钥、服务器发送用的会话密钥等

TLS 1.2的连接 – ⑦

⑦ Change Cipher Spec

- 告知服务器：之后的通信会采用计算出来的会话密钥进行加密

```
TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
```

TLS 1.2的连接 – ⑧

⑧ Finished

- 包含连接至今全部报文的整体校验值（摘要），加密之后发送给服务器
- 这次握手协商是否成功，要以服务器是否能够正确解密该报文作为判定标准

```
TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 40
    Handshake Protocol: Encrypted Handshake Message
```

TLS 1.2的连接 – ⑨、⑩

⑨ Change Cipher Spec

⑩ Finished

- 到此为止，客户端服务器都验证加密解密没问题，握手正式结束
- 后面开始传输加密的HTTP请求和响应

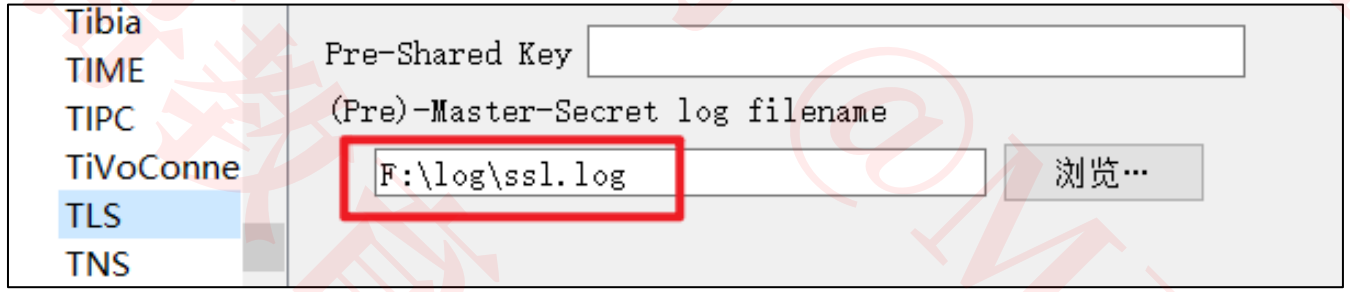
```
TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
```

```
TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 40
    Handshake Protocol: Encrypted Handshake Message
```

Wireshark解密HTTPS

- 设置环境变量SSLKEYLOGFILE（浏览器会将key信息导出到这个文件）

- 设置完成后，最好重启一下操作系统
- 在Wireshark中选择这个文件
  - 编辑 → 首选项 → Protocols → TLS



Wireshark解密HTTPS

■ 如果环境变量不管用，可以直接设置浏览器的启动参数（下图是使用了Rolan进行启动）

配置服务器HTTPS – 生成证书

- 环境：Tomcat9.0.34、JDK1.8.0_251

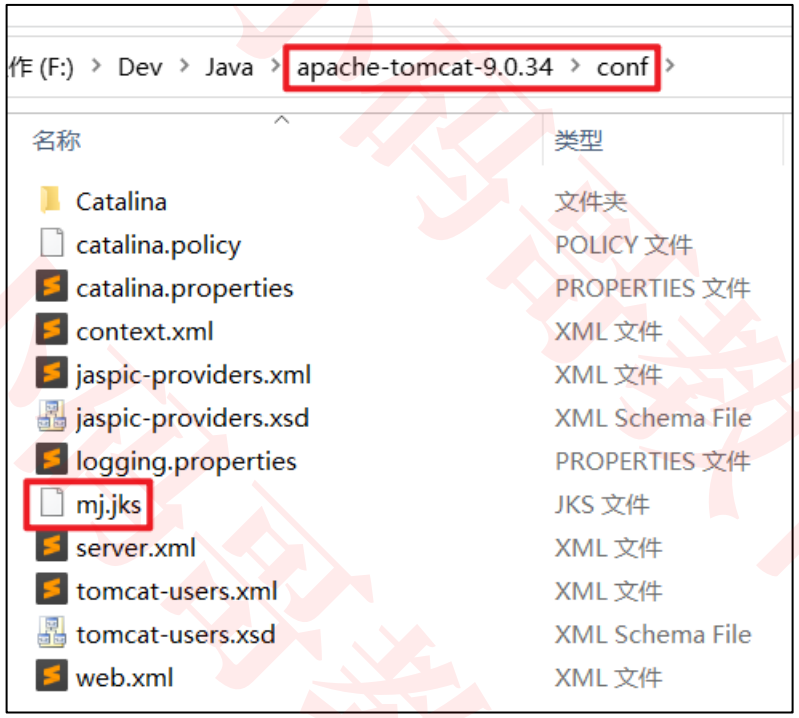
- 首先，使用JDK自带的keytool生成证书（一个生成免费证书的网站： https://freessl.org/）
  - keytool -genkeypair -alias mj -keyalg RSA -keystore F:/mj.jks

```
C:\Users\MJ\Desktop>keytool -genkeypair -alias mj -keyalg RSA -keystore F:/mj.jks
输入密钥库口令:
再次输入新口令:
您的名字与姓氏是什么?
  [Unknown]:
您的组织单位名称是什么?
  [Unknown]:
您的组织名称是什么?
  [Unknown]:
您所在的城市或区域名称是什么?
  [Unknown]:
您所在的省/市/自治区名称是什么?
  [Unknown]:
该单位的双字母国家/地区代码是什么?
  [Unknown]:
CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown是否正确?
  [否]: y

正在为以下对象生成 2,048 位RSA密钥对和自签名证书 (SHA256withRSA) (有效期为 90 天):
        CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
```

配置服务器HTTPS – 配置Tomcat

- 将证书*.jks文件放到TOMCAT_HOME/conf目录下



配置服务器HTTPS – 配置Tomcat

- 修改TOMCAT_HOME/conf/server.xml中的Connector

```
<Connector port="8443"
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           SSLEnabled="true">
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/mj.jks"
                     type="RSA"
                     certificateKeystorePassword="123456"/>
    </SSLHostConfig>
</Connector>
```