密码学原理
实验四：Web PKI 与 TLS

学号：2022113564　　姓名：张哲恺

实验目的：通过搭建一个 HTTPS 网站并分析 Web PKI 与 TLS 协议来理解密码学的现实应用。

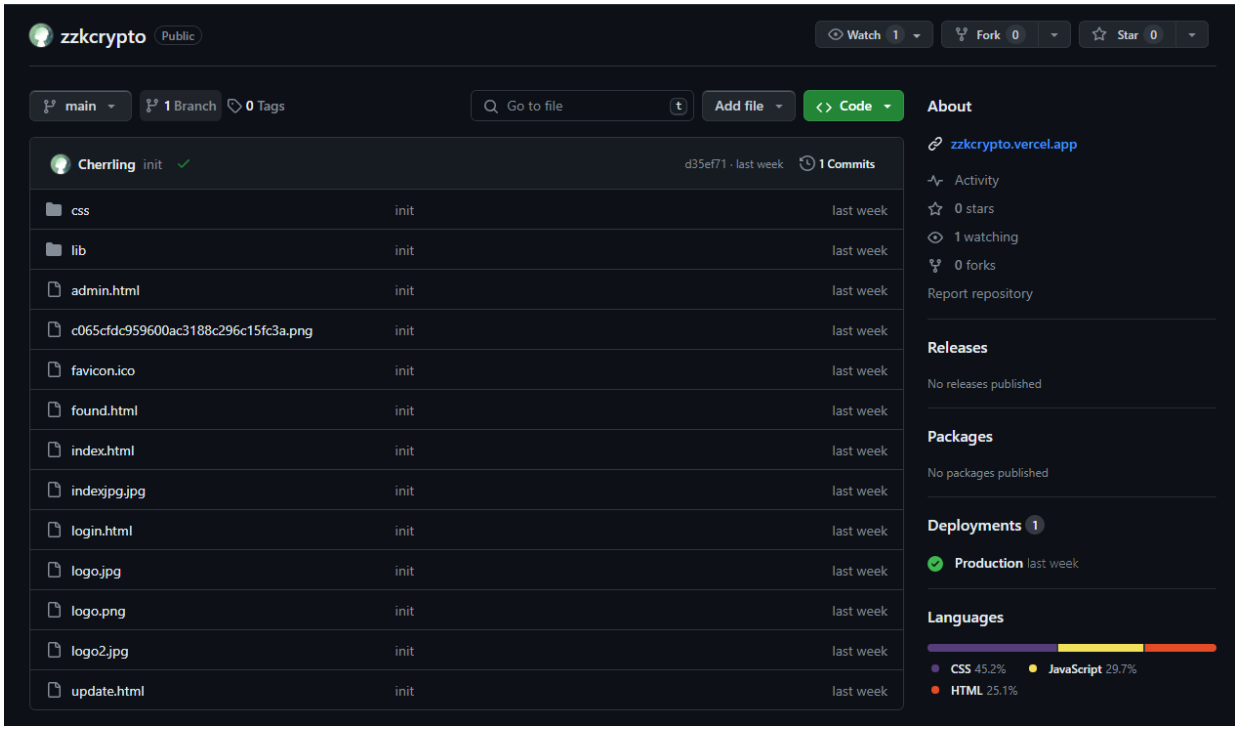
实验内容：

1. 建立一个部署数字证书的网站

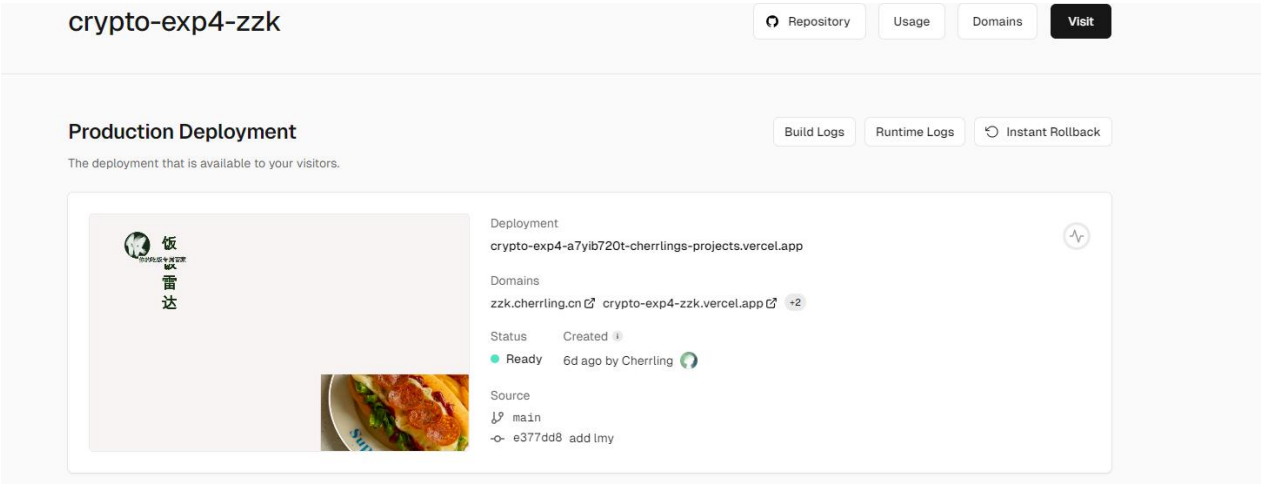
要求：注册一个域名，建立一个 Web 网站，生成该网站的数字证书并部署，通过 HTTPS 访问该网站。域名可免费申请。Web 服务器推荐购买云服务。证书可以通过自建 CA 颁发（需在浏览器部署 CA 信任锚），或者向第三方 CA（Let's Encrypt、GoDaddy 等）申请。

首先是在阿里云上注册域名然后进行域名解析。

使用 vercel 协助部署网站，首先把自己的网页文件上传到 GitHub 仓库。

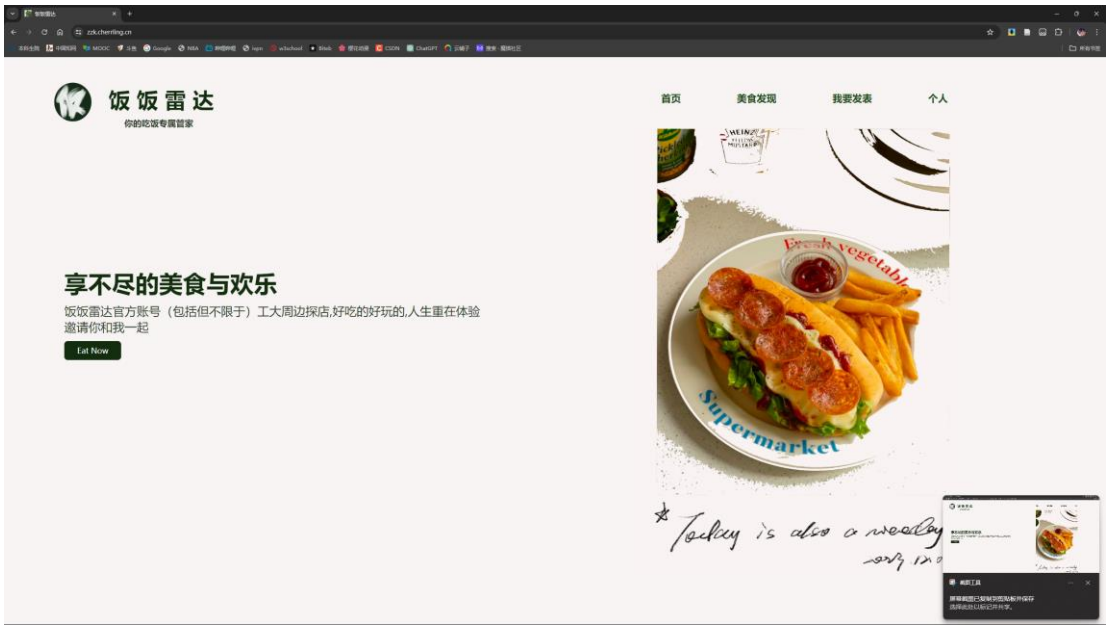


然后在 vercel 上导入刚刚的 GitHub 仓库进行网页部署。



然后将我们刚刚注册的域名与网页绑定即可。Vercel 会自动给我们颁发证书。

我的网页界面：



证书：

2. TLS 协议的密码学要素分析

要求：参考 TLS1.3 协议，通过浏览器自带功能和网络抓包等协议分析方法，详细展示其中的密码学要素信息，包括所涉及的密钥协商协议、数字签名（证书）、

非对称加密和对称加密方案。

访问支持 TLS1.3 协议的网站，使用浏览器自带功能控制台查看其密码学相关信息：

使用 wireshark 抓包，过滤 ip 地址。

抓包结果：

对握手的各个阶段进行分析

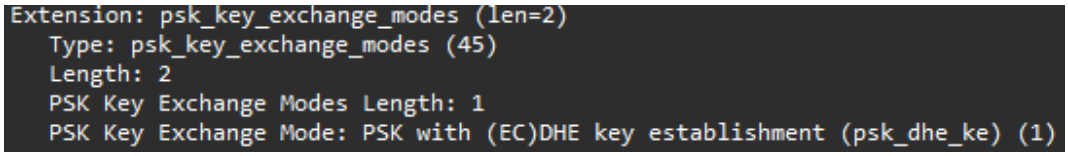
Client Hello：客户端发送 Hello，传输支持的密码学套件和公钥。

Random 随机数

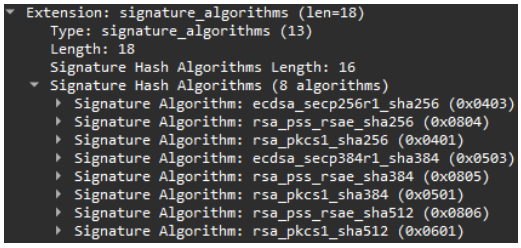
Session ID 会话 ID，

Cipher Suites 客户端发送自己支持的密码学套件

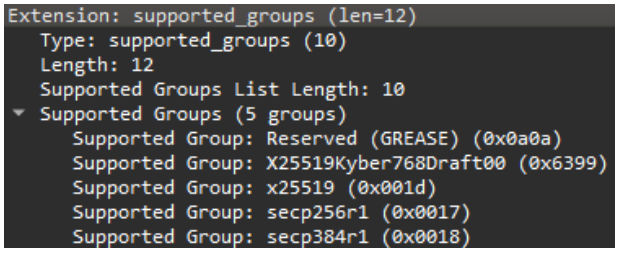
客户端指定使用 ECDHE 密钥交换模式：



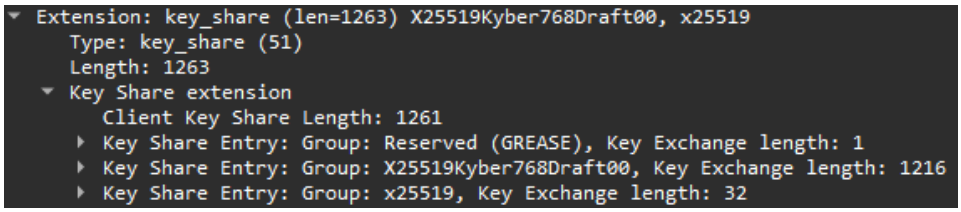
客户端提供的签名算法：



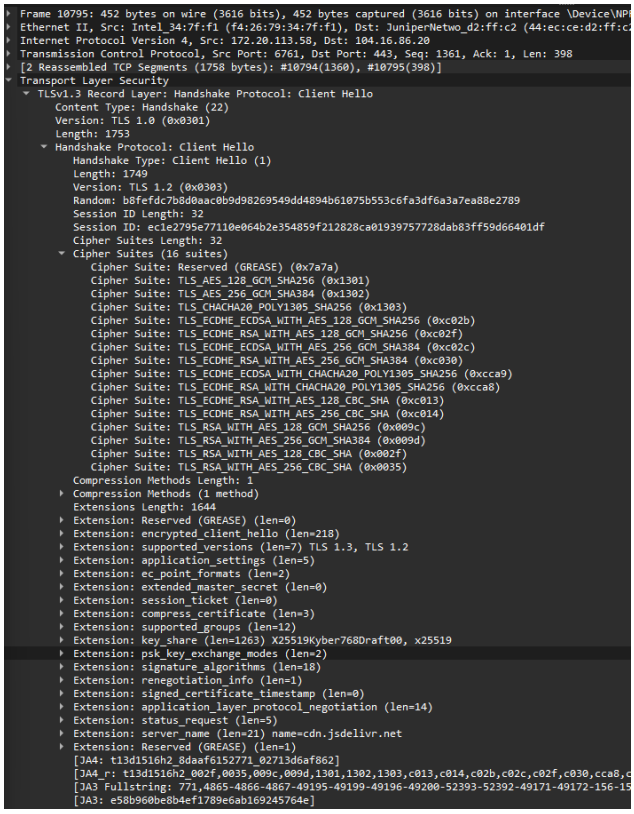
客户端支持的椭圆曲线类型



客户端发送共享密钥数据，包括使用的椭圆曲线和对应的公钥。服务器收到后可以根据接受的 ECDHE 和客户端的公钥生成自己的公钥和对称密钥



全部抓包结果：



Sever Hello：服务器根据客户端的加密套件、公钥算出自己公钥和私钥。

服务器选择 AES_GCM 作为对称加密算法

```
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
```

key_share 拓展，服务器把自己的公钥也发送给客户端，选择了 x25519 椭圆曲线

```
Extension: key_share (len=1124) X25519Kyber768Draft00
   Type: key_share (51)
   Length: 1124
 ▾ Key Share extension
   ▾ Key Share Entry: Group: X25519Kyber768Draft00, Key Exchange length: 1120
        Group: X25519Kyber768Draft00 (25497)
        Key Exchange Length: 1120
        Key Exchange [truncated]: 75ae6965c580362a68760e9d8c96f27369ee65717e3ba88af37149db
```

服务器告诉客户端改变加密方式，使用对称密钥加密。

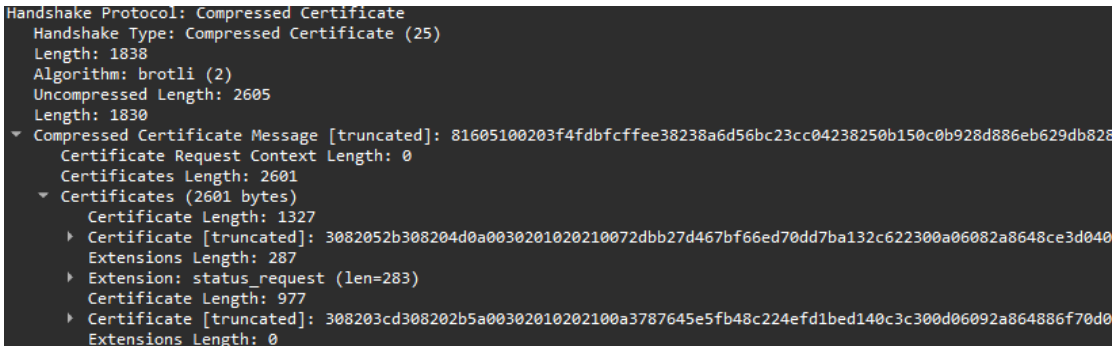
```
TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
   Content Type: Change Cipher Spec (20)
   Version: TLS 1.2 (0x0303)
   Length: 1
   Change Cipher Spec Message
```

全部抓包结果：

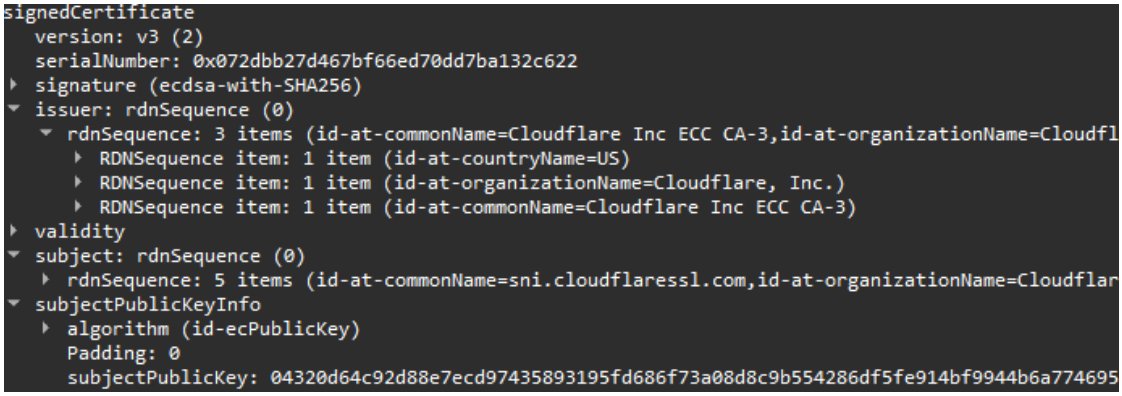
```
▸ Frame 10808: 1414 bytes on wire (11312 bits), 1414 bytes captured (11312 bits) on interface \Device
▸ Ethernet II, Src: JuniperNetwo_d2:ff:c2 (44:ec:ce:d2:ff:c2), Dst: Intel_34:7f:f1 (f4:26:79:34:7f:f1
▸ Internet Protocol Version 4, Src: 104.16.86.20, Dst: 172.20.113.58
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 6761, Seq: 1, Ack: 1759, Len: 1360
▾ Transport Layer Security
  ▾ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 1210
    ▾ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 1206
        Version: TLS 1.2 (0x0303)
        Random: d25fb92d2e5d2754bdaabd815d4608b9cb7f7feb4e268820db167c2324928180
        Session ID Length: 32
        Session ID: ec1e2795e77110e064b2e354859f212828ca01939757728dab83ff59d66401df
        Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        Compression Method: null (0)
        Extensions Length: 1134
      ▾ Extension: key_share (len=1124) X25519Kyber768Draft00
          Type: key_share (51)
          Length: 1124
        ▾ Key Share extension
          ▸ Key Share Entry: Group: X25519Kyber768Draft00, Key Exchange length: 1120
      ▾ Extension: supported_versions (len=2) TLS 1.3
          Type: supported_versions (43)
          Length: 2
          Supported Version: TLS 1.3 (0x0304)
        [JA3S Fullstring: 771,4865,51-43]
        [JA3S: eb1d94daa7e0344597e756a1fb6e7054]
  ▾ TLSv1.3 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 segment data (139 bytes)
```

Certificate：为了确认身份，服务器把证书发给客户端，客户端再用第三方权威认证中心的公钥解析证书。

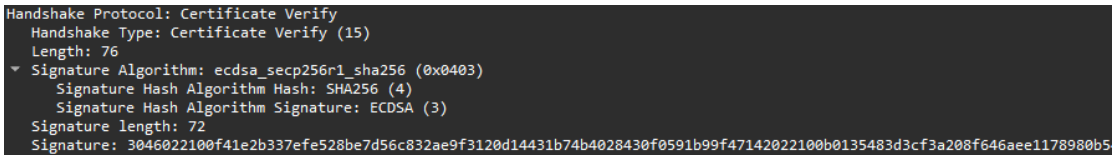
可以发现服务器发送的是 Compressed Certificate，发送压缩过后的证书能有效减少握手时的带宽消耗。



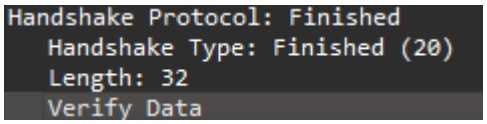
每个证书里还包含了证书的颁发者 issuer，有效期 validity，颁发给谁的 subject，服务器的公钥 subjectPublicKeyInfo



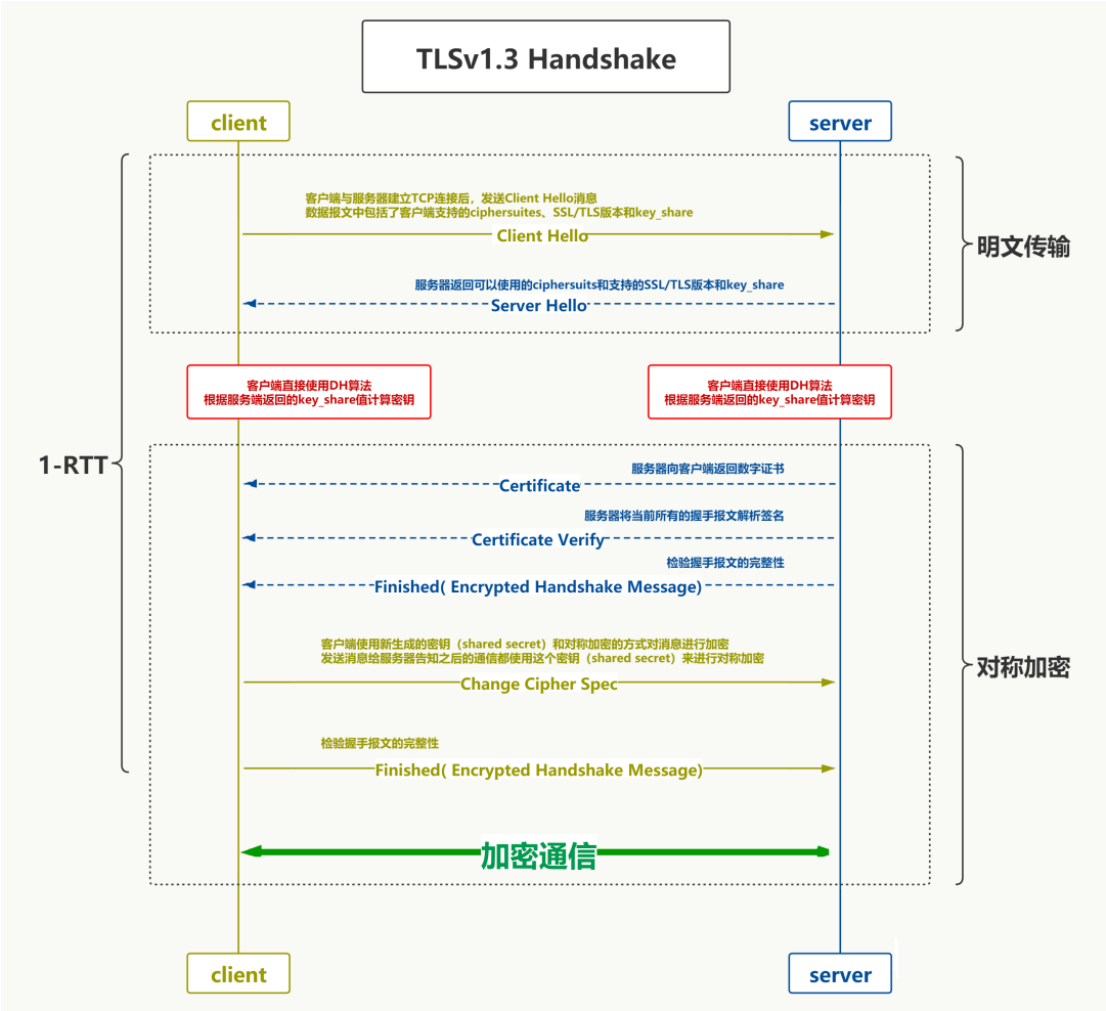
Certificate verify：客户端发送证书验证，使用 ecdsa_secp256r1 椭圆曲线来做认证，确保证书没有被篡改，hash 值是用 SHA256 计算出来的。



Finished：这是身份验证阶段的最后一条消息，也是第一个使用协商的算法进行加密和防篡改保护的信息，Verify Data 使用 HMAC 计算得来的，包含 finished_key 和握手消息的 hash。



TLS1.3 握手全过程：


TLSv1.3 Handshake
client
server
客户端与服务器建立TCP连接后，发送Client Hello消息
数据报文中包括了客户端支持的ciphersuites、SSL/TLS版本和key_share
Client Hello
明文传输
服务器返回可以使用的ciphersuits和支持的SSL/TLS版本和key_share
Server Hello
客户端直接使用DH算法
根据服务端返回的key_share值计算密钥
客户端直接使用DH算法
根据服务端返回的key_share值计算密钥
1-RTT
服务器向客户端返回数字证书
Certificate
服务器将当前所有的握手报文解析签名
Certificate Verify
检验握手报文的完整性
Finished(Encrypted Handshake Message)
客户端使用新生成的密钥（shared secret）和对称加密的方式对消息进行加密
发送消息给服务器告知之后的通信都使用这个密钥（shared secret）来进行对称加密
Change Cipher Spec
对称加密
检验握手报文的完整性
Finished(Encrypted Handshake Message)
加密通信
client
server