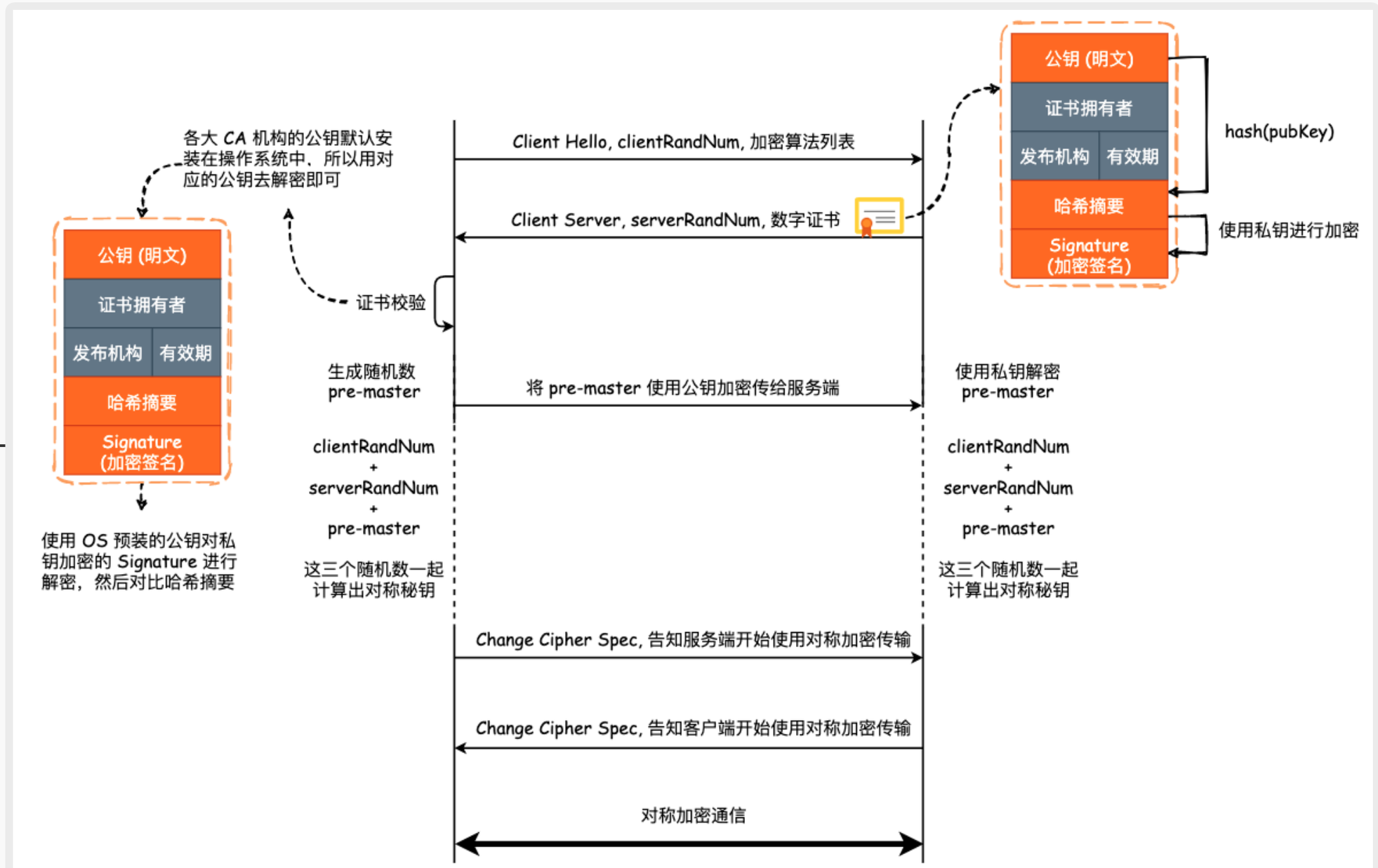
HTTPS

概述

加密方式

- 对称加密
  - 在对称加密算法中，加密、解密时需要的密钥是相同的，也就是说，使用 "小鸡炖蘑菇" 这句话既可以对信息进行加密，也可以用来解密加密后的字符串。因此，在使用对称加密时，密钥一定不能被别人知道
  - 对称加密因为加密、解密逻辑比较简单，所以性能很高，速度比较快
- 非对称加密
  - 在非对称加密中，加密和解密的密钥是不相同的
    - 公钥加密的内容，只有私钥才能解开
    - 私钥加密的内容，只有公钥才能解开
  - 也就是说，如果这世上只有唯一的一个人拥有某一个私钥，而你拥有对应的公钥。那么如果你可以使用公钥解开对方使用私钥加密的内容，那么对面的那个人就一定是接头儿的，我们可以放心的把谍报给他了
  - 因为非对称加密比较复杂，所以其性能赶不上对称加密
- 结合上述优缺点，HTTPS 在握手阶段采用非对称加密，而在正式的数据交互过程中则使用对称加密来提高效率



HTTPS 数据通信需要经历 4 个阶段：DNS 查询，建立 TCP 连接，建立 TLS 连接，最后才是 HTTP 发送数据，上图主要描述了建立 TLS 连接阶段

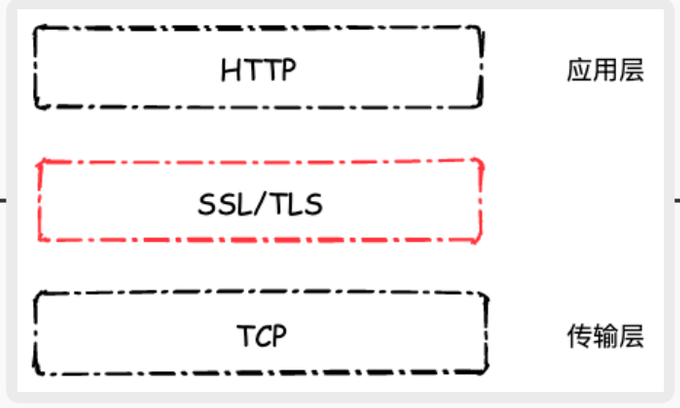
TLS 握手过程

1. 客户端向服务端发送 Client Hello 信息，并将自己所支持的 TLS/SSL 协议版本号、加密算法、压缩算法，以及一个随机数 clientRandNum 一起发送给服务端
2. 服务端收到客户端发来的 Client Hello 以后，向其回送 Server Hello 信息，并将选择的 TLS/SSL 版本号、加密算法、生成的随机数 serverRandNum 以及数字证书发送给客户端（这个过程会有多个数据包传输）
3. 客户端收到服务端的协议版本号、加密方法以及数字证书以后，开始验证数字证书的有效性
   - 包括证书是否过期，域名是否一致等，还会使用自身预装的 CA 公钥对 Signature 进行机密，对比哈希摘要
   - 证书确认无误后再次生成一个随机字符串，称为 Pre Master Secret，预主密钥，并使用公钥进行加密，传输给服务端
   - 而后，客户端使用 clientRandNum、serverRandNum 以及 pre-master 这三个随机数生成对称加密密钥
4. 服务端收到客户端加密的 Pre Master Secret 后，使用其私钥对其进行解密，并使用同样的方式对三个随机字符串生成对称加密密钥

而后，客户端与服务端均使用对称加密密钥对发送数据进行加密与通信

HTTPS 常见问题

- TLS 握手与 TCP 握手
  - TLS/SSL 和 TCP 属于完全不同的网络层面，TCP 属于传输层，不再乎用户发送的到底是加密数据还是非加密数据，只有应用层才会在意



  - 也就是说，SSL/TLS 并不是 TCP 协议的一部分，它只是为应用层数据提供加密、解密以及压缩的功能
  - 也就是说，在 TLS 握手之前，必须要经过 TCP 三次握手建立正常的连接，才能继续往下
- HTTPS 的优化问题
  - 从上面可以看出，一个 HTTPS 应用数据的发送，需要经过 TCP 三次握手 + TLS 四次握手才能建立起基本的连接，然后再发数据。协议比较复杂，沟通过程太繁复，这样会导致效率问题
  - 优化 TCP 三次握手 —— 如果"狠"一点，可以使用基于 UDP 实现的 QUIC，直接把三次握手省略掉
  - 优化 TLS 四次握手 —— 升级 TLS 1.2 至 TLS 1.3