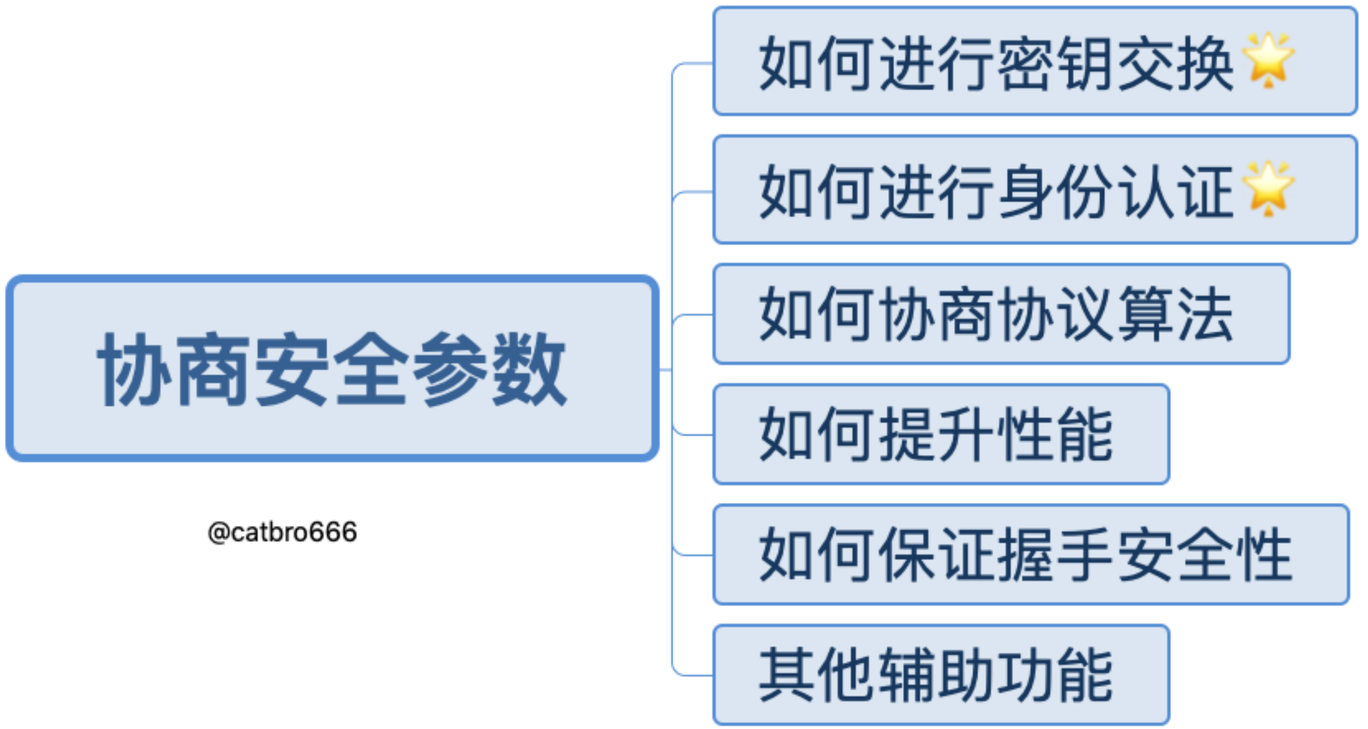
SSL握手协议详解

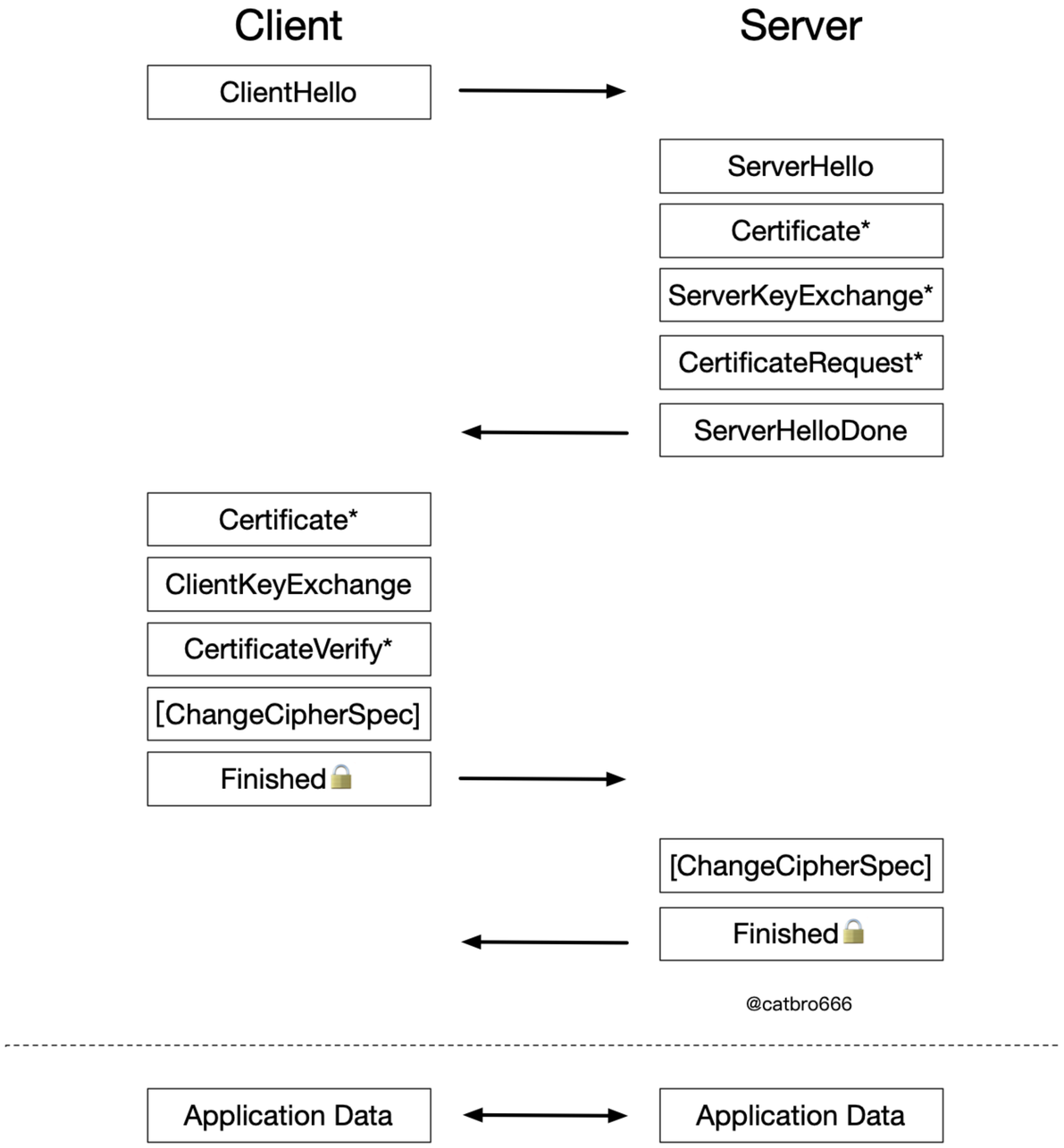
陈浙锋 20211027

概览


协商安全参数
@catbro666
如何进行密钥交换🌟
如何进行身份认证🌟
如何协商协议算法
如何提升性能
如何保证握手安全性
其他辅助功能

完整握手流程

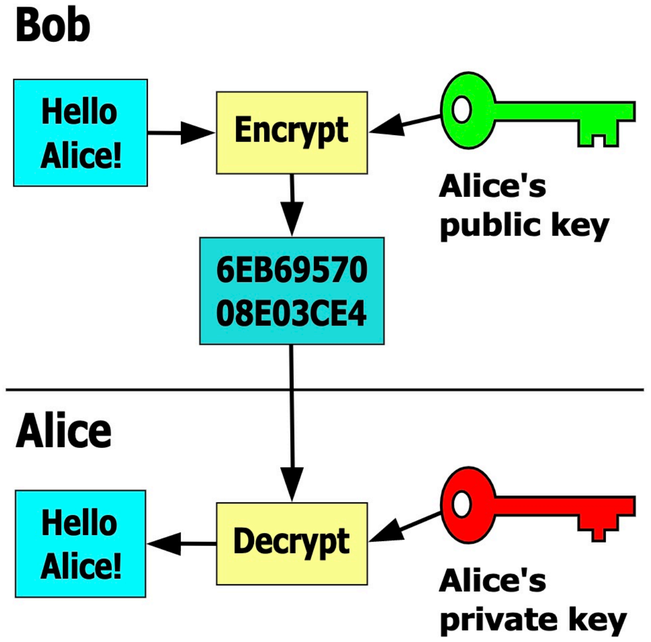
- SSL协议分为两个阶段:
  - 握手阶段
  - 应用阶段



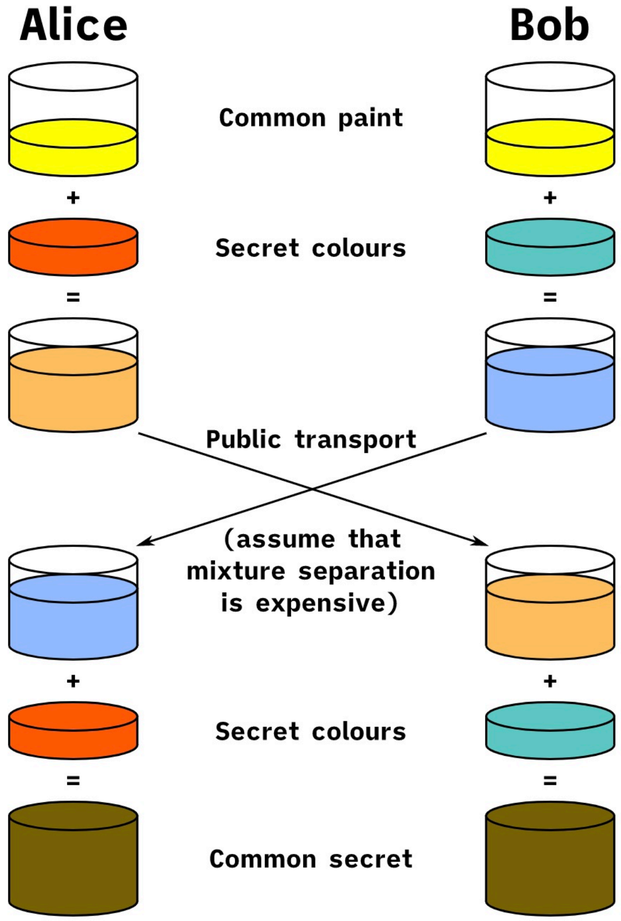
如何进行密钥交换

- 基于加密（RSA、国密ECC）
- 基于DH（ECDHE、国密ECDHE）

公钥加密原理

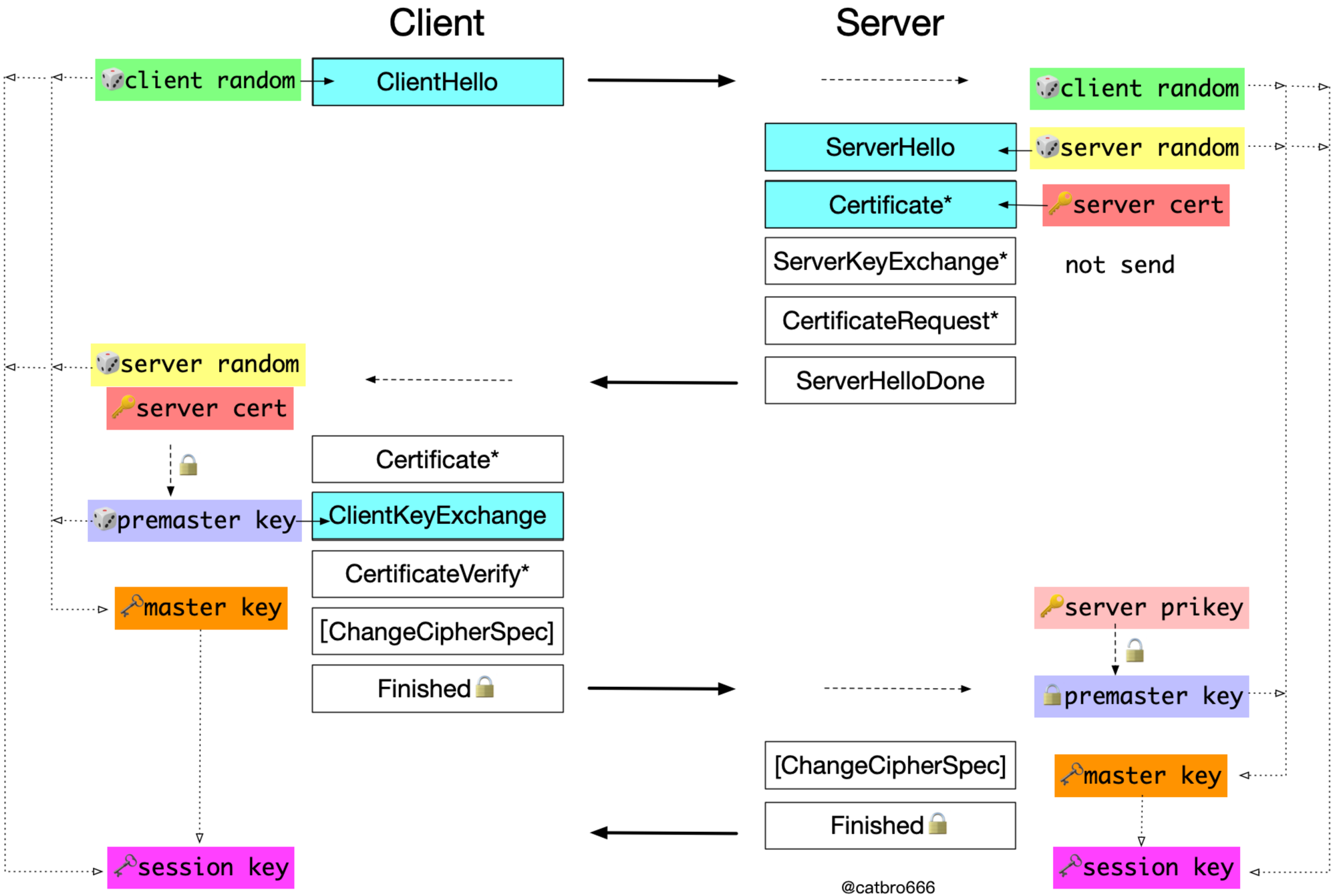


DH原理



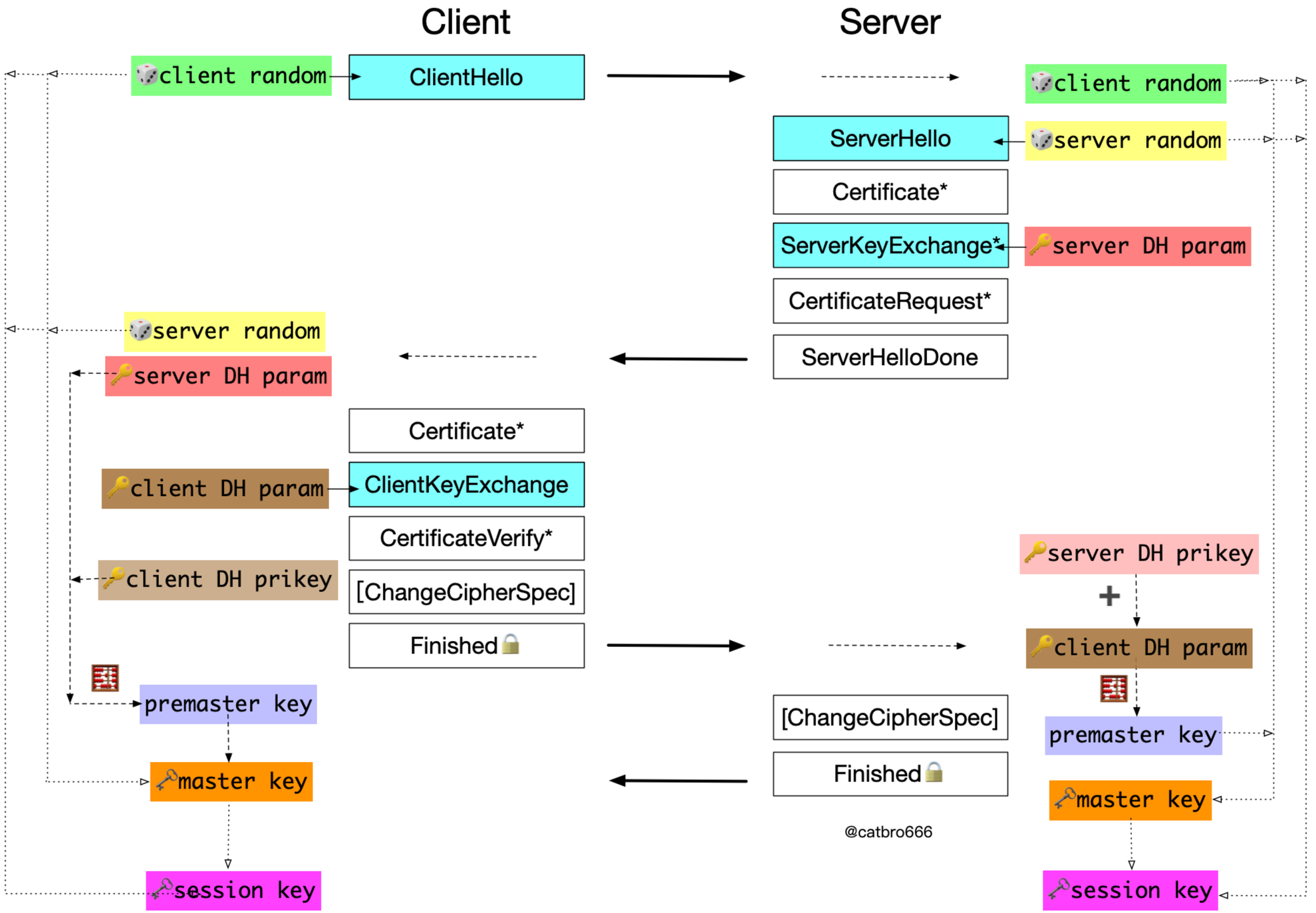
TLS RSA密钥交换

- 典型算法套件：
  - AES256-SHA
- 要点：
  - 客户端用服务端的公钥加密预主密钥，然后发送给服务端
  - 服务端用对应的私钥进行解密，得到预主密钥



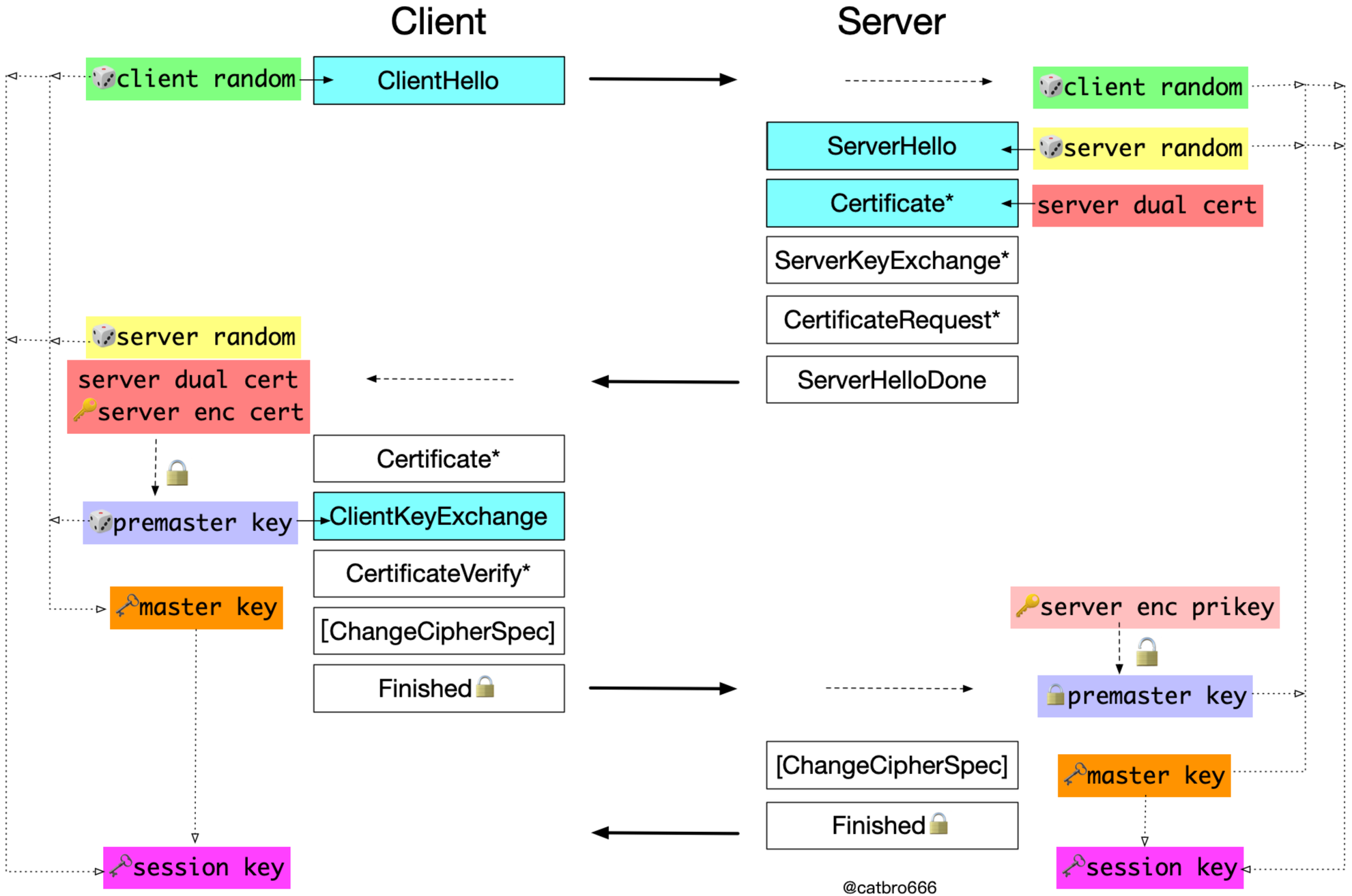
TLS ECDHE密钥交换

- 典型算法套件：
  - ECDHE-RSA-AES256-GCM-SHA384
  - ECDHE-ECDSA-AES256-GCM-SHA384
- 要点：
  - 双方将自己的临时DH参数发送给对方
  - 然后各自利用自己的私钥和对方的公钥计算出预主密钥



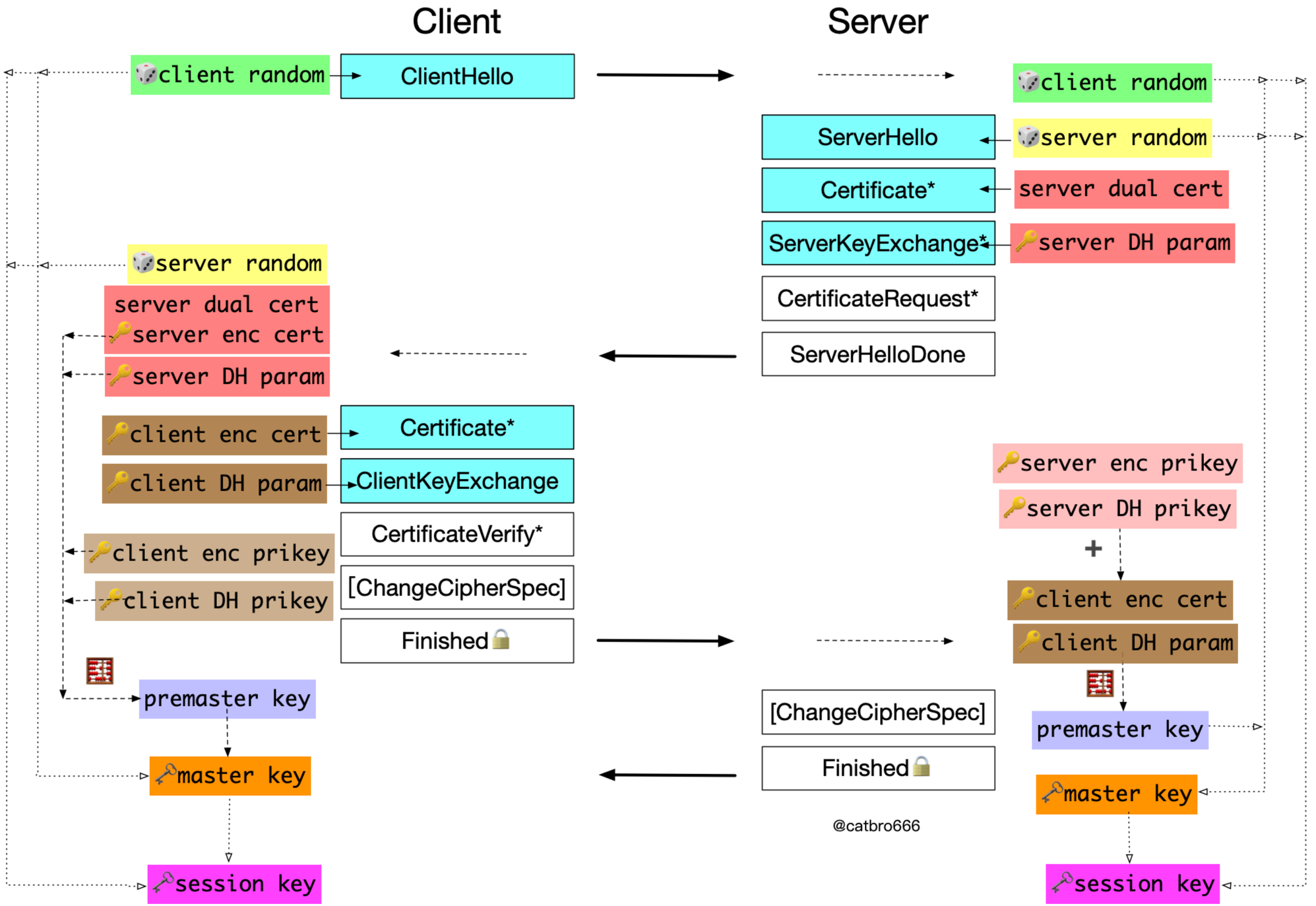
GM ECC密钥交换

- 典型算法套件：
  - ECC-SM4-SM3
- 要点：
  - 客户端用服务端加密证书中的公钥加密预主密钥，然后发送给服务端
  - 服务端用对应的加密私钥进行解密，得到预主密钥



GM ECDHE密钥交换

- 典型算法套件：
  - ECDHE-SM4-SM3
- 要点：
  - 双方将自己的加密证书及临时DH参数发送给对方
  - 然后利用自己的加密私钥、临时私钥及对方加密证书中的公钥及临时公钥，各自计算出预主密钥



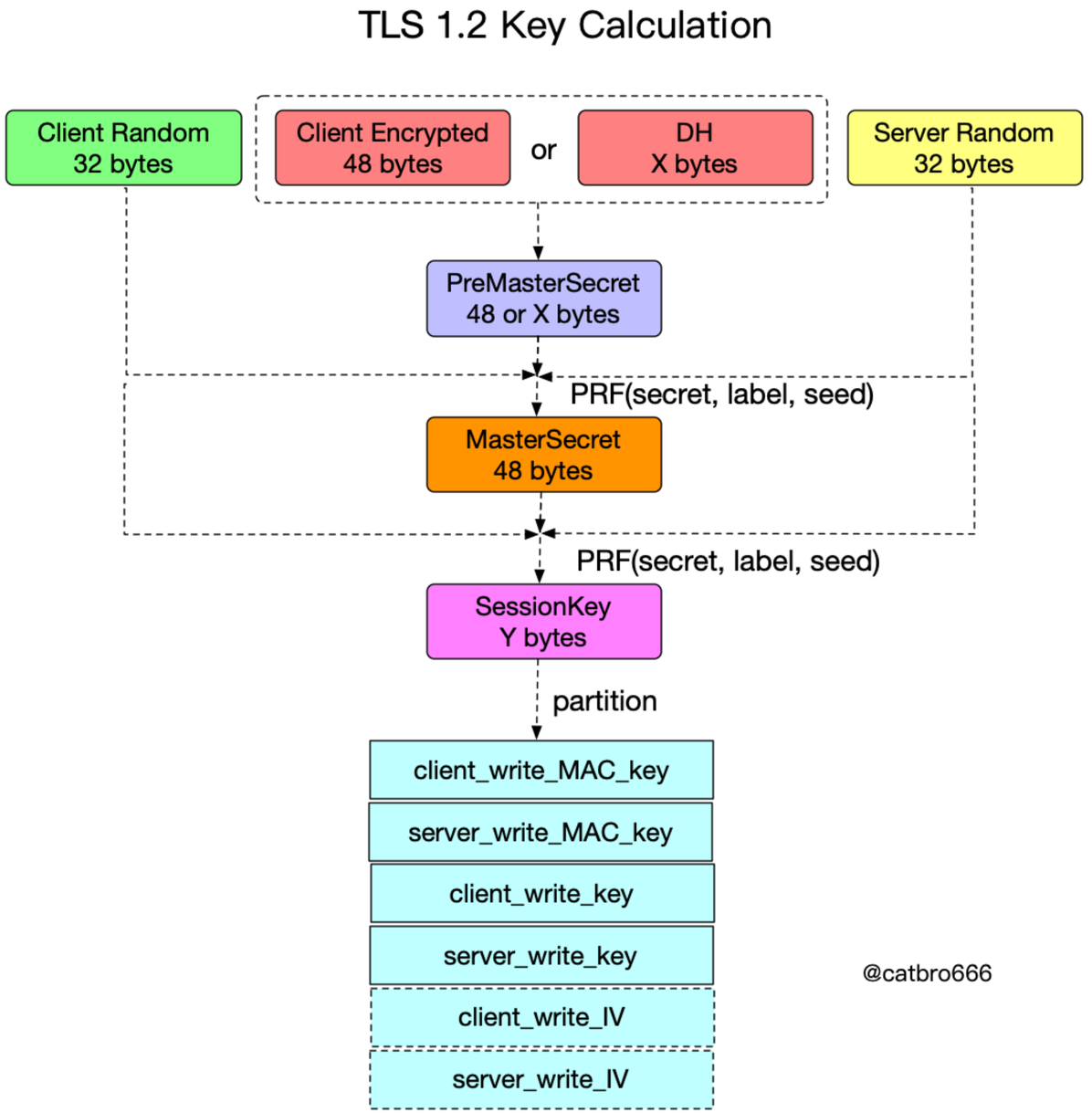
TLS1.2密钥派生流程

- 生成主密钥

```
master_secret = PRF(pre_master_secret, "master secret",
                    ClientHello.random + ServerHello.random)
                    [0..47];
```

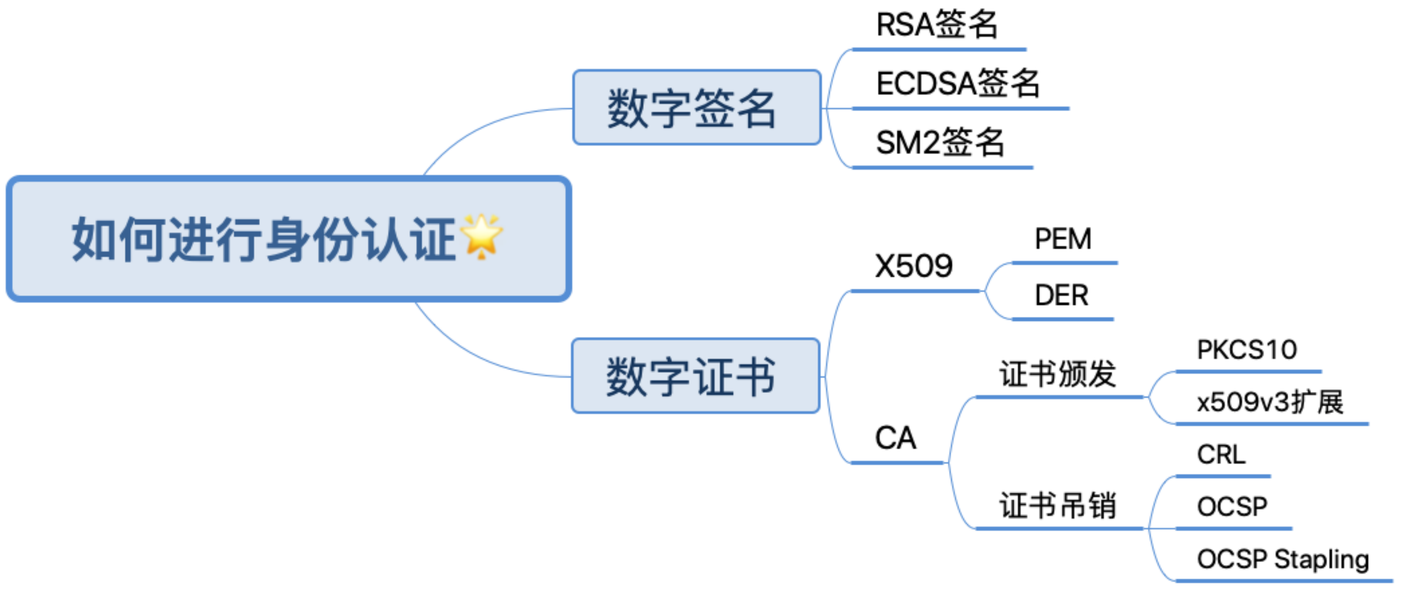
- 生成会话密钥

```
key_block = PRF(SecurityParameters.master_secret,
                 "key expansion",
                 SecurityParameters.server_random +
                 SecurityParameters.client_random);
```

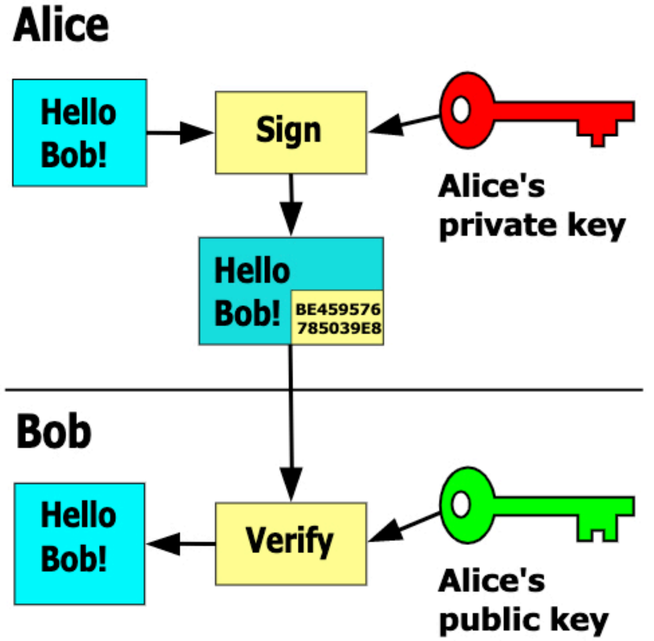


如何进行身份认证

- 确认对方拥有公钥对应的私钥
- 确认公钥所有者的身份

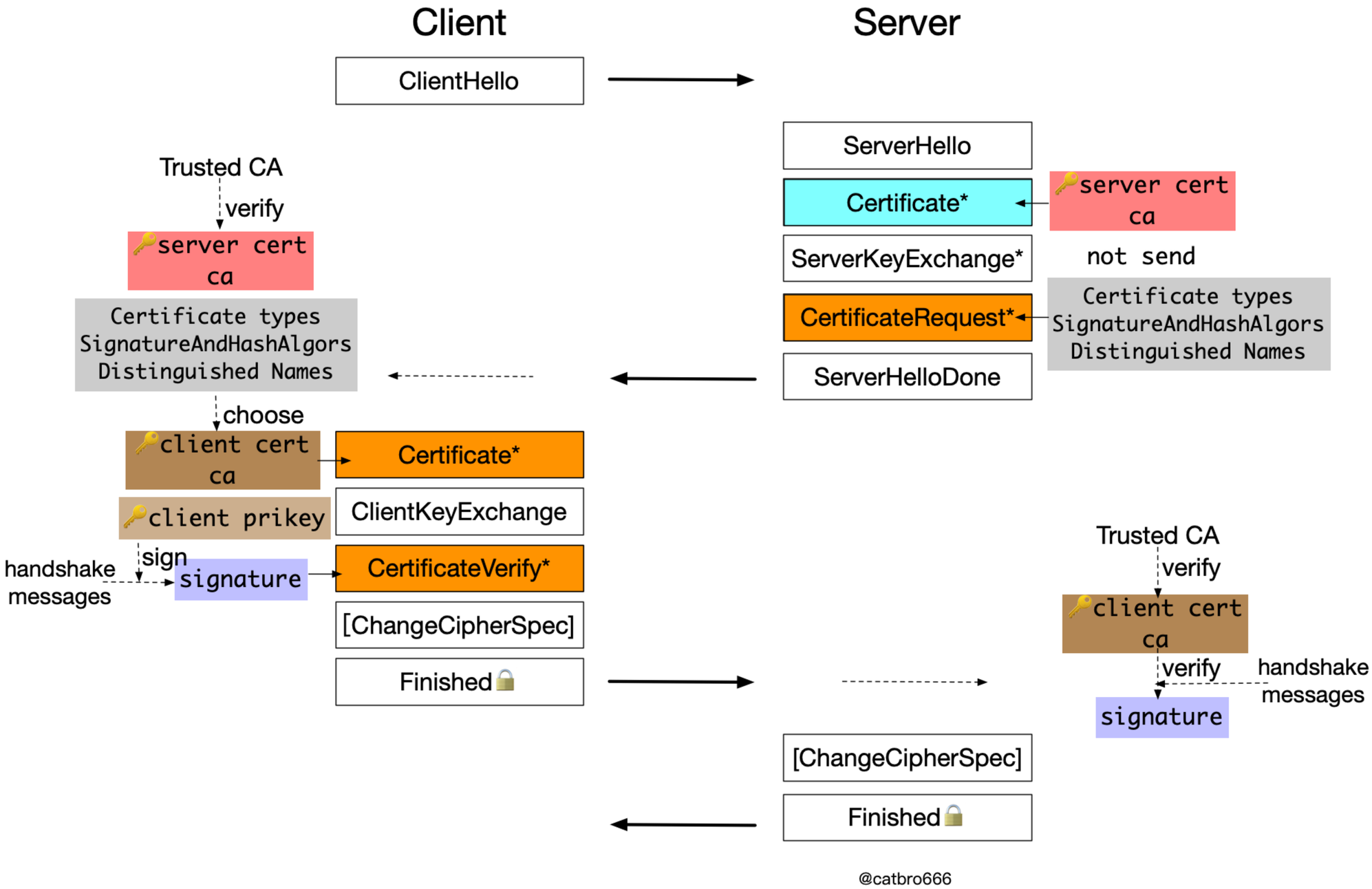


数字签名原理



TLS RSA密钥交换
RSA认证

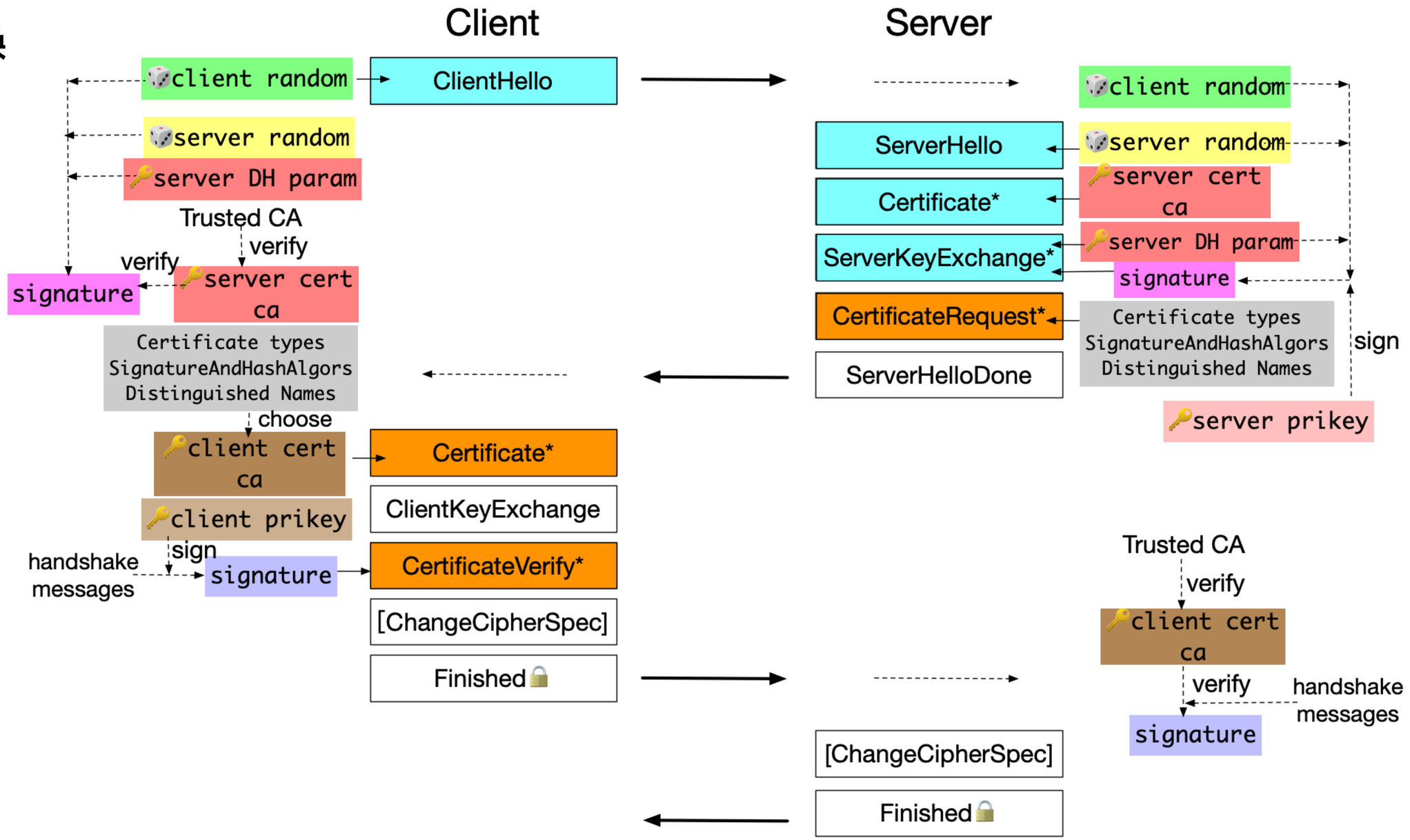
- 典型算法套件：
  - AES256-SHA
- 要点：
  - 服务端将证书及证书链发送给客户端，客户端对证书进行逐级验证，最终的Root CA需要是本机信任的
  - 如果需要双向认证，服务端通过CertificateRequest消息告知客户端，客户端发送自己的证书及证书链，并用私钥做签名之后发送给服务端，服务端对签名及证书链进行验证



@catbro666

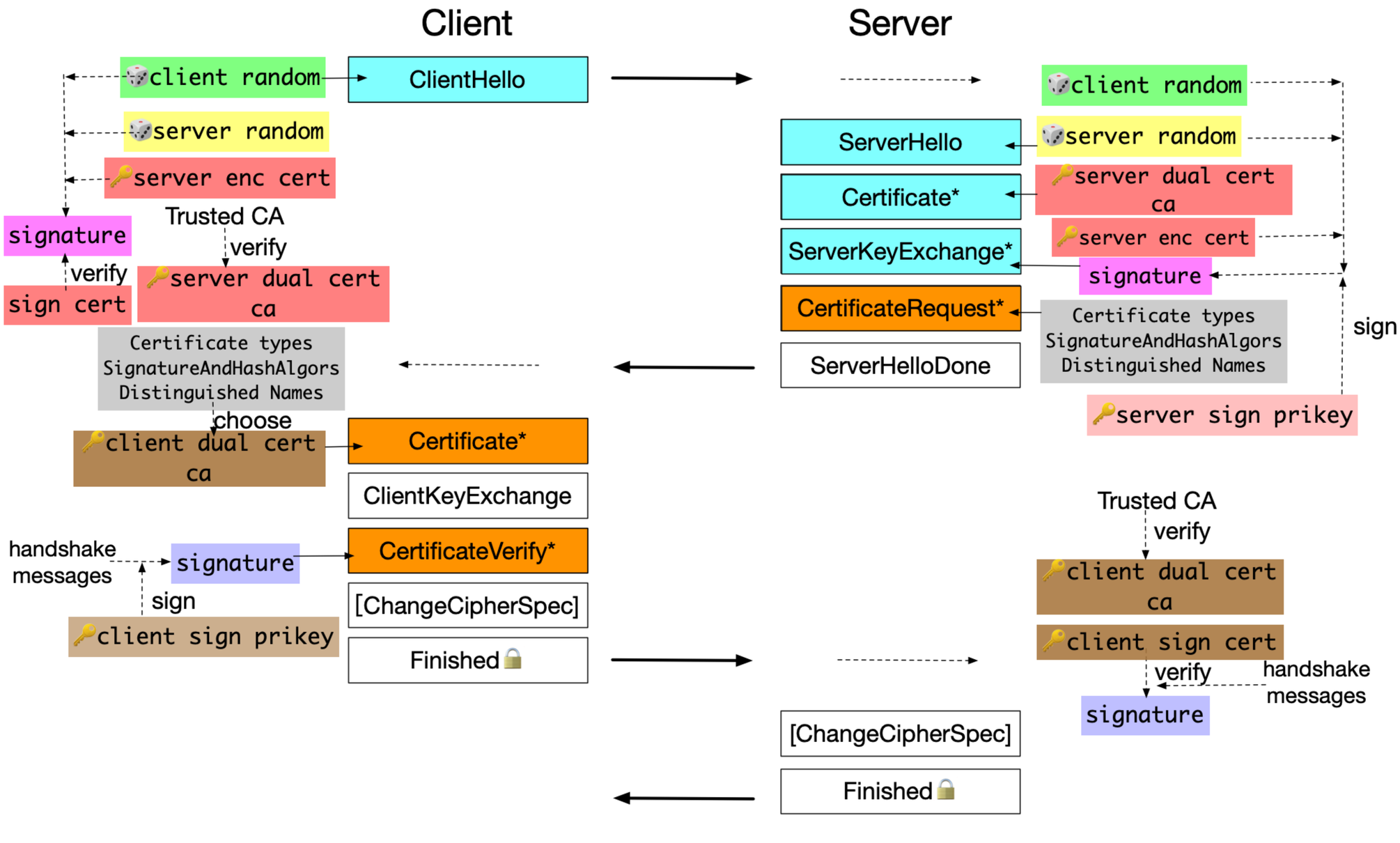
TLS ECDHE密钥交换的身份认证

- 典型算法套件：
  - ECDHE-RSA-AES256-GCM-SHA384
  - ECDHE-ECDSA-AES256-GCM-SHA384
- 要点：
  - 服务端除了发送证书及证书链给客户端，还需要发送一个签名以证明自己拥有对应私钥，客户端对签名及证书链进行验证，最终的Root CA需要是本机信任的
  - 客户端的认证流程跟前一种情况一样



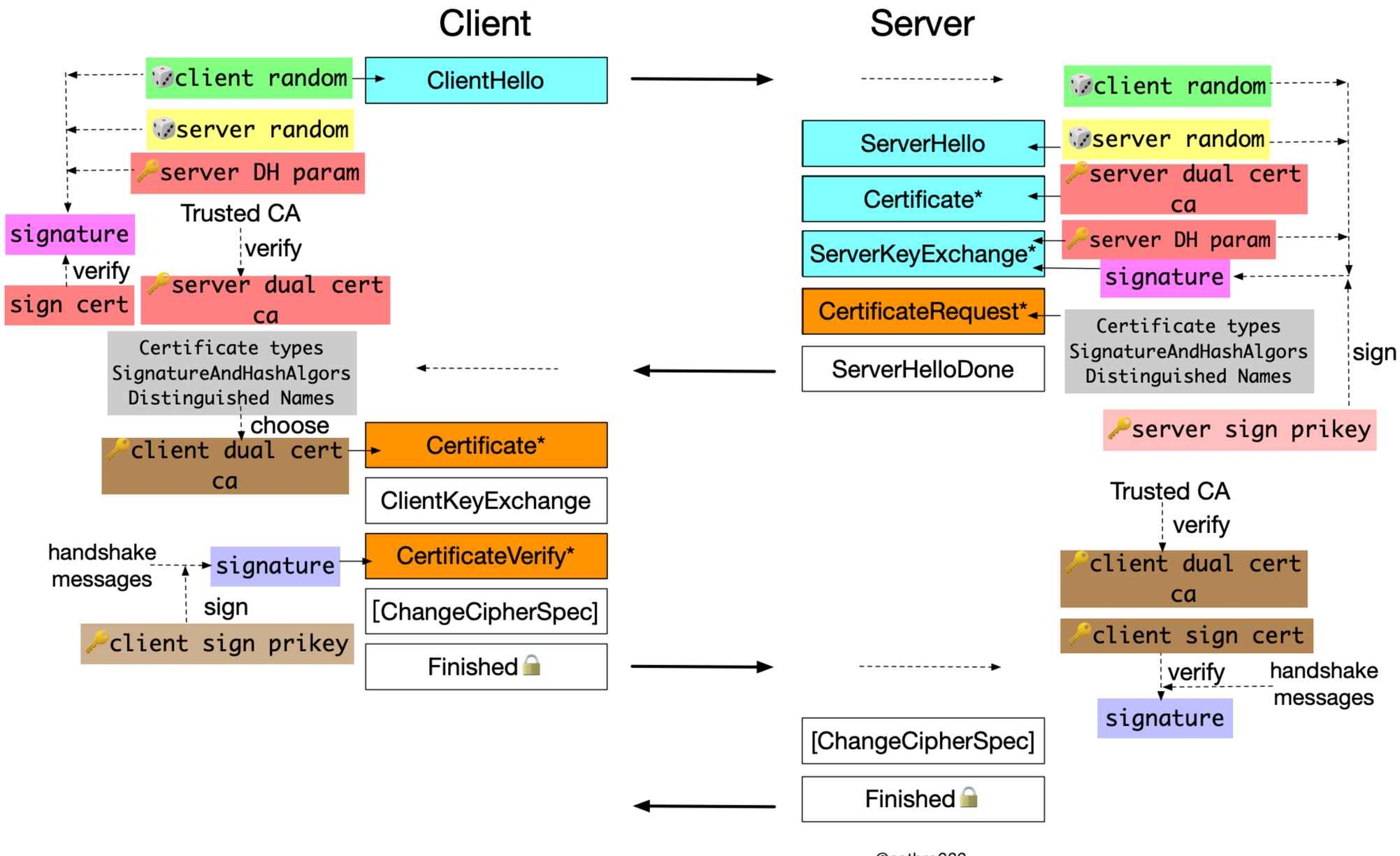
GM ECC密钥交换的身份认证

- 典型算法套件：
  - ECC-SM4-SM3
- 要点：
  - 服务端认证跟前一种情况的区别：发送的是双证书，签名的内容变为两个随机数+加密证书，由签名私钥进行签名
  - 客户端认证跟前一种情况的区别：发送的是双证书，由签名私钥进行签名



GM ECDHE密钥交换的身份认证

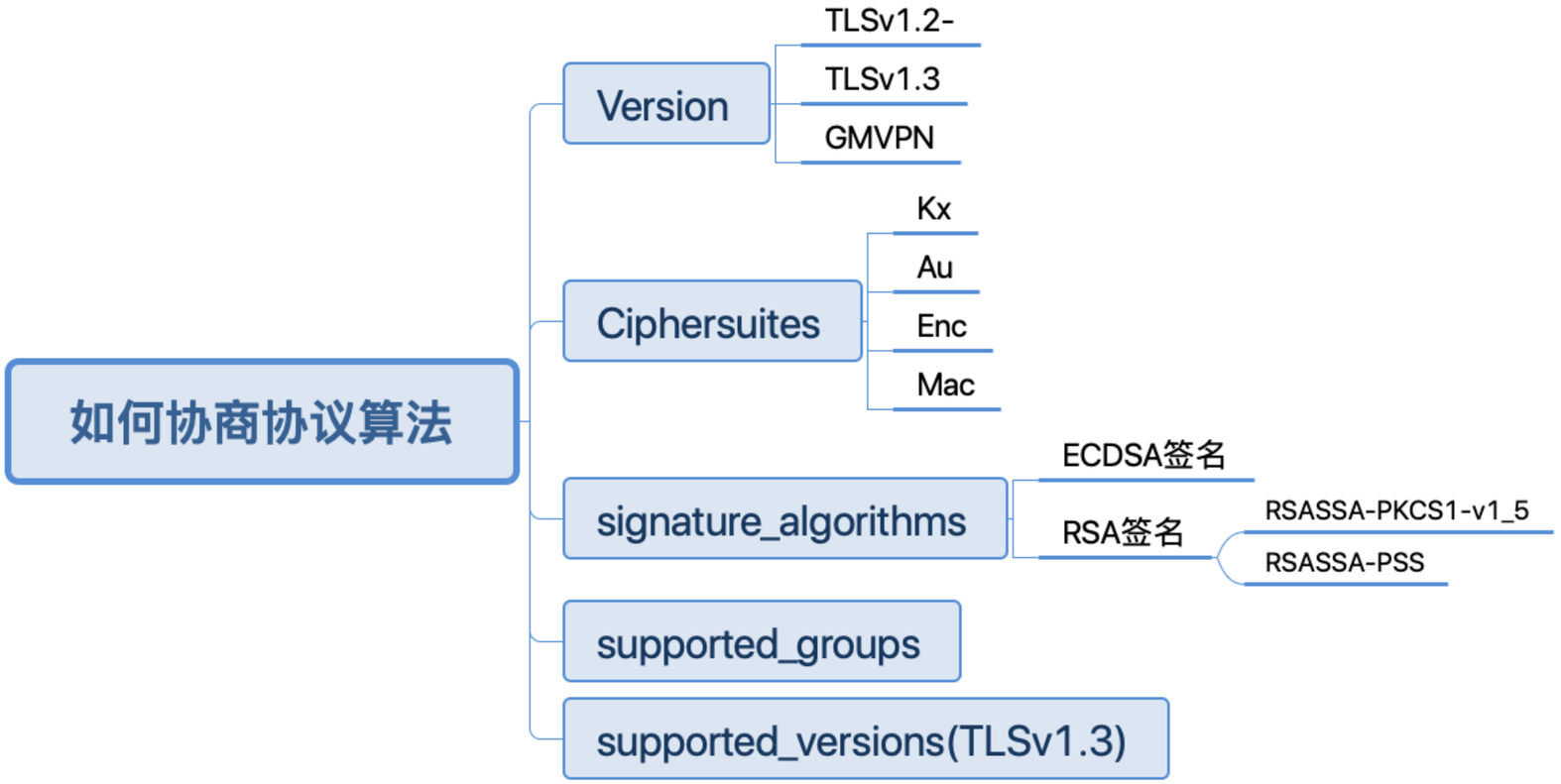
- 典型算法套件：
  - ECDHE-SM4-SM3
- 要点：
  - 服务端认证跟前一种情况的区别：签名的内容变为两个随机数+临时DH参数
  - 客户端认证流程跟前一种情况一样



@catbro666

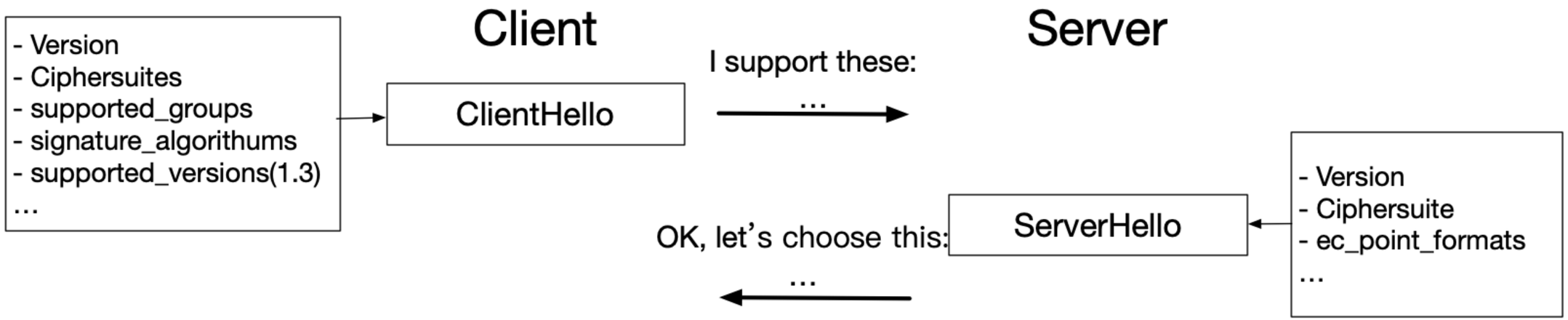
如何协商协议算法

- 不同的协议、算法套件等在处理流程上是不同的，那么双方如何对此达成一致呢？



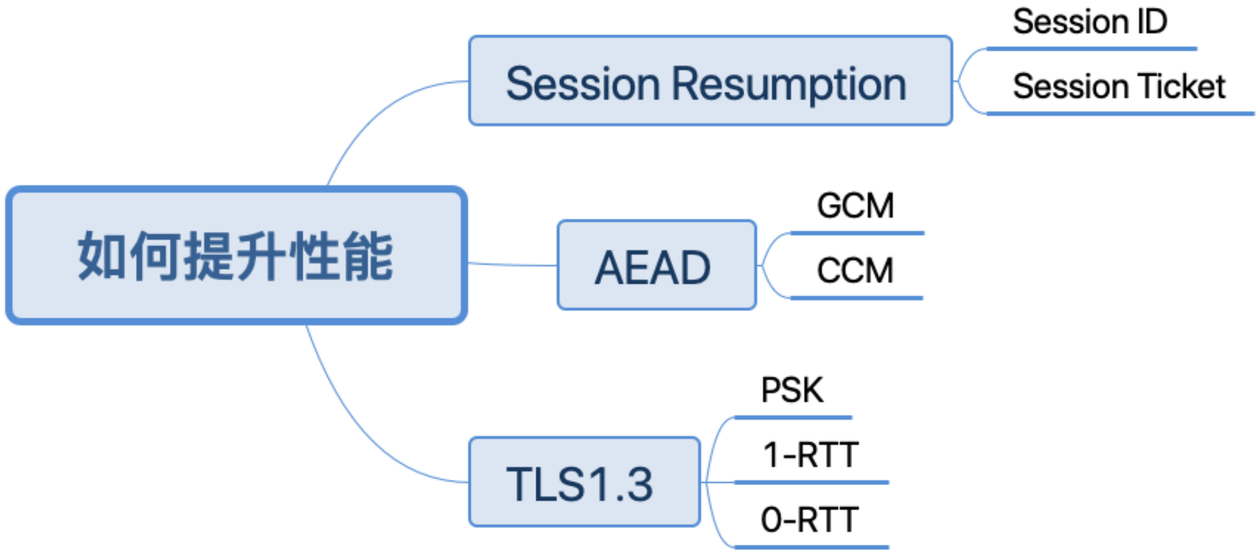
如何协商协议算法

- 增加一次Hello交互
- 利用ClientHello/ServerHello中的扩展项



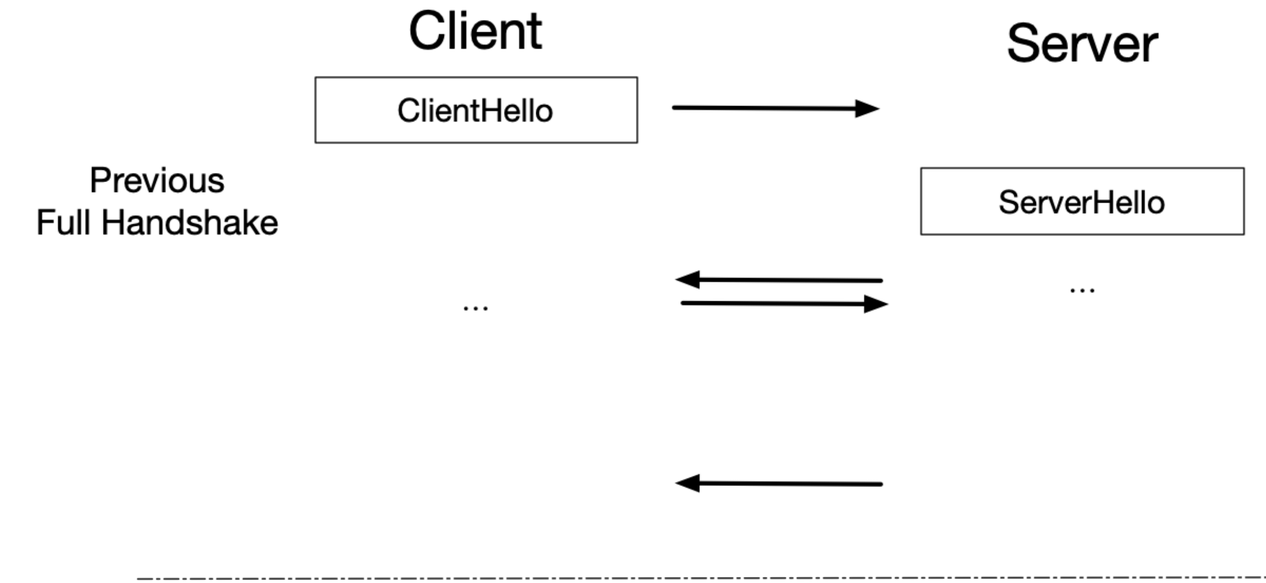
如何提升性能

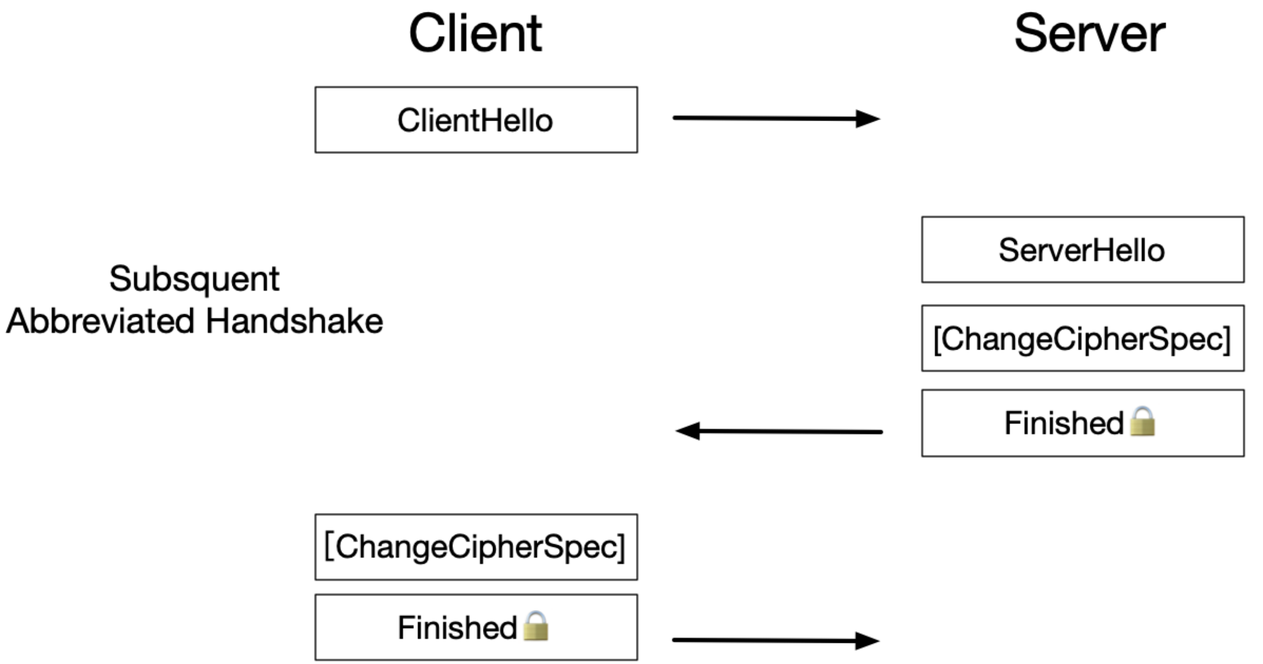
- 握手阶段
  - 握手流程优化
- 应用数据阶段
  - AEAD



会话重用基本流程

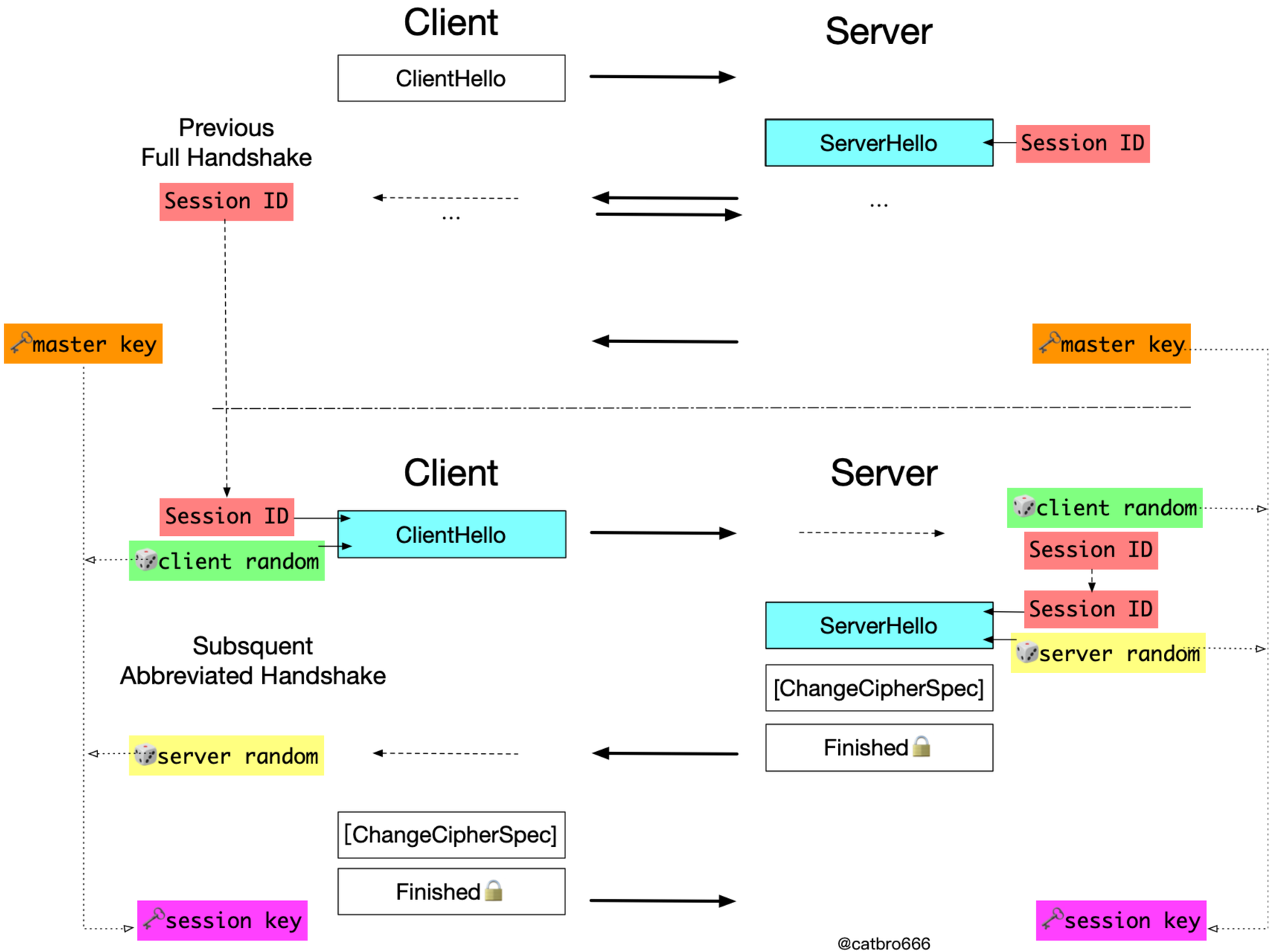
- 前一次完整握手
- 后面直接恢复前面的会话





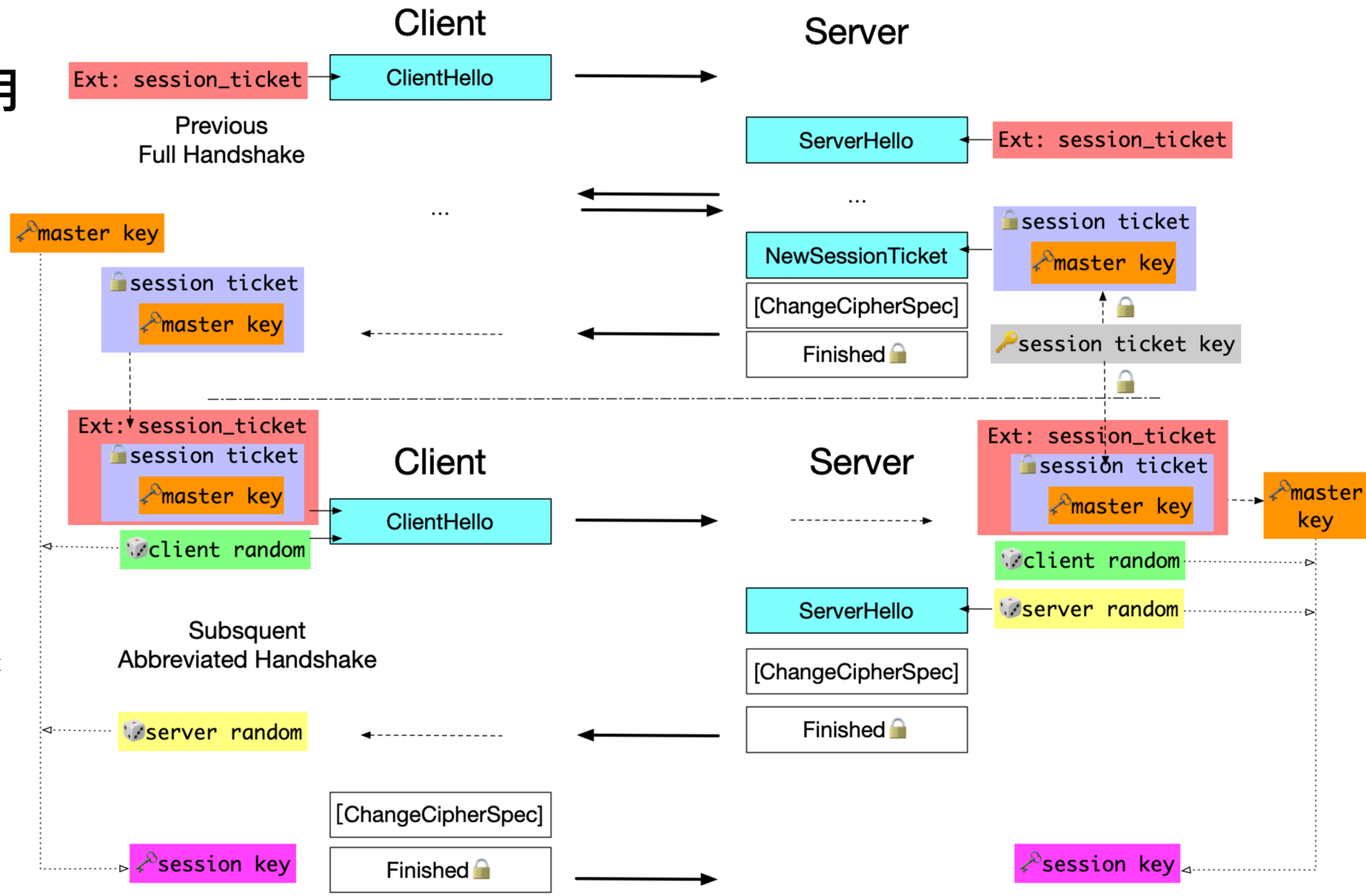
Session ID会话重用

- 要点：
  - 前一次握手完成后，服务端将Session保存
  - 后续客户端想要重用该Session，在ClientHello中包含其Session ID
  - 服务端在cache中进行查找，如果查找到且未过期，则进行会话重用；否则回退到完整的握手流程
  - 根据重用的主密钥重新派生会话密钥

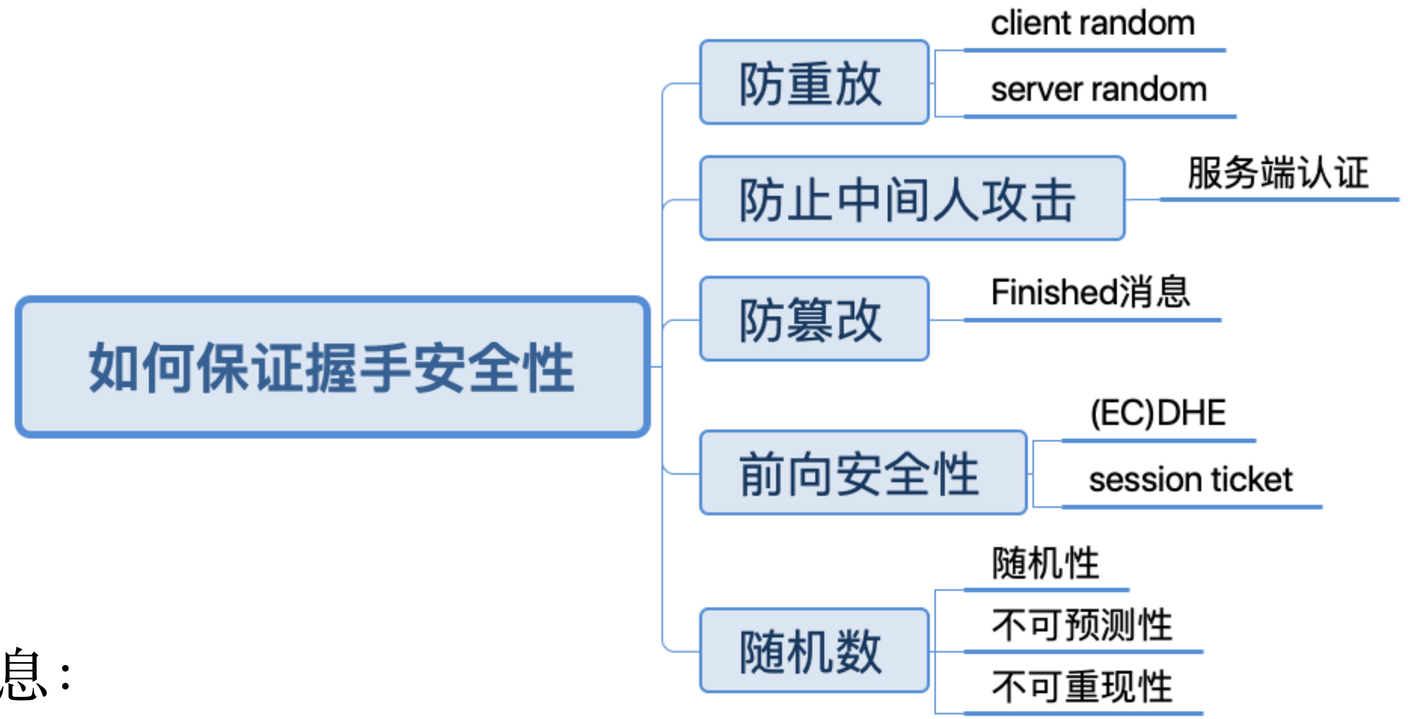


Session Ticket会话重用

- 要点：
  - 客户端首先在ClientHello中包含session_ticket扩展项，表示想使用session_ticket功能
  - 服务端同意使用，则在ServerHello中回复session_ticket扩展项。
  - 服务端不保存session，而是加密之后通过NewSessionTicket消息发送给客户端
  - 后续客户端想要重用该Session，在ClientHello中包含对应的session_ticket
  - 服务端在成功解密且验证通过之后，则进行会话重用；否则回退到完整的握手流程
  - 根据重用的主密钥重新派生会话密钥



如何保证安全性



Finished消息：

```
verify_data:
PRF(master_secret, finished_label,
    Hash(handshake_messages))
            [0..verify_data_length-1];
```

其他辅助功能

- SNI (Server Name Indication)
  - 多域名
- ALPN (Application-Layer Protocol Negotiation)
  - HTTP2

Thank you ~

作业

- 1写出以下情况对应的报错信息
  - 初始化时失败
    - 证书和私钥不匹配
    - 证书和算法套件不匹配
  - 握手时失败
    - 协议不支持
    - 算法套件不匹配
    - 签名算法不支持
    - 客户端身份验证失败（没提交证书、证书过期、没有对应的上级证书链、证书被吊销CRL）
- 2协议理解
  - SSL握手协议是如何保证安全性的，谈谈你的理解
  - 国密协议与TLS协议的主要区别
  - 统计使用前面几种典型算法套件进行握手时的密码运算次数（签名/验签/加密/解密/随机数生成）