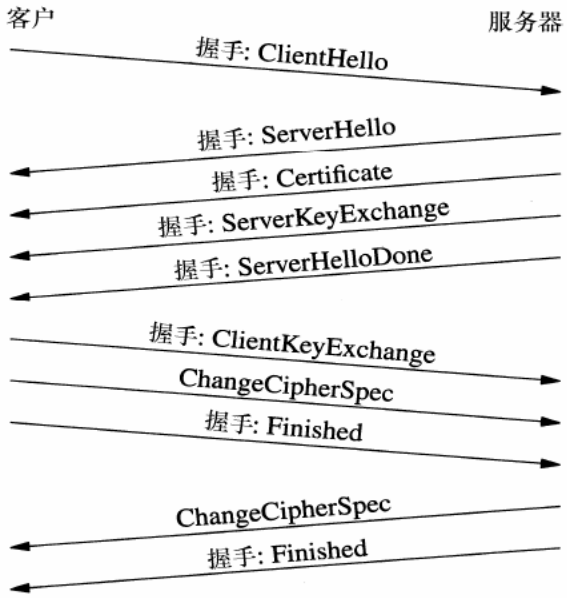
TLS 抓包及 DH 过程分析

TLS，通过非对称密钥加密法来交换会话密钥，通过对称密钥加密法来加密信息。



第一次 Client Hello，客户端向服务器端发送 Hello 消息

```
372 5.806460    192.168.1.101   17.248.158.114  TLSv1…   583 Client Hello
374 5.874551    17.248.158.114  192.168.1.101   TLSv1…  1494 Server Hello
379 5.877757    17.248.158.114  192.168.1.101   TLSv1…  1494 Certificate [TCP segment of a reassembled PDU]
381 5.880409    17.248.158.114  192.168.1.101   TLSv1…   276 Certificate Status, Server Key Exchange, Server Hello Done
383 5.888635    192.168.1.101   17.248.158.114  TLSv1…   159 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
384 5.942890    17.248.158.114  192.168.1.101   TLSv1…   117 Change Cipher Spec, Encrypted Handshake Message
385 5.942891    17.248.158.114  192.168.1.101   TLSv1…   135 Application Data
387 5.946125    192.168.1.101   17.248.158.114  TLSv1…   419 Application Data, Application Data, Application Data, Application Data
388 5.946216    192.168.1.101   17.248.158.114  TLSv1…   104 Application Data
389 5.946276    192.168.1.101   17.248.158.114  TLSv1…   594 Application Data
391 6.000911    17.248.158.114  192.168.1.101   TLSv1…   104 Application Data
392 6.000911    17.248.158.114  192.168.1.101   TLSv1…   301 Application Data
```

```
▾ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 508
      Version: TLS 1.2 (0x0303)
    ▾ Random: 5418e50decf3a4fecb4ac8767c10ebdc8c28d521f20e4d12bc01d0d8304e0c7d
        GMT Unix Time: Sep 17, 2014 09:34:05.000000000 CST
        Random Bytes: ecf3a4fecb4ac8767c10ebdc8c28d521f20e4d12bc01d0d8304e0c7d
      Session ID Length: 32
      Session ID: c9eb74b253411e421062db06ba940308b1129e407ca1196aa9358d9e3a6f8a66
      Cipher Suites Length: 54
    › Cipher Suites (27 suites)
      Compression Methods Length: 1
    ▾ Compression Methods (1 method)
        Compression Method: null (0)
```

内容类型：握手🤝

版本：客户端支持最高 TLS 协议的版本，TLS 1.0
随机数：发给服务器端，用于后续协商密钥，random_C，用于计算对称加密时的“主密码”

会话 ID：会话缓存机制，重连时有用
加密套件：有 27 套，按照客户端的偏好从前往后排，服务器会从中选出一个服务器也支持的加密套件
压缩方法：客户端支持的压缩方法，不支持压缩

```
Cipher Suite: Reserved (GREASE) (0x7a7a)
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
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_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
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_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003d)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008)
Cipher Suite: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
```

第二次 Server Hello，服务器返回协商的结果

```
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 110
  Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 106
      Version: TLS 1.2 (0x0303)
    Random: f040219501607d25560293012569e17e431ac87837449907ec532d40132c8903
        GMT Unix Time: Sep 23, 2097 08:32:21.000000000 CST
        Random Bytes: 01607d25560293012569e17e431ac87837449907ec532d40132c8903
      Session ID Length: 32
      Session ID: da99657c879500ec6ad89cd5616ad91fc85bf377bb90e7600de18f41ac9c8741
      Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
      Compression Method: null (0)
```

版本：服务器端确认使用 TLS 协议的版本，TLS 1.2
随机数：发给客户端，用于后续协商密钥，random_S
加密套件：服务器端选择 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384，代号 0xc02c

压缩方法：服务器端选择的压缩方法，不进行压缩

这个阶段之后，客户端和服务端知道了下列内容：
(1) SSL 版本
(2) 密钥交换、信息验证和加密算法
(3) 压缩方法
(4) 有关密钥生成的两个随机数

第三次 Certificate，服务器将数字证书链发给客户端，使客户端通过证书链认证证书的真实性，认证服务器

```
TLSv1.2 Record Layer: Handshake Protocol: Certificate
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 4206
  Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 4202
      Certificates Length: 4199
    Certificates (4199 bytes)
        Certificate Length: 1912
      Certificate: 308207743082065ca003020102021024faa73fa822c6ff9612f3ec057dce81300d06092a… (id-at-countryName=US,id-at-stateOrProvinceName=California,id-at-organizatio…
        Certificate Length: 1186
      Certificate: 3082049e30820386a003020102021005 52c7effeec292ba9f1387b07af929f300d06092a… (id-at-countryName=US,id-at-organizationName=Apple Inc.,id-at-organizational…
        Certificate Length: 1092
      Certificate: 3082044030820328a003020102020323a74300d06092a864886f70d01010b0500304231… (id-at-countryName=US,id-at-organizationName=Apple Inc.,id-at-organizational…
```

我们可以看到 Certificate 字段返回了三个数字证书，服务器的证书必须为证书列表的第一个，其后为签发服务器证书的证书，依次类推，最后一个证书为根证书签署的证书。根证书不在证书列表中，它是通过其他途径给到客户端的。

第四次 Certificate Status, Server Key Exchange, Server Hello done

```
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Certificate Status
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
  TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
```

Certificate Status

```
TLSv1.2 Record Layer: Handshake Protocol: Certificate Status
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 1462
  Handshake Protocol: Certificate Status
      Handshake Type: Certificate Status (22)
      Length: 1458
      Certificate Status Type: OCSP (1)
      OCSP Response Length: 1454
    OCSP Response
      responseStatus: successful (0)
      responseBytes
        ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
        BasicOCSPResponse
          tbsResponseData
            responderID: byKey (2)
                byKey: 668d0ca01867d9c69f7ba47fb9d9e24f5327a536
              producedAt: 2021-11-04 03:33:30 (UTC)
            responses: 1 item
              SingleResponse
                certID
                certStatus: good (0)
                    good
                  thisUpdate: 2021-11-04 03:33:30 (UTC)
                  nextUpdate: 2021-11-04 19:33:30 (UTC)
          signatureAlgorithm (sha256WithRSAEncryption)
              Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
            Padding: 0
            signature: 9012bc5d273f0b802efce617937227cac0b909d6b603d53bda8b79b326599f8b0ca63757…
          certs: 1 item
            Certificate (id-at-countryName=US,id-at-organizationName=Apple Inc.,id-at-commonName=Apple IST CA 2 OCSP Responder PG1 20211014)
```

在客户端和服务器都表明支持 OCSP stapling 后，服务器在发送完 Certificate 消息后紧跟着发送 Certificate Status 消息，提供关于证书吊销的必要信息。status_request 这个 Extension 字段用于表明客户端支持 OCSP staplin

g。OCSP 是一个检查证书吊销信息的协议，OCSP stapling 机制可以使服务器向客户端发送最新的证书吊销信息，而无需客户端去访问 CA 的证书吊销列表。

客户端 status_request 字段

```
Extension: status_request (len=5)
    Type: status_request (5)
    Length: 5
    Certificate Status Type: OCSP (1)
    Responder ID list Length: 0
    Request Extensions Length: 0
```

服务器端 status_request 字段

```
Extension: status_request (len=0)
    Type: status_request (5)
    Length: 0
```

Server Key Exchange

```
TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 115
  Handshake Protocol: Server Key Exchange
      Handshake Type: Server Key Exchange (12)
      Length: 111
    EC Diffie-Hellman Server Params
        Curve Type: named_curve (0x03)
        Named Curve: x25519 (0x001d)
        Pubkey Length: 32
        Pubkey: 6285ace7e39db9832e4856ed55541d0cd3546d2a1f24285a30886787d00d6448
      Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
          Signature Hash Algorithm Hash: SHA256 (4)
          Signature Hash Algorithm Signature: ECDSA (3)
        Signature Length: 71
        Signature: 3045022100d43cf78d468faa820718b078795f8ae22170084e0ad43a4f9514b368faf6a1…
```

服务器发送 Server Key Exchange，消息中包含了服务器这边的 EC Diffie-Hellman 算法相关参数 Pubkey，发送密钥交换算法给客户端。客户端可利用这些算法和服务器端完成“premaster_key”的交换。服务器用 RSA 私钥加密内容，并利用 SHA256 得到摘要哈希值，哈希值再用私钥加密进行签名。客户端收到后，用证书中的公钥解密

Server Hello Done

服务器发送 Server Hello Done，告知客户端服务器这边握手相关的消息发送完毕，可以进入下一个阶段

```
TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 4
  Handshake Protocol: Server Hello Done
      Handshake Type: Server Hello Done (14)
      Length: 0
```

第五次 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message，客户端在验证证书之后，会发送这则报文，CA 证书的公钥包含在操作系统中

```
TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

Client Key Exchange

```
TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 37
  Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 33
    EC Diffie-Hellman Client Params
        Pubkey Length: 32
        Pubkey: 99e188429b97944da3c0ef3aed7d46ca17cf33dad5e39406eea1b314b3843802
```

客户端发送 Client Key Exchange 消息，消息中包含客户端这边的 EC Diffie-Hellman 算法相关参数 Pubkey。前面客户端已经收到了服务器端的 Pubkey，可以立马计算出 premaster_key；等服务器端收到用户端发送的 Pubkey 后，服务器也可以计算出相同的 premaster_key，进而计算出 master_secret，也就是最终协商的密钥

Change Cipher Spec

```
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
```

可以看到 Content Type 从 Handshake 变为了 Change Cipher Spec，客户端切换成密文模式，通知服务器此消息以后客户端会以协商的密钥来加密发送数据

Encrypted Handshake 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
```

结合之前所有通信参数的 hash 值与其它相关信息生成一段数据，采用协商密钥 session secret 与算法进行加密，然后发送给服务器用于数据与握手验证

第六次服务器端 Change Cipher Spec, Encrypted Handshake Message

```
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
```

计算之前所有接收信息的 hash 值，然后解密客户端发送的 encrypted_handshake_message，验证数据和密钥正确性，验证通过之后，服务器同样发送 change

_cipher_spec 以告知客户端后续的通信都采用协商的密钥与算法进行加密通信；
encrypted_handshake_message，服务器也结合所有当前的通信参数信息生成一段数据并采用协商密钥 session secret 与算法加密并发送到客户端。

Deffile-Hellman 加密详解

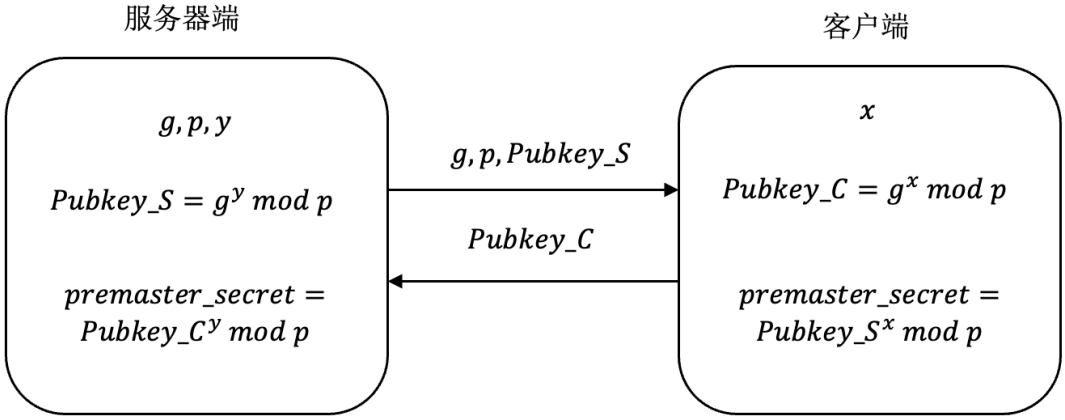
Deffie-Hellman(简称 DH)密钥交换是最早的密钥交换算法之一，它使得通信的双方能在非安全的信道中安全的交换密钥，也就是能在大庭广众之下协商出密钥，用于加密后续的通信消息。

假设 a、p 均为素数，则有以下等式：

$$\{a\text{^}1\ mod\ p, a\text{^}2\ mod\ p, \ldots, a\text{^}(p-1)\ mod\ p\} = \{1, 2, \ldots, p-1\}$$

对于任意一个数 x，若 0<x<p，则必定存在唯一的 y 使得 x=a^y mod p，

其中 0 < y < p。当 p 很大时，很难求出 y，所以它能做为 DH 秘钥交换的基础。



下面证明两者的$premaster_secret$一致

$$Pubkey\_C^y\ mod\ p = (g^x\ mod\ p)^y\ mod\ p = g^{xy}\ mod\ p$$
$$Pubkey\_S^x\ mod\ p = (g^y\ mod\ p)^x\ mod\ p = g^{xy}\ mod\ p$$

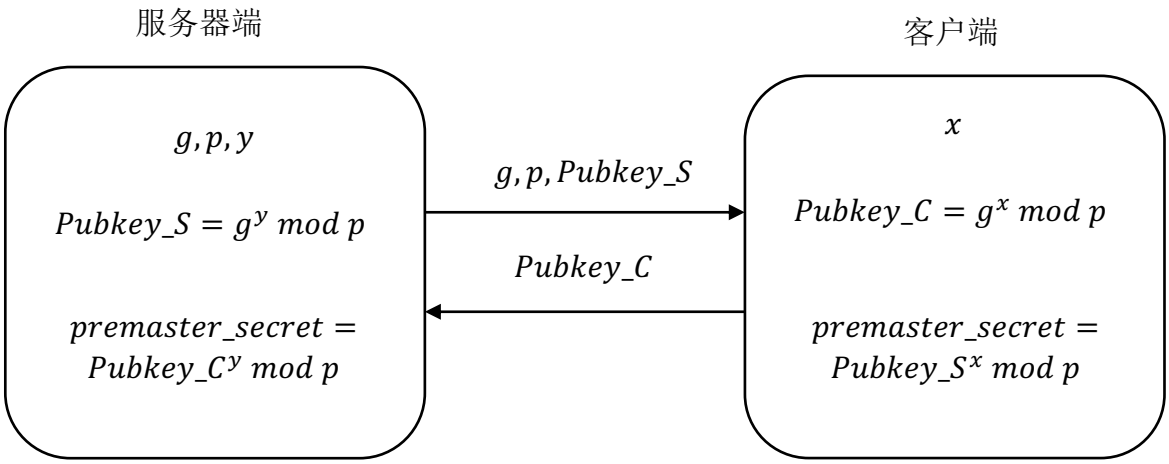
变换规则我在rsa推导过，因此不再赘述
然后再计算出$master_secret$，也就是key
无 Extened Master Secret

$$key = PRF(premaster\_secret, \text{“主密码”}, random\_C + random\_S)[0 \ldots 47]$$

具有 Extened Master Secret

$$key = PRF(master\_secret, \text{”扩展的主密钥”}, random\_S + random\_C)[0 \ldots 47]$$

双方在最开始的 hello 过程中交换了$random$


服务器端
客户端
g, p, y
$Pubkey\_S = g^y \ mod \ p$
$premaster\_secret = Pubkey\_C^y \ mod \ p$
$g, p, Pubkey\_S$
$Pubkey\_C$
x
$Pubkey\_C = g^x \ mod \ p$
$premaster\_secret = Pubkey\_S^x \ mod \ p$