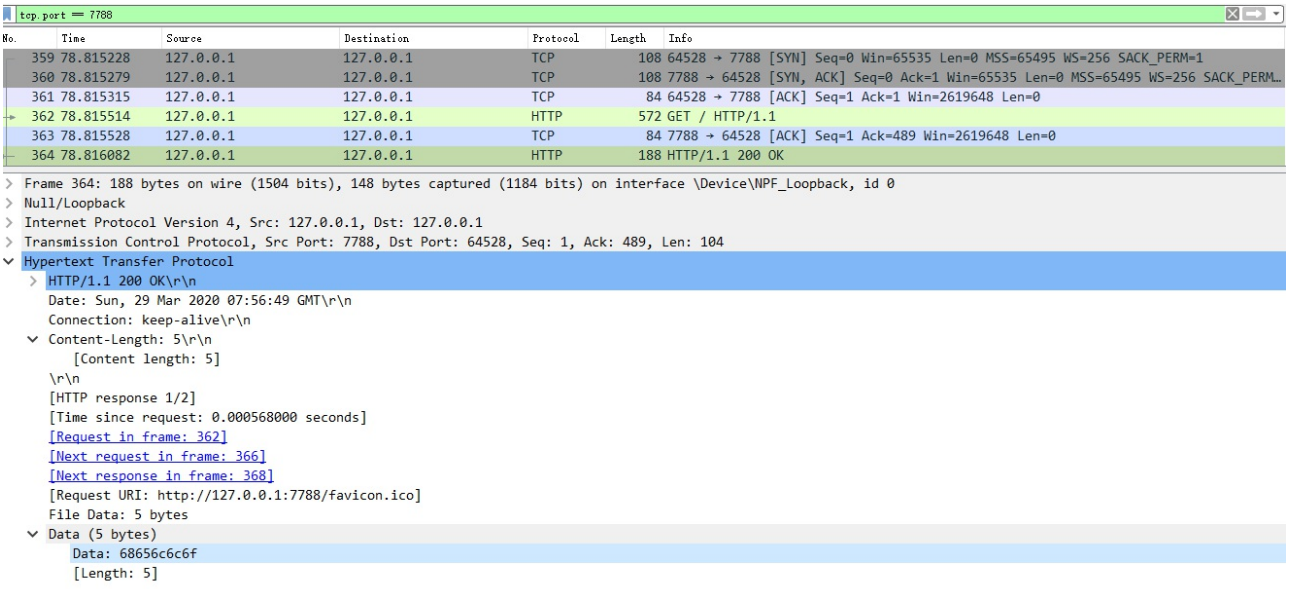
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=148 sentences=197, words=1041

1.HTTP

- HTTP 协议(HyperText Transfer Protocol，超文本传输协议)是客户端浏览器或其他程序与Web服务器之间的应用层通信协议

1.1 HTTP服务器

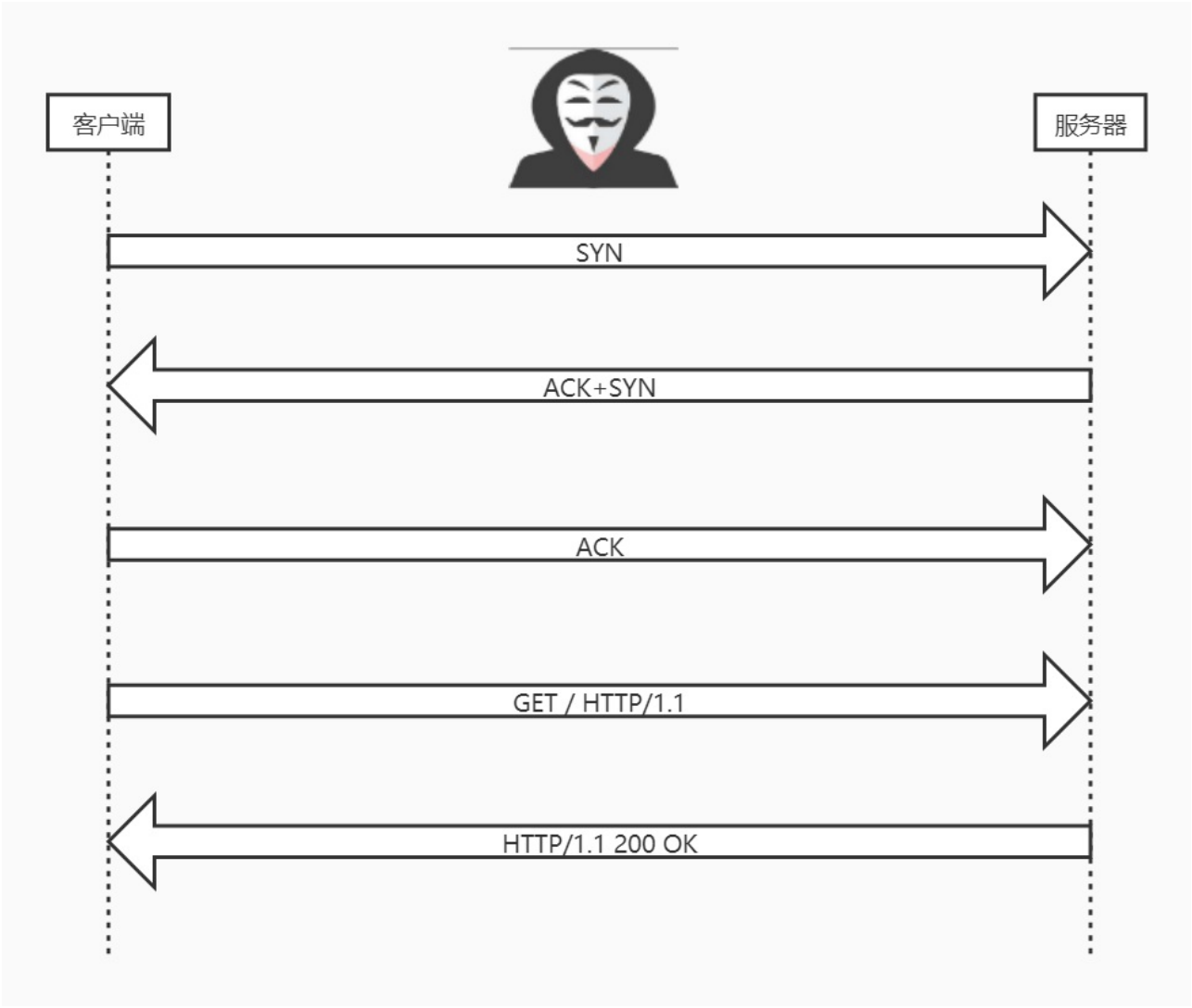



```
tcp.port == 7788
```

```
let http = require('http');
http.createServer(function (req, res) {
    let buffer = Buffer.from('hello');
    console.log(buffer);
    res.end(buffer);
}).listen(7788, () => console.log('listening 7788'));
```

1.2 HTTP三大风险

- (1)窃听风险：黑客可以获知通信内容
- (2)篡改风险：黑客可以修改通信内容
- (3)冒充风险：黑客可以冒充他人身份参与通信



2. HTTPS

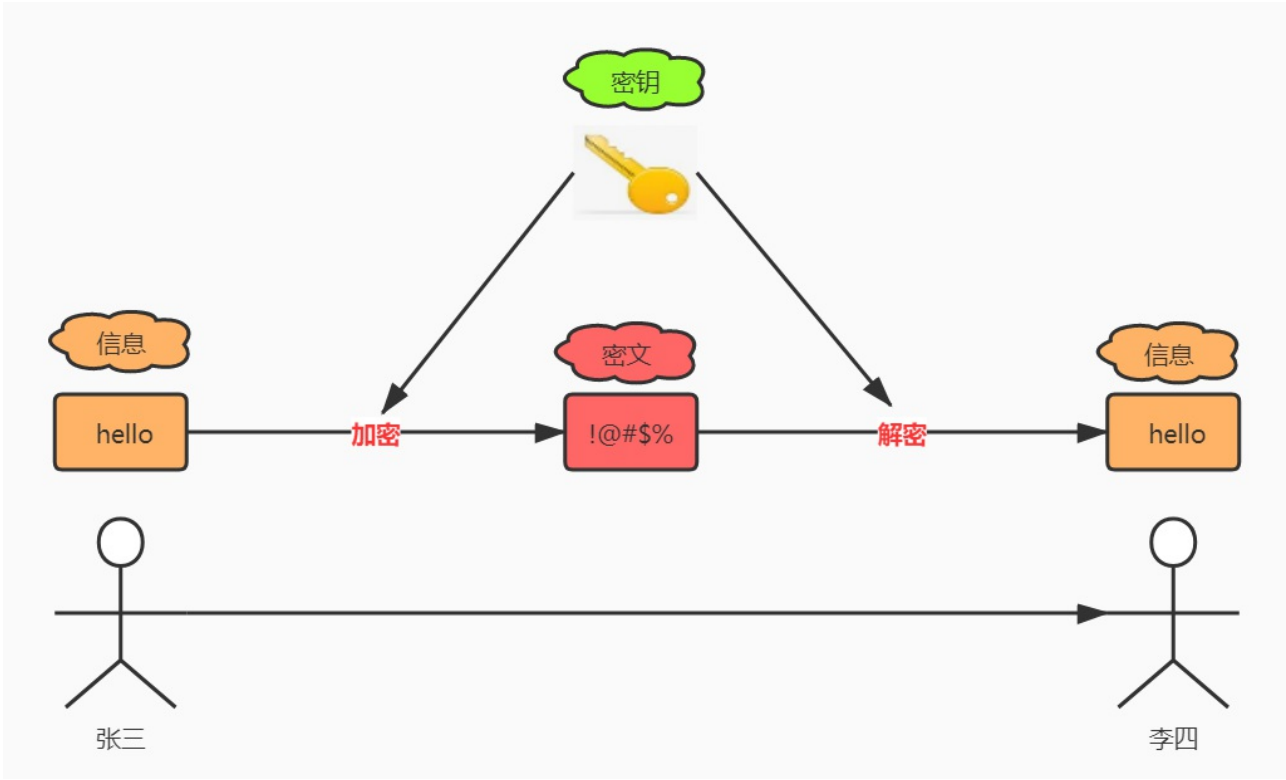
- HTTP = HTTP+TLS/SSL

风险 对策 方法 信息窃听 信息加密 对称加密 AES 密钥传递 密钥协商 非对称加密(RSA和ECC) 信息篡改 完整性校验 散列算法(MD5和SHA) 身份冒充 CA权威机构 散列算法(MD5和SHA)+RSA签名

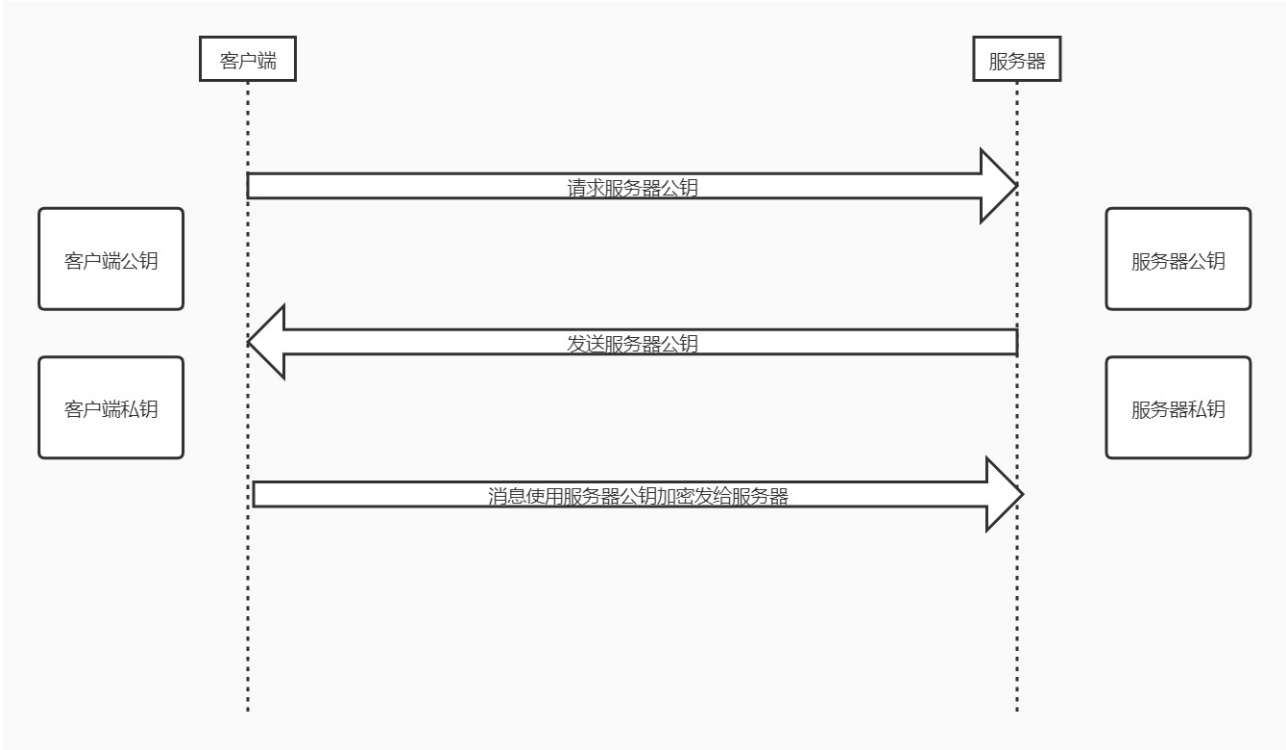
3. 加密算法

3.1 对称加密 AES

- 加密和解密使用同一个密钥



3.2 非对称加密

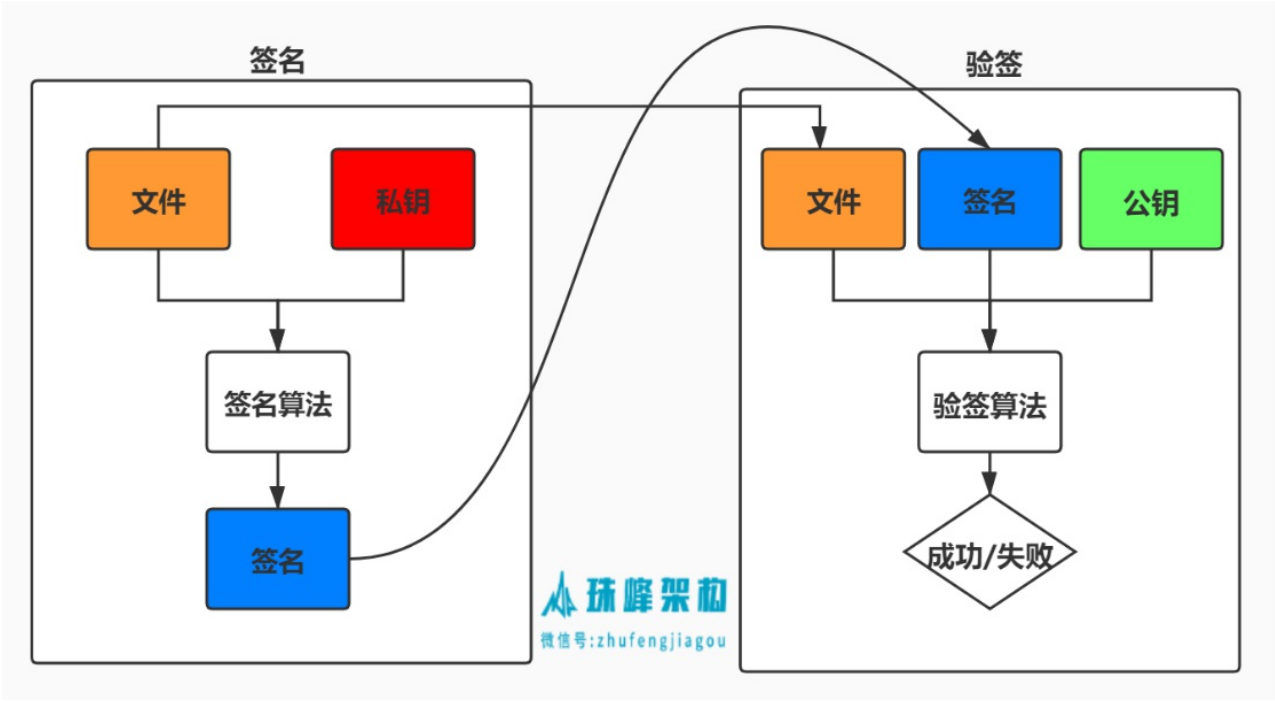


3.3 哈希算法

- 哈希函数的作用是给一个任意长度的数据生成出一个固定长度的数据
  - 安全性 可以从给定的数据X计算出哈希值Y，但不能从哈希值Y计算机数据X
  - 独一无二 不同的数据一定会产出不同的哈希值
  - 长度固定 不管输入多大的数据,输出长度都是固定的

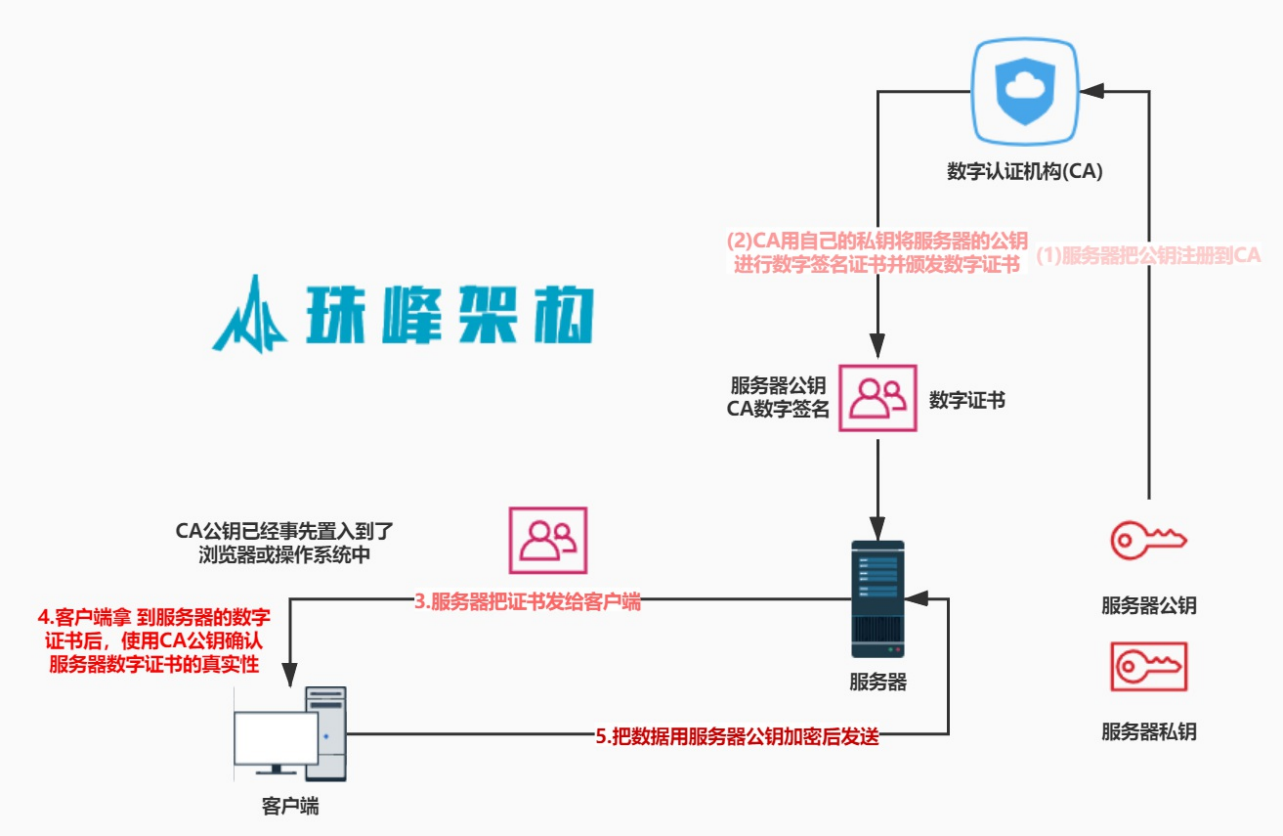
3.4 签名

- 数字签名的基本原理是用私钥去签名，而用公钥去验证签名



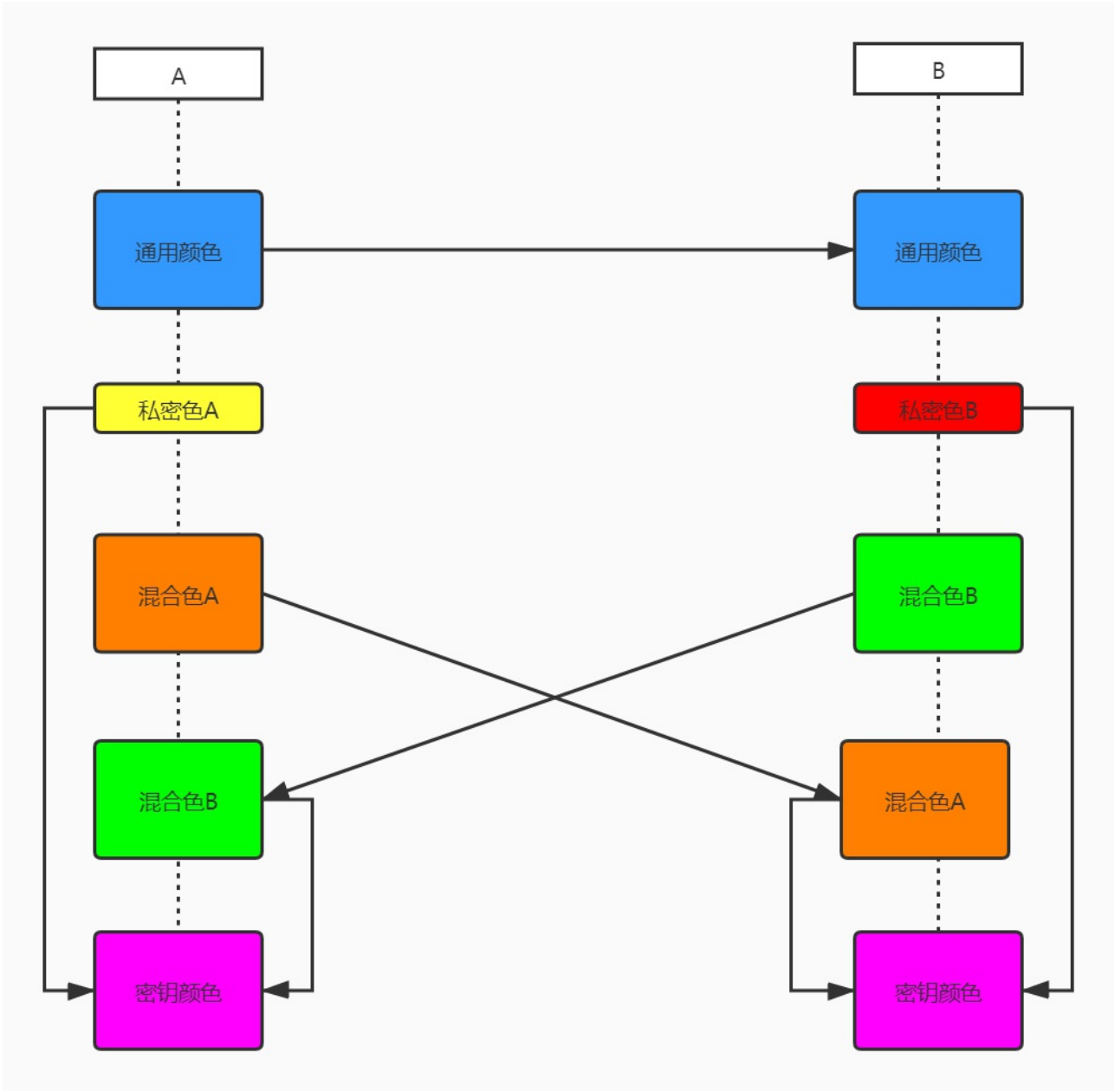
3.5 数字证书

- 数字证书是一个由可信的第三方发出的，用来证明所有人身份以及所有人拥有某个公钥的电子文件



3.6 密钥交换 #

- Diffie-Hellman算法是一种密钥交换协议，它可以让双方在不泄漏密钥的情况下协商出一个密钥来

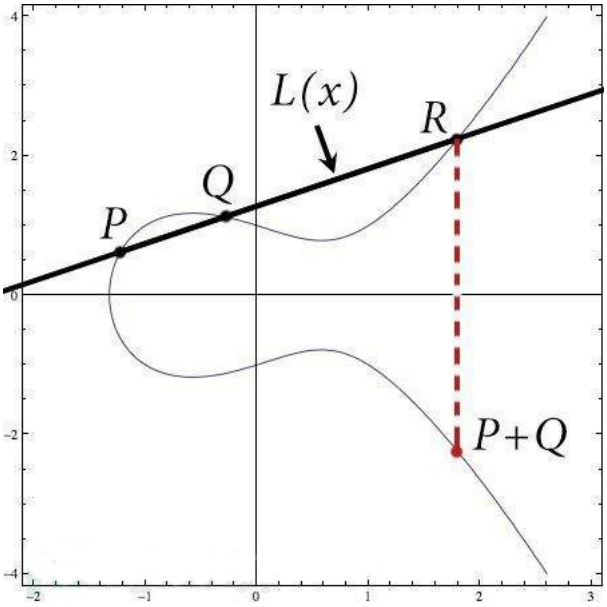


3.7 ECC

- 椭圆曲线加密算法(ECC) 是基于椭圆曲线数学的一种公钥加密的算法

```
let basic = 3;
let a = 5;
let basicA = basic * a;
let b = 7;
let basicB = basic * b;

console.log(a * basicB);
console.log(b * basicA);
```



4. 加密过程

- firefox: https://47.105.191.39 ip.addr ==47.105.191.39 and tls

4.1 ClientHello

- 在一次新的握手流程中，客户端先发送ClientHello
  - Version 协议版本
  - Random 包含32个字节的随机数 28随机数字节+4字节时间戳,随机数是为了保证每一次连接者是独立无二的
  - Cipher Suites 客户端支持的所有密码套件
  - Extensions 扩展的额外数据

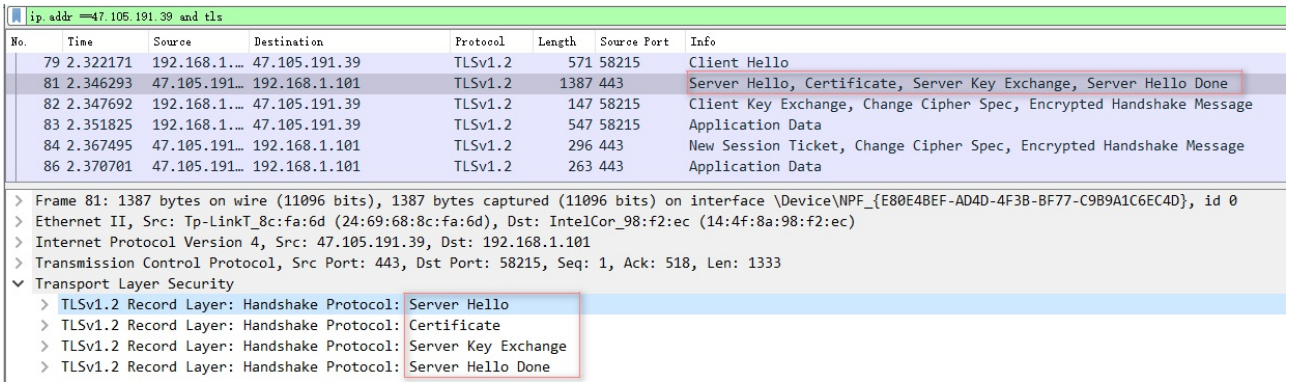
ip.addr ==47.105.191.39 and tls

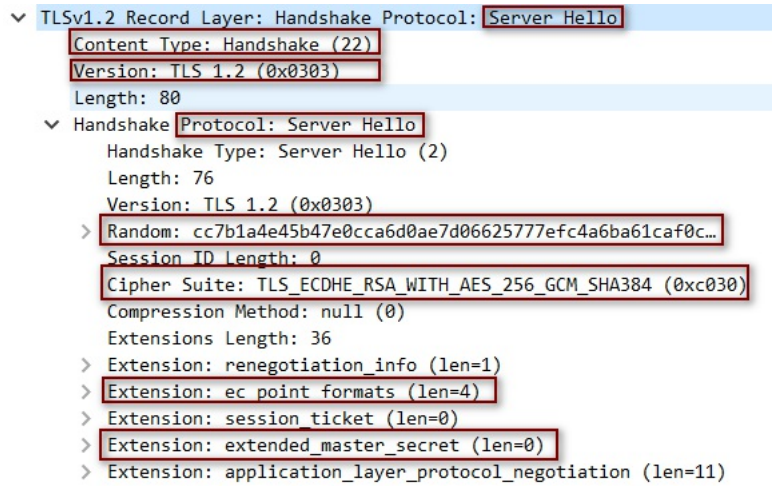
No.	Time	Source	Destination	Protocol	Length	Source Port	Info
79	2.322171	192.168.1.…	47.105.191.39	TLSv1.2	571	58215	Client Hello
81	2.346293	47.105.191…	192.168.1.101	TLSv1.2	1387	443	Server Hello, Certificate, Server Key Exchange, Server Hello Done
82	2.347692	192.168.1.…	47.105.191.39	TLSv1.2	147	58215	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
83	2.351825	192.168.1.…	47.105.191.39	TLSv1.2	547	58215	Application Data
84	2.367495	47.105.191…	192.168.1.101	TLSv1.2	296	443	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
86	2.370701	47.105.191…	192.168.1.101	TLSv1.2	263	443	Application Data

```
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: 52f267d9ab2cacc5d43f36887b49d51452a8550dbf597f78…
    Session ID Length: 32
    Session ID: ac9e05de89b56126fa396ff772bb90a4f69ef300bf18970f…
    Cipher Suites Length: 36
    Cipher Suites (18 suites)
        Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
        Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
```

4.2 ServerHello

- 将服务器选择的连接参数发回给客户端，消息结构和ClientHello类似，每个字段只包含一个选项





4.3 Certificate

- Certificate消息发送X.509证书

```
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Server Hello
  TLSv1.2 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 929
    Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 925
        Certificates Length: 922
      Certificates (922 bytes)
          Certificate Length: 919
        Certificate: 308203933082027ba003020102021022bc066b8e47f6464e… (pkcs-9-at-emailAddress=zf@qq.com,id-at-commonName=47.105.191.39,id-at-organizationalUnitName=zf
          signedCertificate
          algorithmIdentifier (sha256WithRSAEncryption)
          Padding: 0
          encrypted: 1b2c79aaa91cbf0d8018bda03a8e56a8dcbe2cdc4830fe8f…
```

4.4 ServerKeyExchange

- ServerKeyExchange的目的在于发送交换密钥的参数

4.5 Server Hello Done

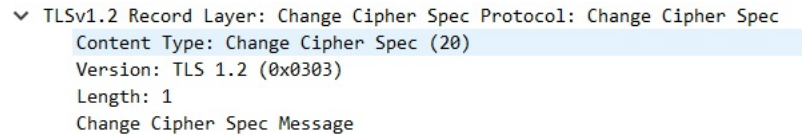
- ClientKeyExchange消息携带客户端为密钥交换的所有信息

4.6 ClientKeyExchange

- ClientKeyExchange消息携带客户端为密钥交换的所有信息

4.7 ChangeCipherSpec

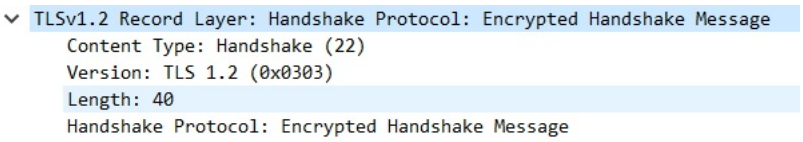
- ChangeCipherSpec表示客户端已经得到了连接参数的足够信息，已生成加密密钥，并切换到了加密模式



4.8 EncryptedHandshakeMessage

- 这个报文的目的就是告诉对端自己在整个握手过程中收到了什么数据，发送了什么数据,来保证中间没人篡改报文
- 其次这个报文作用就是确认秘钥的正确性。因为 Encrypted handshake message是使用对称秘钥进行加密的第一个报文，如果这个报文加解密校验成功，那么就说明对称秘钥是正确的
- 计算方法就将之前所有的握手数据(包括接受和发送)计算哈希运算,然后就是使用协商好的对称密钥进行加密

```
加密(SHA(客户端随机数+服务器随机数))
```



4.9 New Session Ticket

- SSL 中的 session 会跟 HTTP 的 session 类似,都是用来保存客户端和服务端之间交互的一些记录
- 如果服务端允许使用 Session ID,客户端的 Client Hello 带上 Session ID，服务端复用 Session ID 后，会直接略过协商加密密钥的过程，直接发出一个 Change Cipher Spec 报文,然后就是加密的握手信息报文
- 在服务器发送New Session Ticket消息
    - Type 类型
    - Version 版本
    - Length长度
    - Session Ticket Lifetime Hint 表示Ticket的剩余有效时间
    - Session Ticket 会话标识

5.openssl

- linux中的openssl 是SSL/TLS协议和多种加密算法的开源实现
- openssl包括libcrypto实现算法、libssl 实现TLS/SSL协议,libssl是基于会话的，实现了身份认证 ,数据加密, 会话完整性的一个TLS/SSL的库。

5.1 查看版本

```
openssl version -a
```

5.2 摘要算法

```
openssl dgst -help
```

- file... files to digest (default is stdin) 生成摘要的文件
- -out outfile Output to filename rather than stdout 输出文件
- -sign val Sign digest using private key 使用私钥签名
- -verify val Verify a signature using public key 使用公钥验证签名
- -signature infile File with signature to verify 验证的签名的文件
- -hex Print as hex dump 以16进制打印
- -hmac val Create hashed MAC with key 创建hashed过的消息摘要

```
echo 123 > msg.txt
openssl dgst -md5  msg.txt
openssl dgst -sha1  msg.txt
openssl dgst -sha256  msg.txt
```

5.3 对称加密

```
openssl enc -help
```

- -in infile Input file 输入要加密的文件
- -out outfile Output file 输出加密后的文件
- -e Encrypt 加密
- -d Decrypt 解密
- -a Base64 encode/decode, depending on encryption flag Base64编码和解码
- -pass val Passphrase source 指定密码
- -k val Passphrase 指定密码

```
openssl  enc -e -aes128 -a -k 123456789  -in msg.txt  -out enc_msg.txt
openssl  enc -d -aes128 -a -k 123456789  -in enc_msg.txt  -out dec_msg.txt

openssl  enc -e -aes128 -a -pass pass:123456  -in msg.txt  -out enc_msg.txt -P
openssl  enc -d -aes128 -a -pass pass:123456  -in enc_msg.txt  -out dec_msg.txt
```

5.5 RSA非对称加密

5.5.1 RSA

5.5.1.1 RSA生成公私钥

```
openssl genrsa -help
```

- -aes256 是使用aes256算法加密这个私钥
- -passout 指定加密的密钥
- -out output the key to file 指明输出文件
- -in 指定输入文件
- -pubout 输出公钥信息； 根据私钥的信息得出公钥

```
openssl genrsa -aes256 -passout pass:123456 -out private.key 2048

openssl genrsa  -out private.key 1024

openssl rsa -pubout -in private.key  -out  public.key
```

5.5.1.2 RSA加解密

```
openssl rsautl -help
```

- -in infile Input file 输入文件（待加密或待解密的文件）
- -out outfile Output file 输出文件 加密解密后的文件
- -inkey val Input key 输入加密的公钥
- -encrypt Encrypt with public key 使用公钥加密
- -decrypt Decrypt with private key 使用私钥解密
- -pubin Input is an RSA public 表明输入的是公钥(默认是私钥)
- -sign Sign with private key 表明使用私钥签名
- -verify Verify with public key 表明使用公钥验证签名
- -hexdump Hex dump output 以十六进制形式输出

```
openssl rsautl -encrypt -inkey public.key  -pubin  -in msg.txt  -out enc.msg.txt

openssl rsautl -decrypt -inkey private.key -in enc.msg.txt   -out  dec.msg.txt
```

5.5.1.3 数字签名

```
openssl dgst -sign private.key -sha256 -out  sign.msg.txt  msg.txt

openssl dgst -verify  public.key -sha256 -signature sign.msg.txt  msg.txt
```

5.5.2 ECDSA

5.5.2.1 生成公私钥

```
openssl ecparam -genkey  -name secp256k1 -out ec.private.key

openssl ec -in ec.private.key -pubout -out ec.public.key
```

5.5.2.2 数字签名

```
openssl  dgst -sign ec.private.key  -sha256 -out sign.msg  msg.txt

openssl dgst -verify ec.public.key -sha256 -signature sign.msg  msg.txt
```

6.证书体系（PKI）

- 为了检查公钥不是服务器就需要引入权威第三方
- 数字证书就是权威第三方发布的并包括
    - Issuer (证书的发布机构)：哪个权威第三方发布的证书
    - Valid from,Valid to(证书的有效期) 过了有效期限，证书就会作废
    - Public key(公钥) 权威第三方给申请者配发的公钥
    - Subject(主题): 证书所有者,一般是公司名称、机构名称、公司网站的网址等
    - Signature algorithm(签名所使用的算法): 使用哪个算法加密了指纹，指纹的加密结果就是数字签名
    - Thumbprint, Thumbprint algorithm (指纹以及指纹算法) 这是个加密后的结果，是用来确保证书完整性的，保证证书不被篡改

6.1. 生成自签的根证书

- -new 新的签名请求
- -x509 输出x509格式的证书
- -key 私钥
- -out 输出文件
- -days 证书有效期天数
- -subject 请求主体

```
openssl genrsa -out ca.private.key  2048

openssl req -new -x509 -key ca.private.key  -out ca.crt  -days 365  -subj /C=CN/ST=BeiJing/L=BeiJing/O=ca/OU=ca/CN=www.ca.com/emailAddress=ca@qq.com
```

6.2.服务器证书申请

```
openssl genrsa -out server.private.key 2048

openssl req -new -key server.private.key -out server.csr -subj
/C=CN/ST=BeiJing/L=BeiJing/O=47.105.67.214/OU=47.105.67.214/CN=47.105.67.214/emailAddress=47.105.67.214@qq.com

openssl x509 -req -days 365  -CA ca.crt -CAkey ca.private.key -CAcreateserial  -in  server.csr  -out server.crt
```

7.nginx

7.1 安装

```
yum -y install gcc gcc-c++ pcre-devel zlib-devel

wget https:
wget https:
tar zxf nginx-1.12.2.tar.gz
tar zxf openssl-1.1.0h
cd nginx-1.12.2

groupadd nginx

useradd nginx  -M -s /sbin/nologin -g nginx

mkdir -p /usr/nginx
mkdir -p /usr/nginx/logs
mkdir -p /usr/nginx/cache

./configure  --prefix=/usr/nginx   --with-http_ssl_module --with-openssl=/root/openssl-1.1.0h        --with-http_ssl_module --user=nginx --group=nginx

make
make install

export PATH=/usr/nginx/sbin:$PATH
nginx -t
nginx -V
nginx
netstat -ntlp
```

7.2 布署证书

- 配置文件 /usr/nginx
- 重启 nginx -s reload

```
    server {
        listen       443 ssl;
        server_name  localhost;

+       ssl_certificate      /root/server.crt;
+       ssl_certificate_key  /root/server.private.key;

        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;

        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        location / {
            root   html;
            index  index.html index.htm;
        }
    }
```

8. tls安全

8.1 证书吊销

- 证书吊销列表分发点 (CRL Distribution Point ，简称 CDP) 是含在数字证书中的一个可以共各种应用软件自动下载的最新的 CRL 的位置信息,一般 CA 每隔一定时间 (几天或几个月) 才发布新的吊销列表
- OCSP(Online Certificate Status Protocol)证书状态在线查询协议，是IETF颁布的用于实时查询数字证书在某一时间是否有效的标准。

8.1 证书链

- 一个证书链就是能溯源到一个可信根证书的有序证书列表
- 使用证书链的原因
    - 保证根证书安全
    - 交叉证书 用已有的根证书签署新的根证书
    - 划分二级CA
    - 委派 签发给一个组织或者公司一个二级CA，但限制用于签署其自用的证书，只能签署他们自己所拥有的域名的证书。

8.2 多域名证书与泛域名证书

- 建议给每个域名一个单独的证书
- *.xxxx.com或*.xxxx.org 就是泛域名

8.2 安全的优化

- 使用服务器优先的更安全的密码套件
- 密钥算法与加密强度
- 注意保护私钥，定期更换
- 选择可靠的CA权威机构
- 前向安全保密