HTTPS原理

对称加密

Kotlin

```
f(k, data) = X 加密
f(k, X) = data 解密

问题，k的安全性问题，如何安全告知客户端k
```

非对称加密

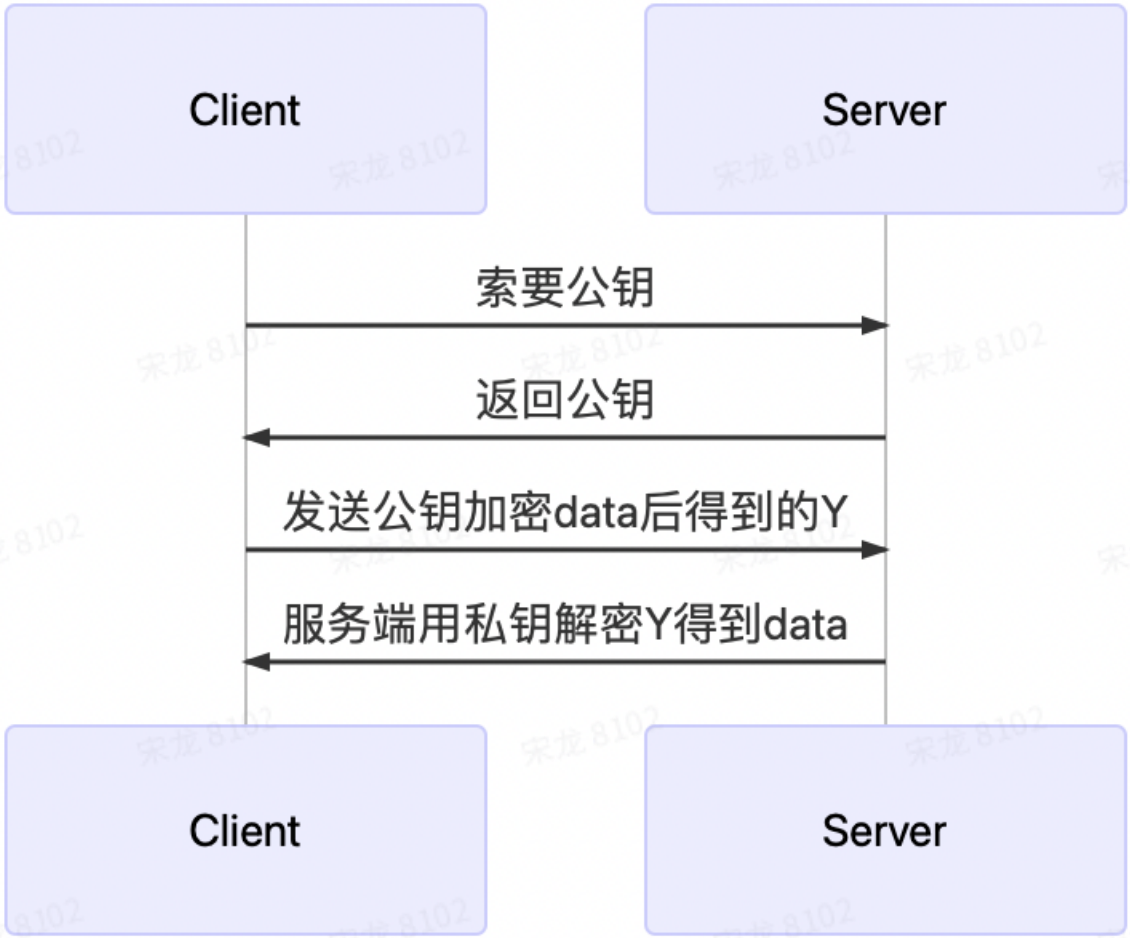
客户端到服务端安全，服务端到客户端不安全

Pk 公钥，sk私钥

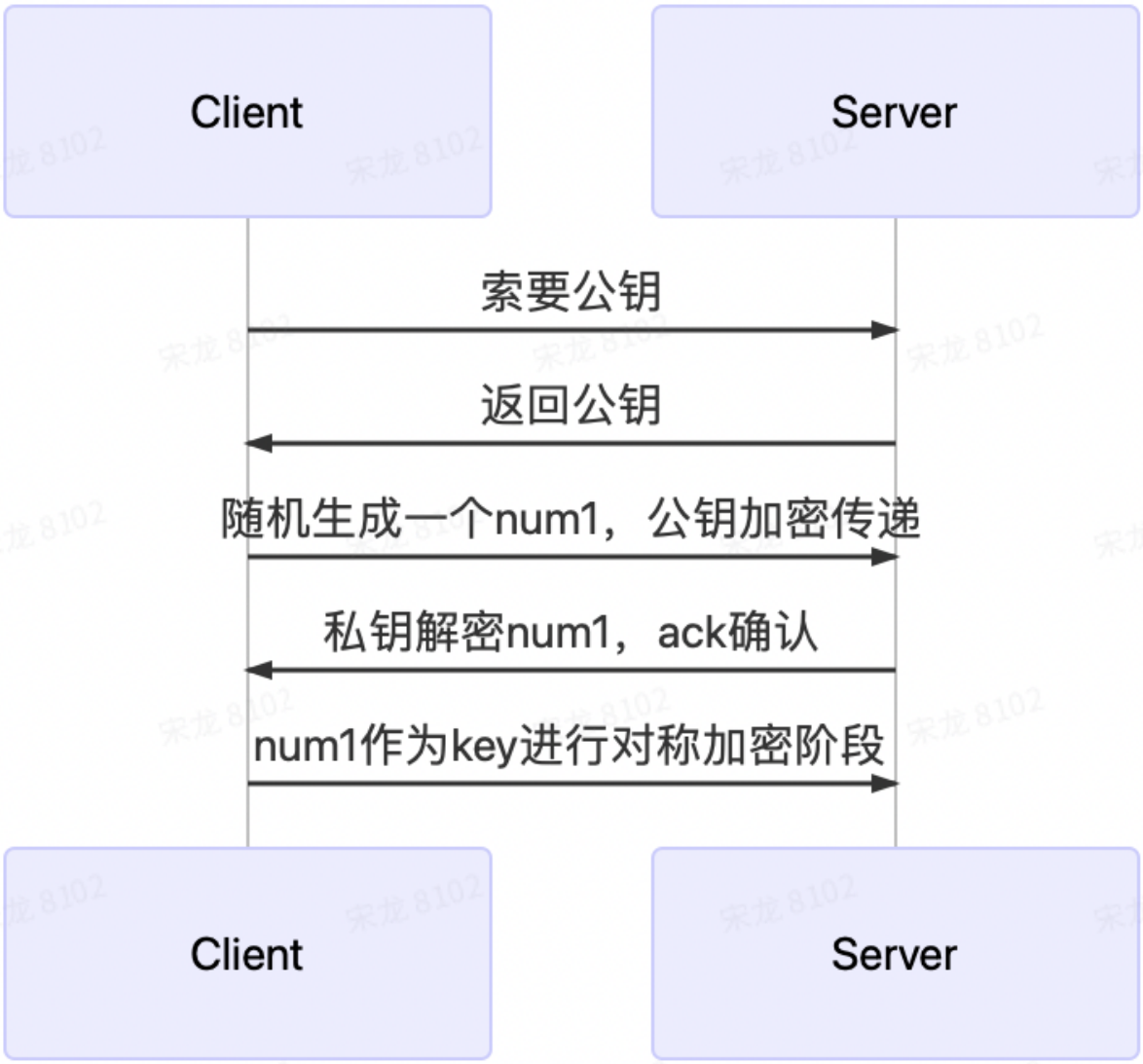
Ruby

```
f(pk, data) = Y 公钥加密
f(sk, Y) = data 私钥解密

f(sk, data) = Y' 私钥加密
f(pubk, Y') = data 公钥解密
```


Client
Server
索要公钥
返回公钥
发送公钥加密data后得到的Y
服务端用私钥解密Y得到data
Client
Server

对称+非对称



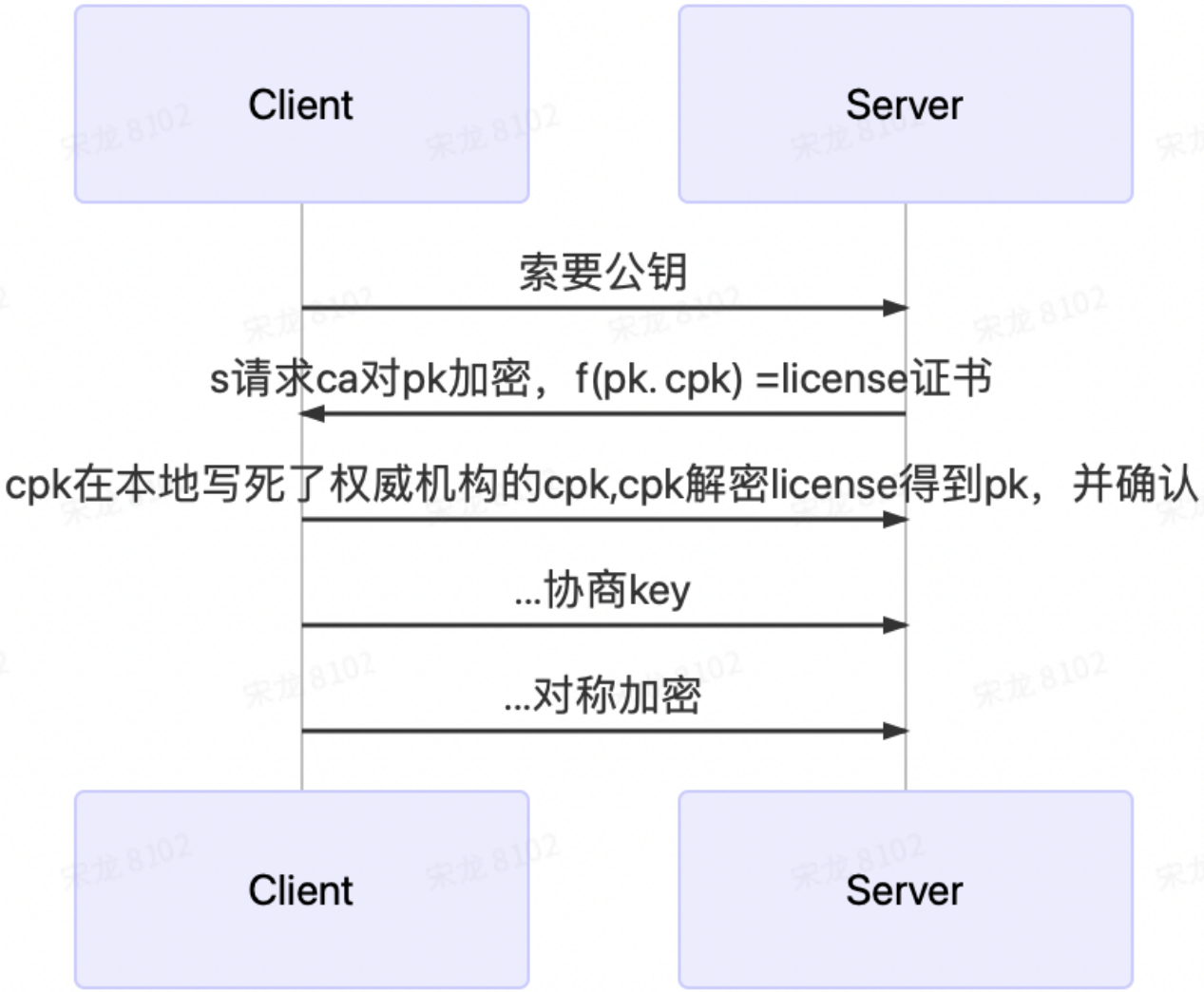
存在问题，中间人攻击，第一次请求，c向s请索要公钥，直接被中间人拦截并返回错误的key。

CA引入

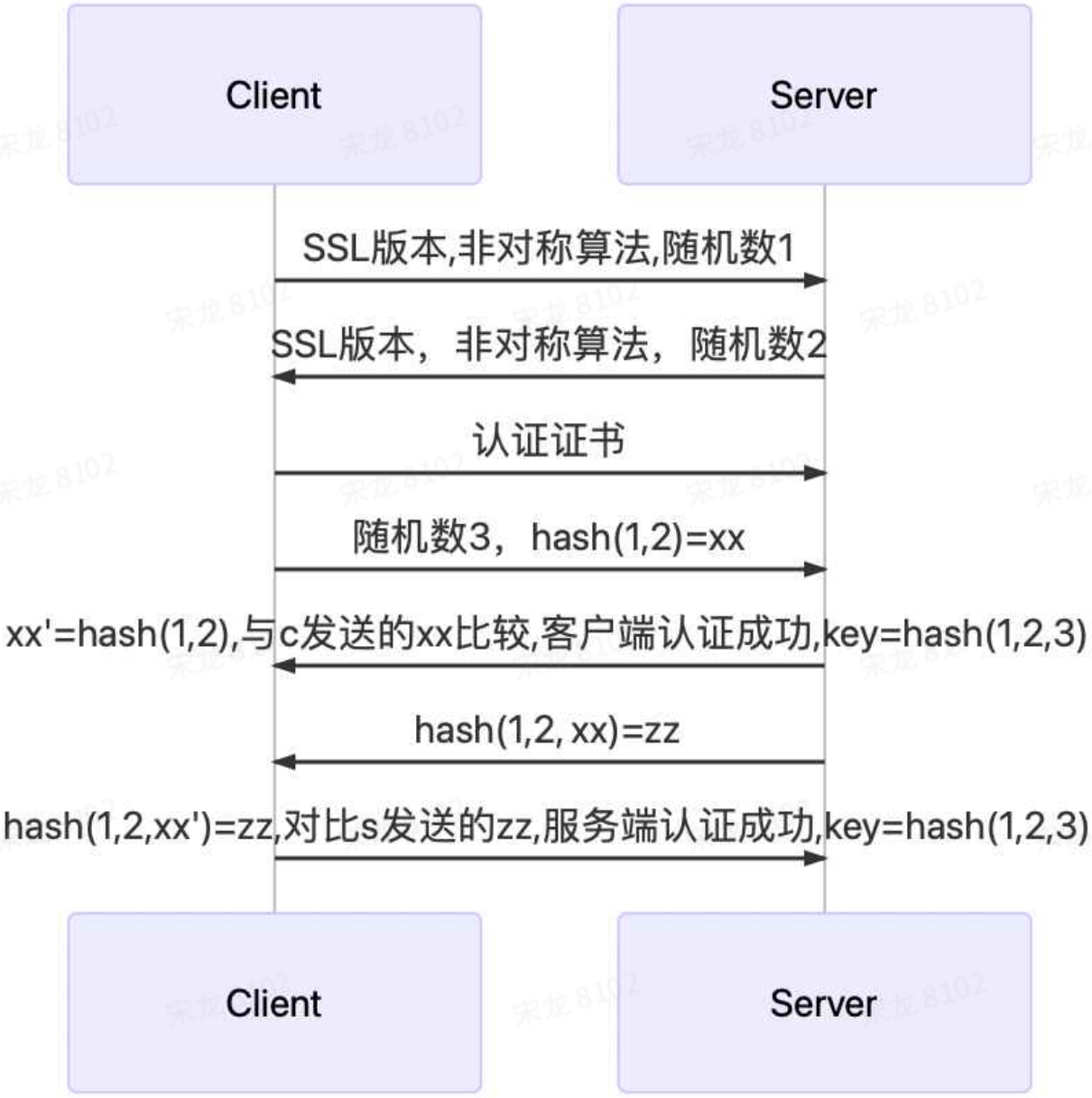
（保证客户端能从license中拿到正确的key，key不被中间人恶意篡改）

CA机构拥有（cpk, csk）

server生成（pk，sk）


Client
Server
索要公钥
s请求ca对pk加密，f(pk. cpk) =license证书
cpk在本地写死了权威机构的cpk,cpk解密license得到pk，并确认
...协商key
...对称加密
Client
Server

key的协商过程



整个过程中，key不参与传递，而是通过前3次通讯生成的随机数的hash结果进行比较鉴别

总结

HTTPS = 非对称加密+对称加密+HASH+CA