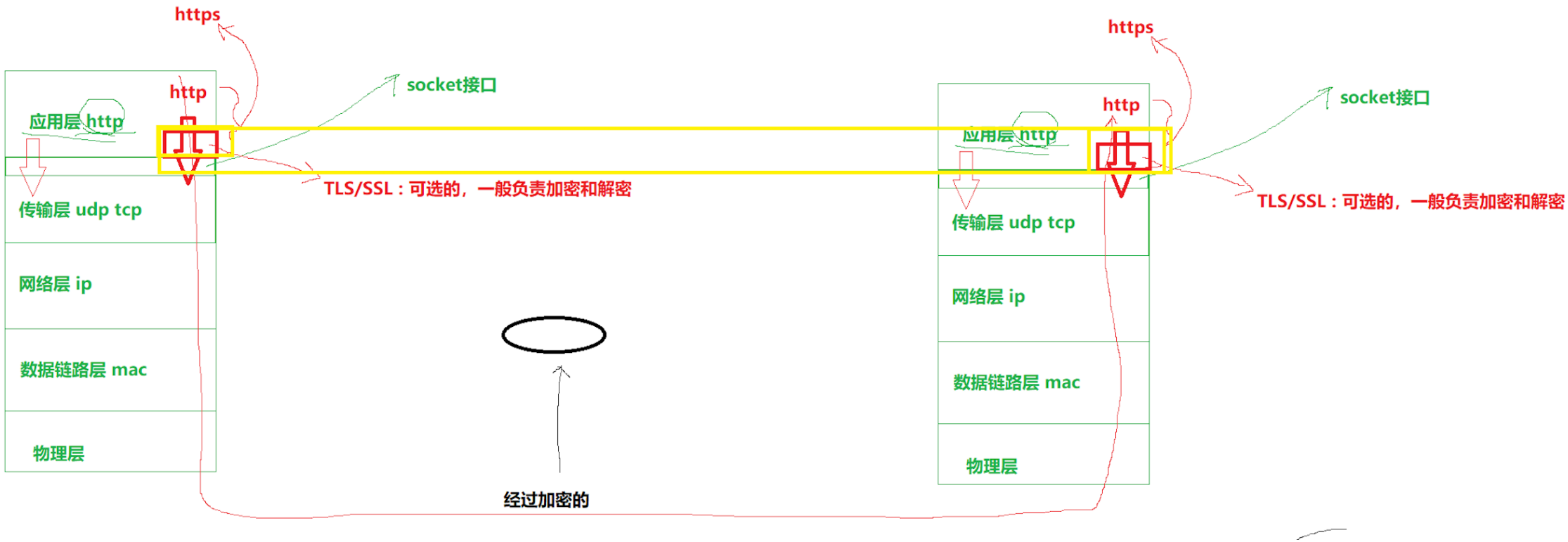
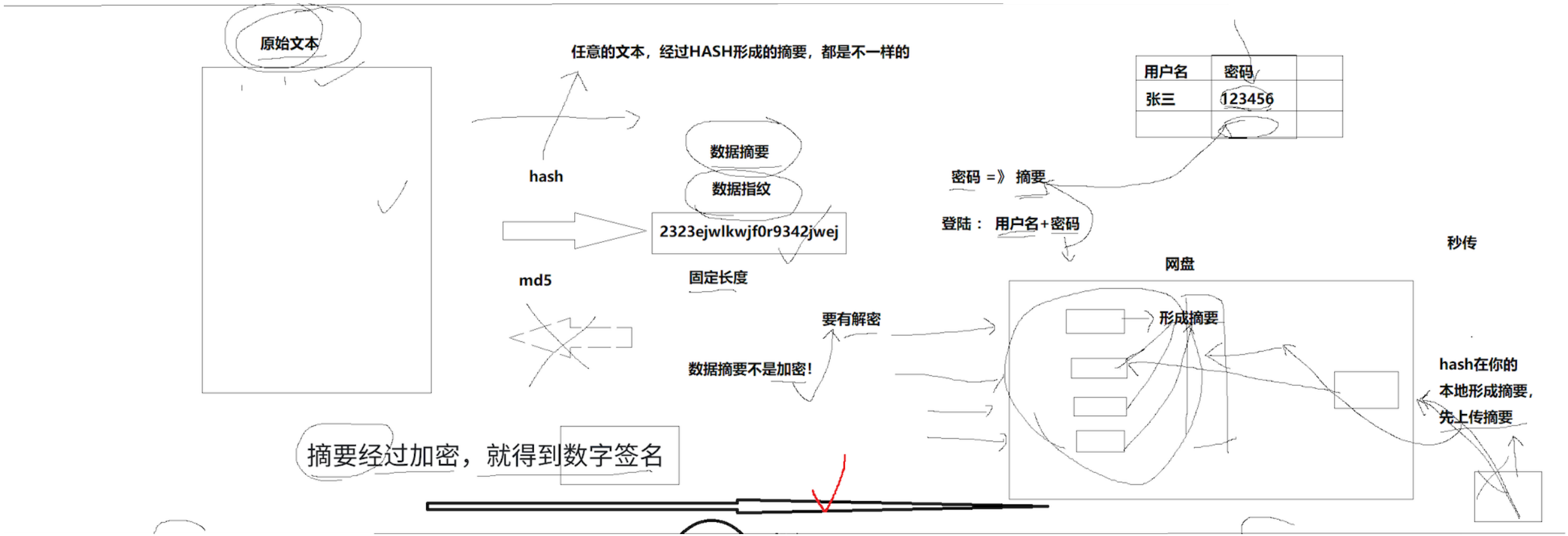
0305https

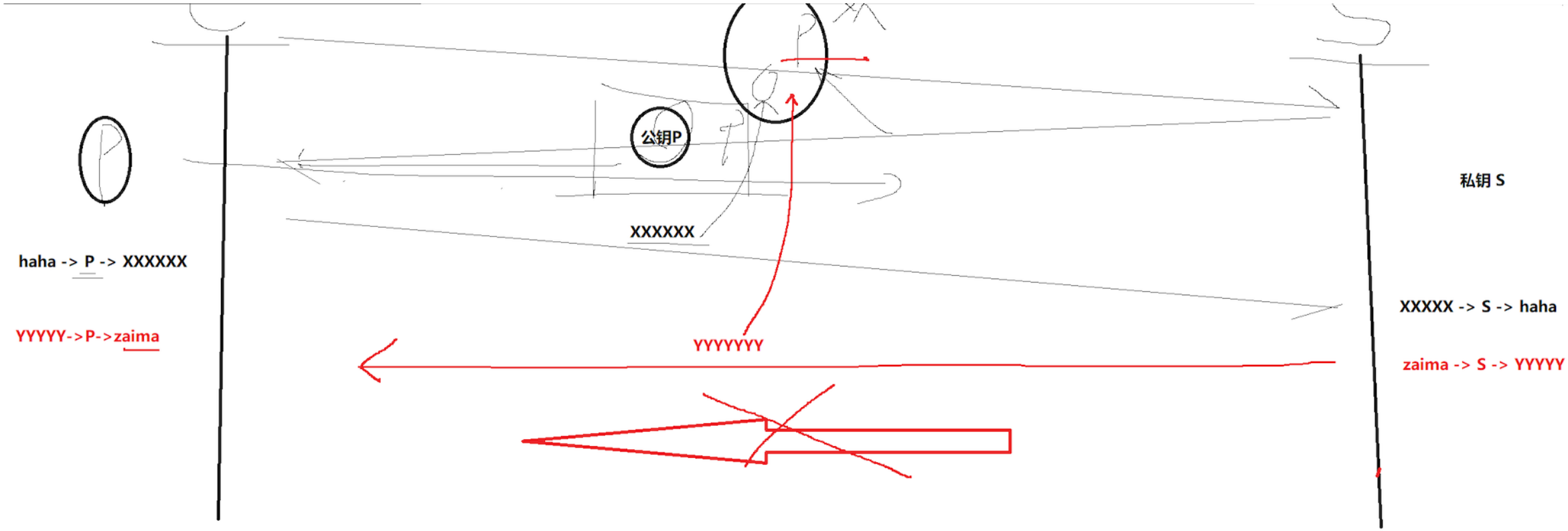
安全：加密 解密

如何正确理解安全：网络安全

安全：破解的成本远远大于破解的收益

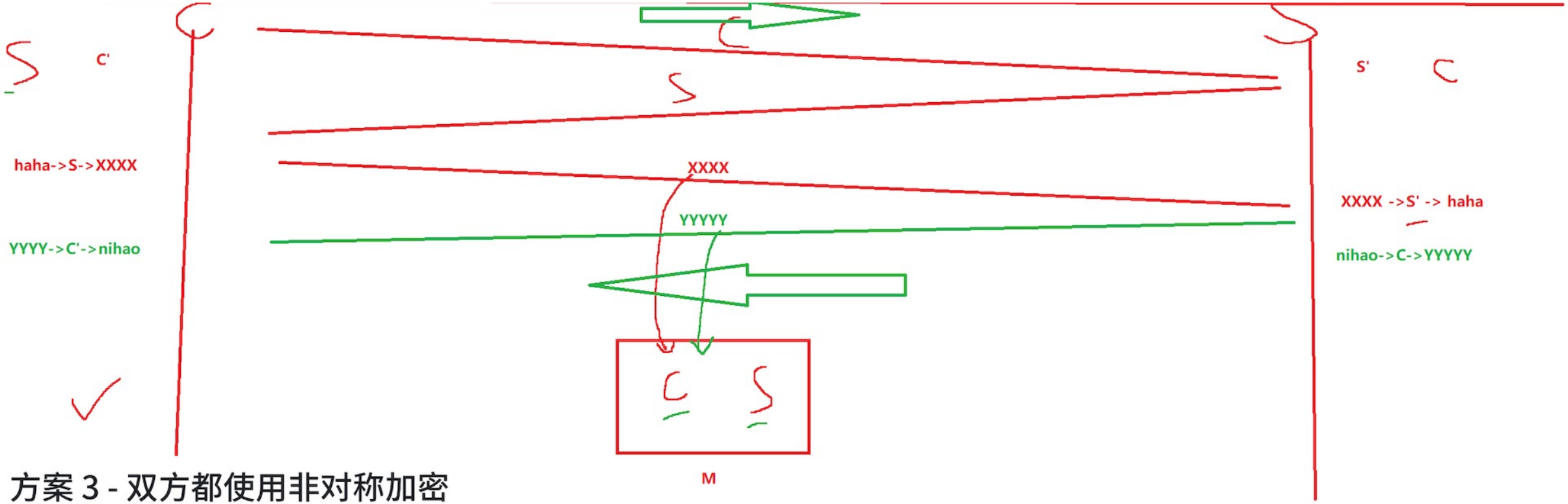



原始文本
任意的文本，经过HASH形成的摘要，都是不一样的
hash
数据摘要
数据指纹
2323ejwlkwjf0r9342jwej
固定长度
md5
用户名
密码
张三
123456
密码 =》 摘要
登陆： 用户名+密码
秒传
网盘
形成摘要
要有解密
数据摘要不是加密！
hash在你的
本地形成摘要，
先上传摘要
摘要经过加密，就得到数字签名



客户端使用非对称加密

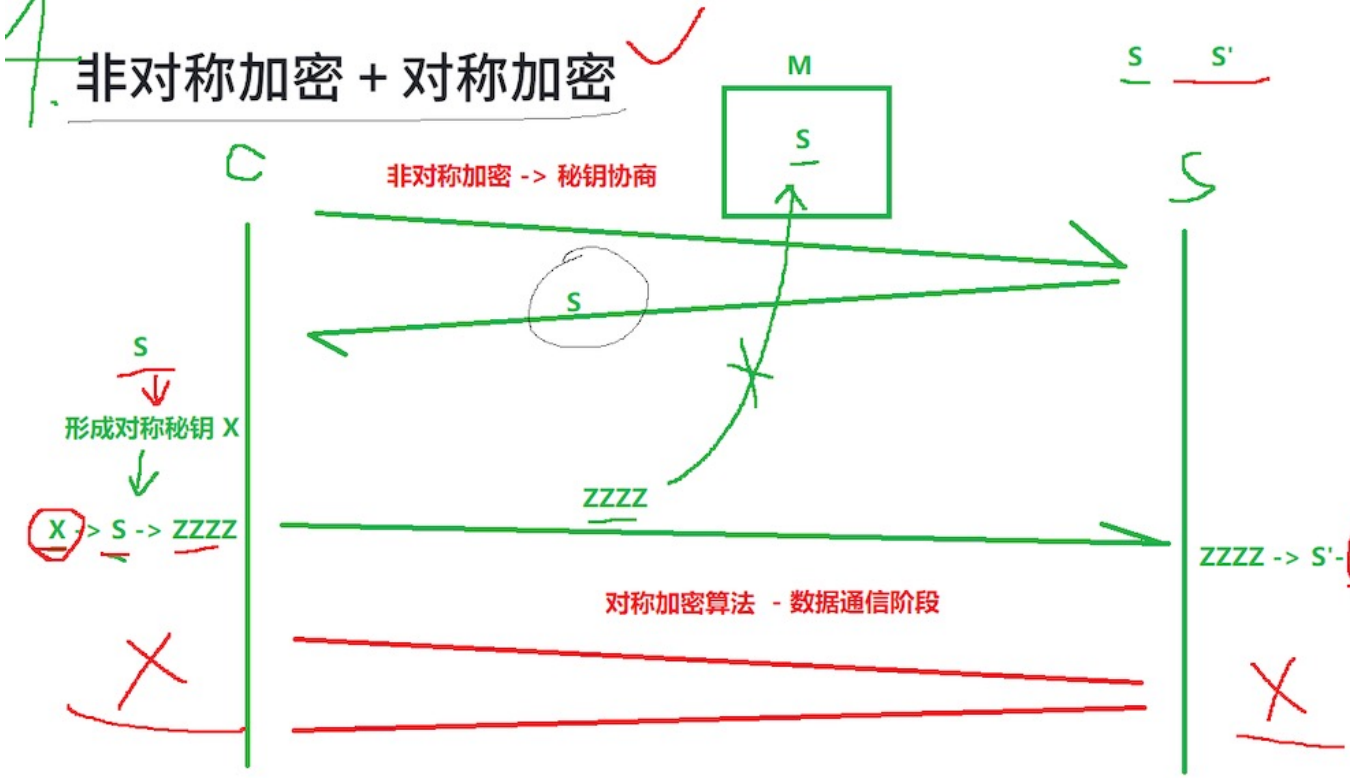
可以保证客户端->服务端的安全
但是保证不了服务端->客户端的安全



方案 3 - 双方都使用非对称加密

效率很低

非对称加密 + 对称加密

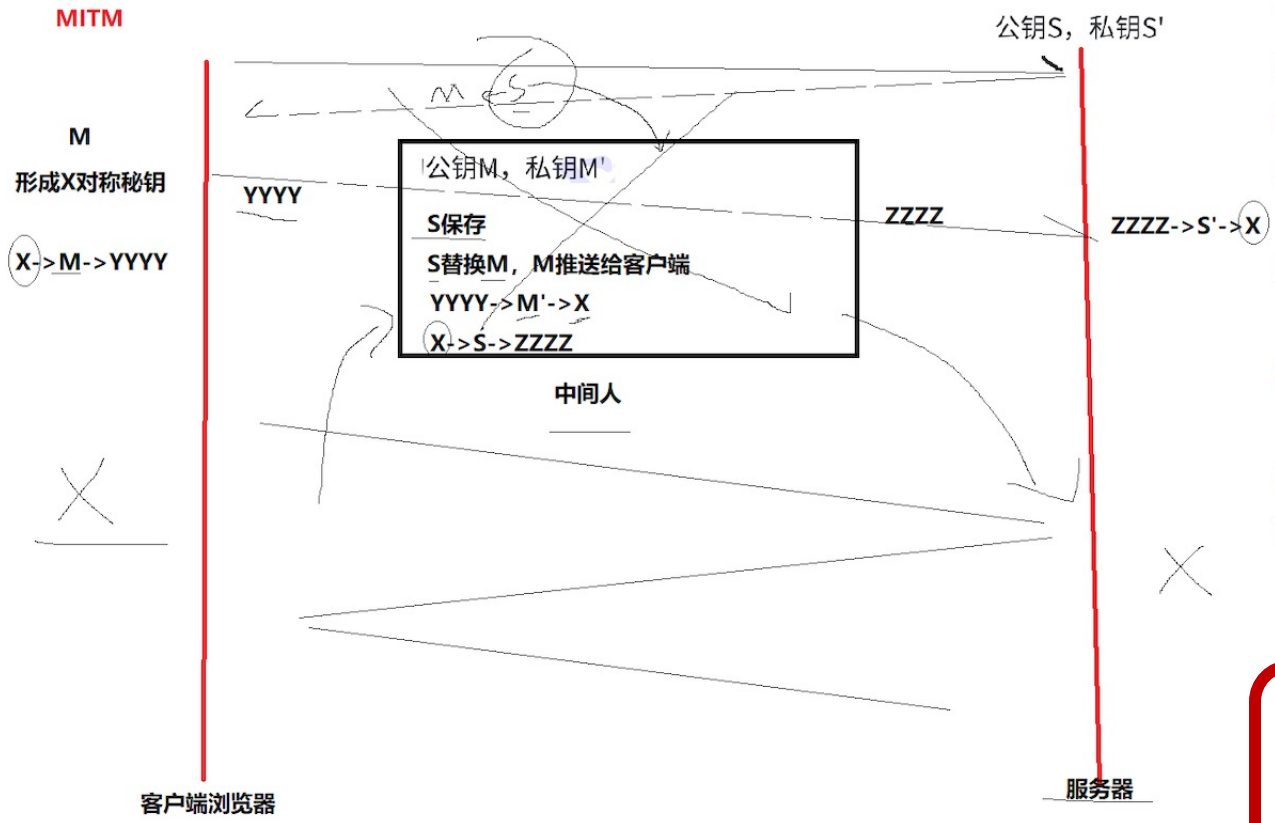


- 服务端具有非对称公钥S和私钥S'
- 客户端发起https请求，获取服务端公钥S
- 客户端在本地生成**对称密钥C**, 通过公钥S加密, 发送给服务器.
- 由于中间的网络设备没有私钥, 即使截获了数据, 也无法还原出内部的原文, 也就无法获取到对称密钥(**真的吗？**)
- 服务器通过私钥S'解密, 还原出客户端发送的对称密钥C. 并且使用这个对称密钥加密给客户端返回的响应数据.

- 后续客户端和服务器的通信都只用对称加密即可. 由于该密钥只有客户端和服务器两个主机知道, 其他主机/设备不知道密钥即使截获数据也没有意义.

一开始为什么我们不单独用对称加密？
就是因为这个对称密钥第一次发过去给服务端的时候可能会不安全

但是，通过非对称加密+对称加密这种方式
第一次发送对称密钥的时候，被非对称加密了一下
这样后面我们传数据的时候，就用对称加密的密钥了



1. 服务器具有非对称加密算法的公钥S，私钥S'
2. 中间人具有非对称加密算法的公钥M，私钥M'
3. 客户端向服务器发起请求，服务器明文传送公钥S给客户端
4. **中间人劫持数据报文，提取公钥S并保存好，然后将被劫持报文中的公钥S替换成为自己的公钥M，并将伪造报文发给客户端**
5. 客户端收到报文，提取公钥M(自己当然不知道公钥被更换过了)，自己形成对称秘钥X，用公钥M加密X，形成报文发送给服务器
6. **中间人劫持后，直接用自己的私钥M'进行解密，得到通信秘钥X，再用曾经保存的服务端公钥S加密后，将报文推送给服务器**
7. 服务器拿到报文，用自己的私钥S'解密，得到通信秘钥X
8. 双方开始采用X进行对称加密，进行通信。**但是一切都在中间人的掌握中，劫持数据，进行窃听甚至修改，都是可以的**

只要已经交换了秘钥了，中间人来了就晚了，中间人在最开始的时候，就可以进行篡改替换

这个的中间人攻击能够成功，本质是什么呢？？本质是中间人能够对数据做篡改&&Client无法验证收到的公钥是合法的就是目标服务器的公钥

所以第四种方案还是有漏洞的！
只要中间人一开始就来，就会出问题

为了解决上面的问题
client需要对服务器的合法性进行认证

为了解决上面的问题，client需要对服务器的合法性进行认证!

权威机构 -- CA机构
颁发证书 -- CA证书

3.签发证书

当服务端申请CA证书的时候，CA机构会对该服务端进行审核，并专门为该网站形成数字签名，过程如下：

1. CA机构拥有非对称加密的私钥A和公钥A'
2. CA机构对服务端申请的证书明文数据进行hash，形成数据摘要
3. 然后对数据摘要用CA私钥A'加密，得到数字签名S

因此中间人没有CA机构的私钥

这个决定了中间人无法制作假证书