基础概念

- **对称加密**：发送方和接收方需要持有同一把密钥，发送消息和接收消息均使用该密钥。相对于非对称加密，对称加密具有更高的加解密速度，但双方都需要事先知道密钥，密钥在传输过程中可能会被窃取，因此安全性没有非对称加密高。常见的对称加密算法有DES、3DES、AES等
- **非对称加密**：有两把密钥，一把叫做公钥、一把叫做私钥，用公钥加密的内容必须用私钥才能解开，同样，私钥加密的内容只有公钥能解开。在数据传输过程中，公钥负责加密，私钥负责解密，数据在传输过程中即使被截获，攻击者由于没有私钥，因此也无法破解。非对称加密算法的加解密速度低于对称加密算法，但是安全性更高。常见的非对称加密算法RSA等
- **消息摘要**：将长度不固定的消息作为参数，运行特定的hash函数，生成固定长度的输出，这个输出就是Hash，也称为这个消息的消息摘要。常见的hash算法有MD5, SHA-1等

对称加密方案

服务端拥有一个对称加密秘钥A：

1. 客户端向服务端请求，服务端把秘钥A明文传输给客户端。
2. 之后双方都用秘钥A加密消息后通信。

缺点：在第一步被中间人劫持后，中间人可以用该秘钥解开任何双方要传输的内容

非对称加密方案

服务端拥有一个非对称加密的公钥A、私钥A'：

1. 客户端向服务端请求，服务端把公钥A明文传输给客户端。
2. 之后客户端发送的消息都用公钥A加密，服务端用私钥A'解密

缺点：在第一步被中间人劫持后，中间人仍然可以拿到公钥，然后冒充客户端与服务端通信

改良的非对称加密方案

服务端拥有一个非对称加密的公钥A、私钥A'，客户端有一个非对称加密的公钥B、私钥B'：

1. 客户端向服务端请求，把公钥B明文传输给服务端，服务端把公钥A明文给传输客户端。
2. 之后客户端向服务端传输的所有东西都用公钥A加密，服务端收到后用私钥A'解密。由于只有服务端拥有这个私钥A'可以解密，所以能保证这条数据的安全。

服务端向客户端传输的所有东西都用公钥B加密，客户端收到后用私钥B'解密。同上也可以保证这条数据的安全。

缺点：非对称加密算法非常耗时，双向的非对称加密对性能影响太大

对公钥用非对称加密传输 + 对消息内容用对称加密传输方案

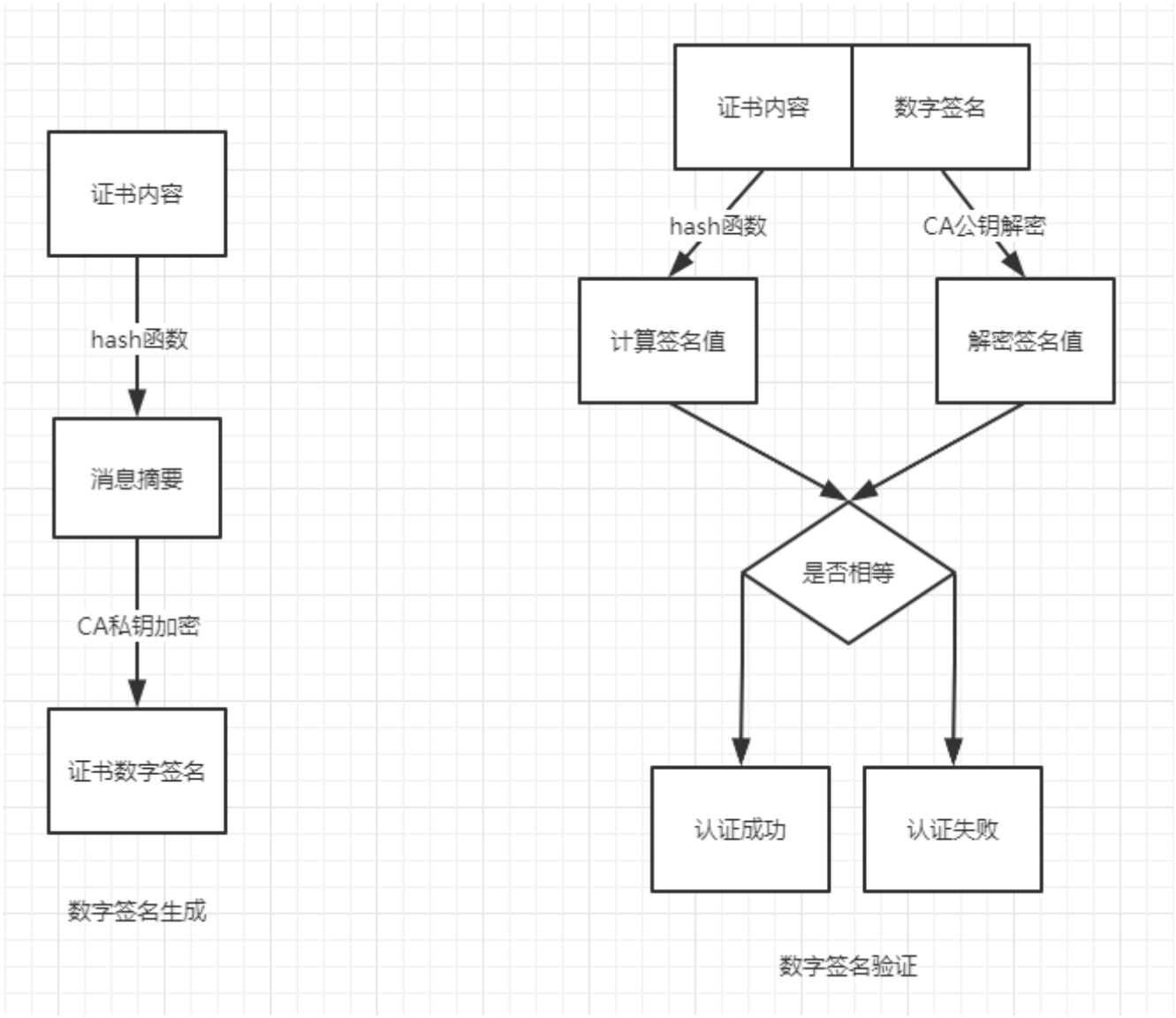
服务端拥有用于非对称加密的公钥A、私钥A'。

1. 客户端向服务端请求，服务端把公钥A明文给传输客户端。
2. 客户端随机生成一个用于对称加密的密钥X，用公钥A加密后传给服务端。
3. 服务端拿到后用私钥A'解密得到密钥X。
4. 这样双方就都拥有密钥X了，且别人无法知道它。之后双方所有数据都用密钥X加密解密。

缺点：其实是上面的所有方案共有的缺点，中间人可以在服务端明文告知客户端自己的公钥A时，把这个公钥A保存下来，替换为自己的公钥C，真正的客户端就会用公钥C加密消息发送给中间人，中间人用自己的私钥C'解密后，可以对消息任意篡改，然后用之前的保存的公钥A加密后发送给服务端

非对称加密 + 对称加密 + 数字证书 + 数字签名方案

- **数字证书**：网站在使用HTTPS前，需要向"CA机构"申请颁发一份数字证书，数字证书里有证书持有者、证书持有者的公钥、证书持有者的证书签名hash算法等信息，服务器把证书传输给浏览器，浏览器从证书里取证书持有者的公钥就行了，证书就如身份证一样，可以证明“该公钥对应该网站”。
- **数字签名**：把证书内容用Hash算法进行加密后得到一个摘要，然后将该摘要用CA私钥加密后附在证书的最后，成为证书的数字签名



服务端拥有用于非对称加密的公钥A，私钥A':

1. 服务端把公钥A交给CA（一般是以CSR文件的形式），CA据此生成证书，并进行数字签名
2. 客户端向服务端发请求，服务端把证书发送给客户端
3. 客户端拿到证书后，进行签名验证，验证通过后随机生成一个用于对称加密的秘钥X（真正的https其实是与服务端进行对称加密算法协商），用从证书中拿到的公钥A将秘钥X加密后发送给服务端
4. 之后客户端和服务端的请求都用秘钥X加密解密

彻底搞懂https的加密机制