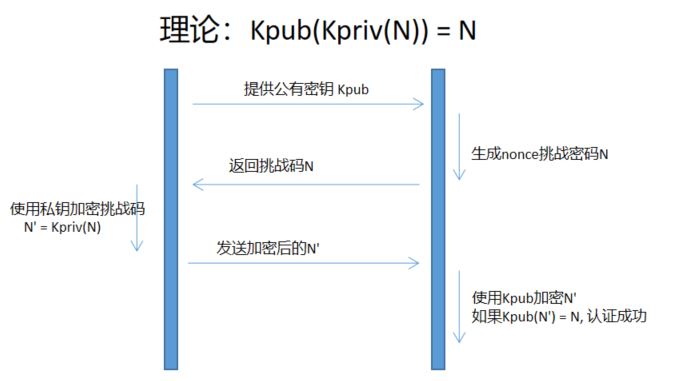
加密算法

|  |  |  |
| --- | --- | --- |
| Hash加密 | 对称加密 | 非对称加密 |
| MD5 | ~~3-DES 已经攻破~~ | RSA |
| SHA-1 | AES (128 & 256) | DH |

以上的3中加密方式都是自己的计算逻辑，可以同时使用两种的结合

其中必须满足的原则是：非对称加密中生成一对密钥，公开其中的一部分，私钥一定在自己的手中 !

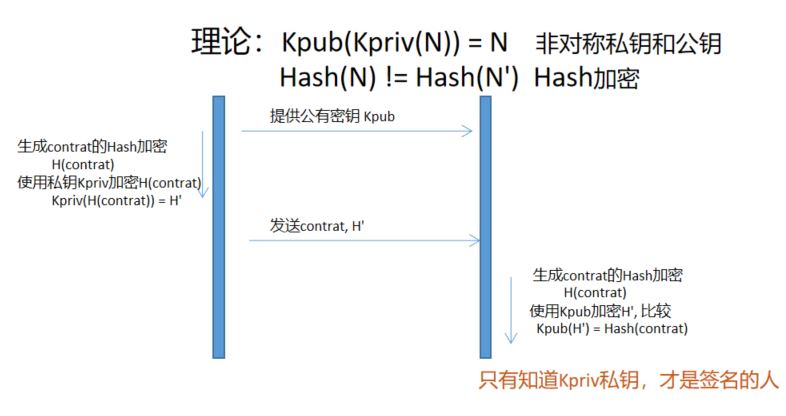
身份认证模型 Authentification



加密算法模型 Chiffrement



签名模型 Signature (私钥加密，公钥解)



**跨域认证的问题 ====> 面试问题 !!!**

互联网服务离不开用户认证。一般流程是下面这样：

1.1用户向服务器发送用户名和密码。

1.2 服务器验证通过后，在当前对话（session）里面保存相关数据，比如用户角色、登录时间

1.3 服务器向用户返回一个 session\_id，写入用户的 Cookie

1.4用户随后的每一次请求，都会通过 Cookie，将 session\_id 传回服务器

1.5 服务器收到 session\_id，找到前期保存的数据，由此得知用户的身份

这种模式的问题在于，扩展性（scaling）不好。单机当然没有问题，如果是服务器集群，或者

是跨域的服务导向架构，就要求 session 数据共享，每台服务器都能够读取 session

举例来说，A 网站和 B 网站是同一家公司的关联服务。现在要求，用户只要在其中一个网站登

录，再访问另一个网站就会自动登录，请问怎么实现？

一种解决方案是 session 数据持久化，写入数据库或别的持久层。各种服务收到请求后，都向

持久层请求数据。这种方案的优点是架构清晰，缺点是工程量比较大。另外，持久层万一挂了，

就会单点失败。

另一种方案是服务器索性不保存 session 数据了，所有数据都保存在客户端，每次请求都发回

服务器。JWT 就是这种方案的一个代表。

## Cookies设计模型



## JSON Web Token设计模型

用户使用用户名密码来请求服务器

服务器进行验证用户的信息

服务器通过验证发送给用户一个token

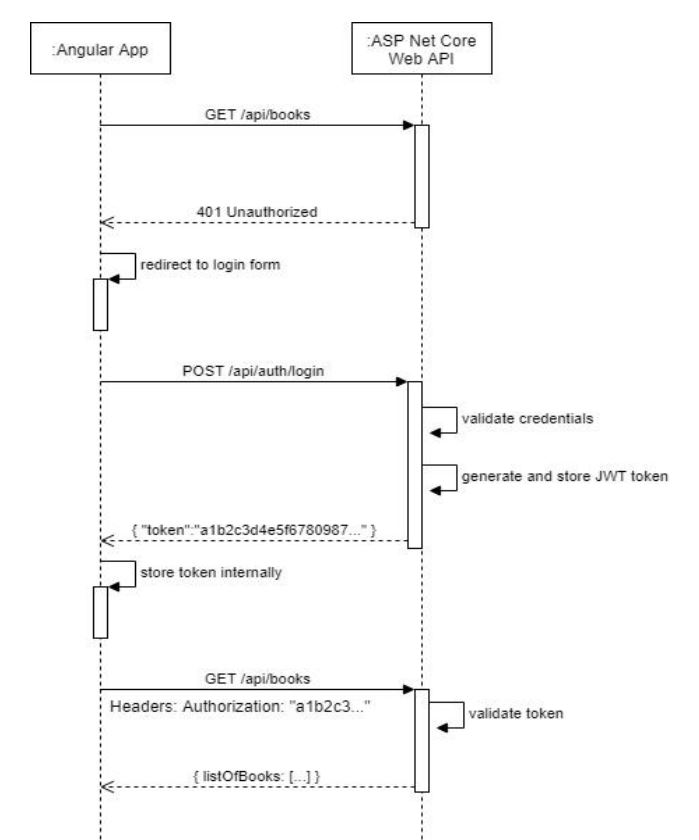
**客户端存储token**，并在每次请求时附送上这个token值

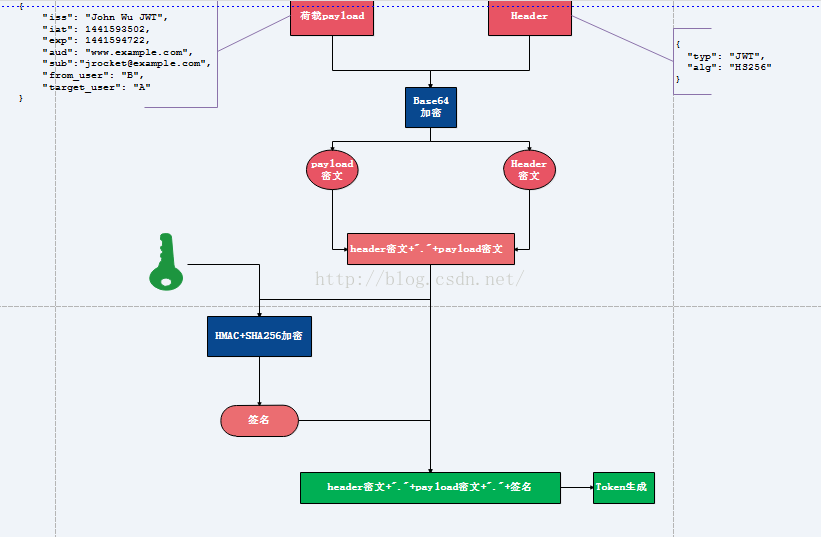
服务端验证token值，并返回数据

这个token必须要在每次请求时传递给服务端，它应该保存在请求头里， 另外，服务端要支持CORS(跨来源资源共享)策略，一般我们在服务端这么做就可以了Access-Control-Allow-Origin: \*

第一部分我们称它为头部（header),

第二部分我们称其为载荷（payload, 类似于飞机上承载的物品)，第三部分是签证（signature).





3.1头部进行base64加密（该加密是可以对称解密的)

3.2 playload不建议存放敏感信息，因为base64是对称解密的（该加密是可以对称解密的)

3.3 secret是保存在服务器端的，jwt的签发生成也是在服务器端的，secret就是用来进行jwt 的签发和jwt的验证，所以，它就是你服务端的私钥，在任何场景都不应该流露出去。

一旦客户端得知这个secret, 那就意味着客户端是可以自我签发jwt了, 也就可以生成

有效的Token登录

3.4用户请求时携带此token（分为三部分，header密文，payload密文，签名）到服务端，服 务端解析第一部分（header密文），用Base64解密，可以知道用了什么算法进行签名，此 处解析发现是HS256

3.5服务端使用原来的秘钥与密文(header密文+"."+payload密文)同样进行HS256运算，然后用 生成的签名与token携带的签名进行对比，若一致说明token合法，不一致说明原文被修改。

3.6判断是否过期，客户端通过用Base64解密第二部分（payload密文），可以知道荷载中授权 时间，以及有效期。通过这个与当前时间对比发现token是否过期。

## JWT的特点

JWT 默认是不加密，但也是可以加密的。生成原始 Token 以后，可以用密钥再加密一次

JWT 不加密的情况下，不能将秘密数据写入 JWT

JWT 不仅可以用于认证，也可以用于交换信息。有效使用 JWT，降低服务器查询数据库的次数。

JWT 的最大缺点是，由于服务器不保存 session 状态，因此无法在使用过程中废止某个 token，

或者更改 token 的权限。也就是说，一旦 JWT 签发了，在到期之前就会始终有效，除非服务

器部署额外的逻辑

JWT 本身包含了认证信息，一旦泄露，任何人都可以获得该令牌的所有权限。为了减少盗用，

JWT 的有效期应该设置得比较短。对于一些比较重要的权限，使用时应该再次对用户进行认证。

为了减少盗用，JWT 不应该使用 HTTP 协议明码传输，要使用 HTTPS 协议传输