**接口为什么要验证签名？**

防止有人拦截请求，修改请求参数

**如何验证签名？**

把请求中的参数key按照字母排序，通过md5或者其它算法生成一个签名，放到sig字段也当成参数传到后台，后台再通过相同的md5算法把请求中的参数进行计算看到到的签名和前端传来的是否一致，一致的通过，不一致的拒绝调用。

**详细介绍**

**一、get请求，直接将需要的参数写在url链接里面：**

http://[api](https://so.csdn.net/so/search?q=api&spm=1001.2101.3001.7020).services.com/queryDatas?param\_1=value\_1¶m\_2=value\_2...

这种方式虽然简单直接，但是容易被别人任意调用来获取相关的数据信息；

**二、将参数和安全密钥拼接后进行签名，然后调用接口的方式：**

1、客户端申请调用服务，然后服务端给客户端分配对应的 accesskey 和 secretKey；

2、用户将调用接口的参数以及上一步生成的安全[密钥](https://so.csdn.net/so/search?q=%E5%AF%86%E9%92%A5&spm=1001.2101.3001.7020) secretKey 进行MD5加密，生成一个签名 signature（一串十六进制码）***signature=MD5(param\_1value\_1param\_2value\_2secretKey)***；

3、最后将第一步生成的 accesskey 和所有参数以及第二步生成的 signature 写到url链接中，调用接口；

http://api.services.com/queryDatas?key=accesskey¶m\_1=value\_1¶m\_2=value\_2...&sign=signature

这种方式加入了安全密钥机制，将所有请求用到的参数全部加密签名处理了，就算被别人拿到了请求链接，他们也获取不到secretKey和相关参数（*因为MD5加密是不可逆的*）；

这种方式虽然在一定程度上防止了参数篡改的问题，但是只要别人获取到了完整的url链接，就可以一直使用这个链接获取数据，大不了就获取相同参数的数据；

**三、给签名中添加时间戳 timestamp ，防止重放问题：**

为了防止别人拿到链接，重复调用接口的问题（重放问题），我们需要**保证请求的唯一性**，也就是保证这条请求只能调用一次，不能被重复调用。怎么做呢？ ==》 在请求参数中加入时间戳 timestamp（yyyyMMddHHmmss），并一起进行MD5加密签名，***signature=MD5(param\_1value\_1param\_2value\_2timestampsecretKey)***；此时新的调用接口的url地址是：

http://api.services.com/queryDatas?key=accesskey¶m\_1=value\_1¶m\_2=value\_2...timestamp=20200527224000&sign=signature

考虑到客户端与服务端之间的网络延迟问题，我们不能保证客户端与服务端的时间完全一致，所以我们可以折中一下：只要满足客户端请求的时间 timestamp 与服务端接收到请求的当前时间 currentTime 之间相差的绝对值在60s以内，我们就算是一次唯一不重发的请求。

这种靠添加时间戳 timestamp 保证请求唯一性的方式可以说已经很好地防止了重发问题的发生；***但是假如有高手能够在1分钟之内获取到完整链接***，那么他就能够再次利用相同的链接进行重发操作，获取服务端返回的数据！！！

**四、给签名中同时添加时间戳 timestamp 和随机数 nonce，防止重放问题：**

最终的方案是给请求链接中再添加一个nonce（number once的简称，意思是一次性数字）；

1、客户端可以利用 timestamp 作为 nonce 数的种子随机生成一串16位的字符串；

2、然后将 nonce 作为 sign 签名的参数，用MD5加密生成一串signature（***signature=MD5(param\_1value\_1param\_2value\_2timestampnoncesecretKey***）；

3、最后将参数、时间戳、nonce已经signature 写在请求的 url 链接中，调用服务端的接口，请求数据；

http://api.services.com/queryDatas?key=accesskey¶m\_1=value\_1¶m\_2=value\_2...timestamp=20200527224000&nonce=\*\*\*\*\*\*&sign=signature

服务端接收到请求后，会解析出 url 链接中的 timestamp、nonce 以及其它参数，然后：

1、将这些参数用相同的加密算法 MD5 加密，得到的一串字符串与 url 中的 signature 值做验证；

2、***接着讲 nonce 作为 key ，存储在 redis 中***（存在数据库中也可以）；

假如别人在 60s 内拿到了完整的链接，他想用相同的链接调用接口，此时，服务端会解析出 nonce ，然后带着这个 nonce 去redis中查询一遍，如果有这个 nonce 作为key的值，则说明此次调用时非法调用，不予通行；如果这个 nonce 在redis中还没有对应的 value值，则说明是用户第一次调用接口，则放行；

**五、总结：**

上面介绍的4种方式，最后一种通过 timestamp +nonce 双重验证的方式，可以说是当前应用最广泛也最安全的；

另外在阅读别人文章的时候，***发现 MD5 加密方式没有 SHA(1,256,512) 加密方式安全（SHA1算法也被破解了）***，大家在操作的时候可以尝试用SHA的加密方式，确保安全性的提升。

**参考链接：https://blog.csdn.net/weixin\_42109071/article/details/102390232**