# ASIHTTPRequest实现https双向认证请求

什么是双向认证呢？简而言之，就是服务器端对请求它的客户端要进行身份验证，客户端对自己所请求的服务器也会做身份验证。服务端一旦验证到请求自己的客户端为不可信任的，服务端就拒绝继续通信。客户端如果发现服务端为不可信任的，那么也中止通信。

双向认证的算法理论是RSA，(点击此处了解RSA算法原理)。 双向认证具体又是通过安全证书的方式来实现的，安全证书可用openssl或java程序来生成，用于双向认证的安全证书中保存了密钥对，证书颁发机构信 息，签名信息，签名算法，颁发对象，有效期等信息。双向认证中安全证书分为服务器端证书和客户端证书，用服务器端证书中的私钥对客户端证书进行签名，并把 签名信息写到客户端证书中，就得到了被服务端信任的证书。当客户端请求该服务端时，服务端为拿到客户端证书信息，然后取出证书中的签名信息，用服务器端证 书的公钥验证，如果发现这个客户端证书确实是服务器端证书签名颁发的，那么通信就可以继续进行，否则中断。

上面简单介绍了一下双向认证和安全证书，那么我们现在开始正题。

首先，我们用java生成一个服务器端证书库myserverdomain和客户端证书库wenfeng.xu，取出服务器端的证书库中的证书为客户端证 书库签名并生成PKCS12格式的证书文件wenfeng.xu.pfx。然后我们将服务器端证书配置在应用服务器中，并启用客户端认证。以jetty为 例，以下为配置方法：

启动应用服务器，并用与生成服务端证书一致的域名访问应用（注意这点非常重要，ASIHTTPRequest如果不这么做是会报错的，这个域名可以随便 取，只要更改系统的host配置，让域名指向服务端ip就行了）。如果你用浏览器访问已启动的应用，如果看到以下信息，就可以开始oc客户端的编码了。

在引入了ASIHTTPRequest框架的项目中新建测试类Https.m

```
1. @implementation Https
2.
3. + (void)testClientCertificate {
4.     NSURL *httpsUrl = [NSURL URLWithString:@"https://www.myserverdomain.com:8443/smvcj"];//访问路径
5.
```

```
    ASIHTTPRequest *request = [ASIHTTPRequest requestWithURL:httpsUrl];

    SecIdentityRef identity = NULL;
    SecTrustRef trust = NULL;


    NSData *PKCS12Data = [NSData dataWithContentsOfFile:
[[NSBundle mainBundle] pathForResource:@"wenfeng.xu" ofType:@"pfx"]];
    [Https extractIdentity:&identity andTrust:&trust fromPKCS12Data:PKCS12Data]
;

    request = [ASIHTTPRequest requestWithURL:httpsUrl];

    [request setClientCertificateIdentity:identity];
    [request setValidatesSecureCertificate:NO];


    [request startSynchronous];

    NSError *error = [request error];
    if (!error) {
        NSString *response = [request responseString];
        NSLog(@"response is : %@",response);
    } else {
        NSLog(@"Failed to save to data store: %@", [error localizedDescription]);
        NSLog(@"%@",[error userInfo]);
    }
}

+ (BOOL)extractIdentity:(SecIdentityRef *)outIdentity andTrust:
(SecTrustRef*)outTrust fromPKCS12Data:(NSData *)inPKCS12Data {


    OSStatus securityError = errSecSuccess;

    CFStringRef password = CFSTR("p@ssw0rd");
    const void *keys[] =   { kSecImportExportPassphrase };
    const void *values[] = { password };
```

```
41.
42.     CFDictionaryRef optionsDictionary = CFDictionaryCreate(NULL, keys,values, 1,
        NULL, NULL);
43.
44.     CFArrayRef items = CFArrayCreate(NULL, 0, 0, NULL);
45.
46.     securityError = SecPKCS12Import((CFDataRef)inPKCS12Data,optionsDictionar
        y,&items);
47.
48.     if (securityError == 0) {
49.         CFDictionaryRef myIdentityAndTrust = CFArrayGetValueAtIndex (items, 0);
50.         const void *tempIdentity = NULL;
51.         tempIdentity = CFDictionaryGetValue (myIdentityAndTrust, kSecImportItemI
        dentity);
52.         *outIdentity = (SecIdentityRef)tempIdentity;
53.         const void *tempTrust = NULL;
54.         tempTrust = CFDictionaryGetValue (myIdentityAndTrust, kSecImportItemTru
        st);
55.         *outTrust = (SecTrustRef)tempTrust;
56.     } else {
57.         NSLog(@"Failed with error code %d",(int)securityError);
58.         return NO;
59.     }
60.     return YES;
61. }
62.
63. @end
```

在项目中调用 testClientCertificate方法，发现会报以下错误

怎么会这样？分析最后一句“Invalid certificate chain”意思是无效的证书链。因为每一个证书中都有一个证书链，来表示这个证书的层次结构。报这个错是因为这个客户端证书的最顶层是我们自己创建的证书，而不是合 法的证书机构颁发的。每个操作系统默认会把一些公认的证书机构颁发的公钥证书存在系统信认的根证书库中，以便信任由这些公认的证书机构签名给其它用户的证 书。那么如何在测试环境中避免这个错？我们只要修改ASIHTTPRequest框架中的相关配置就行了，打开ASIHTTPRequest.m文件，查找“https”关健字，找到

将其注掉，然后换成以下代码

解决我们的错误的关键代码是

[NSNumber numberWithBool:NO], kCFStreamSSLValidatesCertificateChain 表示不校验证书链。

保存一下再运行就可以正常访问应用了。

HTTPS 双向认证构建移动设备安全体系 对于一些高安全性要求的企业内项目,我们有时希望能够对客户端进行验证.这个时候我们可以使用Https的双向认证机制来实现这个功能. 单向认证:保证server是真的,通道是安全的(对称密钥):双向认证:保证client和server是真的,通道是安全的(对称密 ...