HTTPS 从零开始

在下的iOS应用需要从Server接收脚本语言，传递的信息中有点儿不想为外人知的技术细节。咋办呢？HTTPS。

OK，现在我们有了一个关键词。那么HTTPS能完成这个任务吗？根据维基百科的解释：

1. HTTPS在发起回话之前是要进行身份验证的，这保证了Server不会给没有证书的Client返回数据。
2. HTTPS在回话建立后通信的数据是被加密的，这样数据如果被抓包的话，信息也不会泄露。

So，good choice，let's on it。但是从何入手呢？

HTTPS是建立在SSL上的HTTP连接，所以真正的麻烦是SSL。SSL既安全套接层（Secure Sockets Layer），真正的安全工作都是由它完成。这里，有两个主要部分——身份验证和数据加密。

身份验证

这个过程需要名为证书链的技术。证书链是由数字证书认证机构（CA, Certificate Authority）开始的承载信任关系的链状结构。这有点像保媒拉纤的，本来不认识的两个人，小伙子要搭讪，就说"我七大姑的三舅姥姥"是"你八大姨的二叔母"，姑娘一想，确实有这么个人，信任关系就建立了。

CA分为收费的（不是十块二十块能解决的）和免费的，当然我们知道Server和Client都是可信的，所以也可以自己充当CA。

数据加密

加密用的是非对称加密算法。这个算法中有一个Server和Client各持有一个密钥对。密钥对由公钥和私钥组成。公钥就好像一种规范，可以发给对方，用于加密。私钥必须自己妥善保管，用于解密。好比，Server将自己的公钥发给Client，并对Client说，你只要用这个密钥加密，我就能用我自己的私钥解密。

SSL有一套操作流程，能将以上两个系统结合起来。而且因为本身实在够成熟，这套流程被绝大多数扯到网络的软件、类库、框架给实现了。

就在下的情况而言，Server用的Tornado，

```
http_server = tornado.httpserver.HTTPServer(applicaton, ssl_options={
    "certfile": os.path.join(data_dir, "mydomain.crt"),
    "keyfile": os.path.join(data_dir, "mydomain.key"),
})

http_server.listen(443)
```

iOS端网络部分借助了MKNetworkKit，

```
  MKNetworkOperation *op = [self operationWithPath:@"/" params:nil httpMethod:nil
ssl:YES];
  op.clientCertificate = [[[NSBundle mainBundle] resourcePath]
stringByAppendingPathComponent:@"client.p12"];
  op.clientCertificatePassword = @"test";
```

它们都有实现SSL。据我了解，Server这边Apache与Nginx都能提供SSL服务，iOS这边Cocoa本身的URL Loading System也是这样。
从上面的源码中能发现，无论如何实现，都离不开证书文件的支持。上面出现的后缀crt, key, p12只是证书文件的冰山一脚。关于这个庞大的家族，我从这两篇博文了解了个大概：
使用OpenSSL生成证书

电子证书 DER vs. CRT vs. CER vs. PEM
在下在实现过程中只用到了key, csr, crt, p12这四种。其中：
.key是私钥文件,
.csr（Certificate Signing Request）是证书签署请求文件,
.crt（certificate）是证书文件，包含证书链信息和公钥,
.p12则是为了方便管理创造的key与crt的组合体。

它们的生成过程是这样滴：
用RSA（大素数运算，不用管）算法生成私钥.key，根据.key生成请求文件.csr，拿着.csr跑去给CA签署，签署完拿回来的是.crt。最后，如果愿意吧.key和.crt整合成p12。

如何搞到这些文件呢？祭出神器OpenSSL。此神器包含了SSL能用到的所有功能，就证书生成而言，命令很多，看到眼花，用到手软。在下的操作系统是OS X，集成了此软件，用Linux的朋友apt-get一个吧。
1. 自己充当CA，创建CA的私钥与证书。CA本身跟证书链上的其他节点没啥区别，该有的东西都得有，只不过信誉比较好

这一步需要先在当前目录建好需要的目录结构否则会报找不到文件的错

```
mkdir demoCA
cd demoCA
mkdir newcerts
touch index.txt
echo '01' > serial
cd ..
openssl req -new -x509 -keyout MyCA.key -out MyCA.crt -days 3650
```

-x509是一种证书标准。-keyout后面是输出的私钥名，-out后面是输出的证书名，都在当前目录。-days是过期天数，10年，估计够用了。

2. 生成Server私钥

```
openssl genrsa -out MyServer.key 1024
```

-out后面是输出的密钥。1024，此数越大，加密越难被破解，但是运算量也越大。

3. 使用Server密钥生成Server证书请求文件

```
openssl req -new -key MyServer.key -out MyServer.csr
Enter pass phrase for server.key:
Country Name (2 letter code) [GB]:CN
```

State or Province Name (full name) [Berkshire]:LiaoNing
Locality Name (eg, city) [Newbury]:DaLian
Organization Name (eg, company) [My Company Ltd]:Dreamer
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:secret.domin.com
Email Address []:coderdreamer@gmail.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:Dreamer
-key后面是上一步生成的密钥，-out后面数是输出的证书请求文件名。

这一步会要求填入大量个人信息。其中 Common Name (eg, your name or your server's hostname) 要填成网站的域名，这不会影响SSL通信，但似乎会显示在浏览器的证书信息中。

4. 拿着证书请求文件去让自己的CA签

openssl ca -in MyServer.csr -out MyServer.crt -cert MyCA.crt -keyfile MyCA.key
-in后面是上一步的证书请求文件名，-out是要输出的证书名。-cert和-keyfile是CA的证书和私钥，第1步生成过了。

5. 生成Client密钥

openssl genrsa -des3 -out MyClient_enc.key 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
这里比生成Server密钥的时候多了一个参数，-des3指的是加密方式。因为我的情况是，Client的证书要放在iOS端里，不给个加密方式不安全。输入该命令后，会提示输入密码，而且以后用到该密钥时都会要求输入密码。

如果嫌麻烦的话，也可以去掉密码：

openssl rsa -in MyClient_enc.key -out MyServer.key

6、7. 跟服务器一样，生成证书请求文件，拿去给CA签一下

openssl req -new -key MyClient_enc.key -out MyClient.csr

openssl ca -in MyClient.csr -out MyClient.crt -cert MyCA.crt -keyfile MyCA.key

8. 如果需要，合并成同时包含私钥，公钥，证书的p12文件

openssl pkcs12 -export -clcerts -in MyClient.crt -inkey MyClient_enc.key -out MyClient.p12

还要提一点，如果第3和6步所填写的信息是完全一样的会报错：

openssl TXT_DB error number 2 failed to update database

产生的原因是:

This thing happens when certificates share common data. You cannot have two certificates that look otherwise the same.
修改demoCA下 index.txt.attr

将

unique_subject = yes

改为

unique_subject = no

以上，就得到了所有需要的证书文件，把它们安放在Server与客户端中，通信将得到保护。

最后，其实要不要确认对方证书可信再开始通信对于Server和Client都是可选项。如果真的要做到这点，需要额外的配置。

参考文档：

http://hi.baidu.com/leeboysam/item/969db6057e913918ebfe3837
http://sech.iteye.com/blog/1144288
http://wiki.atollon.com/en/tech/Httpd_(Apache)/How_to_create_and_install_a_self-signed_SSL_Certificate
http://blog.csdn.net/fym0121/article/details/7992340
http://wangye.org/blog/archives/732/
http://blog.creke.net/762.html