==1==编译openssl

下载 OpenSSL 源代码：

wget https://www.openssl.org/source/openssl-1.1.1t.tar.gz

解压源代码：

tar -zxvf openssl-1.1.1t.tar.gz

进入解压后的目录：

cd openssl-1.1.1t

配置 OpenSSL：

./config --prefix=/usr/local/ssl

编译 OpenSSL：

make

安装 OpenSSL：

make install

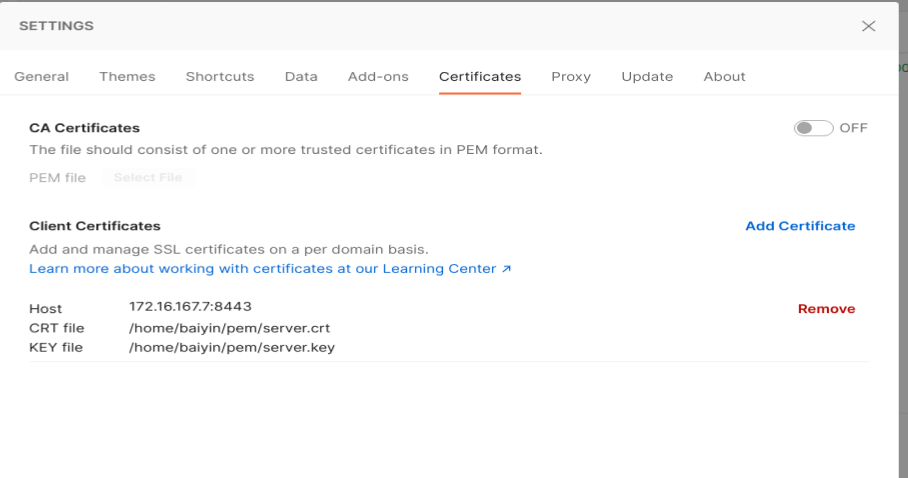
==2==生成证书

一步生成自签名证书和私钥：

openssl req -x509 -newkey rsa:2048 -keyout server.key -out server.crt -days 365 -nodes

==3==使用证书

Postman使用：



REST服务端开启OPENSSL：

主线程：

if (m\_bOpenSSL && !m\_strCertPath.empty() && !m\_strKeyPath.empty())

{

// 初始化OpenSSL库

SSL\_library\_init();

SSL\_load\_error\_strings();

OpenSSL\_add\_all\_algorithms();

}

HTTP线程：

static struct bufferevent\* bevcb(struct event\_base\* base, void\* arg)

{

struct bufferevent\* r;

SSL\_CTX\* ctx = (SSL\_CTX\*)arg;

r = bufferevent\_openssl\_socket\_new(base,

-1,

SSL\_new(ctx),

BUFFEREVENT\_SSL\_ACCEPTING,

BEV\_OPT\_CLOSE\_ON\_FREE);

return r;

}

static void server\_setup\_certs(SSL\_CTX\* ctx, const char\* certificate\_file, const char\* private\_key)

{

qDebug() << "Loading certificate chain = "<< certificate\_file <<", private key = "<< private\_key;

if (1 != SSL\_CTX\_use\_certificate\_file(ctx, certificate\_file, SSL\_FILETYPE\_PEM))

qDebug() << "SSL\_CTX\_use\_certificate\_chain\_file";

if (1 != SSL\_CTX\_use\_PrivateKey\_file(ctx, private\_key, SSL\_FILETYPE\_PEM))

qDebug() << "SSL\_CTX\_use\_PrivateKey\_file";

if (1 != SSL\_CTX\_check\_private\_key(ctx))

qDebug() << "SSL\_CTX\_check\_private\_key";

}

//==

if (m\_bOpenSSL && !m\_strCertPath.empty() && !m\_strKeyPath.empty())

{

//------OPENSSL------

/\* 创建SSL上下文环境 ，可以理解为 SSL句柄 \*/

SSL\_CTX\* ctx = SSL\_CTX\_new(SSLv23\_server\_method());

SSL\_CTX\_set\_options(ctx, SSL\_OP\_SINGLE\_DH\_USE | SSL\_OP\_SINGLE\_ECDH\_USE | SSL\_OP\_NO\_SSLv2);

SSL\_CTX\_set\_verify(ctx, SSL\_VERIFY\_NONE, NULL);

/\*选择椭圆曲线与椭圆曲线密码套件一起使用\*/

EC\_KEY\* ecdh = EC\_KEY\_new\_by\_curve\_name(NID\_X9\_62\_prime256v1);

if (!ecdh)

qDebug() << "EC\_KEY\_new\_by\_curve\_name";

if (1 != SSL\_CTX\_set\_tmp\_ecdh(ctx, ecdh))

qDebug() << "SSL\_CTX\_set\_tmp\_ecdh";

/\* 设置服务器证书 和 服务器私钥 到 OPENSSL ctx上下文句柄中 \*/

server\_setup\_certs(ctx, m\_strCertPath.c\_str(), m\_strKeyPath.c\_str());

/\* 使我们创建好的evhttp句柄 支持SSL加密。实际上，加密的动作和解密的动作都已经帮我们自动完成，我们拿到的数据就已经解密之后的 \*/

evhttp\_set\_bevcb(http, bevcb, ctx);

//------OPENSSL------

}