《第十四章 密钥协商》示例代码

作者：韩露露、杨波

日期：2019年3月1日

说明

本电子文档来源于书籍《深入浅出CryptoPP密码学库》，它最初被存放于GitHub上。任何人都可以复制、传播、使用本示例代码。

⇓

简介

《深入浅出CryptoPP密码学库》内容简介：

本书向读者介绍密码学库CryptoPP（或Crypto++）的使用方法和设计原理。CryptoPP是一个用C++语言编写的、开源的、免费的密码程序库，它最初由Wei Dai开发，现由开源社区维护。CryptoPP库广泛应用于学术界、开源项目、非商业项目以及商业项目，它几乎包括了目前已经公开的所有密码算法，支持当前主流的多种系统平台，并且具有良好的设计结构和较高的执行效率。

全书共15章，主要内容包括随机数发生器、Hash函数、流密码、分组密码、消息认证码、密钥派生和基于口令的密码、公钥加密系统、数字签名、密钥协商等，本书涵盖C++程序设计、设计模式、数论和密码学等知识。

本书最大的特点就是以应用为导向、以解决实际工程问题为目标，理论结合实践，将抽象的密码学变成保障信息安全的实际工具。

本书可以作为密码学、网络安全等专业在校学生的上机实验教材，也可以作为信息安全产品开发者、科研人员、密码算法实现者的参考手册。

⇓

资源

本书更多示例代码：https://github.com/locomotive-crypto
Crypto++网站：https://www.cryptopp.com/
Crypto++库GitHub地址：https://github.com/weidai11/cryptopp
Crypto++库SourceForge地址：https://sourceforge.net/projects/cryptopp/
Crypto++库Google论坛：
⇒公告通知地址：https://groups.google.com/forum/#!forum/cryptopp-announce
⇒用户群组地址：https://groups.google.com/forum/#!forum/cryptopp-users

目录

1 使用经典的DH密钥协商算法

下面以DH（Diffie-Hellman）密钥协商算法为例，演示产生共享密钥的方法。

```
#include<integer.h>//使用Integer
#include<iostream>//使用cout、cin
#include<osrng.h>//使用AutoSeededRandomPool
#include<nbtheory.h>//使用PrimeAndGenerator
#include<dh.h>//使用DH
using namespace std;//std是C++的命名空间
using namespace CryptoPP;//CryptoPP是CryptoPP库的命名空间
int main()
{
    AutoSeededRandomPool rng;//定义一个随机数发生器对象
    Integer p, q, g;//定义三个大整数对象
    //定义一个产生特殊形式素数的PrimeAndGenerator对象
    PrimeAndGenerator pg;
    //产生p=r*q+1形式的素数，其中q也为素数
    pg.Generate(1, rng, 512, 511);
    p = pg.Prime();//获取p的值
    q = pg.SubPrime();//获取q的值
    g = pg.Generator();//获取g的值
    //定义两个DH对象，分别表示通信实体的双方
    DH dhA(p, q, g),dhB(p, q, g);
    //申请存储空间，以存放A的协商密钥对（临时密钥）
    SecByteBlock dhA_pri(dhA.PrivateKeyLength());
    SecByteBlock dhA_pub(dhA.PublicKeyLength());
    //申请存储空间，以存放B的协商密钥对（临时密钥）
    SecByteBlock dhB_pri(dhB.PrivateKeyLength());
    SecByteBlock dhB_pub(dhB.PublicKeyLength());
    dhA.GenerateKeyPair(rng,dhA_pri,dhA_pub);//通信方A产生公私钥对
    dhB.GenerateKeyPair(rng,dhB_pri,dhB_pub);//通信方B产生公私钥对
    if(dhA.AgreedValueLength()==dhB.AgreedValueLength())
        cout<<"双方密钥长度相等"<<endl;
    else
    {
        cout<<"双方密钥长度不相等"<<endl;
        return 0;
    }
    //申请存储空间，分别存放A和B协商的共享值
    SecByteBlock sharedA(dhA.AgreedValueLength()),sharedB(dhB.
        AgreedValueLength());
    //A根据自己的私钥和B的公钥计算出一个共享值
    if(dhA.Agree(sharedA,dhA_pri,dhB_pub))
        cout<<"A生成协商密钥成功"<<endl;
    else
    {
```

```
43          cout << "A生成协商密钥失败" << endl;
44          return 0;
45      }
46      //B根据自己的私钥和A的公钥计算出一个共享值
47      if(dhA.Agree(sharedB,dhB_pri,dhA_pub))
48          cout << "B生成协商密钥成功" << endl;
49      else
50      {
51          cout << "B生成协商密钥失败" << endl;
52          return 0;
53      }
54      Integer A,B;//定义两个大整数对象
55      //将sharedA存储区的内容解码成大整数
56      A.Decode(sharedA.BytePtr(),sharedA.size());
57      cout << "A协商的共享值: " << A << endl;//打印输出
58      //将sharedB存储区的内容解码成大整数
59      B.Decode(sharedB.BytePtr(),sharedB.size());
60      cout << "B协商的共享值: " << B << endl;//打印输出
61      return 0;
62  }
```

执行程序，程序的输出结果如下：

```
双方密钥长度相等
A生成协商密钥成功
B生成协商密钥成功
A协商的共享值：125445819742208856882144107272029248764041569476335196833945792031512695910480757066377377976372864145318747226854536696027110096343276754071271942288
41086.
B协商的共享值：125445819742208856882144107272029248764041569476335196833945792031512695910480757066377377976372864145318747226854536696027110096343276754071271942288
41086.
请按任意键继续. . .
```

2 使用具有认证功能的ECMQV密钥协商算法

下面以具有认证功能的ECMQV算法为例，演示产生共享密钥的方法。

```
#include<integer.h>//使用Integer
#include<iostream>//使用cout、cin
#include<osrng.h>//使用AutoSeededRandomPool
#include<dh.h>//使用DH
#include<string>//使用string
#include<asn.h>//使用OID
#include<oids.h>//使用secp256r1()
#include<eccrypto.h>//使用ECMQV
#include<ecp.h>//使用ECP
using namespace std;//std是C++的命名空间
using namespace CryptoPP;//CryptoPP是CryptoPP库的命名空间
using namespace CryptoPP::ASN1;//使用ANS1命名空间
int main()
{
    OID CURVE = secp256r1();//获得ANS.1标准的椭圆曲线参数
    AutoSeededRandomPool rng;//定义随机数发生器对象
    //定义两个密钥协商对象，分别表示通信的双方
    ECMQV<ECP>::Domain mqvA(CURVE), mqvB(CURVE);
    //申请存储空间，以存放A的长期密钥对和临时密钥对
    //长期密钥对（mqvA_stpri,mqvA_stpub）
    //临时密钥对（mqvA_eppri，mqvA_eppub
    SecByteBlock mqvA_stpri(mqvA.StaticPrivateKeyLength()),
        mqvA_stpub(mqvA.StaticPublicKeyLength());
    SecByteBlock mqvA_eppri(mqvA.EphemeralPrivateKeyLength()),
        mqvA_eppub(mqvA.EphemeralPublicKeyLength());
    //申请存储空间，以存放B的长期密钥对和临时密钥对
    //长期密钥对（mqvB_stpri,mqvB_stpub），
    //临时密钥对（mqvB_eppri，mqvB_eppub）
    SecByteBlock mqvB_stpri(mqvB.StaticPrivateKeyLength()),
        mqvB_stpub(mqvB.StaticPublicKeyLength());
    SecByteBlock mqvB_eppri(mqvB.EphemeralPrivateKeyLength()),
        mqvB_eppub(mqvB.EphemeralPublicKeyLength());
    //产生A的长期密钥对和临时密钥对
    mqvA.GenerateStaticKeyPair(rng,mqvA_stpri,mqvA_stpub);
    mqvA.GenerateEphemeralKeyPair(rng,mqvA_eppri,mqvA_eppub);
    //产生B的长期密钥对和临时密钥对
    mqvB.GenerateStaticKeyPair(rng,mqvB_stpri,mqvB_stpub);
    mqvB.GenerateEphemeralKeyPair(rng,mqvB_eppri,mqvB_eppub);
    if(mqvA.AgreedValueLength()==mqvB.AgreedValueLength())
        cout<<"双方密钥长度相等"<<endl;
    else
    {
        cout<<"双方密钥长度不相等"<<endl;
```

```
44          return 0;
45      }
46      SecByteBlock sharedA(mqvA.AgreedValueLength()),sharedB(mqvB.
            AgreedValueLength());
47      if(mqvA.Agree(sharedA,mqvA_stpri,mqvA_eppri,mqvB_stpub,
            mqvB_eppub))
48          cout << "A生成协商密钥成功" << endl;
49      else
50      {
51          cout << "A生成协商密钥失败" << endl;
52          return 0;
53      }
54      if(mqvA.Agree(sharedB,mqvB_stpri,mqvB_eppri,mqvA_stpub,
            mqvA_eppub))
55          cout << "B生成协商密钥成功" << endl;
56      else
57      {
58          cout << "B生成协商密钥失败" << endl;
59          return 0;
60      }
61      Integer A,B;
62      A.Decode(sharedA.BytePtr(),sharedA.size());
63      cout << "A的协商信息: " << A << endl;
64      B.Decode(sharedB.BytePtr(),sharedB.size());
65      cout << "B的协商信息: " << B << endl;
66      return 0;
67  }
```

执行程序，程序的输出结果如下：

```
双方密钥长度相等
A生成协商密钥成功
B生成协商密钥成功
A的协商信息：665626302203691702021923363440240325835789947557972637198382575769878
321738.
B的协商信息：665626302203691702021923363440240325835789947557972637198382575769878
321738.
请按任意键继续. . .
```

3 声明

Cryptography

⇓

⇓

⇓

此为《深入浅出CryptoPP密码学库》随书电子文档，它仅包含书籍中示例程序的源代码。关于示例代码的解释说明，详见书籍相应章节内容。

由于作者水平有限，错误之处在所难免。欢迎通过如下方式反馈相关问题：

⇒ QQ：1220195669

⇒ 微信：cc1220195669

⇓

⇓

⇓

《深入浅出CryptoPP密码学库》