《第十二章 公钥加密》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 使用非集成公钥加密算法RSAES

下面以CryptoPP库的RSA算法为例，来演示公钥非集成加密算法的使用方法。

```
#include<iostream>//使用cout、cin
#include<osrng.h>//使用AutoSeededRandomPool
#include<rsa.h>//使用RSA相关的算法
#include<hex.h>//使用HexEncoder
#include<files.h>//使用FileSink
#include<filters.h>//使用StringSource、StringSink
#include<string>//使用string
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()
{
    try
    {
        AutoSeededRandomPool rng;//定义一个随机数发生器对象
        //定义RSA的可逆函数对象，与RSA::PrivateKey等价
        InvertibleRSAFunction rsa_param;
        //产生1024比特、随机的加密参数
        rsa_param.GenerateRandomWithKeySize(rng,1024);
        Integer n = rsa_param.GetModulus();//获取参数n
        Integer p = rsa_param.GetPrime1();//获取参数p
        Integer q = rsa_param.GetPrime2();//获取参数q
        Integer d = rsa_param.GetPrivateExponent();//获取参数d
        Integer e = rsa_param.GetPublicExponent();//获取参数e
        cout << "n(" << n.BitCount() << "):" << n << endl;//打印输出
        cout << "p(" << p.BitCount() << "):" << p << endl;//打印输出
        cout << "q(" << q.BitCount() << "):" << q << endl;//打印输出
        cout << "d(" << d.BitCount() << "):" << d << endl;//打印输出
        cout << "e(" << e.BitCount() << "):" << e << endl;//打印输出
        string plain = "I like Cryptograpyh.";//待加密的明文
        cout << "plain:" ;//以十六进制形式输出明文
        StringSource plainSrc(plain,true,
            new HexEncoder(
                new FileSink(cout)));
        //定义两个string对象，分别用于存储加密后的密文和解密后的明文
        string cipher,recover;
        RSA::PrivateKey prikey(rsa_param);//定义RSA算法的私钥对象
        RSA::PublicKey pubkey(prikey);//定义RSA算法的公钥对象
        //RSAES_OAEP_SHA_Encryptor enc(pubkey);
        RSAES_PKCS1v15_Encryptor enc(pubkey);
        //执行加密-加密字符串plain并且将加密的结果存储于cipher对象
        StringSource encSrc(plain,true,
            new PK_EncryptorFilter(rng,enc,
                new StringSink(cipher)));
```

```
        cout << endl << "cipher:";//以十六进制的形式输出密文
        StringSource cipherSrc(cipher, true,
            new HexEncoder(
                new FileSink(cout)));
        //RSAES_OAEP_SHA_Decryptor dec(prikey);
        RSAES_PKCS1v15_Decryptor dec(prikey);
        //执行解密-加密字符串recover并且将解密的结果存储于recover对象
        StringSource decSrc(cipher, true,
            new PK_DecryptorFilter(rng, dec,
                new StringSink(recover)));
        cout << endl << "recover:";//以十六进制的形式输出解密后的明文
        StringSource recoverSrc(recover, true,
            new HexEncoder(
                new FileSink(cout)));
        cout << endl;
    }
    catch(const Exception& e)
    {//出现异常
        cout << endl << e.what() << endl;//异常原因
    }
    return 0;
}
```

执行程序，程序的输出结果如下：

```
n(1024):14108626074333018210009101476637455997575419366624876678836436678871748759938068850262679534764221801791526942201739731312616685253559128209892885692712588594974481651934528281833410182503813115287631819470676733023190082818374729663724851575388473997832147454398084530375587556865832092766437902368916760938 9.
p(512):11642164186464467404927353684435918223451966135745145780877880490445981957157713837856595061203851576645754747969144652103641124080271820770290395677655637.
q(512):12118559615175444852137480925762287500765995188090422796295294332279551684116479034590259353549714664120147433593676424499185425502960705300464482998219897.
d(1022):31121969281616951933843606198464976465239895661672522085668610321040622264569269522638263679626959856893074137209719995542536805706380429874763718439807175482872282693992579210330770211801266235643013461574600769939061346172817504950965309316059555826194273965102782900591016046053398020986391529884667294233.
e(5):17.
plain:49206C696B652043727970746F6772617079682E
cipher:797444C39998B8F8FAED1138391E231C2D47B21249A55FFE337A1542CC8743D08808320537E14C6588B93544A999C80C66DF806CE9E57909B66A3599CF16787C252F2BDBA4D3B8EE3CB4FE88E707E38EC11F90BE9F11BD150FCFB672226A8E441055E40004702B7CFE0C4F69531D91C69D44A018B16F66D871C09A120F78DBCC
recover:49206C696B652043727970746F6772617079682E
请按任意键继续. . .
```

2 使用集成公钥加密算法ECIES

下面以CryptoPP库的ECIES算法为例，来演示公钥集成加密算法的使用方法。

```
#include<integer.h>//使用Integer
#include<iostream>//使用cout、cin
#include<osrng.h>//使用AutoSeededRandomPool
#include<string>//使用string
#include<ecp.h>//使用EC2P
#include<eccrypto.h>//使用ECIES等
#include<oids.h>//使用secp256r1()
#include<files.h>//使用FileSink
#include<hex.h>//使用HexDecoder
#include<filters.h>//使用StringSource、PK_EncryptorFilter等
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
//打印私钥信息
void PrintPrivateKey(const DL_PrivateKey_EC<ECP>& key);
int main()
{
    try
    {
        AutoSeededRandomPool rng;//定义随机数发生器对象
        //定义私钥类对象，与直接使用ECIES:: PrivateKey等价
        DL_PrivateKey_EC<ECP> ecies_param;
        ecies_param.Initialize(rng,ASN1::secp160r1());//初始化私钥对象
        PrintPrivateKey(ecies_param);//打印私钥信息
        //通过私钥类对象来构造解密器类对象
        ECIES<ECP,SHA1,NoCofactorMultiplication,true,true>::
            Decryptor dec(ecies_param);
        //通过解密器类对象来构造加密器类对象
        ECIES<ECP,SHA1,NoCofactorMultiplication,true,true>::
            Encryptor enc(dec);
        string message("I like Cryptography.");//待加密的明文
        cout << "message: " ;//以十六进制的形式输出明文
        StringSource messageSrc(message,true,
            new HexEncoder(
                new FileSink(cout)));
        //定义两个string对象，分别用于存储加密后的密文和解密后的明文
        string ciphere,recover;
        //执行加密–将明文message加密后存储于ciohere对象中
        StringSource encSrc(message,true,
            new PK_EncryptorFilter(rng,enc,
                new StringSink(ciphere)));
        cout << "ciphere: " ;//以十六进制的形式输出密文
        StringSource ciphereSrc(ciphere,true,
```

```
            new HexEncoder(
                new FileSink(cout)));
        //执行解密-将密文ciphere解密后存储于recover对象中
        StringSource decSrc(ciphere,true,
            new PK_DecryptorFilter(rng,dec,
                new StringSink(recover)));
        cout << endl << "recover: " ;
        //以十六进制的形式输出解密后的明文
        StringSource recoverSrc(recover,true,
            new HexEncoder(
                new FileSink(cout)));
        cout << endl;
    }
    catch(const Exception& e)
    {//出现异常
        cout << e.what() << endl;//异常原因
    }
    return 0;
}
void PrintPrivateKey(const DL_PrivateKey_EC<ECP>& key)
{
    //获得群参数
    const DL_GroupParameters_EC<ECP>& params =
        key.GetGroupParameters();
    //基本的预计算
    const DL_FixedBasePrecomputation<ECPPoint>& bpc =
        params.GetBasePrecomputation();
    //计算公钥
    const ECPPoint point =bpc.Exponentiate(
        params.GetGroupPrecomputation(),key.GetPrivateExponent());
    cout << "模: " << params.GetCurve().GetField().GetModulus()
        << endl;//打印模值
    //打印乘法因子
    cout << "乘法因子: " << params.GetCofactor() << endl;
    cout << "系数: " << endl;
    cout << "A: " << params.GetCurve().GetA() << endl;//打印参数A
    cout << "B: " << params.GetCurve().GetB() << endl;//打印参数B
    cout << "基点: " << endl;
    //打印参数X
    cout << "X: " << params.GetSubgroupGenerator().x << endl;
    //打印参数Y
    cout << "Y: " << params.GetSubgroupGenerator().y << endl;
    cout << "公共点: " << endl;
    cout << "x: " << point.x << endl;//打印参数x
    cout << "y: " << point.y << endl;//打印参数y
    cout << "秘密指数: " << endl;
```

```
88      cout << key.GetPrivateExponent() << endl;
89  }
```

执行程序，程序的输出结果如下：

```
模：14615016373309029182036848327162830196537850593 27.
乘法因子：1.
系数：
A：14615016373309029182036848327162830196537850593 24.
B：16323579130616811054660491940327157953054834541 3.
基点：
X：42582623172388835044654159270140906591363556877 0.
Y：20352011416290410787399145795734689202798264197 0.
公共点：
x：47833518382929478243623365396233697910673430597 1.
y：14697312571199342018609975334836914538255152190 8.
秘密指数：
20172169402967204665594385487243632431810305387 4.
message：49206C696B652043727970746F6772617068792E
ciphere：04A6CC206C150538507713211272DB1A9DBED21EB6BAC08E9F26B6723C0134A1A
3C8D6460DD3F93A1D5B785E58F41EC66E61608F2354DA30973DEC0F62047E92EFCD90EF
62835F054884CDD2919F44902D
recover：49206C696B652043727970746F6772617068792E
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》