《第五章 随机数发生器》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LC_RNG算法

以LC_RNG为例演示RandomNumberGenerator类各成员函数的使用方法，代码如下：

```
#include<rng.h> //包含LC_RNG算法的头文件
#include<iostream> //使用cout、cin
using namespace std; //std是C++的命名空间
using namespace CryptoPP; //CryptoPP是CryptoPP库的命名空间
#define Array_Size 64
int main()
{
    //定义一个LC_RNG随机数发生器对象，并设置其种子为123456
    LC_RNG rng(123456);
    //继承自Algorithm类的方法
    cout << "算法的标准名称: " <<rng.AlgorithmName() << endl;
    cout << "是否允许额外的熵输入: "
        << boolalpha << rng.CanIncorporateEntropy() << endl;
    if(rng.CanIncorporateEntropy())
    {//允许额外的熵输入，则输入"asdffq42"
        try
        {
            byte str[] = "asdffq42";
            rng.IncorporateEntropy(str,sizeof(str));
        }
        catch(const exception& e)
        {//出现异常
            cout << e.what() << endl; //打印异常原因
        }
    }
    cout << "产生一个比特的随机数: " << rng.GenerateBit() << endl;
    cout << "产生一个字节的随机数: " << rng.GenerateByte() << endl;
    byte output[Array_Size+1]={0};//定义一个缓冲区
    //产生Array_Size字节长度的随机数
    rng.GenerateBlock(output,Array_Size);
    cout << "产生Array_Size长度的随机数（十六进制）: " << endl ;
    for(int i=0; i < Array_Size;++i)
    {//将获得的随机数转换成十六进制并输出
     printf("%02X",output[i]);
    }
    cout << endl;
    cout << "产生一个100到1000之间的随机数: "
        << rng.GenerateWord32(100,1000) << endl;
    //丢弃掉随机数发生器接下来产生的100个字节数据
    rng.DiscardBytes(100);
    int arry[] = {1,2,3,4,5,6,7,8,9,10};
    rng.Shuffle(arry,arry+10); //打乱数组arry中元素的顺序
    cout << "打乱arry数组中元素的顺序: ";
```

```
44      for(int i=0; i < 10;++i)
45          cout << arry[i] << " "; //输出打乱结果
46      cout << endl;
47      return 0;
48  }
```

执行程序，程序的输出结果如下：

```
算法的标准名称：unknown
是否允许额外的熵输入：false
产生一个比特的随机数：0
产生一个字节的随机数：K
产生Array_Size长度的随机数（十六进制）：
5EBF970F871F8E3D63F902384677BCD20162F1C4E05696B278DDCB2C264432C9509EFF776
A55108F5E144D2F7BB9D6F64F687B539188693A7565546691EBEBBB
产生一个100到1000之间的随机数：105
打乱arry数组中元素的顺序：6 2 8 5 7 3 10 4 9 1
请按任意键继续. . .
```

2 使用AutoSeededX917RNG算法

```cpp
#include<osrng.h> //包含AutoSeededX917RNG算法的头文件
#include<aes.h> //AES算法的头文件
#include<iostream> //使用cout、cin
#include<fstream> //使用ofstream
#include<hrtimer.h> //使用Timer
using namespace std; //std是C++的命名空间
using namespace CryptoPP; //CryptoPP是CryptoPP库的命名空间
#define Array_Size 64
int main()
{
    //定义一个AutoSeededX917RNG对象
    //用分组密码算法AES根据X917RNG框架构造一个随机数发生器
    AutoSeededX917RNG<AES> rng; //不需要外部输入种子
    //定义一个文件对象
    ofstream file("data.dat",ios_base::binary | ios_base::out);
    byte output[Array_Size];
    cout << "开始生成数据..." << endl;
    Timer tmer;//定义一个Timer对象，用于统计时间
    tmer.StartTimer(); //开始计时
    for(int i=0; i < 1953125 ; ++i)
    {//64*8*1953125=1000*1000*1000=1G(bits)
        //每次产生Array_Size字节长度的随机数，并存入文件data.dat中
        rng.GenerateBlock(output,Array_Size);
        file.write((const char*)output,Array_Size); //将数据写入文件中
    }
    cout << "数据生成完毕..." << endl;
    cout << "生成1Gbits数据总共耗时（秒）："
        << tmer.ElapsedTimeAsDouble() << endl;
    file.close();//关闭文件
    return 0;
}
```

执行程序，程序的输出结果如下：

```
开始生成数据...
数据生成完毕...
生成1Gbits数据总共耗时（秒）：5.40707
请按任意键继续. . .
```

3 以Pipeling范式方式使用AutoSeededX917RNG算法

```
#include<iostream> //使用cout、cin
#include<osrng.h> //使用AutoSeededX917RNG算法
#include<aes.h> //使用AES算法
#include<files.h> //使用FileSink、StringSource
#include<filters.h> //使用RandomNumberSource、RandomNumberSink
#include<hrtimer.h> //使用Timer
using namespace std; //std是C++的命名空间
using namespace CryptoPP; //CryptoPP是CryptoPP库的命名空间
#define Array_Size 64
int main()
{
    //定义一个AutoSeededX917RNG对象
    //用分组密码算法AES根据X917RNG框架构造一个随机数发生器
    AutoSeededX917RNG<AES> rng; //不需要外部输入种子
    //定义一个string对象
    string str = "I like Cryptography.";
    //以字符串str为外部熵，输入至rng对象中
    StringSource sSrc(str,true,new RandomNumberSink(rng));
    //定义一个文件对象
    ofstream file("data.dat",ios_base::binary | ios_base::out |
        ios_base::app);
    cout << "开始生成数据..." << endl;
    Timer tmer; //定义一个Timer对象，用于统计时间
    tmer.StartTimer(); //开始计时
    for(int i=0; i < 1953125 ; ++i)
    {//64*8*1953125=1000*1000*1000=1G(bits)
        //每次产生Array_Size字节长度的随机数，并存入文件data.dat中
        RandomNumberSource rSrc(rng,Array_Size,true,new FileSink(
            file));
    }
    cout << "数据生成完毕..." << endl;
    cout << "生成1Gbits数据总共耗时: "
        << tmer.ElapsedTimeAsDouble() << endl;
    return 0;
}
```

4 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》