**《现代密码学》实验报告**

|  |  |
| --- | --- |
| 实验名称：实验8 | 实验时间：2022.09.01 |
| 学生姓名：陈浚铭 | 学号：12345678 |
| 学生班级：20网安 | 成绩评定： |
| **一、实验目的：**  实现基于离散对数的伪随机数生成器， 深入理解快速的modular exponentation 的c语言代码 | |
| **二、实验内容：**  用C\C++实现基于离散对数的伪随机数生成器  参数要求：  p=382769750272491 1605128590080133 13791412554337635849923335522166 11 485936638623; q=1701411834604692317316885826830 58187039; g=1559302665980118342229759829014  54727 795072371119369283727864229 55200978648323;  seed = 学号 (20337021) | |
| **三、实验原理：**  我们需要实现一个对于大数快速求modular exponentation的算法。从维基百科可以找出相关的算法。  伪代码：  **function** modular\_pow(base, exponent, modulus) **is**  **if** modulus = 1 **then**  **return** 0  [Assert](https://en.wikipedia.org/wiki/Assertion_(computing)) :: (modulus - 1) \* (modulus - 1) does not overflow base  result := 1  base := base **mod** modulus  **while** exponent > 0 **do**  **if** (exponent **mod** 2 == 1) **then**  result := (result \* base) **mod** modulus  exponent := exponent >> 1  base := (base \* base) **mod** modulus  **return** result  这个代码是有效率的。它使用了exponential by squaring。 注意到指数e可以转换成binary的形式    当中e 为n bit, a\_i 为0 或者1，0 < I < n, 有定义， a\_(n-1) = 1  因为a^(m +n) = a^m a^n，我们发现有b^e可以写成  b^e = b^()∑ n-10 ai2i) = Π n-1i = 0 ba\_i 2^i  从而得到解 c = Π n-1i=0 ba\_i 2^i  （mod m）  我们从公钥密码体制与离散对数理解到基于离散对数的密码体制。当中， (G, .)为乘法群， alpha ∈ G阶为n，元素beta ∈ <a>， 满足：  alpha^a = beta  类似我们在这里的随机生成器定义序列关系：  p为256bit的素数， q 是128bit素数，小于p-1, g∈ Z\*q 是一个q阶元素。  种子s0 是 Z\*q 是一个元素对于 0 <= I <= l-1, l = 512  有  si+1 = (gs\_i mod p) mod q, 这种递推关系，  定义z\_i = s\_i mod 2，我们就可以定义了i 从0 到511 这么多bit的01序列，作为2进制的伪随机数。  注意到伪随机数生成器的特征：在我们的生成器是基于递归的公式来实现的。所以伪随机数是每一个比特是依赖于种子以及伪随机数前面前面的比特的。 | |
| **四、实验步骤（源代码）： 实现代码再bigint.c**  main函数：  modulus exponential 运算代码：    摸2运算： | |
| **五、实验结果：**  **Random number:**  ** of zeroes: 270Num of ones: 242**  **Run time: 0.048000** | |
| **六、实验总结：**  通过这次实验，我能够掌握到伪随机数的大概形式(递归关系），这也让我可以类比其他伪随机生成器，比如bbs。这次实验让我更深入理解到modular exponentation运算的c语言实现。  reference: https://en.wikipedia.org/wiki/Modular\_exponentiation | |