**密码学原理与实践**

**实验-2-密码执行(1)**

**学院名称 智能与计算学部**

**专 业 网络空间安全**

**学生姓名 石子跃**

**学 号 3020244294**

**年 班 级 2020级1班**

**2022年 10 月 29 日**

# 实验问题-3 设计生成强素数的算法

## 1.1实验原理





****

## 1.2 实验要求

设计一个生成强素数的算法

要求：生成一个指定长度的强素数。

输入：强素数的比特长度0<l<=32

输出：比特长度为l的强素数p， p+1的大素因子s， p-1的大素因子r， r-1大素因子t

(要求：s，r，t比特长度约为l/2，比特长度误差±4；如无符合要求的强素数程序应该给出明确说明)

## 1.3关键技术的程序实现方法与代码：

#### 算法1 MILLER-RABIN

1. *//Miller-Rabin输入一个奇整数n>3，和一个安全参数t>=1*
2. *//输出 1为素数，0为合数*
3. bool MILIERRABIN(ul n,int t){
4. if(n<=3 || t<1) return 0;
5. ul r = n-1;
6. int s = 0;
7. while((r & 0x1)==0 && r){*//(1)写n-1=2^s \* r*
8. r = r >> 1;
9. s++;
10. }
11. *//cout<<"s:"<<s<<endl;*
12. *//cout<<"r:"<<r<<endl;*
13. for(int i=0;i<t;i++){
14. srand((ul)time(0));
15. ul a = (rand() % (n-2-2+1)) + 2; *//选择一个随机整数 a, 2<=a<=n-2*
16. *//cout<<"a:"<<a<<endl;*
17. ul y = rsa\_modExp(a,r,n);*//计算 y = a^r(mod n)*
18. *//cout<<"y:"<<y<<endl;*
19. if(y!=1 && y!=n-1){
20. int j = 1;
21. while(j<=s-1 && y!=n-1){
22. y = rsa\_modExp(y,y,n);*//计算 y = y^2(mod n)*
23. if(y==1){
24. *//cout<<n<<"为合数"<<endl;*
25. return 0;
26. }
27. j = j + 1;
28. }
29. if(y!=n-1){
30. *//cout<<n<<"为合数"<<endl;*
31. return 0;
32. }
33. }
34. }
35. *//cout<<n<<"为素数"<<endl;*
36. return 1;
37. }

#### 算法2 随机搜索可能的素数

1. ul gen\_prime\_limit\_L(int L){
2. ul limt\_s = 0x1<<L; *//限制s的规模*
3. ul s; *//保证s的比特长度正好等于所计算的*
4. int i;
5. while (true)
6. {
7. s = (rand()|limt\_s)%(limt\_s-1)|limt\_s>>1;
8. bool cont = true;
9. for(i=0; i<45; i++){
10. if(s%save\_prim[i]==0&&s!=save\_prim[i]){
11. cont = false;
12. break;
13. }
14. }
15. if(cont){
16. if(MILIERRABIN(s,CONFIDENCE\_t)){
17. return s;
18. }
19. }
20. }
21. }

#### 算法3 Gordon算法

1. *//Gordon的产生强素数算法，产生强素数p*
2. *//指定长度L 的强素数 0<L<=32*
3. *//输出比特长度为L的强素数p，p+1的大素因子s,p-1的大素因子r,r-1大素因子t*
4. *//(要求：s，r，t比特长度约为⎡l/2⎤，比特长度误差±4；如无符合要求的强素数程序应该给出明确说明)*
5. ul Gordon(int L){
6. srand((ul)time(0));*//用于产生随机数*
7. int L\_s,L\_t;*// s的比特长度，t的比特长度*
8. gen\_bit\_L(L,L\_s,L\_t);
9. *// cout<<"s的比特长度L\_s:"<<L\_s<<endl;*
10. *// cout<<"t的比特长度L\_t:"<<L\_t<<endl;*
11. L\_s = L/2;
12. L\_t = L/2-1;
13. cout<<"The bit length of sL\_s:"<<L\_s<<endl;
14. cout<<"The bit length of t -> L\_t:"<<L\_t<<endl;
15. ul s = gen\_prime\_limit\_L(L\_s);
16. ul t = gen\_prime\_limit\_L(L\_t);
17. *// s = 51427;*
18. *// t = 32119;*
19. while (1)   *//直到跑出符合规格的p*
20. {
21. cout<<"s:"<<s<<endl;
22. cout<<"t:"<<t<<endl;
23. cout<<"The bit length of s -> L\_s:"<<check\_L\_num(s)<<endl;
24. cout<<"The bit length of t -> L\_t:"<<check\_L\_num(t)<<endl;
25. *//选择一个整数i0，在序列i0,i0+1,i0+2...中发现第一个素数i，r = 2\*i\*t+1*
26. ul i0 = 0;
27. ul r;
28. for(i0;;i0++){
29. r = 2\*i0\*t+1;
30. if(MILIERRABIN(r,CONFIDENCE\_t))
31. break;
32. }
33. int L\_r = check\_L\_num(r);
34. cout<<"The bit length of r -> Lforr:"<<L\_r<<endl;
35. cout<<"r:"<<r<<endl;
36. *//计算p0 = 2\*(s^(r-2)(mod r)\*s-1*
37. ul p0 = 2\*rsa\_modExp(s,r-2,r)\*s-1;
38. int L\_p0 = check\_L\_num(p0);
39. cout<<"The bit length of p0 ->  Lforp0:"<<L\_p0<<endl;
40. cout<<"p0:"<<p0<<endl;
41. if(L\_p0>L){*//如果p0都大的话，直接取小*
42. *//s = gen\_prime\_limit\_L(L\_s-1);*
43. t = gen\_prime\_limit\_L(check\_L\_num(t)-1);
44. continue;
45. }
46. ul p;
47. ul j0 = 0;
48. for(j0;;j0++){
49. p = p0+2\*j0\*r\*s;
50. *// cout<<"p:"<<p<<endl;*
51. if(MILIERRABIN(p,CONFIDENCE\_t))break;
52. }
53. int L\_p = check\_L\_num(p);
54. cout<<"The bit length of p -> L\_p:"<<L\_p<<endl;
55. cout<<"p:"<<p<<endl;
56. if(L\_p==L){
57. cout<<"FIND IT!!!!!!!!!!!!!!!!!!!!!"<<endl;
58. cout<<"\ts:"<<s<<endl;cout<<"\tThe bit length of s -> L\_s:"<<check\_L\_num(s)<<endl;
59. cout<<"\tt:"<<t<<endl;cout<<"\tThe bit length of t -> L\_t:"<<check\_L\_num(t)<<endl;
60. cout<<"\tr:"<<r<<endl;cout<<"\tThe bit length of r -> L\_r:"<<L\_r<<endl;
61. cout<<"\tp:"<<p<<endl;cout<<"\tThe bit length of p -> L\_p:"<<L\_p<<endl;
62. return p;
63. }
64. else if (L\_p>L)
65. {
66. t = gen\_prime\_limit\_L(L\_t-1);
67. cout<<"P too big"<<endl;
68. }
69. else {
70. t = gen\_prime\_limit\_L(check\_L\_num(t)+1);
71. cout<<"P too small"<<endl;
72. }
73. }
74. }

## 1.4数据测试结果

由于算法问题，随机性太大，导致可能生产素数的时间非常长

可以得到正确结果

