常见素性检验算法的比较分析*

许 斌 张艳硕 吕正宏
北京电子科技学院,北京市 100070

摘 要: 在现代密码系统中,大素数对一些加密系统的建立来说有着不可忽视的作用,如 RSA 密码系统和椭圆曲线密码体制 ECC,作为应用最广和最具有发展潜力的两个密码体系,其安全性都是建立在大素数之上。而大素数的检验显得尤为重要,常见的素性检验算法包括 Fermat 素性检验、Solovay-Strassen 素性检验、Miller-Rabin 素性检验、Pocklington 素性检验、Lucas-Lehmer 素性检验、Pepin 素性检验、Lucas 素性检验、AKS 素性检验等算法。素性检验算法可按照待检验数的形式分为一般形式素性检验算法和特殊形式素性检验算法,也可以按照检验结果的准确性分为概率型素性检验算法以及确定型素性检验算法。本文介绍了上述常见的素性检验的理论算法,并从不同分类、软件实现等方面对这些素性检验算法进行了比较分析,最后得出 Miller-Rabin 素性检验算法的综合效率最高。

关键词: 素性检验; Miller-Rabin 素性检验; 伪素数; 比较分析

中图分类号: TN918.1 **文献标识码:** A **文章编号:** 1672-464X(2021)4-25-37

1 引言

随着科技的发展,网络与信息安全的重要性在生活中体现得越来越明显。目前大多数信息在传输前都会进行加密和存储,而密码学中的加密系统很多都是建立在大素数基础之上的,比如 RSA 公钥密码系统和 ECC(Elliptic Curve Cryptography)椭圆曲线密码系统等。所以在进行加密的过程中,如何确定一种高效准确的素性检验算法对密码系统尤为重要。

从素数的诞生到现在,有关素性检验的研究从未间断过。关于素性检验算法的历史最早可以追溯到两千多年前的埃拉托色尼筛选法,该算法主要通过把自然数中的合数筛除来达到目的,Plaksin[10] 对该算法的统计特性进行了相关阐述。到了近代,有关素性检验的研究迅速发展。1636 年,Fermat[1] 提出了 Fermat 小定理,为后来的素性检验的理论研究奠定了坚实的基础。1876 年,E.Lucas[2] 提出 Lucas 定理,据此,Lucas 的 n-1 检验(后文简称 Lucas 检验)诞生。1914 年,H.C.Pocklington[5] 改进了该算法,只要求知道 n-1 的部分分解(Lucas 检验要求知道 n-1 的完全分解),创造出一种确定型素性检验算法 Pocklington 素性检验。1930 年,Lehmer[13] 完善

* 基金项目: 2020 年教育部新工科项目"新工科背景下数学课程群的教学改革与实践","信息安全"国家级一流本科专业建设点和基本科研业务费(项目编号: 328202008)

** 作者简介: 许斌(2001-),男,本科生在读,信息与计算科学专业。Email: 412684729@qq.com
张艳硕(1979-),男,通信作者,博士,副教授,从事密码数学理论研究。Email: zhang_yanshuo@163.com
吕正宏(1999-),男,本科生在读,信息安全专业。Email: 1963422201@qq.com

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

了 Lucas 关于梅森数的素性检验，创造出可针对于这类特殊形式数的素性检验算法 Lucas-Lehmer 素性检验，检验效率大大提高。1975 年，Pratt [11] 证明了整数的素性测试可以在非确定性多项式时间内完成。1976 年，Miller [12] 提出了一种以 Extended Reimann Hypothesis（广义黎曼假设）为条件获得了一个多项式时间复杂度的确定型的素性检验算法，后来 Rabin [3] 在此基础上将其改进为一个无条件的概率型素性检验算法，也就是著名的 Miller-Rabin 算法。1986 年，Goldwasser 和 Kilian 提出了著名的 GK 算法，这是一个基于椭圆曲线的确定型素性检验算法。1990 年，Baillie、Pomerance、Selfridge 和 Wagstaf [14] 结合 Fermat 检验和 Lucas 检验的强形式提出 Baillie-PSW 素性检验，这是一种概率型素性检验算法。2002 年，印度数学家 Agrawal 和他的两个学生 Kayal、Saxena [6] 提出了著名的 AKS 算法，这是一个多项式时间的确定型素性检验算法，也是素性检验的一个重大突破。2003 年，Bernstein [15] 针对 AKS 算法中关于 r 的寻找进行改进，从而得到改进的 AKS 算法，后来 Berrizbeitia [16] 针对 AKS 算法中的第二步和第五步进行了相关改进，使其时间复杂度最大为 $O(log^6 n)$。Adleman 和 Huang 利用亏格为 2 的超椭圆曲线替换 GK 算法中的椭圆曲线找到了一种超椭圆曲线素性检验算法，Atkin 利用 CM（Complex-Multiply）的方法改进 GK 算法得到了椭圆曲线素性证明算法（Proof of Primality of Elliptic Curve 算法，ECPP 算法）[17]。至今，ECPP 算法仍是当前大整数素性检验最快的算法之一。2020 年 Moshonkin 和 Khamitov [4] 提出一种新的概率型素性检验算法。

这些算法使得在计算机上检验一个超过 1000 位的奇数变得非常容易。但目前如何在多项式时间内检验一个素数以及如何分解一个大数，仍是素数研究的中心。就像前面提到的，素性检验研究的唯一目的就是确定一种高效准确的检验方法，这也是 RSA、ECC 这类密码体制中最为关键的一环。目前关于素性检验的研究成果都比较分散，彼此的联系也很少，虽然关于素性检验的算法有很多，但是仍然没有一个比较系统的比较分析。本文全面地介绍了素性检验，对一些常见的素性检验算法给出具体阐述，并按照它们的特性进行分类，从不同分类上比较分析这些素性检验算法，同时也从软件实现方面对 Lucas 素性检验、Miller-Rabin 素性检验、AKS 素性检验这三类算法的运行时间和运行结果进行了比较分析，最后得出 Miller-Rabin 素性检验算法的综合性能最为突出。

2 常见素性检验算法

素数又称质数，是指一个大于 1 的自然数，除了 1 和它自身外，不能被其他自然数整除的数。检验一个整数是否为素数即为素性检验。最简单的素性检验算法便是试除法，利用素数的定义，用小于待检验数的非 1 整数去一一试除，如果有一个数能整除待检验数，则待检验数为合数。统计显示，大约有四分之三的奇数有 100 以内的素因子，但是对于大素数来说，试除法无法应用在具体的系统中，这就需要效率更高的素性检验算法来实现了。

常见素性检验算法可按照待检验数的形式分为一般形式素性检验算法和特殊形式素性检验算法，也可以按照检验结果的准确性分为概率型素性检验算法以及确定型素性检验算法。这两种分类方法并不是相互独立的，而是相互交叉的。一般形式素性检验是针对于全体整数而言的，它适用于所有的整数。而根据每一个具体算法是否能确定待检验数的素性这一思路，又可以将一般形式素性检验算法分为确定型素性检验算法与概率型素性检验算法。同样的，特殊形式素性检验算法也可以分为确定型和概率型素性检验算法。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

2.1　常见的一般形式素性检验算法

一般形式素性检验算法适用于所有的大数，但会因为每个待检验数自身的独特性而表现出不同的效率。下面介绍一些常见的一般形式素性检验算法。

2.1.1　Fermat 素性检验

Fermat 小定理[1] 若 n 为素数，对任意数 a，只要 gcd(a,n) = 1 就有 $a^{n-1} \equiv 1(modn)$。

根据该定理，任取整数 a，如果 (a,n) = 1 且 $a^{n-1} \neq 1(modn)$，则输出 n 为合数，否则输出 n 可能为素数，这就是 Fermat 素性检验基本思想。

算法描述:

输入: n;

输出: n 是否是素数;

(1) $a = 1$;

(2) *if* (a,n) $\neq 1$,a ++;

(3) *if* $a^{n-1} \equiv 1(modn)$; a ++ *goto*(2);

(4) *if* $a \geqslant n$ 输出 n 可能是素数;

(5) *else* 输出 n 是合数。

2.1.2　Solovay-Strassen 素性检验

定理 2.1[2] 设 $p > 2$ 是一个素数，则对于任意的 $a \geqslant 0$，$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}}(mod\ p)$，其中 $\left(\frac{a}{p}\right)$ 为 Jacobi 符号。

定理 2.2[2] 如果 $n > 2$ 是一个奇合数，则至少有 50%的 a $\in Z$，即 $1 \leqslant a \leqslant n-1$，使得同余式 $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}}(modn)$ 不成立。

根据以上两个定理，Solovay-Strassen 素性检验描述如下:

输入: n;

输出: n 是否是素数;

(1) 随机均匀的选取 a $\in \{1,2,3,\cdots,n-1\}$;

(2) 计算 gcd(a,n);

(3) 如果 gcd(a,n) $\neq 1$，则 n 不是素数;

(4) 计算 $\left(\frac{a}{n}\right)$ 和 $a^{\frac{n-1}{2}}(modn)$;

(5) 如果 $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}}(modn)$，则 n 可能是素数; 否则，n 不是素数。

2.1.3　Miller-Rabin 素性检验

Miller-Rabin 素性检验算法[3] 是目前实际应用中使用最普遍的算法，它是由 Fermat 检验算法改进而来的: 设 n 为奇素数，令 $n-1 = 2^l m$，其中 l 是非负整数，m 是正奇数，如 a 与 n 互素，则 $a^l \equiv 1(modn)$ 或 $\exists r, 0 \leqslant r < l$，使得 $a^{2^r m} \equiv -1(modn)$。

其原理为当 n 为素数时，方程 $x^2 \equiv 1(modn)$ 只有 ± 1 两个解。

算法描述:

输入: 一个大于 3 的奇整数 n 和一个大于等于 1 的安全参数 t(用于确定测试的轮数);

输出: 返回 n 是否是素数;

(1) 将 n-1 表示成 $2^s r$;

(2) 对 i 从 1 到 s 循环下面的操作:

1) 选择一个随机整数 a($2 \leqslant a \leqslant n-2$);

2) 计算 $y \leftarrow a^r(modn)$;

3) 如果 y $\neq 1$ 并且 y $\neq n-1$ 做下面的操作，否则跳转到(c);

(a) j←1;

(b) 当 j $\leqslant s-1$ 并且 $y \neq n-1$ 时做下面操作，否则跳转到(c);

{

计算 $y \leftarrow y^2(modn)$;

如果 $y = 1$ 返回“合数”;

否则 $y \leftarrow y+1(modn)$;

}

(c) 如果 $y \neq n-1$ 则返回“合数”;

(3) 返回“素数”。

通常认为 Miller-Rabin 算法误判概率为 1/4，但是目前最新的研究表明，关于 Miller-Rabin 算法误判概率，可以进行进一步的修正[3]:

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

若 n 以 a 为基通过了 Miller-Rabin 素性检验，则称 a 为奇数 n 的素性见证数。定义 $w(n)$ 为奇数 n 的素性见证数的集合，$\varphi(n)$ 为奇数 n 的欧拉函数值，$N_w(n)$ 为集合 $w(n)$ 的元素个数，在最初的误判概率计算中，根据 $N_w(n) < \frac{1}{4}\varphi(n)$，可以得出误判概率为 1/4。如果能够得到 $N_w(n)$ 的精确计算公式，便能很详细地描述 Miller-Rabin 素性检验的误判概率。表 1 是不同形式合数的 $N_w(n)$ 的精确公式，其中若 $m = 2^s \cdot u$，定义 $bin(m) = s$，$odd(m) = u$。

表 1 不同形式 n 的 $N_w(n)$ 表达式

n 的形式	$N_w(n)$
$n = p^k$	$p - 1$
$n = pq$，$d = gcd(p-1, q-1)$	$\frac{(odd(d))^2 \cdot (4^{bin(d)} + 2)}{3}$
$n = p_1 \cdot p_2 \cdots p_k$，$s = min\{bin(d_1), \cdots, bin(d_k)\}$，$d_i = gcd(p_i - 1; \prod_{j \neq i} p_j - 1)$，$u_i = odd(d_i)$	$u_1 \cdot u_2 \cdots u_k \cdot (1 + \frac{2^{ks} - 1}{2^k - 1})$

根据表 1 可以看出，误判概率和待检验数长度呈负相关关系。待检验数越长，误判概率越低；待检验数越短，误判概率越高。

2.1.4 Pocklington 素性检验

定理 2.3[5] 若 a 使得 $a^{n-1} \equiv 1(modn)$，且对 n-1 的任一素因子 p，有 $a^{\frac{p-1}{n}} \not\equiv 1(modn)$，则 n 为素数。

定理 2.4[5] 假设 $n = ab + 1 > 1$，$0 < a \leqslant b + 1$，若对 b 的任何一个素因子 p，都存在一个整数 x，使得 $\gcd(x^{\frac{n-1}{p}} - 1, n) = 1$ 和 $x^{n-1} \equiv 1(mod\ n)$ 同时满足，则 n 为素数。

基于以上两个定理，Pocklington 素性检验描述如下：

输入：n ≥ 3；

输出：n 是否是素数；

首先对 n-1 进行素因子分解，即 $n-1 = q_1^{s_1} q_2^{s_2} \cdots q_t^{s_t}$，$q_1, q_2 \cdots q_t$ 为不同的素数，再依次提取 $q_t^{s_t}$、$q_{t-1}^{s_t-1} \cdots\cdots$，直到其中 $F = q_1^{s_1} q_2^{s_2} \cdots q_j^{s_j}$（j <t）满足 $F \leqslant sqrt(n)$。

如果对于每一个 i，$i \in [1, j]$，存在一个 a_i，满足：

（1）$a_i^{n-1} \equiv 1(modn)$；

（2）$gcd(a_i^{\frac{n-1}{q_i}} - 1, n) = 1$；

则输出 n 为素数。

该算法主要是围绕定理 2.4 进行展开的，下面对定理 2.4 进行证明。

已知 b 的所有素因子，不妨设 a 与 b 互素，反证，假设 n 有素因子 q，且 $q \leqslant \sqrt{n}$，对 b 的每个素因子 p，存在整数 x_p，使得 $x_p^{n-1} \equiv 1(\text{mod } q)$，$x_p^{\frac{n-1}{p}} \not\equiv 1(modq)$，即 $ord_q(x_p) \mid (n-1)$，$ord_q(x_p) \nmid \frac{n-1}{p}$。如果 $p^k \mid b$，可知 $ord_q(x_p^a) \mid b$，$ord_q(x_p^a) \nmid \frac{b}{p}$，令 $x = \prod_{p \mid a} x_p^a$，令 $ord_q(x) = b$，由此推出 $q-1 \geqslant b$ 及 $q^2 \geqslant (b+1)^2 \geqslant a(b+1) \geqslant ab + a \geqslant n$，又因为 $q \leqslant \sqrt{n}$，所以有 $q^2 = n$，$a = b + 1$，$a = 1$，与假设矛盾，证毕。

上述算法中，Fermat 检验、Solovay-Strassen 检验、Miller-Rabin 检验都是概率型素性检验算法，它们都存在相应的误判概率，而 Pocklington 检验是一种确定型素性检验算法。可以看出前三个概率型素性检验算法的检验条件是越来越严格的，其实它们的准确率也是越来越高的，这在后文的比较分析中有详细介绍。

2.2 常见的特殊形式素性检验算法

由于一般形式素性检验算法的效率较低，一些针对特殊形式数的素性检验算法便产生了。特殊形式的素性检验算法，顾名思义，主要是针对具有特殊形式的奇数进行的素性检验算法。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

下面介绍三种常见的特殊形式的素性检验算法。

2.2.1　Lucas-Lehmer 素性检验

梅森数是指形如 $2^p - 1$ 的正整数，其中指数 p 是素数。

针对梅森数，有以下检验算法：

定理 2.5[13] 对于一个梅森数 M_p，有这样的一个数列 $L_{n+1} = L_n^2 - 2$，$L_0 = 4$，当且仅当 $L_{p-2} \equiv 0(modM_p)$ 时，M_p 为质数。

基于以上思想，有以下算法描述：

输入：梅森数 n；

输出：n 是否为素数；

(1) $p = \log_2 n + 1$；

(2) 对于 i 从 0 到 $p - 2$，根据 $L_0 = 4$，计算 $L_{i+1} = L_i^2 - 2$。

(3) 如果 $L_{p-2} \equiv 0(modn)$，则输出 n 为素数，否则输出 n 为合数。

2.2.2　针对形式为 $h \cdot 2^n \pm 1$ 的素性检验

该算法是依据 Lucas-Lehmer 素性检验推广得出的，时间复杂度为 $O(log^2 M)$ [7]。具体描述如下：

输入：h，n（其中 h 是小于 2^n 的奇数，n 是正整数）；

输出：$M = h \cdot 2^n \pm 1$ 是否为素数；

(1) 找到一个 $p \equiv 1(mod4)$，使得 $\left(\frac{M}{p}\right) = -1$，其中 $\left(\frac{M}{p}\right)$ 是勒让德符号。

(2) 找到一个复数 α，$\bar{\alpha}$ 是其共轭复数，使得 $p = \alpha\bar{\alpha}$。

(3) 计算 $s_0 = (\alpha/\bar{\alpha})^h + (\bar{\alpha}/\alpha)^h$，并按照 $s_k = {s_{k-1}}^2 - 2(k \geqslant 1)$ 生成 Lucas 序列。

判断 $s_{n-2} \equiv 0(modM)$ 是否成立，如果成立，则说明 M 是一个素数，否则，M 不是一个素数。

2.2.3　Pepin 素性检验

费马数是指形如 $2^{2^n} + 1$ 的正整数，其中指数 n 是非负整数。

针对费马数，有以下检验算法：

定理 2.6[8] 设 $n = 2^k + 1$，$k > 1$，则 n 为素数的充要条件为 $3^{\frac{n-1}{2}} \equiv -1(modn)$。

输入：费马数 F_n；

输出：F_n 是否为素数；

(1) 计算 $y = 3^{2^{(2^n-1)}} + 1$；

(2) 如果 $y \equiv 0(modF_n)$，则说明 F_n 为素数；否则，F_n 为合数。

对于这些特殊形式的大数，站在效率优先的落脚点，可以优先考虑上述算法，这样在实际运用中可以起到事半功倍的效果。

2.3　常见的概率型素性检验算法

概率型素性检验是指通过该种方法只能以某个概率说明待检验数为素数。常见的概率型素性检验算法如 Fermat 素性检验、Solovay-Strassen 素性检验、Miller-Rabin 素性检验已经在上述一般形式的素性检验算法中描述过，这里介绍一种 2020 年由 Moshonkin 和 Khamitov[14] 提出的一种新的素性检验算法。

输入：n 为一个奇正整数；

输出：n 是否为素数；

(1) 寻找是否存在 a，$b \in N$，使得 $n = a^b$。其中，$b \in [2, log_2 n - 1]$。如果能找到，说明 n 不是一个素数，否则进行(2)。

(2) 随机选取 r(mod n)，如果 $(r, n) \neq 1$，则说明 n 不是一个素数，否则进行(3)。

(3) $C = r^2(modn)$，寻找 b 满足 $b^2 - 4c \neq 0(modn)$ 和 $\left(\frac{b^2 - 4c}{n}\right) = 0$ 或 -1。其中 $\left(\frac{b^2 - 4c}{n}\right)$ 是雅可比符号。

1) 若 $\left(\frac{b^2 - 4c}{n}\right) = 0$，说明 n 是一个合数；

2) 若 $\left(\frac{b^2 - 4c}{n}\right) = -1$，令 $f(x) = x^2 + bx + c$。如果 $x^{(n+1)/2} \neq \pm r(modf(x))$，说明 n 是合数，否则进行(2)。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

该算法一轮下来的误判概率不超过 $\frac{1}{2}$，即通过一轮该算法的检验，待检验数是素数的概率大于 $\frac{1}{2}$，虽然这一检验的效率不如 Miller-Rabin 素性检验，但是一些启发性论证表明，对该算法的效率估计是很粗略的，可以关注该算法的实际运行效率[4]。

概率型素性检验算法在目前的实际应用上相比确定型检验算法要更胜一筹。在实际应用中，选择合适的概率型素性检验算法往往是一个值得深思的问题。其中，误判概率是一个很重要的指标。如果计算机计算能力很强，可以选择误判概率较低但算法复杂度较高的算法；同样如果计算能力较差，可以选择误判概率高但算法复杂度较低的算法，通过多次检验降低误判概率。

2.4 常见的确定型素性检验算法

确定型素性检验算法所检验出来的结果即为确定的，所以确定型算法的优势在于其去除了误判概率这一缺点，对于一些要求必须是素数的情况，大都采用该类算法。但在算法复杂度上，确定型素性检验算法无法达到概率型素性检验算法的优化。因此，目前关于确定型素性检验算法的时间复杂度优化也是素性检验研究这一领域的一个重点。常见的确定型素性检验算法如上述 Pocklington 素性检验、Lucas-Lehmer 素性检验、Pepin 素性检验等。下面介绍 Lucas 素性检验和 AKS 素性检验算法。

2.4.1 Lucas 素性检验

Lucas 定理[2] 设 n ∈ N，若存在一个整数 a（$1 < a < n, a^{n-1} \equiv 1(modn)$），且对（n-1）的每一个素数因子 q 都满足 $a^{\frac{n-1}{q}} \neq 1(modn)$，则 n 为素数。

算法描述：

输入：n；

输出：n 是否是素数；

设 n 是待检验的正整数，假定已知 $(n-1) = p_1^{b1} \times p_2^{b2} \times \cdots\cdots \times p_r^{br}$，其中 $p_1, p_2, \cdots\cdots, p_r$ 是（n-1）的互素因子。任选一个整数 a，a ∈（1，n），若以下条件能够得以满足：

（1）n 通过了 a 为基数的伪素性检验：

$a^{n-1} \equiv 1(modn)$；

（2）a 的实数阶能整除（n-1）并且对每一个 i，i ∈ [1，r]，有

$a^{\frac{n-1}{p_i}} \neq 1(modn)$；

则 n 为一个素数（当然 a 亦为模数 n 的一个原根）。

2.4.2 AKS 素性检验

AKS 素性检验算法[4] 以一个从 Fermat 小定理派生出的恒等式为理论基础，该等式可描述如下：

令 a ∈ Z，$n \in Z$，$X \in Z$，并且（a，n）= 1，则 n 是素数，当且仅当：$(X+a)^n = X^n + a(modn)$ 成立。

算法描述：

输入：整数 n>1；

输出：n 是否是素数；

（1）如果存在 a，b（a ∈ N，b>1）使得 $n = a^b$，则 n 为合数；

（2）找出最小的 r 使得 $O_r(n) > 4log^2 n$；

（3）如果对一些 $a \leqslant r$，使得 $1 < (a, n) < n$ 成立，则 n 为合数；

（4）如果 $n \leqslant r$，则 n 为素数；

（5）从 a = 1 到 $a = 2\sqrt{h(r)}\log n$，如果 $(X+a)^n \neq X^n + a(modn)$，则 n 为合数；

（6）输出 n 为素数。

上述算法中，h（r）为欧拉函数，它返回小于 r 并与 r 互质的数的个数。$O_r(n)$ 表示 n 模 r 的阶，即满足 $n^r \equiv 1(modr)$ 的最小 r 值。

Lucas 素性检验其实是利用了原根的特殊性质来对大奇数进行检验的，其检验过程类似于验证其原根的特殊性质，所以该算法的核心是寻找原根。AKS 检验算法主要是依据 Fermat 小定

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

理衍生而来的，目前关于确定型素性检验算法的研究大多是从改进 AKS 检验出发的，如何提升 AKS 检验算法的运行效率成为一大热点。

3　三类常见的一般形式检验算法比较分析

Fermat 素性检验、Solovay-Strassen 素性检验、Miller-Rabin 素性检验这三类检验算法是针对一般形式的素性检验算法，它们都可以快速地、低开销地检验大素数。由于这三类算法都是概率型素性检验算法，这里将分别从原理、对应产生的伪素数、误判概率三个方面对它们进行比较分析，详见表 2。

表 2　三类概率型素性检验算法比较

检验算法	原理(当 n 为素数时)	产生的伪素数	单轮误判概率
Fermat 素性检验	$a^n \equiv a(modn)$	费马伪素数	$\frac{1}{2}$
Solovay-Strassen 素性检验	$\left(\frac{a}{p}\right) \equiv a^{\frac{n-1}{2}}(modn)$	欧拉伪素数	$\frac{1}{2}$
Miller-Rabin 素性检验	$x^2 \equiv 1(modn)$ 只有 ± 1 两个解	强伪素数	$\frac{1}{4}$

若 n 满足 Fermat 小定理却仍为合数，此时称 n 为对基 a 的 Fermat 伪素数。当底为 2 时，最小的 Fermat 伪素数为 341，可以验证 $2^{340} \equiv 1(mod341)$，$341 = 11 \times 31$。特别的是，对于合数 n，如果对于所有与 n 互质的正整数 b，都有同余式 $b^{n-1} \equiv 1(modn)$ 成立，将这类数称为卡麦克尔数(Carmichael 数)，最小的卡麦克尔数为 561。正是因为卡迈克尔数的存在导致 Fermat 素性检验算法并不是一个确定型检验算法。当 n 为卡麦克尔数时，该算法会完全判断错误，对于非卡麦克尔数的合数，单轮的误判概率为 $\frac{1}{2}$，t 轮检验的误判概率至多是 $\frac{1}{2^t}$。

类似 Fermat 伪素数，存在一类数称为欧拉伪素数，对于奇合数 n 以及与其互素的自然数 a，如果 $a^{\frac{n-1}{2}} \equiv \pm 1(modn)$ 成立，则称 n 为关于 a 的欧拉伪素数。对于这类数，无法通过 Solovay-Strassen 素性检验来判断其是否为素数。如果一个待检验整数 n(n 不是欧拉伪素数)，通过了一次该检验算法，那么其为素数的概率至少为 50%[2]。当对 n 进行 t 次检验时，如果每次检验结果都为素数，那么该整数 n 是素数的概率为 $1-\frac{1}{2^t}$。当 t 足够大的时候，$\frac{1}{2^t}$ 可以被降到很小，在目前强大计算能力的环境下，该算法一般能够满足大数的素性检验。

同样，通过 Miller-Rabin 素性检验的合数被称为强伪素数，具体定义如下：

设 $n-1 = 2^l m$，m 为奇数，gcd(a,n) = 1，

若

1. $a^m \equiv 1(modn)$；

2. 存在 t，$1 \leqslant t < l$，$a^{2^t m} \equiv -1(modn)$；

则称 n 为以 a 为基的强伪素数。特殊的是，以任意 a 为基的强伪素数是素数。

定理 3.1[3] 对于一个合数 n，N(n) = {a | n 以 a 为基的强伪素数}，$\varphi(n)$ 表示 n 的欧拉函数，则 $|N(n)| < \frac{1}{4}\varphi(n)$。

一般情况下，Miller-Rabin 算法的时间复杂度都是基于定理 3.1 计算的。所以 Miller-Rabin 算法将一个合数检验成素数的 a，至多有 $\frac{1}{4}\varphi(n)$ 个，选择 t 个不同的 a，做 t 轮检验，误判概率至多为 $\frac{1}{4^t}$，因此正确率至少为 $1-\frac{1}{4^t}$。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

4 三类常见的特殊形式检验算法比较分析

在素数中有两类特殊的素数,分别是梅森素数和费马素数。表 3 介绍了一些关于不同形式数对应的检验算法。

表 3 三类不同形式的数对应的检验算法

算法	特殊形式
Lucas-Lemher 素性检验	梅森数
Pepin 素性检验	费马数
改进的 Lucas-Lemher	$h \cdot 2^n \pm 1$

梅森数是形式为 $2^n - 1$(其中 n 为素数) 的数,Lucas-Lemher 素性检验主要是依靠定理 4. 1。

定理 4. 1 [18] 对于一个梅森数 M_P,有这样的一个数列 $L_{n+1} = L_n^2 - 2$,$L_0 = 4$,当且仅当 $L_{p-2} \equiv 0(modM_p)$ 时,M_p 为质数。

下面对 Lucas-Lemher 所依赖的定理 4. 1 进行证明:

引理 4. 1 [22]:定义 $w = 2 + \sqrt{3}$,$\bar{w} = 2 - \sqrt{3}$;对 $\forall m \in N$,$L_m = w^{2^m} + \bar{w}^{2^m}$。

引理 4. 2 [22]:设 G 为有限群,则 $\forall a \in G$,有 $|a| \leqslant |G|$(其中 $|a|$ 代表群 a 中元素的个数)。

引理 4. 3 [23]:设 q 是素数,Z_q 为模 q 的剩余类加群,设 $X_q = \{a + b\sqrt{3}: a, b \in Z_q\}$,依自然方式定义 X_q 中的加法“+”和乘法“* ”,记 X_q^* 为 X_q 中对运算“* ”的可逆元之集,则(X_q^* ,*) 为交换群。

充分性:设 $M_P \mid L_{p-2}$,若 M_P 是合数,则可取其素因子 q' 满足 $q'^2 \leqslant M_P$。由引理 4. 1 知,存在 $l \in Z$,使 $w^{2^{p-2}} + \bar{w}^{2^{p-2}} l = lq'$。从而

$$w^{2^{p-1}} = lq' w^{2^{p-2}} - 1 \quad (1)$$

对(1) 两端平方,得

$$w^{2^p} = (lq' w^{2^{p-2}} - 1)^2 \quad (2)$$

由引理 4. 3,在环 $X_{q'}$ 中,式(1) ,(2) 化为 $w^{2^{p-1}} = -1$ 及 $w^{2^p} = 1$。于是,$w \in X_{q'}^*$,且 w 在群 $X_{q'}^*$ 中之阶 $|w| = 2^P - 2$,又易见 $|X_{q'}^*| \leqslant q^{12} - 1$,再由引理 4. 2,得 $z^p \leqslant q^{12} - 1$,但 $q^{12} - 1 \leqslant M_P - 1 = z^p - z$,矛盾,得证。

必要性: 记 $q = M_P$,则 q 为素数。又记 $\tau = \frac{1 + \sqrt{3}}{\sqrt{2}}$,则依引理 4. 3 中的记号,有 $\sqrt{2}\tau = 1 + \sqrt{3} \in X_q$。从而

$$2 \cdot 2^{\frac{q-1}{2}} \tau^{q+1} = (1 + \sqrt{3})^q (1 + \sqrt{3}) \in X_q \quad (3)$$

因 $q \equiv -1(mod8)$,由 Euler 准则及 Legendre 符号计算式,有 $2^{\frac{q-1}{2}} \equiv \left(\frac{2}{q}\right) = 1(\mathrm{mod}q)$。

又由 $q \equiv 1(mod3)$,可得 $3^{\frac{q-1}{2}} \equiv \left(\frac{3}{q}\right) = -1(\mathrm{mod}q)$。

从而在环 X_q 中,有 $2 \cdot 2^{\frac{q-1}{2}} = 2 \cdot \tau^{q+1} = 2 \cdot w^{\frac{q-1}{2}}$,及 $(1 + \sqrt{3})^q (1 + \sqrt{3}) = (1 + 3^{\frac{q-1}{2}}\sqrt{3})(1 + \sqrt{3}) = (1 - \sqrt{3})(1 + \sqrt{3}) = -2$。故在环 X_q 中,式(3) 化为 $2 \cdot w^{\frac{q-1}{2}} = -2$。

又易见 $2 \in X_q^*$,所以,在群 X_q^* 中,$w^{\frac{q-1}{2}} = -1$。由此易见,在环 X_q 中,有 $L_{p-2} = w^{2^{p-2}} + \bar{w}^{2^{p-2}} \equiv 0(\mathrm{mod}M_p)$,即 $M_P \mid L_{p-2}$,得证。

关于 Lucas-Lemher 素性检验算法的改进也经历了很长一段时间的研究,见表 4。1993 年,Bosma [19] 提出针对形式为 $h \cdot 2^n - 1$ 的素性检验算法,其中 h 要求不能整除 3。2004 年,Berrizbeitia 和 Berry [20] 同时提出针对形式为 $h \cdot 2^n \pm 1$ 的素性检验算法,其中 h 要求不能整除 5 [20]。2019 年黄丹丹和康云凌 [7] 总结了前人的理论,对形式为 $h \cdot 2^n \pm 1$ 的数的素性检验的条件进行了优化。

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

表 4　Lucas-Lemher 素性检验的不同改进版本

针对形式	要求	Lucas 序列种子
$h \cdot 2^n - 1$	h 不能整除 3	$-\left((2+\sqrt{3})^h + (2-\sqrt{3})^h\right)$
$h \cdot 2^n \pm 1$	h 不能整除 5	$(\alpha/\bar{\alpha})^h + (\bar{\alpha}/\alpha)^h$ $\alpha = -1 + 2i$
$h \cdot 2^n \pm 1$	$h < 2^n$	$(\alpha/\bar{\alpha})^h + (\bar{\alpha}/\alpha)^h$ $\left(\frac{M}{p}\right) = -1$ 且 $p = \alpha\bar{\alpha}$

针对 Lucas-Lemher 素性检验算法的改进主要体现在 Lucas 序列产生的种子上，即 Lucas 序列的首项。如在 Bosma [19] 的素性检验算法中，Lucas 序列首项为 $-\left((2+\sqrt{3})^h + (2-\sqrt{3})^h\right)$；在 Berrizbeitia 和 Berry [20] 的素性检验中，Lucas 序列的首项为 $(\alpha/\bar{\alpha})^h + (\bar{\alpha}/\alpha)^h$，$\alpha = -1 + 2i$；在黄丹丹 [7] 的素性检验中，α 的取值要依据具体的待检验数来决定。

5　三类素性检验算法的软件实现

Fermat 素性检验算法不同于以往的试除法，它首先提出从素数的性质去检验素性的思路，大大优化了素性检验流程，使得素性检验算法从呆板的、低效的转变为更具普适性的、高效的，可以称为概率型素性检验算法的鼻祖。Lucas 素性检验算法作为试除法后第一个确定型素性检验算法，对后续的确定型素性检验算法具有重要的参考意义。Miller-Rabin 素性检验算法是如今应用最广泛的素性检验算法。研究这三种具有代表性的素性检验算法，能够帮助我们更深地认识素性检验算法的本质。在本章，我们将对这三类算法进行软件实现，随后对它们的效率进行分析。

5.1　Fermat 素性检验的软件实现

Fermat 运用软件实现的具体流程是，对于待检验数 n，任意取整数进行 Fermat 检验，当取了 n-1 次，n 仍通过检验，那么则认为 n 很大概率是素数。

根据图 1，可以看出 Fermat 检验无法对卡麦克尔数进行检验，这是由卡麦克尔数的特殊性质决定的。图 2 展示了 Fermat 检验的核心思想。

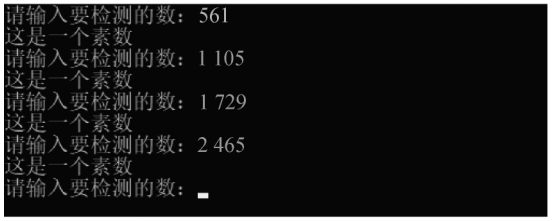

图 1　Fermat 检验卡麦克尔数的示例

```
int isprime(int n)// 费马检测
{
int i;
for(i=1; i<n; i++)//检验n-1次
{if(gcd(i,n)!=1)
    continue;
if(quickmod(i,n-1,n)!=1//快速幂
break;
}
if(i==n)
return 1;
else
return 0;
}
```

图 2　Fermat 检验部分代码截图

5.2　Lucas 素性检验的软件实现

Lucas 检验对基数 a 的要求是比较特殊的，如果待检验的 n 为素数，由于该算法的检验实际上是依靠原根的性质的，所以当选取的基数 a 不是循环群 Z_n^* 的原根的时，就需要重新选取 a 的值。图 5 是 Lucas 检验部分代码截图。

举个例子，如图 3 和图 4，当检验奇数 49417 时，只有当选择基底为 5 时才可以检验出其素性，而当检验卡麦克尔数 561 时，选遍所有可能的基底都不能通过 Lucas 算法表示其为一个素数，因此 561 是一个合数，很显然 561 作为一个

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

卡麦克尔数确实是合数。

```
请输入要检测的奇数n：49 417
当基底为2时：不能通过
当基底为3时：不能通过
当基底为4时：不能通过
当基底为5时：yes,n为素数
```

图 3 Lucas 检验通过的示例

```
请输入要检测的奇数n：561
当基底为2时：不能通过
当基底为3时：不能通过
当基底为4时：不能通过
当基底为5时：不能通过
当基底为6时：不能通过
当基底为7时：不能通过
当基底为556时：不能通过
当基底为557时：不能通过
当基底为558时：不能通过
当基底为559时：不能通过
当基底为560时：不能通过
所有基底都不通过，所以n为合数
```

图 4 Lucas 检验不通过的示例

5.3 Miller-Rabin 素性检验的软件实现

根据 Miller-Rabin 检验的流程，令 $n-1=2^f m$，其中 f 是非负整数，m 是正奇数。若 $b^m \equiv 1(modn)$ 或 $b^{2^f m} \equiv -1(modn)$，则称通过以 b 为基的 Miller-Rabin 检验。图 6 是 Miller-Rabin 检验部分代码截图。

5.4 三类算法软件性能比较分析

首先分析时间复杂度，从之前的代码截图中可以看出，Fermat 检验算法主要是快速幂算法的应用，常见的快速幂算法都是基于传统的二进制拆分思想，由于 n 的二进制位数为 $\lfloor \log_2 n \rfloor + 1$，所以单次利用快速幂算法的时间复杂度为 $O(\log n)$，在 Fermat 素性检验中重复 n-1 次，总的时间复杂度为 $O(n\log n)$。

Lucas 检验算法主要由三部分构成，首先是数的唯一分解，针对整数 n，注意到它的因子一半小于 $\sqrt{n}$，另一半大于 $\sqrt{n}$，所以只需要枚举其前一半的因子即可，时间复杂度为 $O(\sqrt{n})$。对一个基数 a，需要进行以下验证：伪素性检验，时间复杂度为 $O(\log n)$；将分解的因子作为指数进行伪素性检验，时间复杂度最大为 $O(\sqrt{n}\log n)$。基数 a 又有 n-2 种，所以总的时间复杂度为 $O(n^{\frac{3}{2}}\log n)$。

Miller-Rabin 检验算法的计算主要集中在刚选取基数 a 和 a 的模指数运算以及平方后模指数运算。最坏的情况即为所有可能的指数都计算了，所以该算法的时间复杂度为 $O(log^3 n)$。

```
int main()
{
    int prime[]={2,3,5,7,11,13,17,19,23,29,31,37,
    41,43,47,53,59,61,67,71,73,79,83,89,97,101,103,
    107,109,113,127,131,137,139,149,151,157,163,167,
    173,179,181,191,193,197,199,211,223,227,229,233,
    239,241,251,257,263,269,271,277,281,283,293,307,
    311,313,317,331,337,347,349,353,359,367,373,379,
    383,389,397,401,409,419,421,431,433,439,443,449,
    457,461,463,467,479,487,491,499,503,509,521,523};
    int n;
    int num[100]={0};
    int i=0,j=0;
    int a=2;
    printf("请输入要检测的奇数n: ");
    scanf("%d",&n);
    int m=n;
    n--;
    while(n!=1)
    {

        if(n%prime[j]==0)
        {
            num[i++]=prime[j];
            j++;

        while(n%prime[j-1]==0)
            n/=prime[j-1];
        }
        else j++;
    }
    int flag=0;
    while(a<m)
    {
        printf("当基底为%d时: ",a);
        if(fastPower(a,m-1,m)!=1)
        {
            printf("不能通过\n");
            a++;
            continue;
        }
        else
        {
            int k;
        for(k=0;k<i;k++)
        {
            if(fastPower(a,(m-1)/num[k],m)==1)
            {
                printf("不能通过\n");
                a++;
                break;
            }
        }
        if(k==i)
           {printf("yes,n为素数\n");
        break;}
        }
    }
    if(a==m)
        printf("所有基底都不通过，所以n为合数");
    return 0;
}
```

图 5 Lucas 检验部分代码截图

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

```
int Miller_Rabbin(int n,int a)// 米勒拉宾测试
{
int r = 0,s = n-1,j;
long long k;
if(n%a == 0)
return 0;
while((s&1)== 0)
{
s >>= 1;
r++;
}
k = quickmod(a,s,n);
if(k == 1)
return 1;
for(j = 0; j < r; j++,k=k*k%n)
if(k == n-1)
return 1;
return 0;
}
int Isprime(int n)// 判断是否是素数
{
int i;
for(i=0;i<4;i++)
{
if(n == tab[i])
return 1;
if(!Miller_Rabbin(n,tab[i]))
return 0;
}
return 1;
}
```

图 6　Miller-Rabin 检验部分代码截图

表 5　三类算法的时间复杂度

算法名称	时间复杂度	算法类别
Fermat 检验	$O(n\log n)$	概率型
Lucas 检验	$O(n^{\frac{3}{2}}\log n)$	确定型
Miller-Rabin 检验	$O(log^3 n)$	概率型

根据表 5，可以看出当对一个大数进行素性检验时，运行效率上，Miller-Rabin 检验> Fermat 检验> Lucas 检验。

下面对这三个算法的容错率进行分析:

运用 Fermat 检验来分别检验 100、10000、1000000、100000000 以内的数据，得到如下数据。

表 6　Fermat 素性检验结果统计

检验数据	成功报合%	错误报素%	成功报素%
100	75	0	25
10000	87. 64	0. 07	12. 29
1000000	92. 1459	0. 0044	7. 8503
100000000	94. 238088	0. 000255	5. 761657

根据表 6，当检验 10^8 以内的数据时，有 255 个数导致 Fermat 检验算法错误报素，这 255 个数便是上文中提到的卡麦克尔数。

由于 Lucas 检验算法是确定型素性检验算法，所以其容错率视为零。

Miller-Rabin 检验算法的运算主要是模指数运算，它的时间复杂度达到了 $O(log^3 n)$。上文中也提到其正确率至少为 $1-\frac{1}{4^t}$，实际上，一般情况下 $\frac{1}{4^t}$ 这个理论错误上界是个很难达到的界。分别用基底 a = 2 和 a = 3 进行数据检验，可以得到如下结果。

表 7　检验 a = 2 时 Miller-Rabin 检验结果统计

检验数据	成功报合%	错误报素%	成功报素%
100	74	0	26
10000	87. 66	0. 05	12. 29
1000000	92. 1456	0. 0046	7. 8498
100000000	94. 238057	0. 000488	5. 761455

表 8　检验两轮时 Miller-Rabin 检验结果统计

检验数据	成功报合%	错误报素%	成功报素%
100	74	0	26
10000	87. 71	0	12. 29
1000000	92. 1502	0	7. 8498
100000000	94. 238524	0. 000021	5. 761455

根据表 7 和表 8，可以看出，在只选择一种基底 a 去判断时，它容错率是相对大于 Fermat 素性检验的; 当选择两种不同的基底 a = 2 和 a = 3 判断两轮时，对于小于10^6 的 n 都不会报错，对于小于 10^8 的数的容错率也是很低，远低于 Fermat 素性检验。

显而易见，当检验更大的数时，容错率上，Lucas 检验>多轮 Miller-Rabin 检验> Fermat 检验>单轮 Miller-Rabin 检验。

综合运行效率和容错率两个方面，Fermat 素性检验的时间复杂度适中，其容错率也适中; Lucas 素性检验的时间复杂度最高，但是其准确度也是最高; Miller-Rabin 素性检验的综合性能最好，在多次检验提高容错率的前提下，还能保证拥有一个乐观的运行效率。

6　结论

素性检验算法作为密码学中一个重要基础，

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

它的研究具有非常繁荣的前景。本文深入分析了几种常见的素性检验算法，首先，对常见的素性检验算法的核心思想与算法流程进行完整的描述，对一些定理也进行了相关证明；其次，将它们按照不同标准进行分类和比较分析；最后，通过对具有代表性的三种素性检验算法进行实验比较分析，得出 Miller-Rabin 素性检验算法的综合性能最好。

实际上，Miller-Rabin 素性检验算法是目前最适合被用来构建密码安全体系的素数生成模块的算法，有关其误判概率修正的研究也越来越多。随着现代计算机的发展，计算速度不断突破瓶颈，越来越多的计算要求较高的问题逐渐被解决，AKS 作为确定型素性检验算法的代表，如果在运行速度上能够追赶上概率型算法，且能够进一步将其应用到密码应用系统的话，那么对于现代加密系统可靠性的发展将会是一个质的飞跃。同时，基于椭圆曲线的素性检验算法也正在迅速发展，相信在不远的将来也会进一步走上素性检验研究的舞台。

参考文献

[1] M，C，Wunderlich. A performance analysis of a simple prime-testing algorithm [J]. Math Comp，1983.

[2] Solovay R，Strassen V. A Fast Monte-Carlo Test for Primality [J]. Siam Journal on Computing，1977，6(1)：84-85.

[3] Michael O Rabin，Probabilistic algorithm for testing primality. Journal of Number Theory 12，(1980)，no.1，128-138.

[4] Moshonkin A G， Khamitov I M. A New Probabilistic Primality Test [J]. Journal of Mathematical Sciences，2020，(249)：2-4.

[5] Pocklington H C. The determination of prime of composite nature of large numbers by fermat's theorem [J]. Proc Cambridge Philo s Soc，1914(18)：29-30.

[6] Manindra Agrwal and Neeraj Kayal and Nitin Saxena，PRIMES is in P，Annals of Mathematics 160(2004)，no.2，781-793.

[7] Dandan Huang，Yunling Kang. Primality Testing for Numbers of the Form h 2^n ± 1 [J]. Journal of Systems Science and Complexity，2019，32(5).

[8] 潘承洞，潘承彪. 初等数论 [M]. 北京：北京大学出版社，1994.

[9] Ishmukhametov S T，Mubarakov B G，Rubtsova R G. On the Number of Witnesses in the Miller-Rabin Primality Test [J]. Symmetry，2020，12(6)：890.

[10] Plaksin V A. A Statistical Property of the Sieve of Eratosthenes [J]. Theory of Probability and Its Applications，1991，36(3)：587-593.

[11] Pratt，Vaughan R. Every Prime Has a Succinct Certificate [J]. Siam Journal on Computing，1975，4(3)：214-220.

[12] Gary L.Miller. Riemann's hypothesis and tests for primality [J]. Journal of Computer and System Sciences，1976，13(3)：300-317.

[13] Lehmer D H. An extended theory of Lucas' functions [J]. Annals of Mathematics，1930，31(3)：419-448.

[14] C. Pomerance，J. L. Selfridge and Samuel S. Wagstaff，Jr.，The pseudoprimes to 25 10^9，Math. Comp. 35 (1980)，1003 - 1026. MR 0572872 (82g：10030).

[15] D. J. Bernstein，Proving Primality after Agrawal-Kayal-Saxena，http://cr. yp. to/papers/aks.ps，January 2003.

[16] Berrizbeitia P. Sharpening "Primes Is in P" for a Large Family of Numbers [J]. Mathematics of Computation，2005，74(252)：2043-2059.

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

[17] 陈燕. 基于椭圆曲线的素性检验研究 [D]. 昆明: 云南大学, 2008.

[18] 郭常超, 唐仙芝. Lucas-Lehmer 检验法的一个证明 [J]. 河南大学学报(自然科学版), 2005(3): 14-15.

[19] Wieb, Bosma. Explicit primality criteria for h · 2^k ± 1 [J]. Mathematics of Computation, 1993, 61(203).

[20] Berrizbeitia P, Berry T G. Biquadratic reciprocity and a Lucasian primality test [J]. Mathematics of Computation, 2004, 73(247): 1559-1564.

[21] Lucas, Edouard.Theorie des Fonctions Numeriques Simplement Periodiques [J]. Amer. J. Math., 1878, 2: 184-196.

[22] J. W. Bruce(1993). A Really Trivial Proof of the Lucas-Lehmer Test. The American Mathematical Monthly. 100(4): 370-371.

[23] J Rödseth. A note on primality tests for $N = h \cdot 2^n - 1$ [J]. Bit Numerical Mathematics, 1994, 34(3): 451-454.

Comparative Analysis of Common Primality Testing Algorithms

XU Bin　ZHANG Yanshuo　LV Zhenghong

Beijing Electronic Science and Technology Institute, Beijing 100070, P.R.China

Abstract: In the modern cryptosystem, large prime number is essential to establishing some encryption systems, e.g. the RSA cryptosystem and the ECC cryptosystem. As the two most popular and potential cryptosystems, the RSA and the ECC build the security based on the large prime number. Testing the large prime number is particularly important and common primality testing algorithms include the Fermat, the Solovay Strassen, the Miller Rabin, the Pocklington, the Lucas Lehmer, the Pepin, the Lucas, the AKS, etc. A primality testing algorithm could be classified as a general form or a special form primality testing algorithm according to the form of the tested number, or as a probabilistic or a deterministic primality testing algorithm according to the accuracy of the test result. In this paper, abovementioned algorithms are introduced, and compared and analyzed from the aspects of classification and software implementation. Theoretical analysis indicates that the Miller Rabin primality testing algorithm has the highest comprehensive efficiency.

Keywords: primality testing; Miller Rabin primality testing; pseudo prime; comparative analysis

(责任编辑: 夏　超)

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net