PRIMALITY PROVING WITH CYCLOTOMY

WIEB BOSMA

MARC-PAUL VAN DER HULST

STELLINGEN

Wieb Bosma

een

Zij $n \in \mathbf{Z}_{\geq 2}$ een samengesteld getal, en zij $n-1 = r2^k$, met r oneven en $k \geq 0$. Noem een positief geheel getal a getuige voor het samengesteld zijn van n, als a niet deelbaar is door n en zowel $a^r \not\equiv 1 \bmod n$ als $a^{r2^i} \not\equiv -1 \bmod n$ voor $i = 0, 1, \ldots, k-1$. Het aantal elementen van $\{1, 2, \ldots, n-1\}$ dat getuige is voor het samengesteld zijn van n is groter dan of gelijk aan $\frac{3}{4}n - 1$, en gelijkheid geldt dan en slechts dan als $n = 4$.

twee

Er bestaat een kollektie eenheidswortels die over $\mathbf{Q}$ lineair onafhankelijk is en een geheelheidsbasis omvat voor elk cyclotomisch lichaam.

drie

Voor elke eindige verzameling $\mathcal{D}$ van positieve gehele getallen bestaat er een positief geheel getal k zodanig dat voor alle $D \in \mathcal{D}$ het Jacobisymbool $\left(\frac{D}{3 \cdot 2^k - 1}\right)$ gelijk aan 1 is.

vier

Zij $h \in \mathbf{Z}_{\geq 1}$ oneven. Definieer $n_k = h \cdot 2^k - 1$, voor alle $k \in \mathbf{Z}_{\geq 1}$. Laat het startwaardeprobleem *voor h het probleem zijn een eindige kollektie K van drietallen gehele getallen (a, b, D) met $D \neq 0 \neq a^2 - b^2 D$ te vinden zodat voor elke k waarvoor $2^k > h$ en $n_k > 1$ is, er een drietal $(a, b, D) \in K$ bestaat met de eigenschap dat*

$$\left(\frac{D}{n_k}\right) \neq 1 \neq \left(\frac{a^2 - b^2 D}{n_k}\right).$$

Het startwaardeprobleem is onoplosbaar voor $h = 4^m - 1$, met $m \in \mathbf{Z}_{\geq 1}$. Als $h \leq 2^{10}$ en $h \notin S = \{3, 15, 63, 255, 1023\}$, dan is het startwaardeprobleem voor h oplosbaar. Als $h \in S$, dan bestaat er voor elke k waarvoor $2^k > h$ is, een $D_k \in \mathbf{Z}_{\geq 1}$ zodat

$$\left(\frac{D_k}{n_k}\right) \neq 1 \neq \left(\frac{1 - D_k}{n_k}\right).$$

H. Riesel, *Prime numbers and computer methods for factorization*, Boston: Birkhäuser, Progr. Math. **57**, 1985, pp. 132–137.
W. Borho, *Große Primzahlen und befreundete Zahlen: über den Lucas-test und Thabit-regeln*, Mitt. Math. Ges. Hamburg **11** (1983), 232–256.

De beide volgende stellingen betreffen halfregelmatige kettingbreukontwikkelingen.
Zij x een irrationaal reëel getal. Een *halfregelmatige* kettingbreukontwikkeling kan worden gedefinieerd als een rij $\epsilon_0, \epsilon_1, \ldots$ van getallen $\epsilon_i \in \{-1, +1\}$. De *wijzergetallen* van x met betrekking tot deze ontwikkeling zijn de niet-negatieve gehele getallen $b_0, b_1, \ldots$, bepaald door de konditie $0 < \epsilon_i(x_i - b_i) < 1$, waar $x_0 = x$ en $x_i = \epsilon_{i-1}/(x_{i-1} - b_{i-1})$ voor $i \geq 1$. De *konvergenten* met betrekking tot deze ontwikkeling zijn de rationale getallen p_i/q_i die voor $i \geq 0$ worden gegeven door:

$$p_{-2} = 0,\ p_{-1} = 1,\ p_i = b_i p_{i-1} + \epsilon_i p_{i-2} \quad \text{en} \quad q_{-2} = 1,\ q_{-1} = 0,\ q_i = b_i q_{i-1} + \epsilon_i q_{i-2}.$$

Er geldt dat $\lim p_i/q_i \to x$ voor $i \to \infty$. De *regelmatige* kettingbreukontwikkeling van x is de halfregelmatige met $1 = \epsilon_0 = \epsilon_1 = \ldots$, en de *ontwikkeling naar dichtstbijzijnde gehele* is degene waarvoor $|x_i - b_i| < \frac{1}{2}$; van de laatste geven we de konvergenten aan met R_i/S_i. Een halfregelmatige ontwikkeling heet *snelst mogelijk* indien $q_i \geq S_i$ voor oneindig veel $i \geq 0$.

vijf

Voor elke irrationale x bestaat er een unieke halfregelmatige kettingbreukontwikkeling, welke we de optimale ontwikkeling noemen en waarvan we de konvergenten met r_i/s_i aangeven, met de volgende eigenschap. Voor elke halfregelmatige ontwikkeling van x, met konvergenten p_i/q_i, en voor elke $k \geq 1$ geldt:

$$\frac{1}{n-1}\sum_{i=1}^{n-1} q_i^2\left|x - \frac{p_i}{q_i}\right| \geq \frac{1}{k-1}\sum_{i=1}^{k-1} s_i^2\left|x - \frac{r_i}{s_i}\right|$$

waar n zodanig is dat $q_{n-1} < s_k \leq q_n$. De optimale ontwikkeling is snelst mogelijk, en wordt volledig bepaald door de konditie dat $\epsilon_{i+1} = -1$ dan en slechts dan als met $b = \lfloor x_i \rfloor$:

$$((b+1)q_{i-1} + \epsilon_i q_{i-2})^2\left|x - \frac{(b+1)p_{i-1} + \epsilon_i p_{i-2}}{(b+1)q_{i-1} + \epsilon_i q_{i-2}}\right| < (bq_{i-1} + \epsilon_i q_{i-2})^2\left|x - \frac{bp_{i-1} + \epsilon_i p_{i-2}}{bq_{i-1} + \epsilon_i q_{i-2}}\right|.$$

W. Bosma, *Optimal continued fractions*, Indag. Math. **49** (1987), 353–379.
W. Bosma, C. Kraaikamp, *Optimal approximation by continued fractions*, J. Australian Math. Soc. (1990), to appear.

zes

De enige gehele getallen $m \in \mathbf{Z}_{\geq 2}$ waarvoor $\sqrt{m}$ een halfregelmatige ontwikkeling van periode een of twee toelaat, zijn $m = a^2 \pm b$, met $b \mid 2a$. Voor zulke m geldt dat:

$$\sqrt{m} = \sqrt{a^2+b} = [a; \overline{\frac{2a}{b}, 2a}] \quad \textit{of} \quad \sqrt{m} = \sqrt{a^2-b} = [a; \overline{\frac{-2a}{b}, -2a}],$$

en dit zijn snelst mogelijke kettingbreukontwikkelingen. Bovendien is dit altijd de optimale ontwikkeling, met dien verstande dat in het speciale geval $b = a$ geldt:

$$\sqrt{a^2+a} = [a; \overline{2, 2a}] = [a+1; \overline{-2, -2a}] = \sqrt{(a+1)^2 - (a+1)},$$

en de eerste is de optimale ontwikkeling.

zeven

De enige waarden voor $k \leq 320$ waarvoor $2 \cdot (5+2i)^k + 1$ priem is in de ring van gehelen van Gauss, zijn 0, 1, 2, 4, 7, 21, 28, 70, 71, 84, 163 *en* 263.

W. Bosma, *Primality testing with elliptic curves*, Report **85–12** (1985), Mathematisch Instituut, Universiteit van Amsterdam.

acht

Laat, over een lichaam K van karakteristiek ongelijk 2 en 3, de elliptische kromme E gedefinieerd zijn door de homogene Weierstraßvergelijking $y^2z = x^3 + axz^2 + bz^3$, met als basispunt $(0:1:0)$. Het volgende tweetal tripels formules voor het bepalen van de som $(x_3 : y_3 : z_3)$ van twee punten $(x_1 : y_1 : z_1)$ en $(x_2 : y_2 : z_2)$ op E vormt een volledig stelsel van optelwetten:

$$
\begin{aligned}
x_3 &= (x_1y_2 + x_2y_1)(y_1z_2 - y_2z_1) + a(x_1z_2 + x_2z_1)(x_1z_2 - x_2z_1) + \\
&\quad - (y_1y_2 - 3bz_1z_2)(x_1z_2 - x_2z_1) \\
y_3 &= 2\Big((y_1y_2 - 3bz_1z_2)(y_1z_2 - y_2z_1) - a(x_1z_2 + x_2z_1)(y_1z_2 - y_2z_1) + \\
&\quad + (3x_1x_2 + az_1z_2)(x_1y_2 - x_2y_1)\Big) \\
z_3 &= (y_1z_2 + y_2z_1)(y_1z_2 - y_2z_1) + (3x_1x_2 + az_1z_2)(x_1z_2 - x_2z_1)
\end{aligned}
$$

$$
\begin{aligned}
x_3 &= (y_1y_2 - 3bz_1z_2)(x_1y_2 + x_2y_1) - 3b(x_1z_2 + x_2z_1)(y_1z_2 + y_2z_1) + \\
&\quad + a(az_1z_2 - x_1x_2)(y_1z_2 + y_2z_1) - a(x_1y_2 + x_2y_1)(x_1z_2 + x_2z_1) \\
y_3 &= 2\Big((y_1y_2 - 3bz_1z_2)(y_1y_2 + 3bz_1z_2) + a(3x_1x_2 + az_1z_2)(x_1x_2 - az_1z_2) + \\
&\quad - a^2(x_1z_2 + x_2z_1)^2 + 3b(3x_1x_2 - az_1z_2)(x_1z_2 + x_2z_1)\Big) \\
z_3 &= (y_1y_2 + 3bz_1z_2)(y_1z_2 + y_2z_1) + (3x_1x_2 + az_1z_2)(x_1y_2 + x_2y_1) + \\
&\quad + a(x_1z_2 + x_2z_1)(y_1z_2 + y_2z_1).
\end{aligned}
$$

H. Lange, W. Ruppert, *Complete systems of addition laws on abelian varieties*, Invent. Math. **79** (1985), 603–610.

negen

Externe druk en de interne neiging haarkloverij tot buiten het eigen vakgebied voort te zetten, doen een promovendus er maar al te vaak toe besluiten een akelig betweterige laatste stelling te produceren.

STELLINGEN

van Marc-Paul van der Hulst

behorende bij het proefschrift "Primality proving with cyclotomy"

van Wieb Bosma en Marc-Paul van der Hulst

1. *Laat de getuigetest voor een positief getal n gedefinieerd zijn als de methode die controleert of er geen getal bestaat in $\{1, 2, 3, \ldots, \lfloor 2 \cdot \log_e^2(n) \rfloor\}$ dat getuige is voor het samengesteld zijn van n.*

   Als de gegeneraliseerde Riemann-hypothese waar is, is de getuigetest een polynomiale primaliteitstest. Pas als n meer dan 1950 decimale cijfers heeft kan de getuigetest sneller zijn dan het primaliteit testen met behulp van cyclotomie.

   E. Bach, *Explicit bounds for primality testing and related problems*, Math. Comp. **55** (1990), 355–380.
   G. L. Miller, *Riemann's hypothesis and tests for primality*, J. Comput. System Sci. **13** (1976), 300–317.
   M. O. Rabin, *Probabilistic algorithm for testing primality*, J. Number Theory **12** (1980), 128–138.
   W. Bosma, M. P. M. van der Hulst, dit proefschrift, I.8, V.(2.2).

2. *Zij $t_0 = 6983776800$. Er zijn 618 priemen q zodanig dat $q-1 | t_0$, waarvan de grootste gelijk is aan $q_{618} = 1745944201$. Het grootste priemgetal dat $q_{618} - 1$ deelt is gelijk aan $p = 19$. Voor een character χ van orde $p = 19$ en conductor gelijk aan q_{618}, kunnen de quotiënten van Gauss-sommen $\tau(\chi)^i/\tau(\chi^i)$ voor $0 < i < p$ en de Gauss-som $\tau(\chi)^p$ worden uitgedrukt in de Jacobi-sommen $J(\chi, \chi)$, $J(\chi, \chi^2)$ en $J(\chi^2, \chi^3)$. Indien $\zeta_{19} = \chi(23)$ dan zijn deze Jacobi-sommen gelijk aan*

$$\begin{aligned}
J(\chi,\chi) = &-23252\cdot\zeta_{19}^{17} + 14\cdot\zeta_{19}^{16} - 16764\cdot\zeta_{19}^{15} - 20606\cdot\zeta_{19}^{14} - 4842\cdot\zeta_{19}^{13} - 15192\cdot\zeta_{19}^{12}\\
&- 1222\cdot\zeta_{19}^{11} - 15844\cdot\zeta_{19}^{10} - 1676\cdot\zeta_{19}^{9} - 17230\cdot\zeta_{19}^{8} - 19352\cdot\zeta_{19}^{7} - 20074\cdot\zeta_{19}^{6}\\
&- 2328\cdot\zeta_{19}^{5} + 2449\cdot\zeta_{19}^{4} + 4548\cdot\zeta_{19}^{3} - 11734\cdot\zeta_{19}^{2} - 10704\cdot\zeta_{19} + 5506,\\
J(\chi,\chi^2) = &-16357\cdot\zeta_{19}^{17} - 26771\cdot\zeta_{19}^{16} - 23056\cdot\zeta_{19}^{15} - 17627\cdot\zeta_{19}^{14} - 13765\cdot\zeta_{19}^{13}\\
&- 29563\cdot\zeta_{19}^{12} - 9324\cdot\zeta_{19}^{11} - 10263\cdot\zeta_{19}^{10} - 18731\cdot\zeta_{19}^{9} - 40374\cdot\zeta_{19}^{8}\\
&- 22705\cdot\zeta_{19}^{7} - 26798\cdot\zeta_{19}^{6} - 23958\cdot\zeta_{19}^{5} - 17903\cdot\zeta_{19}^{4} - 28794\cdot\zeta_{19}^{3}\\
&- 27622\cdot\zeta_{19}^{2} - 5808\cdot\zeta_{19} - 23774 \quad \textit{en}\\
J(\chi^2,\chi^3) = &\ 23159\cdot\zeta_{19}^{17} + 23159\cdot\zeta_{19}^{16} + 4692\cdot\zeta_{19}^{15} + 5258\cdot\zeta_{19}^{14} + 4692\cdot\zeta_{19}^{13} + 19200\cdot\zeta_{19}^{11}\\
&+ 4692\cdot\zeta_{19}^{10} + 23545\cdot\zeta_{19}^{9} + 19200\cdot\zeta_{19}^{7} + 23545\cdot\zeta_{19}^{6}\\
&+ 23159\cdot\zeta_{19}^{5} + 23545\cdot\zeta_{19}^{4} + 5258\cdot\zeta_{19}^{3} + 5258\cdot\zeta_{19}^{2} + 19200\cdot\zeta_{19} + 10602.
\end{aligned}$$

   W. Bosma, M. P. M. van der Hulst, dit proefschrift, IV.(2.5), V.(3.6), Tables.

3. *Voor gehele getallen $n \geq 0$ is M_n gedefinieerd door $M_n = 2^n - 1$. De grootste k waarvoor de factorisatie van M_{2^k} volledig bekend is, heeft de waarde* 10.

$M_{1024} = 3 \cdot 5 \cdot 17 \cdot 257 \cdot 641 \cdot 65537 \cdot 274177 \cdot 6700417 \cdot 2424833 \cdot 67280421310721 \cdot 1238926361552897 \cdot 59649589127497217 \cdot 5704689200685129054721 \cdot 7455602825647884208337395736200454918783366342657 \cdot 934616397153579777691635581996068965840512375416381885802803 21 \cdot 7416400626275308015247871419019374740599407810975190239058213161444157595047050080928187116939407 37.$

J. Brillhart, D. H. Lehmer, J. L. Selfridge, B. Tuckerman, S. S. Wagstaff, Jr., *Factorizations of $b^n \pm 1$, $n = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers*, Providence: A.M.S., Contemp. Math. **22** (1988).
A. K. Lenstra, M. S. Manasse, rn sci.crypt message 550 (15 Juni, 1990).

4. *Voor een rationaal getal r met $0 \leq r < 1$ en een geheel getal $b \geq 2$ zij $v(r, b)$ de oneindige vector die gevormd wordt door de cijfers van r geschreven in basis b; bijvoorbeeld $v(\frac{1}{3}, 10) = (3, 3, 3, \ldots)$. Dan geldt voor iedere r dat de vectorruimte opgespannen door $\{v(r, b) : b \geq 2\}$ eindigdimensionaal is.*

5. *Zij f een polynoom over* $\mathbf{Q}$ *in k variabelen van graad ten hoogste m in elk der variabelen. Factorisatie van f over* $\mathbf{Q}$ *in irreducibele polynomen kan in tijd polynomiaal in de lengte van de invoer worden uitgevoerd, door voor $k - 1$ variabelen een transcendent getal of een algebraïsch getal van voldoend hoge graad te substitueren, en vervolgens het resulterende polynoom in één variabele te factoriseren door middel van rooster-reductie.*

M. P. van der Hulst, A. K. Lenstra, *Factorization of polynomials by transcendental evaluation*, Proceedings Eurocal (1985), Lecture Notes in Comput. Sci. **204**, Springer-Verlag.
R. Kannan, A. K. Lenstra, L. Lovász, *Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers*, Math. Comp. **50** (1988), 235-250, [eveneens in: Proceedings 16th STOC (1984)].

6. *Voor gehele getallen $n \geq 0$ is het n-de Fermat-getal F_n gedefinieerd door $F_n = 2^{2^n} + 1$. De complexiteit van primaliteit testen van Fermat-getallen F_n door middel van Pepin's test is dubbel exponentieel in de invoer.*

7. *Om snelle vermenigvuldigingsformules, zoals gebruikt in de oude Jacobi-som test, ook in de nieuwe test te gebruiken is het voldoende om alleen formules uit te werken voor de vermenigvuldiging modulo polynomen van priemmacht-graad.*

W. Bosma, M. P. M. van der Hulst, dit proefschrift, II.(4.11), IV.(2.3).
H. Cohen, A. K. Lenstra, *Implementation of a new primality test*, Math. Comp. **48** (1987), 103-121.
A. Karatsuba, Yu. Ofman, **Умножение многозначных чисел на автоматах**, Dokl. Akad. Nauk SSSR **145** (1962), 293-294 [Engelse vertaling: *Multiplication of multidigit numbers on automata*, Soviet Phys. Dokl. **7** (1963), 595-596].

8. *Gebruikers van* LaTeX *zijn meegaande types.*

PRIMALITY PROVING WITH CYCLOTOMY

PRIMALITY PROVING WITH CYCLOTOMY

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof. dr. S. K. Thoden van Velzen
in het openbaar te verdedigen in de Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui),
op donderdag 20 december 1990
te 16.30 uur en te 17.30 uur

door

WIEBREN BOSMA
geboren te Den Burg (Texel)

en

MARCUS PAULUS MARIA VAN DER HULST
geboren te Amsterdam

promotores:	prof. dr. H. W. Lenstra, Jr.
	dr. P. van Emde Boas
copromotor:	dr. A. K. Lenstra
promotiecommissie:	prof. dr. G. B. M. van der Geer
	prof. dr. H. Jager
	dr. R. J. Schoof
	prof. dr. P. M. B. Vitányi

Faculteit Wiskunde en Informatica

Mais ne trouvez vous pas que c'est presque faire trop d'honneur aux nombres premiers que d'y répandre tant de richesses, et ne doit-on aucun égard au goût raffiné de notre siècle?

Daniel Bernoulli, Letter to L. Euler, 18 March 1778.

CONTENTS.

PREFACE.

This thesis consists of seven chapters, and five appendices.

In the first chapter the history of the primality testing problem is outlined. The second chapter comprises a mathematical description of the primality test that is the main subject of this thesis. In particular, it is proved in this chapter that the conditions that an integer n must satisfy in order to pass the test are sufficient to *prove* the primality of n. In the third chapter some problems (and their solutions) are discussed that regard the optimal choice of the parameters in the test. A detailed description of the primality test is given in the fourth chapter. In the fifth chapter the complexity bounds of the algorithm are given and some heuristics necessary to obtain these bounds are presented. In the sixth chapter an overview is given of the performance of the primality test, and the seventh chapter is intended to be a guideline for those people interested in installing and using the computer program that accompanies this thesis. Finally, the appendices consist of some tables that are part of the computer program, a bibliography, a list of symbols, an index, and a Dutch summary.

The primality testing algorithm, as described in Chapter IV, is a modified version of the so-called Jacobi sum test (cf. [29], [30]). The main theoretical improvements are the following. First of all, it turned out to be possible, and very fruitful, to combine the Jacobi sum test with Lucas-Lehmer type tests (which are classically used for primes of a special form), by putting everything in the same mathematical framework; see Sections II.5 and II.7. The second improvement makes the new algorithm faster: a Jacobi sum test, which consists of the verification of an identity (involving Jacobi sums), will in general be done in a smaller ring. The construction of the rings necessary for this is explained in Section II.4. Thirdly, it has been possible to reduce the amount of work involved, by doing several of such Jacobi sum tests simultaneously; this is explained in Section II.8. To find good combinations of tests, and to determine good values for all parameters in the improved primality test, an intricate optimization stage had to be built into the algorithm (see Chapter III).

The effect of these changes on the performance of the algorithm (both in a theoretical and a practical sense), is explained in Chapters V and VI. An important conclusion is, that in practice the improved algorithm performs better than any other general purpose primality proving algorithm that is currently known, in two respects: it is faster, and it is capable of coping with larger primes.

This thesis is the product of joint work. Wieb Bosma is primarily responsible for the first three chapters, containing a description of the theoretical aspects of the subject, while the last three chapters, devoted to the algorithmical aspects, are mainly the responsibility of Marc-Paul van der Hulst. Chapter IV combines both theoretical and algorithmical aspects and the responsibility is therefore shared by the authors.

Most of the research for this thesis has been done between June 1985 and July 1989 at the Faculteit Wiskunde en Informatica of the Universiteit van Amsterdam.

Part of it has been carried out while the authors visited the University of California at Berkeley[1], the Department of Computer Science of the University of Chicago, and while Marc-Paul van der Hulst visited Bell Communications Research in Morristown, New Jersey.

The authors thank all of the above institutions for their hospitality, and for providing the facilities to generate and test the computer programs.

This thesis could not have been written without the support of many. Here, the authors take the opportunity to express their gratitude to the following persons for contributions to their scientific well-being.

We owe many thanks to Peter van Emde Boas for his continuing support and encouragement. We wish to convey special thanks to Arjen Lenstra, for his contributions, his energy, and his inspiration. Finally, it is a pleasure to express very special thanks to Hendrik Lenstra, for pointing it all out with endless patience and good humour.

October 1990,

Wieb Bosma[2]
Department of Pure Mathematics
University of Sydney
Sydney

Marc-Paul van der Hulst[3]
Faculteit Wiskunde en Informatica
Universiteit van Amsterdam
Amsterdam

---

[1] Made possible by teaching assistantships at the Department of Mathematics and a research assistantship at the Department of Electrical Engineering and Computer Science (supported by the National Science Foundation under Grant No. DMS-8706176).

[2] Supported by the Nederlandse organisatie voor wetenschappelijk onderzoek NWO (formerly ZWO) via the Stichting Mathematisch Centrum SMC.

[3] Supported by the Nederlandse organisatie voor wetenschappelijk onderzoek NWO (formerly ZWO) via the Stichting Informatica Onderzoek in Nederland SION.

I. HISTORY.

1. TERMINOLOGY.

The fundamental theorem of arithmetic states that every positive integer n has a unique prime factor decomposition:

$$n = \prod_{p \text{ prime}} p^{k(p)}.$$

This thesis is concerned with a problem that arises when one tries to find prime factor decompositions. In finding such factorizations, three steps can be recognized; these steps are applied recursively if necessary. Loosely speaking they are the following.

(i) Find out whether n is prime or composite.

(ii) If n is prime, prove its primality.

(iii) If n is composite, find n_1 and n_2 in $\mathbf{Z}_{\geq 2}$ such that $n = n_1 n_2$.

The second step, called primality testing, covers the field we will concern ourselves with. In this first section, we will present motivation for interest in the problem, and we will introduce the basic terminology. The rest of this chapter describes some of the historic attempts to conquer the problem, in particular in relation to the algorithm presented in the next chapters.

A *primality testing algorithm*, or *primality test* for short, is an algorithm that, on input a prime number n, outputs a proof for the primality of n; if the input n is a composite number however, the algorithm need not terminate, but if it does, a proof for the compositeness of n is supplied. Thus, as was suggested in the formulation of step (ii) above, a primality test is a primality prover.

This raises several questions; first of all: what is a primality proof? This is closely related to the question: what is an algorithm? We do not want to go into (interesting but distracting) details here, but content ourselves with the following. An algorithm consists, for our purposes, of a set of instructions for constructing certain objects and verifying certain identities between these; in this way, a primality test of n should be thought of as a series of operations, depending on n, showing that certain conditions ensuring the primality of n are satisfied. The correctness of the algorithm and the sufficiency of the conditions form the contents of a theorem, which requires a proof; the algorithm is supposed to be merely a constructive way of checking the conditions.

Of course this ignores practical difficulties, but the point we are making, is that a primality test is ultimately a *theorem*, and that proof of correctness lies in mathematical

rigour. Still, one would like to be able to *verify* primality proofs, preferably without carrying out all the (possibly cumbersome) steps of the algorithm again. Later on such easily verifiable *certificates* for primality will be discussed. In general, algorithms do not provide short certificates; for example, a trial division certificate could consist of all (non-zero) residues $n \bmod p$, for the primes p up to $\sqrt{n}$. Checking these residues comes down to doing trial division again; assuming that these are correct, primality is proven if all residues are non-zero. (The trial division algorithm will be discussed shortly.)

A question that is of greater relevance to our subject is, why steps (ii) and (iii) are put into our description above at all. For, in step (i) we have already found out whether n was prime or not – does this mean that we "know" whether n is prime without a proof, or that we "know" that n is composite without having a non-trivial factor? Indeed it does, and this comes as a surprise on seeing it for the first time. The reason is, that there exist very fast *compositeness tests* (we will describe them later on), that on input an integer n, output one of two possible answers. Either it tells that n is composite, and it furnishes a proof for this as well, or it declares n prime. In the latter case however, it does not furnish a proof, but a probability that the answer is in fact wrong; thus it merely tells that n is *probably prime*. The probability with which a composite number may be declared probably prime by a compositeness test can be made very small, at the cost of performing the test repeatedly; primes will never be declared composite. In practice this means that one knows whether n is prime or composite, but in the prime case a formal proof is lacking, and in the composite case no factors are known!

The aforementioned compositeness tests have led some authors to the use of confusing terminology; regarding compositeness tests as algorithms which output either "prime" or "composite", they call them probabilistic primality tests, because the output has a probability of being wrong. Our view will be that compositeness tests are *compositeness provers*, and cannot serve as a primality test in our sense.

The name *probabilistic primality test* will be reserved for primality testing algorithms that have some "random" aspect built in, in which certain constructions are made depending on random choices. Probabilistic algorithms in our sense may rely on randomness for obtaining their output, but the output itself must be correct. This notion is thus opposed to that of so-called *deterministic* algorithms, in which all steps are determined once the input is known. In either case, if a primality test declares n prime, it is *proven* to be so. Practical algorithms are usually probabilistic, but sometimes an effort is made to give a

deterministic version.

We want to mention the so-called *conditional primality tests* here too, since we will also encounter those in this chapter. A conditional primality test provides primality proofs of which the correctness is conditional upon some unproved (but likely) hypothesis. The only examples we know of are those in which conditions are checked that are sufficient for primality if certain generalized Riemann hypotheses are correct. Often the complexity bounds are conditional in the above sense too.

In a way, one could say that primality proving deals with a problem that does not exist. From a practical point of view, compositeness tests provide the answer: who cares about the negligible possibility of erring? And from a theoretical point of view the question is resolved before it arises, since primality can be proved by using the definition of prime number.

In fact we try to find a practical solution to a theoretical question: find an *efficient* algorithm for primality proving. This leads us to considerations of computational complexity; complexity theory provides the tools to distinguish between efficient and inefficient algorithms. But even on this level it is not clear that the primality proving problem is of interest. As we will point out further on in this chapter, from a strictly theoretical point of view the primality proving problem has been solved: there exists an algorithm for distinguishing primes from composite numbers that is efficient in the theoretical sense of the word. However, so far it has not been (and it is doubtful that it ever will be) useful for proving the primality of one single integer. This type of algorithm gives the theoretical solution in an unpractical way. Even more subtly, there are algorithms that can be used in practice, but that can only be proven to yield the correct answer under the assumption that certain mathematical hypotheses hold – hypotheses that many number theorists are willing to accept, let alone the customer in the prime shop. Such algorithms form a practical solution in a theoretically unsatisfactory way.

The algorithm we will describe in later chapters satisfies both the theoretical need for algorithms that give a correct answer, and the practical need to find that answer in a reasonable amount of time. This also suggests that we will not be satisfied with a merely theoretical algorithm: we want to apply the ideas in practice, which means that algorithms should be implemented on electronic computers.

2. COMPLEXITY.

With the theory of computational complexity one attempts to measure how hard computational problems are. In this section we give some basic complexity results concerning primality proving and factorization.

A natural measure for the difficulty of problems seems to be the number of *arithmetic operations* that are needed to solve the problem; here arithmetic operations are additions, subtractions, multiplications and divisions (with remainder) of integers. The following two theorems apply this measure to both of our problems and serve to show the shortcomings.

From now on, n will denote an integer greater than 1.

(2.1) Theorem. *If n is composite, this can be proved by one integer multiplication.*

The proof will be clear: if n is composite, there exist $n_1, n_2 \in \mathbf{Z}_{\geq 2}$ such that $n = n_1 n_2$. Carrying out this multiplication (and comparison) proves compositeness.

This proof shows the first flaw of complexity measures: the existence of a short proof for compositeness is shown, but not the way to find this short proof, that is, how to factor a number! This is overcome by the following theorem of Shamir [143]. The $g(n) = O(f(n))$ notation denotes (as usual) an upper bound $|g(n)| \leq C f(n)$, with a constant C independent of n. In complexity matters one often uses the binary number system, and taking the logarithm with base 2, the *size* of an integer (the number of binary digits, or bits) is bounded by $\log n + 1$, and is thus $O(\log n)$.

(2.2) Theorem. *If n is composite, a non-trivial factor can be found using $O(\log n)$ arithmetic operations.*

The algorithm that finds a factor in $O(\log n)$ arithmetic operations, essentially uses the computation of factorials, and has the same defect for practical purposes as the algorithm behind the following theorem, dealing with primality proofs.

(2.3) Theorem. *If n is prime, this can be proved by at most 87 integer additions, subtractions and multiplications.*

The proof of this uses that the set of primes is Diophantine: there exists a polynomial $f \in \mathbf{Z}[X_1, X_2, \ldots, X_k]$ with the property that n is prime if and only if n is positive and there exist non-negative integers $x_1, x_2, \ldots, x_k$ such that $n = f(x_1, x_2, \ldots, x_k)$. Several of these

polynomials, of varying degrees, are known (see [101], [59]). It turns out that evaluating the polynomial in 26 variables, explicitly given in [59], takes at most 87 arithmetic operations.

Similarly as in (2.1), the proof does not tell how to find the values of the variables for which the polynomial represents n. But in this case there is another problem, just as for (2.2), that has to do with our choice of complexity measure. Although the primality of a prime n can be proved by at most 87 integer operations, and a composite n can be factored in $O(\log n)$ integer operations, the theorems do not tell how *large* the integers involved are. It turns out that the size of the variables grows exponentially with n; in fact, according to Lenstra in [91], the largest variable for (2.3) will exceed

$$n^{n^{n^n}}$$

for the polynomial in 26 variables mentioned before. It is clearly not reasonable to consider multiplication of integers of this size to be as basic as multiplying 2 and 3.

A much better complexity measure is provided by the notion of *bit operations*. A bit operation is an arithmetic operation on integers consisting of one bit. Since the size of n is $O(\log n)$, the addition and subtraction can be done in $O(\log n)$ bit operations, while multiplication and division with remainder take at most $O((\log n)^2)$ bit operations. (We should mention here that using fast Fourier transforms one can asymptotically achieve an $O((\log n)^{1+\epsilon})$ bound for multiplication, with arbitrary small ϵ, see [140], and Chapter V. This affects all theoretical complexity bounds below.)

Usually an algorithm is called *efficient* if the number of bit operations is bounded by a function that is polynomial in the size of the input; these are the *polynomial time* algorithms, which in our applications thus require $O((\log n)^k)$ bit operations, for some $k \in \mathbf{Z}_{\geq 0}$). The class of problems for which polynomial time solutions exist, is often indicated by P.

Before we describe more "practical" results, we state analogues of the previous theorems for bit operations.

(2.4) Theorem. *If n is composite, this can be proved using $O((\log n)^2)$ bit operations.*

This follows immediately from the proof of (2.1), since the factors of n are bounded by n. This result shows that "compositeness" is in the complexity class NP of problems that have a non-deterministic polynomial time solution, i.e., problems for which a suggested

solution for any instance can be verified in polynomial time. Though finding a solution for such problems need not be possible in polynomial time, at least checking the correctness of a solution is. Obviously the class P of problems for which a solution can be found in polynomial time, is contained in NP. One of the major unsolved problems in complexity theory is whether NP is really larger than P see e.g. [43].

As remarked above, the proof of (2.3) will not lead to polynomial primality proofs. Pratt however, was the first to show that $O((\log n)^4)$ proofs do always exist [125]. Thus "primality" is in NP as well as the complementary property "compositeness"; this is sometimes expressed by saying that "primality" is in $NP \cap$co-NP. In fact this is one of the few problems known to be in this class, but not known to be in P. We will explain the idea behind Pratt's theorem later on in this chapter. Pomerance [121] improved upon this, using elliptic curves (see Sections 10 and 11), to obtain Theorem (2.5). Again, this result only concerns the existence of proofs, not ways of finding them.

(2.5) Theorem. *If n is prime, this can be proved by $O((\log n)^3)$ bit operations.*

The following theorem shows that, assuming the extended Riemann hypothesis, primes and composites can be distinguished in polynomial time. The generalized Riemann hypotheses referred to in (2.6) prescribe the position of the zeroes of the L-series of the characters on $(\mathbf{Z}/n\mathbf{Z})^*$.

(2.6) Theorem. *There is a deterministic algorithm taking $O((\log n)^5)$ bit operations, that correctly decides whether n is prime or composite if certain generalized Riemann hypotheses hold.*

Results of this type give rise to the conditional primality tests, and are due to Miller and Rabin, (see [102], [128], [36]). If n is composite, certain elements of $(\mathbf{Z}/n\mathbf{Z})^*$ are "witness" to this fact, which means that using one of those elements a polynomial proof for compositeness can be made, without exhibiting a factor of n. Assuming the proper Riemann hypotheses, one proves that a small witness must exist; checking all small possibilities in polynomial time, either a witness will be found (and the problem of factoring n remains), or a conditional proof for primality has been obtained.

In a practical variant of this test, one does not really check every small element of $(\mathbf{Z}/n\mathbf{Z})^*$ for being witness, but random choices; it can be proved (see II.1 below) that for every composite n at least three quarters of all elements are witnesses, and thus after

several failed attempts to find a witness one may be confident that they do not exist. Compositeness tests like this underly the division of the task of finding the complete prime factorization into three steps as in the previous section. The problem of primality proving remains that of finding a rigorous proof, once it seems that witnesses for compositeness do not exist.

Theoretically, primality testing is "easy", not only conditionally, as (2.6) shows, but also in a probabilistic sense, see (2.8) below. Until very recently however, the best algorithms for proving the primality of n relied on the factorization of certain auxiliary numbers, viz. $n-1$ or $n+1$. But it is also widely believed that factoring is "hard", which appears to be based on the fact that, despite many efforts, an efficient factoring algorithm has not been found.

The following theorem describes the best known (and proved) bounds for different types of factoring algorithms. The notation $g(x) = o(f(x))$ means that $(g(x)/f(x)) \to 0$ for $x \to \infty$.

(2.7) Theorem.

(i) *There is a deterministic algorithm that factors n completely in $O(n^{\frac{1}{4}+\epsilon})$ bit operations.*

(ii) *There is a deterministic algorithm that factors n completely, and takes $O(n^{\frac{1}{5}+\epsilon})$ bit operations if the generalized Riemann hypothesis is true.*

(iii) *There is a probabilistic algorithm that factors n completely in an expected number of $L(n)^{1+o(1)}$ bit operations, where $L(n) = e^{\sqrt{\log n \log\log n}}$.*

The algorithm in (2.7)(i) is known as Pollard–Strassen, see [119], [149], and Pomerance's paper in [91].

The conditional result of (2.7)(ii) refers to Schoof's adaptation of Shanks' class group algorithm, see Schoof's paper in [91].

The best probabilistic result mentioned in (2.7)(iii) is due to an analysis by H. W. Lenstra, Jr., and C. Pomerance (to be published) of an algorithm using various methods, including class groups and elliptic curves. We should mention here that a recently developed method that works extremely well for numbers of the form $r^e \pm s$, where r and s are small, has a heuristic expected running time of

$$e^{(c+o(1))\sqrt[3]{\log n(\log\log n)^2}},$$

with $c \approx 1.526$. This refers to the "number field sieve", see [82].

For probabilistic algorithms we use *expected* running time; the precise definition of the bound in (2.7)(iii) is the following. There exists a function $f(n)$, tending to 0 as n tends to infinity, such that for every $\epsilon > 0$ there exists K such that with probability at least $1 - \epsilon$ the number of bit operations for factoring n lies in between

$$\frac{1}{K} L(n)^{1+f(n)} \quad \text{and} \quad K L(n)^{1+f(n)}.$$

For primality proving the following summarizes the best proved theoretical results.

(2.8) Theorem.

(i) *There is a deterministic algorithm that, for prime n, leads to a primality proof in $O((\log n)^{C \log\log\log n})$ bit operations, for some effectively computable constant C.*

(ii) *If the generalized Riemann hypothesis is true, there is a deterministic algorithm that, for prime n, leads to a primality proof in $O((\log n)^5)$ bit operations.*

(iii) *There is a probabilistic algorithm that, for prime n, leads to a primality proof in $O((\log n)^k)$ expected bit operations, for some $k \geq 1$.*

The algorithm referred to in (i) is the Gauss sum primality test, see Section 9, and variants like the Jacobi sum test. A probabilistic variant of this led to the first practical general-purpose primality test, that is, a test that does not rely upon special properties of n (such as $n-1$ being easily factorable) to complete the primality proof (see [29] and [30]). The bound in (2.8)(i) is not polynomial in $\log n$, albeit sub-exponential. Recently one competitor for the Jacobi sum test has emerged, see Section 10, for which a rigorous running time analysis has not been given, but which has sparked hope that it may at some time be proven to be polynomial. A probabilistic variant is the only serious competitor for the Jacobi sum method in practice as well.

The result in (2.8)(ii) is a consequence of (2.6).

The final result alludes to results by Adleman and Huang, see [3] and also Section 10.

3. TRIAL DIVISION.

By the very definition of prime number, as an integer having no non-trivial divisors, one can prove the primality of n by showing that it is not divisible by any r with $1 < r < n$. This is the basic form of the *trial division method* for primality testing. But we can do better, based on the following (obvious) theorem.

(3.1) Theorem. *If n has no divisor r with $1 < r \leq \sqrt{n}$ then n is prime.*

In this explicit form the method is often attributed to Leonardo Pisano ("Fibonacci"), beginning 13th century. Cataldi is said to have used trial division to prove that the numbers $2^{13} - 1$, $2^{17} - 1$ and $2^{19} - 1$ are prime ([22] published 1603, see [5]).

Several more methods have been devised to prevent having to trial divide by all r. Firstly, it suffices to check *primes* r up to $\sqrt{n}$ for divisibility. But to use this, one has to generate a list of primes up to $\sqrt{n}$ first. As a matter of fact, such lists are usually made applying trial division techniques, see below. If n is large, it will be more work to generate these primes than it is to do some unnecessary trial divisions. Therefore "wheel" methods are often used, which generalize the simple idea that it is useless to trial divide by even numbers larger than 2 (see e.g. [163]). In these methods trial division is applied by all r up to $\sqrt{n}$ for which r_i is coprime to the auxiliary number t, where $1 \leq r_i \leq t$ and $r \equiv r_i \bmod t$. The auxiliary number t will be taken as the product of small primes. Taking for instance $t = 2 \cdot 3 \cdot 5 \cdot 7 = 210$, first $\gcd(n, t) = 1$ is checked. Next trial division by the "spokes" r_i smaller than and coprime to t is performed, and then the wheel is turned once, to do trial division by the integers $r_i + t$. This is repeated until the bound $\sqrt{n}$ is reached. Thus t trial divisions have been replaced by $\phi(t)$ of them, where ϕ denotes Euler's function. In this example every 210 trials have been replaced by $\phi(210) = 48$ of them.

No matter how much the method is speeded up however, it remains an exponential method. Basically, one proves primality by showing that one cannot factor n. For several reasons though, trial division remains an important tool. We list some of these reasons below.

(3.2) Prime tables. For generating a list of primes up to a given bound, still the sieve method of Eratosthenes, sometimes with minor modifications, is generally used. This elementary sieve method dates back to 200 B.C. approximately. From the list of all positive

integers up to the required bound B, the multiples of the primes are discarded; the primes appear successively on top of the list. For fast methods, see [126], [148].

(3.3) Restricted divisors. If n is an integer of a special form, the number of trial divisions can often be reduced dramatically for primality proofs.

For example, in the (naïve) case $n = 8191 = 2^{13} - 1$ it will take only two trial divisions to complete a primality proof by the following argument. Since $2^{13} \equiv 1 \bmod n$, also $2^{13} \equiv 1 \bmod p$ for any prime divisor p of n, and therefore (by Fermat's theorem) 13 divides $p - 1$. If n were composite, it would thus have an odd prime factor smaller than $\sqrt{8191} < 91$ and congruent to 1 modulo 13, that is, among the integers 27, 53, 79. Since 27 is composite, the primality proof is finished by calculating the remainder of 8191 upon division by 53 and 79.

A bit more of a landmark was reached by Euler, who proved in 1772 that $n = 2^{31} - 1$ is prime, as follows (see [39]). Fermat's theorem again (much more on that in Section 6), gives a congruence for any prime divisor p of n modulo 31. On the other hand p will divide $2 \cdot (2^{31} - 1) = (2^{16})^2 - 2$, and any odd divisor of $x^2 - 2$ is $\pm 1 \bmod 8$. That restricts p to the residue classes 1 and 63 modulo 248, and Euler completed trial division up to the square root of n for integers in these classes with negative results.

Later on, we shall see that modern primality proving algorithms also often restrict the possible divisors of n to a very limited set, for which trial division remains to be done.

(3.4) Trial division bounds. In certain special purpose primality tests that we will encounter in the next sections, it will sometimes be convenient (or even crucial) to know that n, or an auxiliary integer depending on n, does not have divisors smaller than a bound B, even if B is considerably smaller than $\sqrt{n}$.

4. DIFFERENCE OF SQUARES.

The method of factoring n by looking for integers x and y such that

$$n = x^2 - y^2 = (x+y)(x-y) \tag{4.1}$$

was invented by Fermat. If $n = rs$ is odd and composite, it has a decomposition as the difference of squares, since one may take

$$x = \frac{r+s}{2} \quad \text{and} \quad y = \frac{r-s}{2},$$

in (4.1). Thus an exhaustive search for squares among

$$x^2 - n, \quad \text{for } \sqrt{n} \le x \le \frac{1}{2}(\frac{n}{3} + 3), \tag{4.2}$$

can prove the non-existence of divisors. As a primality test, the difference of squares method is only practical in combination with methods of limiting the search in (4.2).

The first way of achieving this, is by *trial division*, as discussed in the previous section.

Gauss proposed another way of limiting the set of possible divisors of n, called *quadratic exclusion*. Here one simply uses the observation that a quadratic residue modulo n must necessarily be a quadratic residue modulo every prime divisor of n. Thus one tries to generate many (small) quadratic residues a modulo n and excludes primes p from the list of possible prime divisors of n by finding some a that is a quadratic non-residue modulo p. Of course the difficulty lies in generating sufficiently many quadratic residues modulo n. Gauss proposed ways of doing that in [44].

By one, or a combination, of the methods of trial division and quadratic exclusion, one may arrive at a bound B such that n is known to have no factors below B. That leads to an upper bound for x in (4.2), replacing 3 by B:

$$x \le \frac{1}{2}(\frac{n}{B} + B). \tag{4.3}$$

For numbers n that have a particular form, this may be combined with restricting the residue classes in which divisors must lie, as done in (3.3).

We demonstrate this by copying a primality proof given by Lehmer in 1929, see [71].

(4.4) Example. Let

$$n = 11111111111111111111 = \frac{10^{23}-1}{9}.$$

First note that if $p > 3$ is a prime divisor of n, then $10^{23} \equiv 1 \bmod p$, and hence by Fermat's little theorem $p \equiv 1 \bmod 23$. As a consequence, *every* divisor r of n must be 1 mod 23.

Next, Lehmer looked for prime divisors of $n-1$, and found the factors 11^2 and 4093. He showed that both

$$m_1 \equiv 3^{\frac{n-1}{11}} - 1 \quad \text{and} \quad m_2 \equiv 3^{\frac{n-1}{4093}} - 1$$

are coprime to n, while $3^{n-1} \equiv 1 \bmod n$. That shows that for every prime divisor p of n the order of 3 in the group $(\mathbf{Z}/p\mathbf{Z})^*$ is a multiple of both 11^2 and 4093; thus every prime divisor, and as a consequence every divisor, is congruent to 1 modulo $11^2 \cdot 4093$. (More about this method in Section 6.)

Since also every divisor r of n is odd, we arrive at

$$r \equiv 1 \bmod 2B \quad \text{where} \quad B = 11^2 \cdot 23 \cdot 4093. \tag{4.5}$$

Without trial division (or quadratic exclusion) we have at least found the lower bound $2B$ for factors of n. For the possible difference of squares $n = x^2 - y^2$, this leads to the restrictions

$$105409255338 < \sqrt{n} \le x \le \frac{1}{2}\left(\frac{n}{2B} + 2B\right) < 243861122499492. \tag{4.6}$$

The lower bound for x is then raised by Lehmer as follows. Applying (4.5) to the factors $r = n$, $r = x - y$ and $r = x + y$, we find that $n \equiv 1 + fB \bmod B^2$ for some f that is determined modulo B, that $x \equiv 1 \bmod B$ and that $y \equiv 0 \bmod B$. But then

$$x^2 \equiv x^2 - y^2 \equiv n \equiv 1 + fB \bmod B^2,$$

which together with $x \equiv 1 \bmod B$ implies that

$$x \equiv 1 + \frac{fB}{2} \bmod B^2.$$

Explicitly: $x \equiv 115222895547343 \bmod 11^4 \cdot 23^2 \cdot 4093^2$.

Finally, this is combined with information modulo 3. Since $n \equiv 2 \bmod 3$, we must have $x \equiv 0 \bmod 3$. Combined with the previous congruence, this leads to

$$x = 115222895547343 + k \cdot 11^4 \cdot 23^2 \cdot 4093^2 \text{ where } k \equiv 2 \bmod 3.$$

But the smallest such x, with $k = 2$ exceeds the upper bound in (4.6). That proves that n is not the difference of two squares and therefore n is prime.

The fact that in a paper two years earlier Lehmer claimed [69] that $3^{n-1} \not\equiv 1 \bmod n$, with n as in this example, and that therefore n could not be prime, shows once more that mistakes are easily made in finding compositeness or primality proofs.

5. QUADRATIC FORMS.

The primality test described in this section arose, mainly by the efforts of Euler, out of an attempt to generalize the following property, known to Fermat. If p is prime and $p \equiv 1 \bmod 4$, then p is the sum of two squares in a unique way (up to interchanging the summands).

For a primality criterion a converse to this would be required. Some care is needed for that as the following examples show. The only representation of the composite number 45 as a sum of squares is $45 = 36 + 9$; it suggests that we should require that the sum of squares is *proper*, meaning that the summands are coprime. But $125 = 121 + 4$ is the only proper sum of squares representation of 125, which also has the improper $125 = 100 + 25$.

(5.1) Theorem. *An integer $m > 1$ with $m \equiv 1 \bmod 4$ is prime if and only if it is the sum of two squares in a unique way and this sum is proper.*

(5.2) Examples. Euler used this primality criterion (even before he had a formal proof for it), to show for instance that

$$262657 = 129^2 + 496^2$$

uniquely, which shows that 262657 is prime, while

$$32129 = 95^2 + 152^2$$

uniquely but improperly, proving that 32129 is composite.

If we say that an integer m is *represented by a quadratic form* $aX^2 + bXY + cY^2$ when there exist integers x and y such that $m = ax^2 + bxy + cy^2$, then (5.1) is a statement about the representability by the quadratic form $X^2 + Y^2$. The *discriminant* of the general quadratic form is $\Delta = b^2 - 4ac$.

We will say that m is represented by $F = aX^2 + cY^2$ in *essentially* one way if there exist integers x and y with $m = ax^2 + cy^2$, and every solution in integers is among the pairs (x, y), $(-x, y)$, $(x, -y)$, $(-x, -y)$; if there exist other solutions, we say that m is represented by F in essentially more than one way. Generalizing our previous notion, a representation $m = ax^2 + cy^2$ is called *proper* if x is coprime to cy and y is coprime to ax.

The first step towards a generalization of (5.1) by Euler consisted of the following lemma; note that again this is the "wrong direction" for primality testing.

(5.3) Lemma. *Let $F = aX^2 + cY^2$ be a quadratic form of negative discriminant $\Delta = -4ac$. Let $m \in \mathbf{Z}_{>1}$ be coprime to Δ. If m is represented by F in essentially more than one way, then m is composite.*

Two centuries ago, this could be proved by looking at identities between products of quadratic forms; for an account of this (and most of this section) see [159]. Modern proofs of the statements in this section use the correspondence between quadratic forms and modules in (orders of) the ring of integers of $\mathbf{Q}(\sqrt{\Delta})$, see [10].

(5.4) Remark. It is worth remarking that if m is represented in essentially more than one way as in Lemma (5.3), a factor of m is also easily obtained. We prove this in an entirely elementary way as follows.

Suppose that $ax_1^2 + cy_1^2 = m = ax_2^2 + cy_2^2$, with $a, c \in \mathbf{Z}_{\geq 1}$. Without loss of generality we assume that x_1, x_2, y_1, y_2 are all positive integers. We also assume that $\gcd(a, c) = 1$, since otherwise a factor is obtained immediately. Finally, we may assume that $x_1 > x_2$, and hence $y_1 < y_2$. Note that therefore $0 < x_1y_2 - x_2y_1 < m$.

We claim that if the two representations of m are essentially different, then either at least one of them is not proper, or $\gcd(x_1y_2 - x_2y_1, m)$ is a non-trivial factor of m. In the former case, a non-trivial factor is given by $\gcd(ax_1, y_1)$ or $\gcd(ax_2, y_2)$.

Suppose that $\gcd(x_1, y_1) = \gcd(x_2, y_2) = 1$, otherwise we have an improper representation. The core of the proof is contained in the following identities:

$$\begin{aligned} m(x_2y_2 - x_1y_1) &= (ax_1^2 + cy_1^2)x_2y_2 - (ax_2^2 + cy_2^2)x_1y_1 = \\ &= a(x_1^2x_2y_2 - x_1x_2^2y_1) + c((x_2y_1^2y_2 - x_1y_1y_2^2) = \\ &= (ax_1x_2 - cy_1y_2)(x_1y_2 - x_2y_1). \end{aligned}$$

For, suppose first that $x_2y_2 - x_1y_1 \neq 0$; we assume $x_2y_2 - x_1y_1 > 0$, the other case is similar. Now necessarily

$$0 < ax_1x_2 - cy_1y_2 < ax_1^2 - cy_1^2 < ax_1^2 + cy_1^2 = m,$$

and therefore the above identities show that $x_1y_2 - x_2y_1$ has at least one prime factor in common with m.

Suppose, on the other hand, that $x_2y_2 = x_1y_1$. Let $1 \leq s = \gcd(y_1, y_2)$, with $y_1 = rs$ and $y_2 = qs$. Then, since x_1 and y_1 are coprime, $x_2 = pr$, for some $p \geq 1$. But $x_2y_2 = x_1y_1$, so $x_1 = pq$. Next observe that $a(x_1^2 - x_2^2) = c(y_2^2 - y_1^2)$. If $\gcd(a, s) \neq 1$, the above

representations are not both proper, and we have proved our claim. So assume that $\gcd(a,s) = 1$; because $s^2 \mid y_2^2 - y_1^2$ then also $s^2 \mid x_1^2 - x_2^2 = p^2(q^2 - r^2)$. But $\gcd(s,p) = 1$, since x_1 and y_1 are coprime, and hence $s^2 \mid q^2 - r^2$. This implies that in fact $s^4 \mid y_2^2 - y_1^2$, and, as before, $s^4 \mid q^2 - r^2$, etcetera. Therefore, $s = 1$, and, by an analogous argument, $p = 1$. In other words, $x_1 = y_2$ and $x_2 = y_1$. Hence $ax_2^2 + cy_2^2 = ay_2^2 + cx_2^2$, therefore $a(x_2^2 - y_2^2) = c(x_2^2 - y_2^2)$. Since x_2 and y_2 are coprime, either $x_2 = y_2 = 1$ (and then also $x_1 = y_1 = 1$), or $a = c = 1$ (by coprimality). In both cases the two representations are not essentially different, in contradiction with the assumptions.

In 1778 Euler proposed the following theorem concerning the *idoneal numbers*; he did not have a proof, or even a clear way of defining or finding idoneal numbers (by some others called suitable numbers or convenient numbers), apparently, other than that which forms the content of the theorem.

(5.5) Theorem. *Let the set I of idoneal numbers consist of the following 65 integers:*

$I =$ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 18, 21, 22, 24, 25, 28, 30, 33, 37, 40, 42, 45, 48, 57, 58, 60, 70, 72, 78, 85, 88, 93, 102, 105, 112, 120, 130, 133, 165, 168, 177, 190, 210232, 240, 253, 273, 280, 312, 330, 345, 357, 385, 408, 462, 520, 760, 840, 1320, 1365, 1848}.

Let $F = aX^2 + cY^2$, with $a, c \in \mathbf{Z}_{\geq 1}$, and let n be coprime to $\Delta = -4ac$. If $ac \in I$, and n is represented essentially uniquely by F and this representation is proper moreover, then n is prime.

(5.6) Example. Using the largest of his idoneal numbers, Euler "proved" the primality of 18518809, by showing that it is uniquely and properly represented as

$$18518809 = 197^2 + 1848 \cdot 100^2.$$

An understanding (as well as a proof) of this theorem, and of the fact that Euler could not find any other idoneal numbers up to at least 10000, came only with the further development of the theory of quadratic forms, by Lagrange and Gauss.

Two quadratic forms $F(X,Y)$ and $G(X,Y)$ of (the same) negative discriminant are *strictly equivalent*, if integers α, β, γ, δ exist such that

$$F(\alpha X + \beta Y, \gamma X + \delta Y) = G(X,Y) \quad \text{with} \quad \alpha\delta - \beta\gamma = 1. \tag{5.7}$$

We call F and G *rationally* equivalent, if (5.7) holds with α, β, γ, δ in $\mathbf{Q}$; thus strictly equivalent forms are certainly rationally equivalent. Strict equivalence is an equivalence relation, and so is rational equivalence. Strictly equivalent forms are said to be in the same *class*. Rationally equivalent forms are said to be in the same *genus*, provided that the equivalence over $\mathbf{Q}$ is given by a matrix with entries whose denominators are coprime to Δ. Thus each genus will comprise at least one class.

The important fact is that two forms of the same discriminant are in the same genus, if and only if they represent integers in the same residue classes of $(\mathbf{Z}/\Delta\mathbf{Z})^*$. Therefore, given a finite set of classes of forms, the integers represented by these will only have a characterization in terms of residue classes modulo Δ if the classes constitute one or more genera. In terms of class field theory this is merely a statement about the splitting of primes in (subfields of) the genus field of $\mathbf{Q}(\sqrt{\Delta})$. For a detailed exposition of all this, see [32].

As a consequence, when looking at a single quadratic form as above, we will only arrive at a satisfactory description of the prime numbers represented by it, in case the class of the form makes up a whole genus of its own; in that case one will obtain (5.5). The integers i in I are such that every genus of quadratic forms of discriminant $-4i$ consists of one class. (In the (36) cases where $i \in I$ is congruent to 0 or 3 modulo 4, the quadratic forms Euler considered correspond to non-maximal orders (proper subrings of the ring of integers) of $\mathbf{Q}(\sqrt{\Delta})$.)

The search for more idoneal numbers has been continued to over 100000 in 1901 (in [33, p. 552], according to Frobenius [42, p. 574]), and even 10^7 in 1948 (see [151]), using restrictions given by Dickson and Hall, (see [35], [48], [49]). No more examples were found. As a consequence of bounds for L-series associated to quadratic fields, it was proven that their number is finite by Chowla in 1934, and Siegel ([24] and [144]). Chowla and Briggs showed in [25] that there is at most one idoneal number exceeding 10^{60}. In [160] it is shown that under certain hypotheses on the distribution of zeroes of L-series it cannot exist.

6. THE CONVERSE OF FERMAT'S THEOREM.

The ideas invoked to apply some kind of converse to Fermat's (little) theorem to primality testing, have been the most successful of all. In fact, these are the only methods that have survived the advent of the electronic computer, and, until very recently, virtually every practical primality test was a descendant of these.

Let us first cite Fermat's theorem.

(6.1) Theorem. *If n is prime, then every a coprime to n satisfies*

$$a^{n-1} \equiv 1 \bmod n. \tag{6.2}$$

It therefore takes only one a for which (6.2) does not hold to prove that n is *not* prime. Using (6.2) for primality proving is more difficult. Although checking (6.2) for a single a can be done quickly, for large n it is impossible to check it for every residue class a modulo n. Still, one could hope that checking (6.2) for one particular a would suffice to prove that n is prime. It was established by Lucas that the value $a = 2$ cannot serve for this purpose, since he noted that $2^{n-1} \equiv 1 \bmod n$ for $n = 37 \cdot 73$; he apparently overlooked the smaller example $n = 11 \cdot 31$. In 1904 Cipolla [27] proved the following theorem, showing that no fixed value for a in (6.2) will distinguish primes from composites.

(6.3) Theorem. *For every $a \in \mathbf{Z}_{\geq 2}$ there exist infinitely many composite n such that $a^{n-1} \equiv 1 \bmod n$.*

Cipolla's proof (see [163]) was simple and constructive: for every a, any n of the form

$$\frac{a^{2p}-1}{a^2-1}$$

with p prime and not a divisor of $a^2 - 1$ will be composite and will satisfy (6.2).

Soon after this, Carmichael (see [21]) showed that trying to find a single a depending on n is also doomed to fail eventually, since for certain composite n there exist no a coprime to n that violate (6.2).

(6.4) Theorem. *There exist composite numbers $n > 1$ such that every a coprime to n satisfies*

$$a^{n-1} \equiv 1 \bmod n.$$

(6.5) Carmichael numbers. The composite numbers n referred to in (6.4) are now called *Carmichael numbers.*

Since (6.2) is satisfied for a whenever the order of a in the multiplicative group $(\mathbf{Z}/n\mathbf{Z})^*$ divides the exponent of the group, Carmichael numbers are those composite numbers for which $\lambda(n)|n-1$; here λ is the function giving the exponent of $(\mathbf{Z}/n\mathbf{Z})^*$. Carmichael's λ-function is easily computed from the structure of $(\mathbf{Z}/n\mathbf{Z})^*$ as the least common multiple of the $\lambda(p^k)$, where p^k is a maximal prime power dividing n, and where

$$\lambda(p^k) = \begin{cases} \phi(p^k), & \text{if } p \text{ is odd or } k \leq 2; \\ \frac{1}{2}\phi(2^k), & \text{if } p = 2 \text{ and } k > 2. \end{cases}$$

It follows at once that every Carmichael number is odd and squarefree. Moreover it is easily shown (as Carmichael did) that Carmichael numbers are the product of at least 3 distinct primes; cf. [23].

The smallest example of a Carmichael number is $n = 3 \cdot 11 \cdot 17 = 561$; indeed $\lambda(561) = \text{lcm}(2, 10, 16) = 80$, which divides $n - 1 = 560$. (Strangely enough, Carmichael overlooked this example; but he gave several others, the smallest being $n = 5 \cdot 13 \cdot 17 = 1105$.)

It is believed that infinitely many Carmichael numbers exist, cf. [122], but this has not been proved; for a conjectured density function see [120], and for large examples see [37], [152], [156], [166], [167], [168].

Still, it is possible to use the fact that only for prime n the multiplicative group $(\mathbf{Z}/n\mathbf{Z})^*$ has order $n-1$. The first steps towards the following result were, again, taken by Lucas.

(6.6) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$. If for every prime p dividing $n-1$ there exists an integer a such that*

$$a^{\frac{n-1}{p}} \not\equiv 1 \bmod n \quad \textit{and} \quad a^{n-1} \equiv 1 \bmod n, \tag{6.7}$$

then n is prime.

(6.8) Remarks. In [99] it seems that Lucas was the first to note that n must be prime if there exists some a such that $a^k \equiv 1 \bmod n$ for $k = n-1$ but for no $k > 0$ smaller than $n-1$. In his book on number theory in 1891 he mentioned that it suffices to check this property for the divisors k of $n-1$. Lehmer ([69], [70], [75], [76]) remarks that it suffices to check it for all k of the form $\frac{n-1}{p}$, with p a prime divisor of $n-1$. Apparently, the

explicit observation that different a may be used for different p has been made much more recently.

In any case, the primality of n follows from the existence of an element of multiplicative order $n-1$ modulo n.

The most important step forward came with Pocklington's theorem; it shows at the same time all the possibilities as well as the inherent limitations of the method. Before we state his result, we mention two special instances, that historically preluded on it.

(6.9) Pepin's Theorem. *Let $n = 2^m + 1$ with $m \geq 2$. Then:*

$$n \text{ is prime} \quad \Longleftrightarrow \quad 3^{\frac{n-1}{2}} \equiv -1 \bmod n.$$

In (6.9) the implication

$$n \text{ is prime} \quad \Rightarrow \quad 3^{\frac{n-1}{2}} \equiv -1 \bmod n$$

is usually proved using quadratic reciprocity: 3 is a quadratic non-residue for primes of the form $2^m + 1$ so the right hand side follows by Euler's criterion. The implication that is of interest for primality testing

$$n \text{ is prime} \quad \Leftarrow \quad 3^{\frac{n-1}{2}} \equiv -1 \bmod n$$

is merely the observation that the order of 3 in $(\mathbf{Z}/n\mathbf{Z})^*$ equals $n-1$.

(6.10) Fermat numbers. Let again $n = 2^m + 1$, and write $m = 2^k r$, with r odd. From

$$2^{2^k r} \equiv (2^{2^k})^r \equiv (-1)^r \equiv -1 \bmod 2^{2^k} + 1$$

it follows immediately that $n = 2^m + 1$ can only be prime if $r = 1$, so n is one of the *Fermat numbers* $F_k = 2^{2^k} + 1$. Pepin's theorem provides a fast primality test for these.

The first five Fermat numbers $F_0, F_1, \ldots, F_4$ are prime, and Fermat seemed to believe that all of them are. Euler however, found the factorization $F_5 = 2^{32} + 1 = 641 \cdot 6700417$. As an aside, we remark that Euler also succeeded in finding two essentially different ways of representing F_5 as the sum of two squares:

$$F_5 = (2^{16})^2 + 1^2 = 62264^2 + 20449^2.$$

Indeed, as pointed out in (5.4), this yields a factor: $\gcd(2^{16}\cdot 20449 - 62264\cdot 1, F_5) = 6700417$.

The only other Fermat numbers that have been completely factored are F_6 (by Landry [66], 1880), F_7 (by Morrison and Brillhart [17], 1970), F_8 (by Brent and Pollard [14], 1980), F_9 (by Lenstra and Manasse et al. [84], 1990) and F_{11} (by Brent [13], 1989). The latter involved a primality proof for the largest prime factor (564 decimal digits), by an elliptic curve method as implemented by F. Morain (see also Section 10).

The smallest Fermat numbers of which the status is at present unknown are F_{22}, F_{24}, F_{28} and F_{31}.

For F_{14} (Selfridge and Hurwitz [53], 1963) and F_{20} (Young and Buell [169], 1987) a compositeness proof has been given, but no factors are known.

For the remaining k up to 32 (inclusive) a partial factorization of F_k is known, as well as for 76 larger values of k, the largest being $k = 23471$ (see Keller [63]). For all these, see [132] and [18].

(6.11) Remarks. Pepin's original test [113] dates back to 1877; he used the base 5 instead of 3, which will work for all Fermat numbers except $F_1 = 5$. (As a matter of fact, he also mentions 10 as a possible base, because it has the advantage that the actual calculations start only when the power of 10 exceeds the modulus $n = F_k$.)

It is interesting to note that Pepin's paper appeared as a reaction to a paper by Lucas, in the same volume of the same journal ([96]). In the latter, Lucas made an apparently erroneous attempt to apply the new method he was developing (which we will describe in the next section) to the Fermat numbers. Apart from those errors, it was not clear from the theorem as it was stated, for which k it would settle the question of primality of F_k and for which k it would merely recount known information about residue classes of possible divisors of F_k. But anyhow it is interesting to see that the "$n+1$"-methods in fact preceded the "$n-1$"-methods.

The following is a slight generalization of Pepin's theorem, due to Proth (1878) [127].

(6.12) Proth's Theorem. *Let $n = h\cdot 2^m + 1$ with h odd and $h < 2^m$. Then:*

$$n \text{ is prime} \iff \text{there exists } a \in \mathbf{Z} \text{ such that } a^{\frac{n-1}{2}} \equiv -1 \bmod n.$$

In this case the right hand side implies the existence of an element of order 2^m in $(\mathbf{Z}/n\mathbf{Z})^*$; that element must have order 2^m for at least one prime p dividing n. Necessarily, this p is congruent to 1 modulo 2^m. But 2^m exceeds the square root of n by assumption, and therefore $n = p$ is prime itself.

This argument lends itself for generalization to a proof of Pocklington's theorem below.

For testing a particular n, a suitable value of a is generally easily found: find an element that is a quadratic non-residue modulo n if n is prime. A given a typically works for h and m in certain residue classes modulo the value of some function of a.

(6.13) Example. In 1957 Robinson carried out the first extensive (computer) tests based on Proth's theorem and he generated several pages of primes (see [134]). One of these primes is $n = 1575 \cdot 2^{147} + 1$, which was proven prime by calculating

$$47^{\frac{n-1}{2}} \bmod n.$$

The value 47 was an exceptionally large smallest quadratic non-residue.

(6.14) Pocklington's Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$ and let $p^k \mid n-1$, with p prime and $k \in \mathbf{Z}_{\geq 1}$. If there exists $a \in \mathbf{Z}$ such that*

$$\gcd(a^{\frac{n-1}{p}} - 1, n) = 1 \quad \text{and} \quad a^{n-1} \equiv 1 \bmod n,$$

then every divisor r of n satisfies $r \equiv 1 \bmod p^k$.

The main new ingredient in Pocklington's theorem (first given in [118]) is the requirement that the greatest common divisor is trivial, which allows one to draw the conclusion that the order of a modulo any prime divisor (and hence every divisor) of n is divisible by p^k.

In order to be able to draw the conclusion that n is prime, either p^k should in itself exceed $\sqrt{n}$, or we have to combine our knowledge for several primes dividing $n-1$, using the Chinese remainder theorem. In any event, proofs for primality will be obtained only if the factored part of $n-1$ exceeds $\sqrt{n}$.

This shows the weakness of this converse of Fermat type approach from a general point of view: it reduces the primality question for n to a supposedly harder problem, namely that of factoring $n-1$. Before we show the generalizations allowing factors of $n+1$ and other auxiliary numbers as well, we present the most versatile theorem of this section, taking factor bounds into consideration (see [18] and [19]).

(6.15) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$ and let $n-1 = FR$. If for every prime p dividing F there exists $a \in \mathbf{Z}$ such that*

$$\gcd(a^{\frac{n-1}{p}} - 1, n) = 1 \quad \textit{and} \quad a^{n-1} \equiv 1 \bmod n,$$

then every divisor r of n satisfies $r \equiv 1 \bmod F$.

Suppose moreover that $\gcd(F, R) = 1$, that every prime factor of R exceeds B, and there exists $b \in \mathbf{Z}$ such that

$$\gcd(b^{\frac{n-1}{R}} - 1, n) = 1 \quad \textit{and} \quad b^{n-1} \equiv 1 \bmod n;$$

then n is prime if $FB > \sqrt{n}$.

In the first assertion Pocklington's theorem has been combined for all primes in the factored part F of $n-1$; the last assertion combines this with a Pocklington type test for the prime divisors of the unfactored part R of $n-1$, about which the only available information is that they exceed B.

7. TESTS OF LUCAS-LEHMER TYPE.

The primality tests of the previous section can be seen as attempts to prove that n is prime by showing that the multiplicative group $(\mathbf{Z}/n\mathbf{Z})^*$ has order (or rather exponent) $n-1$. If n is indeed prime, $\mathbf{Z}/n\mathbf{Z}$ is a *finite field* of n elements, and extension fields of every positive degree will exist. In the simplest case of a *quadratic extension*, the multiplicative group will have order n^2-1; using that, one is able to utilize divisors of $n+1$ too. An easy way to construct these quadratic extensions of finite fields, is by looking at reductions modulo n of the ring of integers of suitable *quadratic number fields* as follows.

The quadratic field $\mathbf{Q}(\sqrt{\Delta})$ is obtained by adjoining to $\mathbf{Q}$ a root (in an algebraic closure) of $X^2 - PX + Q = 0$, with P, Q in $\mathbf{Z}$ and where the *discriminant* $\Delta = P^2 - 4Q$ is not an integral square. This field $\mathbf{Q}(\sqrt{\Delta})$ has an automorphism σ over $\mathbf{Q}$ of order 2, obtained by sending $\sqrt{\Delta}$ to $-\sqrt{\Delta}$. The *norm* of an element $x \in \mathbf{Q}(\sqrt{\Delta})$ is the element $N(x) = x\sigma x$ of $\mathbf{Q}$. In the ring of integers O_Δ of $\mathbf{Q}(\sqrt{\Delta})$, rational primes p (not dividing Δ) for which Δ is not a square modulo p remain prime, and $O_\Delta/(p)$ forms a field of order p^2. We say that $\alpha \in \mathbf{Q}(\sqrt{\Delta})$ is coprime to $m \in \mathbf{Z}$ if $N(\alpha)$ is.

(7.1) Theorem. *Let $\Delta \equiv 0, 1 \bmod 4$. If n is an odd prime number not dividing Δ, then every $\alpha \in O_\Delta$ coprime to n satisfies*

$$\alpha^{n^2-1} \equiv 1 \bmod n. \tag{7.2}$$

This is the direct generalization of Theorem (6.1). Notice that in case $(\frac{\Delta}{n}) = 1$, the conclusion follows from (6.1).

If n is an odd prime, then for $\alpha = a + b\sqrt{\Delta} \in \mathbf{Z}[\sqrt{\Delta}]$ we have

$$\alpha^n = (a + b\sqrt{\Delta})^n \equiv a^n + b^n\sqrt{\Delta}^n \equiv a + b\Delta^{\frac{n-1}{2}}\sqrt{\Delta} \bmod n.$$

By Euler's criterion

$$\Delta^{\frac{n-1}{2}} \equiv (\frac{\Delta}{n}),$$

so

$$\alpha^n = (a + b\sqrt{\Delta})^n \equiv a^n + b^n\sqrt{\Delta}^n \equiv a + (\frac{\Delta}{n})b\sqrt{\Delta} \bmod n.$$

If we define $\rho_n = \sigma$ in case $\left(\frac{\Delta}{n}\right) = -1$ and $\rho_n = \mathrm{id}$ in case $\left(\frac{\Delta}{n}\right) = 1$, we proved that

$$\alpha^n \equiv \rho_n \alpha \bmod n.$$

Remark that in any case ρ_n induces an automorphism on the finite field $\mathbf{Z}/n\mathbf{Z}[\sqrt{\Delta}]$ generating the automorphism group over $\mathbf{Z}/n\mathbf{Z}$, under the convention that $\mathbf{Z}/n\mathbf{Z}[\sqrt{\Delta}]$ denotes $\mathbf{Z}/n\mathbf{Z}$ in case Δ is a square modulo n.

The next theorem easily follows; note that $2\alpha \in \mathbf{Z}[\sqrt{\Delta}]$ if $\alpha \in O_\Delta$. It shows that powering modulo n has the same effect as applying the automorphism that generates the Galois group of the extension, a theme that will reappear in Chapter II.

(7.3) Theorem. *Let $\Delta \equiv 0, 1 \bmod 4$ and let n be an odd prime number not dividing Δ. Then every $\alpha \in O_\Delta$ satisfies*

$$\alpha^{n+1} \equiv \alpha \rho_n \alpha \bmod n. \tag{7.4}$$

We can rephrase (7.4) in several ways; for instance

$$\alpha^{n+1} - (\rho_n \alpha)^{n+1} \equiv 0 \bmod n. \tag{7.5}$$

If we write $\beta = \alpha / \rho_n \alpha$, which equals 1 if $\left(\frac{\Delta}{n}\right) = 1$, then (7.4) implies

$$\beta^{n+1} \equiv 1 \bmod n. \tag{7.6}$$

From this it is even more obvious that $n+1$ is coming into play.

Of course we need some kind of converse again for primality testing.

The first to exploit (7.3) was Lucas (see [93], [94], [95], [96], [97] and [98]), as we pointed out, in fact even before Pepin and Proth developed the first "plain" $n-1$-techniques. He, and many of the people building upon his work, phrased results in terms of recurring sequences. Basically, these recurring sequences are merely a way of computing the coefficients of the powers of α in (7.3), without leaving $\mathbf{Z}$, as follows. The more algebraic description can be found in [16], [55], [62], [131], [161].

Suppose that α is a zero of $X^2 - Px + Q$, with $P, Q \in \mathbf{Z}$, so $\alpha + \sigma\alpha = P$ and $\alpha\sigma\alpha = Q$, while $\Delta = P^2 - 4Q$. Define integers v_i, u_i for $i \geq 1$ by

$$\alpha^k = \frac{v_k + u_k\sqrt{\Delta}}{2^k}.$$

Then obviously

$$v_k = 2^{k-1}(\alpha^k + (\sigma\alpha)^k) \text{ and } u_k\sqrt{\Delta} = 2^{k-1}(\alpha^k - (\sigma\alpha)^k).$$

That makes it easy to calculate u_k and v_k recursively:

$$\begin{aligned} v_{k+1} &= 2^k(\alpha^{k+1} + (\sigma\alpha)^{k+1}) \\ &= 2^k\big((\alpha + \sigma\alpha)(\alpha^k + (\sigma\alpha)^k) - \alpha\sigma\alpha(\alpha^{k-1} + (\sigma\alpha)^{k-1})\big) \\ &= 2^k\big(Pv_k - Qv_{k-1}\big), \end{aligned}$$

and

$$\begin{aligned} u_{k+1}\sqrt{\Delta} &= 2^k(\alpha^{k+1} - (\sigma\alpha)^{k+1}) \\ &= 2^k\big((\alpha + \sigma\alpha)(\alpha^k - (\sigma\alpha)^k) - \alpha\sigma\alpha(\alpha^{k-1} - (\sigma\alpha)^{k-1})\big) \\ &= 2^k\big(Pu_k\sqrt{\Delta} - Qu_{k-1}\sqrt{\Delta}\big), \end{aligned}$$

so

$$u_{k+1} = 2^k\big(Pu_k - Qu_{k-1}\big).$$

Also, exponentiation can be done efficiently by repeated squaring and multiplication, and so u_n and v_n are quickly found by the above together with the doubling formulas

$$v_{2k} = 2^{2k}(v_k^2 - 2Q^2) \text{ and } u_{2k} = 2^{2k}u_kv_k.$$

Taking everything modulo n, and replacing σ by ρ_n as before, (7.5) states that

(7.7) $$u_{n+1} \equiv 0 \bmod n$$

for odd primes n; thus we will only be interested in v_n and u_n modulo n, and we need not worry about the powers of 2 in particular. for every $k \geq 1$ if $\left(\frac{\Delta}{n}\right) = 1$.

Before stating Lucas's results for primality testing, we say something about pseudoprimes.

(7.8) Lucas pseudoprimes. Again, (7.2) will be satisfied for certain composite numbers too. Every property of primes that does not precisely characterize the primes will give rise to a notion of *pseudoprimes*, composite numbers having that particular property. More on this in the next section.

Here we give an example of an analogue of Carmichael numbers: there exist composite n for which (7.2) holds for every α coprime to n in O_Δ, where $(\frac{\Delta}{n}) = -1$. For this, let $n = 5 \cdot 7 \cdot 13$ and $\Delta = 61$; one easily verifies that

$$(\frac{\Delta}{5}) = (\frac{\Delta}{13}) = 1 \text{ and } (\frac{\Delta}{7}) = (\frac{\Delta}{n}) = -1.$$

Since $5-1$ and $13-1$ divide $n+1$, by Fermat's theorem $\alpha^{n+1} \equiv 1 \bmod p$ both for $p = 5$ and $p = 13$. Also, $7^2 - 1$ divides $n^2 - 1$ and therefore $\alpha^{n^2-1} \equiv 1 \bmod 7$ by (7.1).

Williams studied the following analogue of Carmichael numbers: composite n such that for some Δ with $(\frac{\Delta}{n}) = -1$ the congruence in (7.7) holds (and hence that in (7.5)), for every $\alpha \in O_\Delta$ (see [162]). One example he gives is $n = 17 \cdot 19$ with $\Delta = 5$. Then:

$$(\frac{\Delta}{n}) = (\frac{5}{17 \cdot 19}) = -1 = (\frac{5}{17}) \text{ and } (\frac{5}{19}) = 1,$$

while both $17+1$ and $19-1$ divide $n+1$. Thus (7.5) holds by (7.3) and (6.1) for every α.

See also [8], [122], [136].

(7.9) Lucas's Theorem. *Let $n = 2^m - 1$ with $m > 2$, and define $e_i \in \mathbf{Z}$ for $i \geq 1$ by $e_1 = 4$ and $e_{i+1} = e_i^2 - 2$. Then:*

(7.10) $\qquad n$ *is prime* $\quad \Longleftrightarrow \quad e_{m-1} \equiv 0 \bmod n.$

Proof. If m is even, $n = 2^m - 1$ is divisible by 3, and therefore not prime for $m > 2$; on the other hand $e_{m-1} \equiv 2 \bmod 3$ in this case, so not divisible by n. Thus we may assume that m is odd.

Now

$$(\frac{12}{n}) = (\frac{3}{n}) = -(\frac{2^m - 1}{3}) = -(\frac{1}{3}) = -1,$$

since m is odd. Take take $\Delta = 12$ and $\alpha = 2^{\frac{m-1}{2}}(-1+\sqrt{3}) \in O_\Delta$, then $\sigma\alpha = 2^{\frac{m-1}{2}}(-1-\sqrt{3})$ and $\alpha\sigma\alpha = -2^m$. Furthermore, define for $i \geq 0$:

$$a_i = \alpha^{2^i} + \sigma\alpha^{2^i} \in O_\Delta;$$

then

$$a_{i+1} = \alpha^{2^{i+1}} + \sigma\alpha^{2^{i+1}} = (\alpha^{2^i} + \sigma\alpha^{2^i})^2 - 2(\alpha\sigma\alpha)^{2^i} = a_i^2 - 2(-2^m)^{2^i},$$

and in particular

$$a_1 = (a_0)^2 - 2(-2^m) = (\alpha + \sigma\alpha)^2 + 2^{m+1} = 2^{m+1} + 2^{m+1} = 2^{m+2}.$$

Using that $2^m \equiv 1 \bmod n$, we find that $a_1 \equiv 4 \bmod n$ and that $a_{i+1} \equiv a_i^2 - 2 \bmod n$. As a consequence $a_i \equiv e_i$ holds for $i \geq 1$. Thus

$$\begin{aligned} e_{m-1} \equiv 0 \bmod n \quad &\Longleftrightarrow \quad a_{m-1} \equiv 0 \bmod n \\ &\Longleftrightarrow \quad \alpha^{\frac{n+1}{2}} + \sigma\alpha^{\frac{n+1}{2}} \equiv 0 \bmod n. \end{aligned}$$

If the right hand side in (7.10) holds, this shows that the image of the element $\alpha/\sigma\alpha$ in $O_\Delta/(n)$ has order $n+1$; then $n+1$ divides $p^2 - 1$ by (7.1) for every prime divisor p of n. In particular, $p^2 - 1 \geq n + 1$, so n must be prime itself.

If, on the other hand, n is prime, then

$$\alpha^{\frac{n+1}{2}}(\alpha^{\frac{n+1}{2}} + \sigma\alpha^{\frac{n+1}{2}}) \equiv \alpha^{n+1} + (\alpha\sigma\alpha)^{\frac{n+1}{2}} \equiv \alpha\sigma\alpha + (-1)^{\frac{n+1}{2}} \equiv 0 \bmod n,$$

by (7.4), and because $\alpha\sigma\alpha \equiv -1 \bmod n$, while $n \equiv 3 \bmod 4$.

That proves (7.9).

Lehmer is credited for formulating the Lucas test in the present form (see [72], [73]). In his original papers ([97], [98]), Lucas formulated his test as in (7.9) only for $m \equiv 1 \bmod 4$; for $m \equiv 3 \bmod 4$ he used the same recurrence relation for e_i but starting value $e_1 = 3$, i.e., in that case he used discriminant $\Delta = 5$ and $\alpha = \frac{3+\sqrt{5}}{2}$ instead of the above values. Lehmer erroneously claimed that the latter would not give a necessary condition for primality.

(7.11) Mersenne numbers. Just as Pepin's test was tailor-made for Fermat numbers, Lucas's test works for the *Mersenne numbers* $n = M_m = 2^m - 1$. We saw already that for $m > 2$ these can only be prime if m is odd. Since

$$2^{rs} \equiv (2^r)^s \equiv (1)^s \equiv 1 \bmod 2^r - 1$$

it is clear that only prime exponents m need be considered.

These numbers are named after Marin Mersenne (1568-1648), who made the unsubstantiated claim in 1647 that M_m is prime for those prime exponents m that exceed by at most 3 a power of 2 with even exponent. He claimed moreover for $m \leq 257$, that M_m is prime only if $m = 2$, 3, 5, 7, 13, 17, 19, 31, 67, 127, or 257. The Mersenne primes up

to M_{19} were known by Cataldi, as we pointed out in Section 4. Also, M_{11} was known to be divisible by 23 and Fermat had published non-trivial factors for M_{23} and M_{37} in 1640. Much later, in 1732, Euler found factors of M_{29}, M_{43} and M_{73} (and stated in general, that if the prime m is 3 mod 4 and $2m+1$ is prime too, then M_m is divisible by $2m+1$, and therefore not prime; this was proven by Lagrange, 1775). Still in accordance with Mersenne's list, Euler proved the primality of $2^{31}-1$ in 1772, see (3.3).

Lucas was especially interested in applying his method to M_{127}, and he claimed to have proved its primality by the above method ("mais une seule fois") in 1877. It seems that only after this had been checked by Fauquemberge [40] in 1914, that its primality was put beyond doubt. Lucas also believed (see [100, v1, p. 376]) to have shown the compositeness of M_{89}, which later turned out to be prime. (Additional justification for suspicion in these matters in general is given by the case of M_{167}; Barker [9] published his result in 1945, stating that this number is composite since he found a non-zero residue for a sequence as in (7.9). Lehmer later found that the result was right, but the residue given wrong. See [154].)

In 1883 finally Mersenne's claim was refuted, as Pervouchine [114] proved M_{61} prime. Cole [31] found the (non-trivial) factorization of M_{67} in 1903. Later three more errors emerged: M_{89} (Powers [123], 1911) and M_{107} (Powers [124] and Fauquemberge [40], 1914) are prime, while M_{257} turned out to be composite (Kraitchik, 1922, "but no guarantee", and Lehmer [73], 1932). For all this, and much more (for instance on Lucas's proposal for a primality proving machine) see [5].

At present, many more of the numbers M_m have been checked, including all m up to 150000. For the following values of m Mersenne's numbers are known to be prime: 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091. See [132] and [18].

For numbers of the form $n = h \cdot 2^m - 1$ we have this theorem, analogous to Proth's theorem.

(7.12) Theorem. *Let $n = h \cdot 2^m - 1$ with h odd and $h < 2^m$. Let $\Delta \equiv 0, 1 \bmod 4$ and suppose that $\left(\frac{\Delta}{n}\right) = -1$. Then:*

$$(7.13) \qquad n \text{ is prime} \iff \text{there exists } \alpha \in O_\Delta \text{ such that } \left(\frac{\alpha}{\sigma\alpha}\right)^{\frac{n+1}{2}} \equiv -1 \bmod n.$$

Proof. First suppose that n is prime. Then $\sigma\alpha \equiv \alpha^n \bmod n$ as we saw above, and

$$\left(\frac{\alpha}{\sigma\alpha}\right)^{\frac{n+1}{2}} \equiv \left(\frac{\alpha}{\alpha^n}\right)^{\frac{n+1}{2}} = (\alpha^{n+1})^{-\frac{n-1}{2}} \bmod n.$$

Since $O_\Delta/(n)$ is a finite field of n^2 elements, the multiplicative group is cyclic of order n^2-1, so any non-square α will have the desired property.

Conversely, in case

$$\left(\frac{\alpha}{\sigma\alpha}\right)^{\frac{n+1}{2}} \equiv -1 \bmod n,$$

the order of $\alpha/\sigma\alpha$ modulo p is divisible by 2^m for every prime divisor p of n. Then either $p-1$ or $p+1$ is divisible by 2^m by (6.1) or (7.6), depending on $\left(\frac{\Delta}{p}\right)$. First suppose that $p = 2^m \pm 1$. Then either $p = 2^m - 1$, in which case p divides $n - ph = h - 1$, or $p = 2^m + 1$, in which case p divides $hp - n = h + 1$; in both cases $h \geq 2^m$, contrary to the assumptions. So every prime p dividing n satisfies $p = k \cdot 2^m \pm 1$ with $k \geq 2$; hence

$$\begin{aligned} p^2 - n &\geq (k2^m - 1)^2 - h2^m = (k^2 2^m - h - 2k)2^m - 2 > \\ &> ((k^2-1)2^m - 2k)2^m + 2 > 0, \end{aligned}$$

and p must equal n. If $k = 1$, then

That proves (7.12).

(7.14) Remarks. Theorem (7.12) is easily translated into the language of recurring sequences again. Let the hypotheses be as in (7.12), in particular $n = h \cdot 2^m - 1$. Writing $\beta = \frac{\alpha}{\sigma\alpha}$, the congruence in (7.13) is equivalent to

$$\begin{aligned} \beta^{\frac{n+1}{2}} \equiv -1 \bmod n \quad &\Longleftrightarrow \quad \beta^{\frac{n+1}{4}} + \beta^{-\frac{(n+1)}{4}} \equiv 0 \bmod n \\ &\Longleftrightarrow \quad w_{m-2} \equiv 0 \bmod n, \end{aligned}$$

if we define w_i for $i \geq 0$ as follows:

$$w_0 = \beta^h + \beta^{-h} \quad \text{and} \quad w_i = w_{i-1}^2 - 2, \text{ for } i \geq 1.$$

The starting value w_0 is similarly determined recursively: $w_0 = z_h$ if we put

$$z_0 = 2,\ z_1 = \beta + \beta^{-1}, \quad z_{i+1} = z_1 z_i - z_{i-1} \quad \text{and} \quad z_{2i} = z_i^2 - 2, \text{ for } i \geq 1,$$

(making $z_i = \beta^i + \beta^{-i}$).

The analogue of Pocklington's theorem reads as follows.

(7.15) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$ and let $p^k \mid n+1$, with p prime. Let $\Delta \equiv 0,1 \bmod 4$ and suppose that $\left(\frac{\Delta}{n}\right) = -1$. If there exists $\alpha \in O_\Delta$ coprime to n such that*

$$\gcd(\beta^{\frac{n+1}{p}} - 1, n) = 1 \quad \text{and} \quad \beta^{n+1} \equiv 1 \bmod n, \quad \text{where } \beta = \frac{\alpha}{\sigma\alpha},$$

then every prime divisor r of n satisfies $r \equiv \left(\frac{\Delta}{r}\right) \bmod p^k$.

As we will generalize this type of theorem in Chapter II further, we do not give a formal proof here. But basically, the imposed conditions imply for any prime divisor r of n that the order of α in $O_\Delta/(r)^*$ is divisible by p^k, whence the result by (6.1) and (7.1) (but note that for $p = 2$ an additional argument is required!). Again everything may be phrased in terms of recurring "Lucas"-sequences.

Combining information about divisors of $n-1$ and $n+1$, we have the following theorem (see [131], [111], [18] etc.).

(7.16) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$ and let $n^2 - 1 = FR$. Let $\Delta \equiv 0,1 \bmod 4$ and suppose that $\left(\frac{\Delta}{n}\right) = -1$. If for every prime p dividing F there exists $\alpha \in O_\Delta$ such that*

$$\gcd(\alpha^{\frac{n^2-1}{p}} - 1, n) = 1 \quad \text{and} \quad \alpha^{n^2-1} \equiv 1 \bmod n,$$

then every divisor r of n satisfies $r \equiv \pm 1 \bmod F$. Suppose moreover that $\gcd(F, R) = 1$, that every prime factor of R exceeds B, and there exists $\beta \in O_\Delta$ such that

$$\gcd(\beta^{\frac{n^2-1}{R}} - 1, n) = 1 \quad \text{and} \quad \beta^{n^2-1} \equiv 1 \bmod n;$$

then n is prime if $F \cdot B > \sqrt{n}$.

Lehmer, Williams and others (see [72], [60], [164]) have generalized the resulting primality test in such a way that divisors of $n^{12} - 1$ will help in completing the primality proof for n. We will refer in the sequel to all of these (including those in the present and in the previous section) as *Lucas-Lehmer-type* primality tests. The basic problem with all of these is that they rely on factorization of auxiliary numbers (such as $n^2 - 1$) and are therefore suitable only for primes for which these factorizations can be obtained, such as the Fermat and the Mersenne numbers.

The success of these methods can be seen from the table below. In it we have tried to assemble information about the largest known primes throughout history.

(7.17) Remarks. The first entry in the table that has not been mentioned so far, is the 14-digit prime discovered by Landry. As an appendix to the first of two lengthy papers by Lucas [97], a table of prime factors of numbers $2^m \pm 1$, with m up to 64, compiled by Landry, was published in 1878. The 14-digit divisor of $2^{53}+1$ was the largest of those. He found also four 13-digit primes; all of his results are correct, but it is not clear how Landry obtained his results. Lucas remarks: "M. F. Landry, au moyen d'une méthode inédite, et probablement fort simple, est parvenu à la décomposition de certains grands nombres en leurs facteurs premiers".

In the second paper ([98]), Lucas mentions that since 1859 a 10-digit prime had been known, as Plana claimed to have verified that

$$\frac{3^{29}+1}{2^2 \cdot 6091}$$

is prime. Lucas discovered however that this number is divisible by 523.

We mentioned before that Lucas put some effort into proving $2^{127}-1$ prime. Especially in later years, he did not seem to be too convinced that he had succeeded. At first he was rather confident: "C'est à l'aide de ces théorèmes que je pense avoir demontré que le nombre $A = 2^{127} - 1$ est premier." (1876). In 1877 he made the statement we quoted in (7.11), that he had shown the primality of M_{127}, but only once. In 1887 however, he said about $2^{61}-1$: "C'est le plus grand nombre premier actuellement connu". (We thank J. O. Shallit for pointing these references out to us.)

This number M_{61} was proven prime by Pervouchine, in 1883; the report [114] only mentions "ses longs et fatigants calculs", and also the existence of a document, accompanied by some tables, that should facilitate the verification of the primality proof – in other words, a prime certificate. But nothing is said about the method.

In June of 1951, for the first time an electronic computer (Edsac) produced a prime that was larger than any known before, after searching through numbers of the form $k \cdot M_{127} + 1$. For the period from early July to October 1951, Ferrier was the last to hold the record with a prime that was found by the use of a desk calculator only. After that, electronic computers took over completely.

The largest proven primes throughout history				
Prime	**Digits**	**Prover**	**Year**	**Method**
$2^{13}-1$	4	? [28]	$\leq$1461	(3.1)
$2^{17}-1$	6	Cataldi [22]	1588	(3.1)
$2^{19}-1$	6	Cataldi [22]	1588	(3.1)
$2^{31}-1$	10	Euler [39]	1772	(3.3)
$\frac{2^{53}+1}{3\cdot 107}$	14	Landry [97]	1876	?
$2^{61}-1$	19	Pervouchine [114]	1883	?
$2^{89}-1$	27	Powers [123]	1911	(7.9)
$2^{107}-1$	33	Powers [124]/Fauquembergue [40]	1914	(7.9)
$2^{127}-1$	39	Lucas/Fauquemberge [40]	1914	(7.9)
$934(2^{127}-1)+1$	42	Miller,Wheeler [103]	1951	(6.12)?
$\frac{2^{148}+1}{17}$	44	Ferrier [41]	1951	(4.4)
$180(2^{127}-1)^2+1$	79	Miller,Wheeler [103]	1951	(6.12)?
$2^{521}-1$	157	Lehmer, Robinson [77][133]	1952	(7.9)
$2^{607}-1$	183	Lehmer, Robinson [77][133]	1952	(7.9)
$2^{1279}-1$	386	Lehmer, Robinson [78][133]	1952	(7.9)
$2^{2203}-1$	664	Lehmer, Robinson [79][133]	1952	(7.9)
$2^{2281}-1$	687	Lehmer, Robinson [79][133]	1952	(7.9)
$2^{3217}-1$	969	Riesel [130]	1957	(7.9)
$2^{4253}-1$	1281	Hurwitz, Selfridge [52][53]	1961	(7.9)
$2^{4423}-1$	1332	Hurwitz, Selfridge [52][53]	1961	(7.9)
$2^{9689}-1$	2917	Gillies [45]	1963	(7.9)
$2^{9941}-1$	2993	Gillies [45]	1963	(7.9)
$2^{11213}-1$	3376	Gillies [45]	1963	(7.9)
$2^{19937}-1$	6002	Tuckerman [153]	1971	(7.9)
$2^{21701}-1$	6533	Nickel, Noll [112]	1978	(7.9)
$2^{23209}-1$	6987	Noll [112]	1979	(7.9)
$2^{44497}-1$	13395	Nelson, Slowinski [146]	1979	(7.9)
$2^{86243}-1$	25962	Slowinski	1982	(7.9)
$2^{132049}-1$	39751	Slowinski	1983	(7.9)
$2^{216091}-1$	65050	Slowinski	1985	(7.9)
$391581\cdot 2^{216193}-1$	65087	Brown, Noll, Parady, Smith, Smith and Zarantonello [170]	1989	(7.9)

8. PSEUDOPRIMES.

Apart from the Lucas-Lehmer type tests, there are several other applications in primality testing that grew out of attempts to find a converse to Fermat's theorem. We describe several of them in this section.

We noted in Section 6 that there exist composite n such that $a^{n-1} \equiv 1 \bmod n$ for some a coprime to n. Such composite n are called *pseudoprimes to the base a*. Every fixed integer a admits infinitely many pseudoprimes, as Cipolla showed, cf. (6.3); more recent results about the *density* of pseudoprimes can be found in [122]. We also introduced the Carmichael numbers, composite numbers n that are pseudoprime to every base coprime to n.

Several attempts have been made to strengthen Fermat's theorem in order to restrict the number of pseudoprimes. The first idea is to use Euler's criterion.

(8.1) Theorem. *Let n be an odd prime number. Then every integer a that is not divisible by n satisfies:*

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \bmod n; \tag{8.2}$$

in particular for every such a:

$$a^{\frac{n-1}{2}} \equiv \pm 1 \bmod n. \tag{8.3}$$

An *Euler pseudoprime to the base a* is an odd composite integer n, not dividing a, such that (8.2) holds.

(8.4) Theorem. *Let n be an odd integer. Then:*

$$n \text{ is prime} \iff \{a^{\frac{n-1}{2}} \bmod n: \ a \text{ coprime to } n\} = \{-1, 1\}.$$

Furthermore, if n is composite, then either

$$a^{\frac{n-1}{2}} \equiv 1 \bmod n \quad \text{for every } a \text{ coprime to } n, \tag{8.5}$$

or

$$a^{\frac{n-1}{2}} \not\equiv \pm 1 \bmod n \text{ for at least } \frac{\phi(n)}{2} \text{ of all } a \text{ with } 1 \leq a < n \text{ coprime to } n. \tag{8.6}$$

A proof for the second assertion in (8.4) is given easily as follows (cf. [80], [147], [68]). Let n be odd and composite. Assume that (8.3) holds for every a with coprime to n. In particular, n will be a Carmichael number, hence squarefree, see (6.5). Let $n = q \cdot r$, with coprime q and r exceeding 1. Suppose that for at least one a:

$$a^* = a^{\frac{n-1}{2}} \not\equiv 1 \bmod n.$$

Then by assumption $a^* \equiv -1 \bmod n$. Choose $b \equiv 1 \bmod q$ and $b \equiv a \bmod r$, then b does not satisfy (8.3). This is a contradiction and therefore either $a^* \equiv 1 \bmod n$ for every a, or the assumption that (8.3) holds for every $a \in (\mathbf{Z}/n\mathbf{Z})^*$ must be false. But the a satisfying (8.3) form a subgroup H of $(\mathbf{Z}/n\mathbf{Z})^*$, which must in this case have index at least 2.

The first assertion follows from this and from Euler's criterion (8.1).

In particular, a composite integer is an Euler pseudoprime to at most half of all bases coprime to it. Namely, either (8.6) applies, or (8.5) holds; in the latter case, n is a Carmichael number, therefore squarefree, and the Jacobi symbol forms a non-trivial quadratic character on $(\mathbf{Z}/n\mathbf{Z})^*$, which assumes the values 1 and -1 equally often.

Thus, if an odd n is composite, we have a probability exceeding $1/2$ that a randomly chosen a will enable us to prove that n is composite, by checking (8.2). This will give rise to a compositeness test, an idea we will discuss below.

If n is composite, and (8.3) holds for some a coprime to n, then (8.2) may or may not hold for every such a; this depends on the number of times that 2 divides $n-1$ and each of the $p-1$, for the prime divisors p of n. This is closely related to an idea that was first used by Miller, which we describe next (see [102]).

Again n will be an odd integer. Write $n - 1 = r2^k$, with r odd. If n is prime then $a^{n-1} \equiv 1 \bmod n$. Therefore either $a^r \equiv 1 \bmod n$, or $a^{r2^j} \equiv 1 \bmod n$ for some j with $0 < j \leq k$. If, in the latter case, we take j minimal then $a^{r2^{j-1}} \equiv -1 \bmod n$ since $\mathbf{Z}/n\mathbf{Z}$ is a field.

If we find $a \in \{1, 2, \ldots, n-1\}$ for which the above does not hold, n must be composite. We will call a non-zero element $a \in \mathbf{Z}/n\mathbf{Z}$ a *witness* to the compositeness of $n = r2^k$ if both

$$(8.7) \qquad a^r - 1 \neq 0 \quad \text{and} \quad a^{r2^i} + 1 \neq 0 \text{ for } i = 0, 1, \ldots, k-1.$$

If n is composite but a is a non-witness, then n is also called a *strong pseudoprime to the base a*. Selfridge and others (see [122]) proved the following connection between strong and Euler pseudoprimes.

(8.8) Theorem. *Let n be an odd composite number. If a is a non-witness for n, then n is an Euler pseudoprime to the base a.*

The following theorem is due to Rabin (cf. [128]). We present a proof of this in section II.1. In combination with the observation that testing whether or not a is a witness is cheaper than testing (8.2), Theorems (8.8) and (8.9) show that the concept of witnesses is an improvement over the Euler-pseudoprimes.

(8.9) Theorem. *Let n be an odd composite number. Then at least $\frac{3}{4}(n-1)$ of all a with $1 \leq a < n$ are witness to the compositeness of n.*

The first application of (8.9) is that of the cheap compositeness test, which we announced in Section 1.

(8.10) Compositeness test. *Let $n > 3$ be odd, and $n - 1 = r2^k$, with r odd. Choose $a \in \{2, 3, \ldots, n-2\}$ at random, and compute $b = a^r \bmod n$. If $b \not\equiv \pm 1 \bmod n$, compute $b^2, b^4, b^8 \ldots \bmod n$ until either $b^{2^i} \equiv -1 \bmod n$ or $i = k - 1$.*

The number n is said to *pass the test* if (8.7) holds. If n passes the test, n is composite, and a is a witness for that fact. If n does not pass ("fails") the test, we may repeat it with another choice of a; from (8.8) we see that the probability that a composite n does not pass the test for c independent choices of a is at most 4^{-c}. Therefore we may be reasonably confident that n is prime if it fails this test a few times.

In practice this distinguishes composites from the probable primes.

(8.11) Remarks. Solovay and Strassen (see [147]) proposed a compositeness test based on (8.2) rather than (8.7), and Lehmann [68] suggested to use (8.3). See also [105].

Lehmann also proposes a test that is in our language neither a compositeness test nor a primality test: calculate $a^* = a^{\frac{n-1}{2}} \bmod n$ for c random choices of a. If for all of these $a^* \equiv \pm 1$ and for at least one $a^* \equiv -1$, declare n prime; declare n composite otherwise. In both cases the answer may be wrong, and in both cases the probability that this happens is at most 2^{-c}.

The second application concerns so-called conditional primality tests. These supply sufficient conditions for primality provided that certain unproved (but generally acknowledged to be likely to hold) hypotheses are true, notably certain generalized Riemann hypotheses. These assert that the non-trivial zeroes in the complex numbers of L-series associated to certain characters modulo n all have real part $1/2$. See [102] and Lenstra in [91] for a proper definition of these notions in our cases, and [85] for a small set of characters that will suffice here.

(8.12) Theorem. *Under the assumption of certain generalized Riemann hypotheses there exists an absolute and effectively computable constant C such that an element of $(\mathbf{Z}/n\mathbf{Z})^*$ outside a given proper subgroup G exists smaller than $C(\log n)^2$.*

Proofs can be found in [4] (for the case of index 2) and [106]. In our application we take a subgroup of $(\mathbf{Z}/n\mathbf{Z})^*$ containing all non-witnesses. In fact one may take the subgroup consisting of all a satisfying (8.2). The constant $C = 2$ in the resulting Theorem (8.13) is due to Bach [7]. The logarithm is the natural logarithm.

(8.13) Theorem. *Let n be an odd composite number. Under the assumption of certain generalized Riemann hypotheses there exists a witness for the compositeness of n smaller than $2(\log n)^2$.*

As a final application of the above ideas, one may list all pseudoprimes of a particular kind in a certain range; if n is in that range, it may be subjected to the corresponding test and if it passes that, it will be prime unless it equals one of the pseudoprimes in the list. Here is a popular example, from [122]; another example may be found in [65].

(8.14) Theorem. *Let $n > 1$ be odd and $n < 25 \cdot 10^9$. If n does not have a witness among $2, 3, 5, 7$ it is prime, unless $n = 3215031751 = 151 \cdot 751 \cdot 28351$.*

9. THE GAUSS SUM TEST.

The first major improvement on all methods described so far, came with Adleman and Rumely's introduction of the Gauss sum test (see [1], [2]). It led to the first primality test that does not require the factorization of integers of about the same size as n itself, and the first test with a sub-exponential bound on the running time on every input.

In this section we give a short outline of the deterministic version of the Gauss sum test, following the presentation in [86] and [87] rather than that of [2]. A probabilistic version was also given in [2]; major improvements to make it practical can be found in [29] and [30], and the following chapters will describe all of those and many more in detail. We will also describe the important possibility to combine the improved versions of the Gauss sum test with Lucas-Lehmer type tests. All of the notions necessary to describe the Gauss sum test will recur in Chapter II. Therefore we do not attempt to make this section as self-contained as the previous ones.

The algorithm we will describe leads to the following result, quoted before in (2.8).

(9.1) Theorem. *There exists an algorithm that, for prime n, proves the primality of n in $O((\log n)^{C \log\log\log n})$ bit operations, for some effectively computable constant C.*

Four stages can be distinguished in the algorithm. We will describe each of these, with a brief theoretical justification.

(9.2) First step of the algorithm. *Select the auxiliary integers s and t as follows.*

Let t be the smallest positive, squarefree integer such that $s = s(t)$ exceeds $\sqrt{n}$, where

$$s = \prod_{\substack{q \text{ prime} \\ q-1 \mid t}} q. \tag{9.3}$$

Finding s and t may be done by trying $t = 1, 2, \ldots$ in succession; it should also be checked that $\gcd(st, n) = 1$.

For the second and third step we have to introduce Gauss sums, and their multiplicative and additive properties.

(9.4) Multiplicative properties of Gauss sums. Let $R_{p,q}$ for different primes p and q be the ring $(\mathbf{Z}/n\mathbf{Z})[\zeta_p, \zeta_q]$, obtained by formal adjunction of the zeroes ζ_p and ζ_q of the p-th and q-th cyclotomic polynomial.

We will consider $R_{p,q}$ for every pair (p,q) of primes for which q divides s and p divides $q-1$ and hence t, with s and t as in (9.2).

Let χ be a character of conductor q and order p with values in $R_{p,q}$, that is, a multiplicative homomorphism $\chi\colon (\mathbf{Z}/q\mathbf{Z})^* \to \langle\zeta_p\rangle$ that is surjective. This can be constructed by finding a primitive root g modulo q and by putting $\chi(g) = \zeta_p$.

Furthermore, let $\tau(\chi)$ be the Gauss sum

$$\tau(\chi) = \sum_{x=1}^{q-1} \chi(x)\zeta_q^x \in R_{p,q}.$$

Write $n^{p-1} - 1 = p^h u_p$, where $p \nmid u_p$.

The following lemma expresses a multiplicative property of Gauss sums that holds if n is prime. It justifies the second step of the algorithm.

(9.5) Lemma. *If n is prime, then, with notation as in (9.4):*

$$\tau(\chi)^{n^{p-1}-1} = \chi(n) \in \langle\zeta_p\rangle.$$

(9.6) Second step of the algorithm. *For every pair of primes (p,q) for which q divides s and p divides $q-1$, do the following. Determine $w(\chi)$, the smallest $i \in \{1,2,\ldots,h\}$ for which*

$$\tau(\chi)^{p^i u_p} \in \langle\zeta_p\rangle;$$

also, check that

$$\tau(\chi)^{n^{p-1}-1} \in \langle\zeta_p\rangle; \tag{9.7}$$

otherwise n is declared composite.

(9.8) Additive properties of Gauss sums. In the next step of the algorithm an additive property of Gauss sums will be checked. We introduce the following notation.

Let $w_p = \max(w(\chi))$, the maximum being taken over all characters in (9.4) of order p. Let $m_p = p^{w_p} u_p$, with u_p as in (9.4). Let q_p be such, that the character χ of conductor q_p and order p satisfies $w(\chi) = w_p$.

(9.9) Third step of the algorithm. *For every prime p dividing t verify that:*

(9.10) *one of the following holds:*

(i) $w_p = 1$;

(ii) $\tau(\chi)^{p^{w(\chi)}u_p} \neq 1$;

(iii) $\mathrm{ord}(\tau(\chi)^{p^{w(\chi)-1}u_p} - \zeta) = n$ *for all* $\zeta \in \langle \zeta_p \rangle$,

where ord *denotes the additive order in the ring* R_{p,q_p} *of* (9.4). *If none of* (i)–(iii) *hold for some* p, *declare* n *composite.*

Verifying the property of the additive order in (iii) can simply be done by checking that one of the coordinates of the element, on a basis over $\mathbf{Z}/n\mathbf{Z}$, is coprime to n.

It is easy to see that (9.10) will hold when n is prime. Conversely, using (9.5), one can prove the following.

(9.11) Lemma. *Let notation be as in* (9.8). *If* (9.10) *holds, then*

$$r^{p-1} \equiv 1 \bmod p^{w_p} \text{ for every divisor } r \text{ of } n.$$

(9.12) Fourth step of the algorithm. *Find the (unique) element* $z \in (\mathbf{Z}/s\mathbf{Z})^*$ *satisfying*

$$\chi(z) = \tau(\chi)^{m_p} \text{ for every pair } (p,q) \text{ as before.}$$

Let $f \leq t-1$ *be the order of* z *in* $(\mathbf{Z}/s\mathbf{Z})^*$. *Define for* $1 \leq i \leq f$ *the integer* r_i *by*

$$r_i \equiv z^i \bmod s \text{ and } 0 < r_i < s.$$

Check for all $i < f$ *for which* $r_i \leq \sqrt{n}$ *that* r_i *does not divide* n; *if this does not hold, declare* n *composite.*

If n *has not been declared composite before, it is now declared prime.*

The correctness of the algorithm is a consequence of the following.

(9.13) Proposition. *Let notation be as before. If* (9.10) *holds for every prime* p *dividing* t, *then for every pair* (p,q) *as in* (9.6)

$$(\tau(\chi)^{m_p})^{l_p(r)} = \chi(r) \text{ for every divisor } r \text{ of } n,$$

with $l_p(r)$ defined modulo p by:

$$l_p(r) \equiv \frac{r^{p-1} - 1}{m_p} \bmod p.$$

To see that this implies the correctness of the remaining part of the algorithm, let r be a divisor of n. Once (9.10) has been checked for all p, we find a unique $l(r)$ modulo t by (9.13), with the property that for every character χ modulo s:

$$\chi(r) = \chi(z^{l(r)})$$

with z as in (9.12). But then $r \equiv z^{l(r)} \bmod s$, while $s > \sqrt{n}$ and therefore the only possible divisors of n are to be found among the powers of z modulo s. These are checked in (9.12).

The running time analysis given in (9.1) is based on the following result (cf [2], [30]), together with the observation that all steps of the Gauss sum test as described can be done in time polynomial in t and $\log n$. It also proves that the bound given is essentially best possible.

(9.14) Theorem. *There exist effectively computable positive constants C_1 and C_2 such that the smallest t for which s as in (9.3) exceeds $\sqrt{n}$, satisfies:*

$$(\log n)^{C_1 \log\log\log n} \leq t \leq (\log n)^{C_2 \log\log\log n}.$$

10. ABELIAN VARIETIES.

In recent years the application of methods from computational algebraic geometry has led to breakthroughs in the area of factoring as well as primality testing. In this section we briefly describe the results with respect to primality testing; a fully self-contained account would take more space than we allow ourselves here. For the use of elliptic curves in factoring the reader should consult [90] and [89]. For general results concerning the arithmetic on elliptic curves we refer to [145].

The use of abelian varieties has brought major improvements both on the theoretical and on the practical side of primality testing. From a practical point of view, the most elementary (non-trivial) abelian varieties, the *elliptic curves* have been the most prolific so far.

Roughly speaking, an elliptic curve and a prime number together determine a finite group; the idea is to use this group instead of $(\mathbf{Z}/n\mathbf{Z})^*$ as in Section 6, for primality proving. Two important features of these groups make this idea work so well. Firstly, the group law is very easy and explicit; secondly, choosing another elliptic curve, for the same prime, will give rise to another group with probably a different order. In the classical case, using the multiplicative group $(\mathbf{Z}/n\mathbf{Z})^*$, success of our primality test depends on the properties of $n-1$; in the case of elliptic curves we get a collection of groups, with orders having a known distribution, from which we can pick one with a favourable order.

Without giving all the details we will describe the basic facts.

(10.1) Elliptic curves. In our applications, we need elliptic curves over certain rings, which means that our definitions will have to be slightly more general than in the classical case of elliptic curves defined over fields.

Two restrictions will be imposed on the rings R. The first is that $6 \in R^*$; this condition is merely put in to allow the use of nice Weierstrass models, and is common in the classical case as well. The second condition is that for every primitive $m \times n$ matrix over R for which all 2×2-subdeterminants are zero, an R-linear combination of the rows must exist that is primitive in R^m. Here a finite set of elements of a ring R is called *primitive* if they generate R as an R-ideal, and a matrix is primitive if the set of entries is.

An *elliptic curve* $E = E_{a,b}$ *over a ring* R may be defined, for our purposes, as a pair $a, b \in R$, for which the discriminant $D = 4a^3 + 27b^2 \in R^*$. The *set of points* $E(R)$ *of* E

over R is the set of projective solutions $(x : y : z)$ to the Weierstrass equation

$$Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{10.2}$$

Here a projective point $(x : y : z)$ is an equivalence class of triples $(0,0,0) \neq (x,y,z) \in R \times R \times R$, under the equivalence

$$(x,y,z) \sim (x',y',z') \iff \exists \lambda \in R: \quad x' = \lambda x,\ y' = \lambda y,\ z' = \lambda z.$$

For the rings satisfying the two conditions above, the set $E(R)$ forms an abelian group. This group is usually written additively, and the zero element is the point $O_E = (0 : 1 : 0)$. Two points can be added easily, using explicit formulas for addition in terms of the coordinates of both points, and a (see [89], [11]).

Thus integer multiples of points are also defined, and can be computed efficiently by repeated doubling and addition. Since the addition formulas do not depend on the ring over which the points are defined, multiplication by a fixed integer m gives in fact rise to an endomorphism of E; if there are other endomorphisms than those obtained from $\mathbf{Z}$, we say that E admits *complex multiplication* (since such endomorphisms are obtained from multiplication by an element from a subring of the ring of integers of some complex quadratic field).

(10.3) Finite ground field. We will be especially interested in elliptic curves over finite fields. Often, these are obtained from reductions of elliptic curves over number fields; more precisely, if $E_{a,b}$ is an elliptic curve over a number field K with coefficients a, b in the ring of integers O_K, then taking (10.2) modulo a prime ideal I of O_K, we arrive at an elliptic curve over O_K/I, a finite field of $\mathrm{N}(I)$ elements, provided that $D = 4a^3 + 27b^2 \not\equiv 0 \bmod I$.

Elliptic curves over finite fields have been studied for a long time. The number of points of such a curve over a finite field is obviously finite, but much more can be said. Hasse proved (see [145, Ch. V]), that for an elliptic curve E over $\mathbf{F}_q$:

$$| \#E(\mathbf{F}_{q^k}) - (q^k + 1) | \leq 2\sqrt{q^k} \tag{10.4}$$

for every $k \geq 1$. Also, for most integers m in the interval around $q^k + 1$ indicated by (10.4), elliptic curves over $\mathbf{F}_{q^k}$ of order m exist (see [158]), and the orders of all curves have a known distribution over this interval (see [142]); all possible group structures have been determined (see [142], [155], [137]).

In 1985, an algorithm was published by Schoof [141], to compute the number of points of an elliptic curve over a finite field $\mathbf{F}_q$, that has an expected running time of $O((\log n)^8)$ bit operations ([141], [89]).

(10.5) Reduction modulo n. If n is prime and E is an elliptic curve defined over $\mathbf{Q}$ such that $a, b \in \mathbf{Z}$, reduction modulo n leads to an elliptic curve over the finite field $\mathbf{Z}/n\mathbf{Z}$. In proving the primality of n, we may not use that n is prime though; that is one reason why elliptic curves over rings are of interest to us.

All primality tests using elliptic curves make use of some variant of the following theorem.

(10.6) Theorem. *Let n be coprime to 6, and let E be an elliptic curve defined over $\mathbf{Z}$. Let m and s be positive integers with $s \mid m$. Suppose that for every prime divisor q of s there exists a point $P \in E(\mathbf{Z}/n\mathbf{Z})$ such that*

$$m \cdot P = (0:1:0) \quad \text{and} \quad \gcd(z, n) = 1, \text{where } (x:y:z) = \frac{m}{q} P \in E(\mathbf{Z}/n\mathbf{Z}). \tag{10.7}$$

Then

$$\#E(\mathbf{Z}/p\mathbf{Z}) \equiv 0 \bmod s \text{ for every prime divisor } p \text{ of } n.$$

If moreover $s > (\sqrt[4]{n} + 1)^2$, then n is prime.

The proof is simple: (10.7) implies that the order of $E(\mathbf{Z}/p\mathbf{Z})$ is divisible by s, and the bound in (10.4) does the rest.

The first application of the theory of elliptic curves to primality testing consisted of analogues of Pocklington's theorem in [11] (see also [26]). We quote an easy example here.

(10.8) Theorem. *Let $n \equiv 1 \bmod 4$ and suppose that $n = \nu\bar{\nu} \in \mathbf{Z}[i]$, where $\bar{}$ denotes complex conjugation in $\mathbf{Z}[i]$. Suppose that $\gcd(\nu, 2 \cdot 3 \cdot 5 \cdot 13 \cdot 17 \cdot 29) = 1$.*

If there exist $\delta \in \mathbf{Z}[i]$ coprime to ν, and for every prime divisor π of $\nu - 1$ in $\mathbf{Z}[i]$ a point $P \in E_{-\delta,0}(\mathbf{Z}[i]/(\nu))$, such that:

$$(\nu - 1) \cdot P = (0:1:0) \text{ and } O \neq \frac{(\nu - 1)}{\pi} P \in E_{-\delta,0}(\mathbf{Z}[i]/(\nu)),$$

then n is prime.

(10.9) Remarks. Actually, (10.8) yields a primality test in $\mathbf{Z}[i]$: it is proved that ν is prime in $\mathbf{Z}[i]$ and hence that n is prime in $\mathbf{Z}$. Use is made of the fact that the curves $E_{-\delta,0}: \ Y^2Z = X^3 - \delta XZ^2$ admit complex multiplication by $\mathbf{Z}[i]$, given by $i \cdot (x : y : z) = (-x : iy : z)$.

This theorem is readily generalized to other complex multiplication rings for which corresponding elliptic curves are known.

Note that we need to be able to factor $\nu - 1$ in $\mathbf{Z}[i]$ to apply (10.8), or equivalently, to factor $n - (\nu + \bar{\nu}) + 1$. The chances that we are able to do this, are independent of those for $n - 1$. Also, ν may be multiplied by a unit, giving four possibilities.

The requirement that ν is coprime to some small odd primes is put in because of the existence of certain small pseudoprimes to this primality test (see [11]). For a more general discussion of elliptic pseudoprimes see [47], [104].

In 1986, the first general purpose primality test based on elliptic curves was proposed by Goldwasser and Kilian [46]. We outline this next.

(10.10) The random curve method.

(i) *Selection of a curve E and a point P.* This is done by repeating the steps (a) and (b), until the following conditions are satisfied: $\gcd(4a^3 + 27b^2, n) = 1$, the integer m satisfies $m = kq$ with $k > 1$ small and q declared probably prime by some compositeness test as in (8.10), and $k \cdot P \neq O$ in $E_{a,b}(\mathbf{Z}/n\mathbf{Z})$.

(a) Choose random $x, y, a \in \mathbf{Z}/n\mathbf{Z}$ and compute $b = y^2 - x^3 - ax \in \mathbf{Z}/n\mathbf{Z}$; let $E = E_{a,b}$ and $P = (x : y : 1)$.

(b) Apply Schoof's method to the set $E_{a,b}(\mathbf{Z}/n\mathbf{Z})$ to determine the integer m; if n is prime, $m = \#E_{a,b}(\mathbf{Z}/n\mathbf{Z})$, but if n is not prime this step need not even terminate.

(iii) *Verification of the order.* Finally it is checked that $m \cdot P = (0 : 1 : 0)$, as should be the case if n is prime.

If all these steps have been performed successfully, n is proven prime if q is, by Theorem (10.6). So we apply this algorithm recursively to q.

There is a heuristic argument why (10.10) should lead to a probabilistic primality test running in expected polynomial time. First of all, if n is prime, the recursion depth is $O(\log n)$ since $q \leq (n + 1 + 2\sqrt{n})/2$ by (10.5). There is only one point in this algorithm

for which it was not already known that it can be done in expected polynomial time, and that concerns the number of times the steps in (i) have to be performed until m has the required form. The probability of m being of the right form, is comparable to that of m being prime, and heuristically this probability should be of order $(\log n)^{-1}$. Goldwasser and Kilian proved the following two theorems, even though they only allowed $k = 2$ in (10.10). Here $\pi(x)$ denotes the number of primes smaller than x.

(10.11) Theorem. *If there exist constants $C_1 > 0$ and C_2 such that*

$$\pi(x + 2\sqrt{x}) - \pi(x) \geq \frac{C_1\sqrt{x}}{(\log x)^{C_2}} \quad \text{for all } x \geq 2, \tag{10.12}$$

then the random curve test supplies a primality proof for prime n in an expected number of $O((\log n)^{9+C})$ bit operations.

The main contribution to the running time in (10.11) is due to Schoof's theorem for computing the order of an elliptic curve over a finite field. Although polynomial, it has been asserted that this algorithm is too slow for practical purposes.

The existence of the constants C_1, C_2 in (10.11) has not been proved; the best result that is available, implies that (10.12) holds on the average, (see [57], [46]).

Yet it has been shown that (10.11) will run in expected polynomial time for almost all n, in the following sense.

(10.13) Theorem. *The random curve method provides primality proofs in an expected number of bit operations that is polynomial in $\log n$ for a fraction of at least*

$$1 - O(2^{-(\log_2 n)^{\frac{1}{\log_2 \log_2 \log_2 n}}})$$

of all primes n.

Atkin has proposed another way of choosing an elliptic curve in (10.10). We will briefly describe this below; his method circumvents the use of Schoof's algorithm, and has led to a practical primality test (see [6], [108]). With this test impressive results have been achieved recently, see e.g. [13]. For a heuristic argument that Atkin's complex multiplication method leads to a probabilistic test running in expected time $O((\log n)^7)$, see [83]. A rigorous analysis of this algorithm has not been given yet.

(10.14) The complex multiplication method. For this algorithm the following procedure to find a curve E and the integer m replaces that of (10.10)(i).

Choose a negative discriminant (that is $\Delta \equiv 1 \bmod 4$ or $\Delta \equiv 8, 12 \bmod 16$ and $\Delta < 0$ not divisible by the square of an odd prime), with $\gcd(n, \Delta) = 1$. Next find out whether or not in O_Δ, the ring of integers of $\mathbf{Q}(\sqrt{\Delta})$, there exists ν such that $n = \nu\bar{\nu}$; there is an efficient way of doing this. Moreover, if such $\nu \in O_\Delta$ exists, and if n is prime, ν can be found efficiently using quadratic forms. Compute $m = (\mu\nu - 1)(\bar{\mu}\bar{\nu} - 1)$ for each of the units μ in O_Δ; notice that there are four (if $\Delta = -4$) or six (if $\Delta = -3$) or two (for other Δ) such units. If one of these m is of the required form, $m = kq$ with k small and q probably prime, we continue, and if none is as desired we repeat everything for another choice of Δ.

Once a proper m is found, construct an elliptic curve admitting complex multiplication by O_Δ. This involves finding a root of a polynomial of degree h_Δ over $\mathbf{Z}/n\mathbf{Z}$. For details, see [83], [108], and also [61].

For theoretical purposes, Adleman and Huang have improved upon the random curve method in another direction. Instead of looking at elliptic curves, they consider more general abelian varieties; notably, they utilize abelian varieties obtained from hyper-elliptic curves, defined in general by $Y^2 = f(X)$, where f is a squarefree monic polynomial of degree $2g+1$ over some field K; for $g = 1$ we get the ordinary elliptic curve. The Jacobian of such a hyper-elliptic curve is an abelian variety of dimension g over K; elliptic curves form their own Jacobians. Again, the set of points of such a curve over an extension field of the field of definition is the set of solutions to the equation $Y^2 = f(X)$, replacing (10.2), and the set of points of the Jacobian of the curve forms an abelian group. Over a finite field, its number of elements is bounded by

$$\#J(\mathbf{F}_q) = p^g + O(\sqrt{4p}^{2g-1}). \tag{10.15}$$

In the abelian variety method one uses Jacobians of dimension $g = 2$; again one chooses random varieties, until the number m, that is the order $\#J(\mathbf{Z}/n\mathbf{Z})$ if n is prime, is of the form $m = kq$. If $g = 2$ however, the interval given by (10.15) has length $O(n^{3/4})$, and it has been proven that this contains sufficiently many primes to provide polynomial bounds. Note that (10.15) shows that we reduce the proof of primality of n in this way to that of an integer of roughly size n^2; Adleman and Huang prove that after a few of these steps in the wrong direction, one expects to hit a prime for which a proof can be given in

time polynomial in $\log n$ by the random curve test (10.10). One arrives at the following result, quoted before in (2.8).

(10.16) Theorem. *There exists a positive integer k such that for every prime n the abelian variety method gives a primality proof for n in expected time $O((\log n)^k)$.*

11. MISCELLANEOUS RESULTS.

We start this section with some results concerning short proofs for primality. The first one was already mentioned in Section 2, and states that every prime admits a short proof.

(11.1) Theorem. *If n is prime, this can be proved by $O((\log n)^4)$ bit operations.*

Theorem (11.1) is due to Pratt, see [125]. As we mentioned in Section 2, the exponent 4 can be improved upon by using elliptic curves, see [121], and by using fast multiplication techniques (see also [116]. For some examples see [12].

Here we are mainly interested in the concept of *short certificates*, that is, primality proofs that can be verified in polynomial time.

(11.2) Short certificates. Pratt's certificate for primality of n consists of a tree T, where the nodes are integer triples, that can be constructed as follows.

First of all, T contains a *root* $(n, 1, 0)$.

Then we add a node (n, p, a), one for every prime divisor p of $n-1$, with a such that

$$a^{\frac{n-1}{p}} \not\equiv 1 \bmod n \quad \text{and} \quad a^{n-1} \equiv 1 \bmod n;$$

and we add edges between these nodes and the root. Notice that different nodes may already consist of identical triples, if $n-1$ contains multiple prime factors.

Next, for every odd prime divisor p of $n-1$, we repeat the construction, that is, to (n, p, a) we attach nodes (p, p', b), where the primes p' satisfy $\prod p' = p - 1$, and where

$$b^{\frac{p-1}{p'}} \not\equiv 1 \bmod p \quad \text{and} \quad b^{p-1} \equiv 1 \bmod p; \tag{11.3}$$

moreover, we connect a *leaf* $(2, 1, 0)$ to $(n, 2, a)$.

This process is repeated for every odd prime that is encountered; since the primes decrease, this terminates after finitely many steps. All leaves will be triples $(2, 1, 0)$.

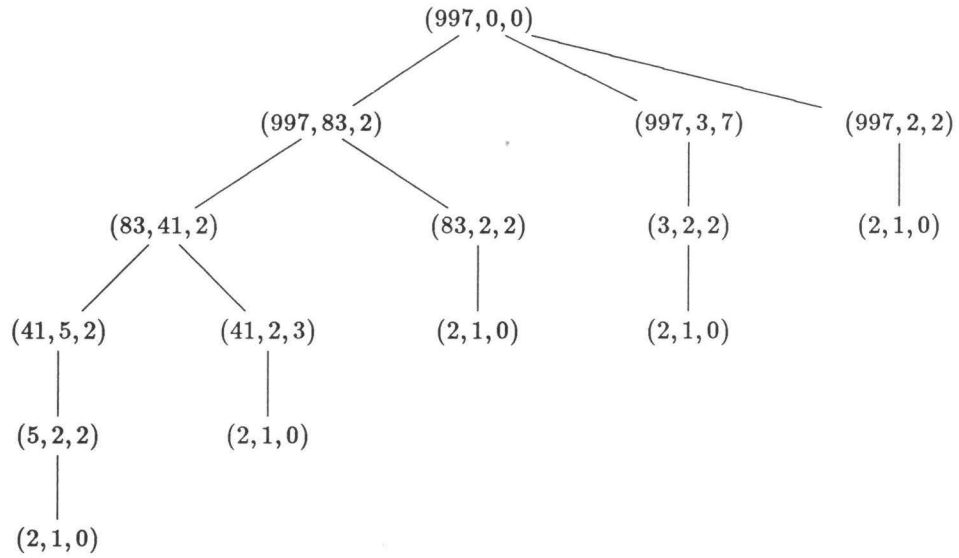
Verifying the primality proof encoded by T means checking that (11.3) holds for every triple (p, p', b) in T that forms a node other than the root or a leaf. That this suffices to prove primality is an immediate consequence of Theorem (6.6).

Every check of (11.3) involves at most $\log_2 p$ multiplications modulo p. Theorem (11.1) can now simply be proved by showing that T has at most $2\log_2 n - 1$ nodes. That can be done by induction; it is certainly true for $n = 2$ and $n = 3$, when the number of

nodes is 1, respectively 2. Suppose it has been proved for all primes smaller than n; let $n-1 = p_1 p_2 \cdots p_k$, with p_i prime. Then $k \geq 2$, and for the number of nodes we have:

$$\#T \leq 1 + \sum_{i=1}^{k} (2\log_2 p_i - 1) = 1 + 2\log_2(n-1) - k < 2\log_2 n - 1.$$

(11.4) Example. The proof tree for the largest prime smaller than 1000 looks like this.



The second result we wish to mention here, concerns the existence of an infinitude of primes for which short proofs can be found quickly.

One has to keep in mind that Theorem (11.1) asserts that short primality proofs exist for every prime, but it does not tell you how to find them. On the other hand, we have seen in Sections 6 and 7 that for certain infinite sets of integers, like the Fermat and the Mersenne numbers, it is possible to decide efficiently whether an element is prime or not; however, it is not known whether any of these sets contain infinitely many primes.

The result from [115] we quote below postulates the existence of a set containing infinitely many primes, such that a given element n of the set can be tested for primality in time polynomial in $\log n$.

(11.5) Theorem. *Let C be some suitably chosen absolute constant. Let N consist of the positive integers n with*

(11.6) $\qquad n \equiv 1 \bmod 3^k \quad \textit{and} \quad 3^{3(k-1)} < n < 3^{3k} \quad \textit{for some } k,$

that satisfy the following property: if n is prime, there exists a with $1 < a \leq C(\log n)^6$ such that

$$\gcd(a^{\frac{n-1}{3}} - 1, n) = 1 \quad \textit{and} \quad a^{n-1} \equiv 1 \bmod n. \tag{11.7}$$

If $n \in N$ and n is prime, this can be proved in $O((\log n)^9)$ bit operations. Moreover, N contains infinitely many primes.

The first assertion is easily proved: let $n \in N$, then to prove that it is prime, we check (11.7) for $a \leq C(\log n)^6$. For every a this involves about $\log n$ multiplications modulo n, so this can be done in $O((\log n)^9)$. If (11.7) does not hold for any such a, then n is composite by the assumptions on N. If we find a such that (11.7) holds, then every divisor r of n satisfies $r \equiv 1 \bmod 3^k$, with k as in (11.6), by Pocklington's theorem (6.14). But since $n < 3^{3k}$, it can have at most two such divisors r, say $x3^k+1$ and $y3^k+1$, with $1 \leq xy < 3^k$; then $n = xy3^{2k} + (x+y)3^k + 1$. So, writing $n = A3^{2k} + B3^k + 1$, we conclude that n is now prime precisely when $B^2 - 4A$ is not an integral square.

The interesting part of Theorem (11.5) is the final assertion. Using results from analytic number theory, a much stronger statement is proved in [115], namely that

$$\{n \in N\colon\ n \leq x,\ n \text{ prime}\} > \frac{cx^{\frac{2}{3}}}{\log x},$$

for every x exceeding some x_0, and for some absolute constant $c > 0$.

The final result we mention in this chapter concerns an efficient algorithm of H.W. Lenstra, Jr., to find all divisors in a given residue class for a sufficiently large modulus. The following theorem is contained in [88].

(11.8) Theorem. *Let $C > 0$ and $\alpha \geq \frac{1}{3}$ be constants, and let r, s, n be integers satisfying:*

$$0 \leq r < s < n, \quad s > Cn^{\alpha}, \quad \gcd(r, s) = 1.$$

Then the number of divisors of n that are congruent to r mod s is bounded by a constant that only depends on C and α, and there exists an algorithm that finds all of these in time polynomial in $\log n$.

We will be more explicit in Section II.9, since this algorithm is used in the final stage of the algorithm of Chapter IV. For $C = 1$ and $\alpha = \frac{1}{3}$, it is shown in [88] that there exist

at most 11 divisors in a given residue class modulo $s > \sqrt[3]{n}$, and that the algorithm for finding them takes $O((\log n)^3)$ bit operations. The running time does not depend on α, and is linear in $1/C$, which leads to a general bound of $O(\frac{1}{C}(\log n)^3)$ bit operations.

The use of Theorem (11.8) for primality testing will be clear: methods based on Pocklington's theorem, or on the Gauss sum method as in Section 9, restrict possible divisors of n to certain residue classes modulo an auxiliary number s. Classically, one is able to finish the primality proof quickly if $s > \sqrt{n}$ since any composite n will have at least one divisor smaller than $\sqrt{n}$; but using (11.8) the same is true if only $s > \sqrt[3]{n}$.

(11.9) Example. Theorem (11.8) was successfully applied in primality testing for the first time for the primality proof of

$$n = \frac{10^{1031} - 1}{9},$$

the number consisting of 1031 decimal digits 1. As reported in [38], an enormous effort produced a completely factored part s of $n^{12} - 1$ larger than the cube root of n, but much smaller than its square root. Using the Lucas-Lehmer methods, briefly mentioned at the end of Section 7, one finds the residue classes modulo s in which the divisors of n must lie. Running the algorithm in (11.8) took only a fraction of the time spent on finding factors of $n^{12} - 1$.

The numbers $(10^k - 1)/9$ are the analogue to the Mersenne numbers in base 10. It is now known that for $k \leq 10000$ (see [38]) there are only 5 primes, namely for $k = 2$, 19, 23 (see (4.4)), 317 (see [165]), and 1031.

II. THEORY.

1. COMPOSITENESS TESTING.

In this chapter we give the mathematical background for the primality testing algorithm that is described in detail in Chapter IV. As we explained in the previous chapter, this algorithm should be thought of as a *primality prover*. It provides rigorous proofs of primality for prime numbers, and one would rather not spend any time on looking for such a proof if the number is composite. Therefore one subjects the integer that is to be tested to a few preliminary tests that will sort out the vast majority of composites. First one performs some trial divisions by small primes and next one can apply a compositeness test based on the theorem below.

(1.1) Lemma. *Let $n \in \mathbf{Z}_{\geq 2}$ and $n-1 = r2^k$, with r odd. If n is prime then for every element $a \in \{1, 2, \ldots, n-1\}$:*

$$a^r \equiv 1 \bmod n \quad \text{or} \quad a^{r2^i} \equiv -1 \bmod n \text{ for some } i \text{ with } 0 \leq i < k. \tag{1.2}$$

Proof. Since n is prime, we know from Fermat's little theorem that $a^{n-1} \equiv 1 \bmod n$. Then either $a^r \equiv 1 \bmod n$, or $a^{r2^j} \equiv 1 \bmod n$ for some j with $0 < j \leq k$. If, in the latter case, we take j minimal, then $a^{r2^{j-1}} \equiv -1 \bmod n$ since $\mathbf{Z}/n\mathbf{Z}$ is a field. That proves (1.1).

This means that if we find $a \in \{1, 2, \ldots, n-1\}$ for which (1.2) does not hold, n must be composite. We will call a non-zero element $a \in \mathbf{Z}/n\mathbf{Z}$ a *witness* to the compositeness of $n = r2^k + 1$ if both $a^r - 1 \neq 0$ and $a^{r2^i} + 1 \neq 0$ for $i = 0, 1, \ldots, k-1$. The following theorem, due to Rabin (cf. [128]), shows that witnesses for composite numbers are abundant.

(1.3) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$ be an odd composite number; write $n - 1 = r2^k$, with r odd. Then at least $\frac{3}{4}(n-1)$ elements a of $\{1, 2, \ldots, n-1\}$ satisfy:*

$$a^r \not\equiv 1 \bmod n \tag{1.4}$$

and

$$a^{r2^i} \not\equiv -1 \bmod n, \qquad \text{for} \quad i = 0, 1, \ldots, k-1. \tag{1.5}$$

Proof. Define j by

$$j = \max\{i \in \mathbf{Z}_{\geq 0}\colon\ b^{2^i} \equiv -1 \bmod n \text{ for some } b \in \mathbf{Z}\}.$$

Notice that every divisor of n, in particular n itself, must be 1 mod 2^{j+1}. Let $m = r2^j$; then $2m$ divides $n-1$.

Suppose that a is a non-witness for n. Then either $a^r = 1$ or $a^{r2^i} = -1$ for some i with $i \leq j$ by definition of j. In both cases $a^m = \pm 1$, which implies that all non-witnesses are contained in the subgroup

$$\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m = \pm 1\}$$

of $(\mathbf{Z}/n\mathbf{Z})^*$. Note that in particular every non-unit of $\mathbf{Z}/n\mathbf{Z}$ is a witness.

Write $(\mathbf{Z}/n\mathbf{Z})^* = \prod_q (\mathbf{Z}/q\mathbf{Z})^*$ where the product ranges over the set S of prime powers q exactly dividing n. Let J be the subgroup $J = \prod_q \langle -1 \rangle$ of $(\mathbf{Z}/n\mathbf{Z})^*$. Since every q satisfies $q \equiv 1 \bmod 2^{j+1}$, it follows that every element of J is a 2^j-th power, and therefore also an m-th power. Hence $\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m = 1\}$ has index $2^{\#S}$ in $\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m \in J\}$. But $\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m = 1\}$ is a subgroup of index 2 in $\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m = \pm 1\}$ and $\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m \in J\}$ is contained in $\{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^{n-1} = 1\}$, so

$$\begin{aligned}[(\mathbf{Z}/n\mathbf{Z})^* \ :\ \{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^m = \pm 1\}] \geq \\ \geq 2^{\#S-1}[(\mathbf{Z}/n\mathbf{Z})^* \ :\ \{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^{n-1} = 1\}].\end{aligned}$$

If $\#S \geq 3$ this implies that the subgroup in which the non-witnesses are contained has at least index 4 in $(\mathbf{Z}/n\mathbf{Z})^*$, which proves the result in that case. If $\#S = 2$, then n can not be a Carmichael number by I.(6.5); in that case $[(\mathbf{Z}/n\mathbf{Z})^* : \{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon\ a^{n-1} = 1\}] \geq 2$ by Theorem I.(8.4) and so again the result follows. Finally, if $\#S = 1$, then $n = q = p^l$ for some odd prime p and some $l \in \mathbf{Z}_{\geq 2}$. Now $(\mathbf{Z}/n\mathbf{Z})^*$ is cyclic of order $(p-1)p^{l-1}$; since p and $n-1$ are coprime, $b^{n-1} = 1$ implies that b is a p^{l-1}-th power, so $[(\mathbf{Z}/n\mathbf{Z})^* : \{a \in (\mathbf{Z}/n\mathbf{Z})^*\colon a^{n-1} = 1\}] \geq p^{l-1}$. But p^{l-1} is at least 4, unless $n = 9$. For $n = 9$ however, only cubes b satisfy $b^8 = 1$, so there are $6/3 = 2$ non-witnesses, and $6 = 3(n-1)/4$ witnesses.

This proves (1.3).

(1.4) Compositeness test. *Choose $a \in \{2, 3, \ldots, n-2\}$ at random, and compute $b = a^r \bmod n$. If $b \not\equiv \pm 1 \bmod n$, compute $b^2, b^4, b^8 \ldots \bmod n$ until either $b^{2^i} \equiv -1 \bmod n$ or $i = k-1$.*

The number n is said to *pass the test* if $b \not\equiv \pm 1 \bmod n$ and $b^{2^i} \not\equiv -1 \bmod n$ for $1 \leq i < k$. If n passes the test, n must be composite by (1.1). If n does not pass the test we may repeat it with another choice of a; from (1.3) we see that the probability that n will not pass the test for c independent choices of a is at most 4^{-c}. Therefore we may be reasonably confident that n is prime if it does not pass this test a few times and in that case we will subject it to our primality prover.

2. CYCLOTOMIC CONSTELLATIONS.

Before giving the first theorem we will indicate the kind of properties of prime numbers we will try to use. For the basic properties of cyclotomic fields and cyclotomic polynomials, we refer the reader to [56].

Suppose that K is a finite Galois extension of $\mathbf{Q}$ with group $\mathrm{Gal}(K/\mathbf{Q})$, of order u. Suppose that a cyclotomic field $\mathbf{Q}(\zeta_m)$ exists such that $\mathbf{Q} \subset K \subset \mathbf{Q}(\zeta_m)$, where ζ_m is a primitive m-th root of unity. It is well known that $\mathrm{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q})$ is canonically isomorphic to $(\mathbf{Z}/m\mathbf{Z})^*$ under the map

$$\mathrm{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q}) \to (\mathbf{Z}/m\mathbf{Z})^*$$

given by

$$\sigma \mapsto b, \quad \text{where } b \text{ is such that } \sigma(\zeta_m) = \zeta_m^b.$$

For n with $\gcd(m,n) = 1$ the *Artin symbol* ϕ_n of $K \supset \mathbf{Q}$ is the element of $\mathrm{Gal}(K/\mathbf{Q})$ obtained by restriction of the inverse image of n mod m under this map; that is, ϕ_n is the restriction to K of the automorphism on $\mathbf{Q}(\zeta_m)$ that sends ζ_m to ζ_m^n. Notice that ϕ_n acts on O_K, the ring of integers of K, and since the ideal nO_K is invariant this induces an action on O_K/nO_K. Below we give three properties involving the Artin symbol ϕ_n, the residue class ring O_K/nO_K and cyclotomic polynomials Φ_t, with $t \in \mathbf{Z}_{\geq 1}$, that hold if n is prime; Theorem (2.8) may be seen as a partial converse: what can be said about n if some of these properties hold for some K?

First we need a lemma on roots of unity. By $\mathrm{ord}\, g$ we will abbreviate the order of an element g in a finite group G.

(2.1) Lemma. *Let R be a commutative ring with 1, let $t \in \mathbf{Z}_{\geq 1}$, and let $\zeta \in R$ satisfy $\Phi_t(\zeta) = 0$. Then $\zeta \in R^*$. If $t \cdot 1 \neq 0$ in R then $\mathrm{ord}\,\zeta = t$ in R^*. If $t \cdot 1 \in R^*$ then $\zeta^i - \zeta^j \in R^*$, for every $i, j \in \mathbf{Z}$ with $i \not\equiv j \bmod t$.*

Proof. In $\mathbf{Z}[X]$

$$X^t - 1 = \prod_{k|t} \Phi_k.$$

Since $\Phi_t(\zeta) = 0$, this implies that $\zeta^t = 1$ so $\zeta \in R^*$ and $d = \mathrm{ord}\,\zeta$ divides t.

Assume that $t \cdot 1 \neq 0$ in R; if $d < t$, then

$$X^t - 1 = \Phi_t \cdot \Big(\prod_{k|d} \Phi_k\Big) \cdot G = \Phi_t \cdot (X^d - 1) \cdot G$$

for some $G \in \mathbf{Z}[X]$. Now

$$\Phi_t \cdot G = \frac{X^t - 1}{X^d - 1} = 1 + X^d + \ldots + X^{(\frac{t}{d}-1)d}$$

and on substituting ζ we see $0 = t/d$ in R. This contradicts our assumption, and therefore $d = t$.

Now assume that $t \cdot 1 \in R^*$. Suppose that $i \not\equiv j \bmod t$; without loss of generality we assume that $i > j$. Since $\zeta \in R^*$, multiplication by ζ^{-j} shows that the final statement of the lemma is equivalent to the assertion $\zeta^k - 1 \in R^*$, for every $k \not\equiv 0 \bmod t$. Fix such k, and let the ring S be $S = R/(\zeta^k - 1)R$. The image of ζ in S still satisfies $\Phi_t(\zeta) = 0$, while the order of ζ in S is clearly smaller than t; that contradicts the previous part of this lemma, unless $t \cdot 1 = 0$ in S. But $t \cdot 1$ is also a unit in S since it is a unit in R by assumption. Therefore S must be the zero ring: $\zeta^k - 1$ is a unit in R.

That proves (2.1).

(2.2) Remarks. Note that it is not always true that an element of order t is a zero of Φ_t: for any $t \in \mathbf{Z}_{\geq 2}$, the element $\zeta = Y$ in the ring $R = \mathbf{Q}[Y]/(Y^t - 1)\mathbf{Q}[Y]$ has order t, but $\Phi_t(Y) \neq 0$ in R.

In the remainder of this chapter, we will also use several times that if h divides k, we may view $R[\zeta_h]$ as a subring of $R[\zeta_k]$, for any commutative ring R with 1. Here we define $R[\zeta_k]$ to be the ring $R[X]/\Phi_k$, in which the element ζ_k is the image of X. Let f be the minimal polynomial of ζ_k over $\mathbf{Q}(\zeta_h)$; this will be a polynomial of degree $d = \phi(k)/\phi(h)$ with coefficients in the ring of integers $\mathbf{Z}[\zeta_h]$. Therefore, $\mathbf{Z}[\zeta_k] = \mathbf{Z}[\zeta_h]/f\mathbf{Z}[\zeta_h]$. As additive groups, we therefore have

$$\mathbf{Z}[\zeta_k] \cong \oplus_{i=0}^{d} \mathbf{Z}[\zeta_h] X^i;$$

in particular, $\mathbf{Z}[\zeta_h]$ is a direct summand of $\mathbf{Z}[\zeta_k]$. This leads not only to an injection of $\mathbf{Z}[\zeta_h]$ into $\mathbf{Z}[\zeta_k]$, but, by taking tensor products with R over $\mathbf{Z}$, indeed to an injection of $R[\zeta_h]$ into $R[\zeta_k]$ for any R.

Note that an injection $\mathbf{Z}[\alpha] \hookrightarrow \mathbf{Z}[\beta]$ does not in general lead to $R[\alpha] \hookrightarrow R[\beta]$ for every R, as the following exaple shows. Take $\alpha = \sqrt{50}$ and $\beta = \zeta_8$; since $\sqrt{2} \in \mathbf{Z}[\zeta_8]$ we have

$\mathbf{Z}[\sqrt{50}] \hookrightarrow \mathbf{Z}[\zeta_8]$. But if we take $R = \mathbf{Z}/5\mathbf{Z}$, then $R[\sqrt{50}] \cong R[X]/X^2$ does not inject into $R[\zeta_8]$ (which does not have nilpotents).

(2.3) Lemma. *Let $\mathbf{Q} \subset K \subset \mathbf{Q}(\zeta_m)$ be an intermediate field. If n is prime then:*

$$\phi_n(\zeta) = \zeta^n \qquad \text{for every} \quad \zeta \in O_K/nO_K.$$

Proof. By Fermat's little theorem and the action of ϕ_n on ζ_m we see that $\phi_n(\alpha) \equiv \alpha^n \bmod n$ for every $\alpha \in \mathbf{Z}[\zeta_m]$, the ring of integers of $\mathbf{Q}(\zeta_m)$. The result follows immediately.

(2.4) Remark. Note that the converse of (2.3) does not hold: for instance, if n is a Carmichael number, then with $K = \mathbf{Q}$ and any m coprime to n, we have that $\phi_n(z) = z \equiv z^n \bmod n$ for every $z \in \mathbf{Z}$.

(2.5) Lemma. *Let $\mathbf{Q} \subset K \subset \mathbf{Q}(\zeta_m)$ be a Galois extension, and let $n \in \mathbf{Z}_{\geq 2}$ be coprime to m. Then:*

$$O_K/nO_K \text{ is a field} \iff n \text{ is prime and } \mathrm{Gal}(K/\mathbf{Q}) = \langle \phi_n \rangle.$$

Proof. Let u denote the degree $[K : \mathbf{Q}]$. Suppose that O_K/nO_K is a field; since $\mathbf{Z}/n\mathbf{Z} \subset O_K/nO_K$ it is clear that n is prime. Then O_K/nO_K is a field of n^u elements and it is well-known that $\mathrm{Gal}(\mathbf{F}_{n^u}/\mathbf{F}_n) = \langle F_n \rangle$, where F_n: $x \mapsto x^n$. Any $\sigma \in \mathrm{Gal}(K/\mathbf{Q})$ induces an $\mathbf{F}_n$-automorphism of $\mathbf{F}_{n^u}$, so we get a homomorphism $\mathrm{Gal}(K/\mathbf{Q}) \to \mathrm{Gal}(\mathbf{F}_{n^u}/\mathbf{F}_n)$ that maps ϕ_n by (2.3) to the generator F_n. Since both Galois groups are of order u this proves the implication $\Rightarrow$.

For the converse, use again that $\phi_n(\alpha) \equiv \alpha^n \bmod nO_K$ for every $\alpha \in O_K$ by (2.3) and so $\phi^i(\alpha) \equiv \alpha^{n^i} \bmod nO_K$. For the norm $\mathrm{N}_{K/\mathbf{Q}}(\alpha) \in \mathbf{Z}$ we have:

$$\mathrm{N}_{K/\mathbf{Q}}(\alpha) \equiv \prod_{i=0}^{u-1} \alpha^{n^i} = \alpha^{1+n+\ldots+n^{u-1}} = \alpha^{\frac{n^u-1}{n-1}} \qquad \bmod nO_K.$$

Since n is prime, $\alpha^{n^u-1} \equiv \mathrm{N}_{K/\mathbf{Q}}(\alpha)^{n-1} \equiv 0$ or $1 \bmod nO_K$. But $\alpha^{n^u-1} \equiv 1 \bmod nO_K$ means that $\alpha \in \left(O_K/nO_K\right)^*$ and $\alpha^{n^u-1} \equiv 0 \bmod nO_K$ implies that $\alpha = \phi^u\alpha \equiv \alpha^{n^u} \equiv 0 \bmod nO_K$. Therefore every non-unit in O_K/nO_K is zero: O_K/nO_K is a field.

This proves (2.5).

(2.6) Lemma. *Let K be a number field of degree $[K : \mathbf{Q}] = u$, and let $n \in \mathbf{Z}_{\geq 2}$. Then O_K/nO_K is a field if and only if:*

$$\text{for every } t \mid n^u - 1: \quad \Phi_t(\zeta) = 0 \text{ for some } \zeta \in \left(O_K/nO_K\right)^*.$$

Proof. If O_K/nO_K is a field, it consists of n^u elements and it is the splitting field over $\mathbf{F}_n$ of $X^{n^u-1} - 1 = \prod \Phi_d$, the product ranging over all divisors of $n^u - 1$. That proves $\Rightarrow$.

For the other implication, apply Lemma (2.1) with $t = n^u - 1$.

That proves (2.6).

Next we investigate what can be said about n if for some t we can find a cyclic field L of degree u (with u such that $t \mid n^u - 1$) and an element ζ in O_L/nO_L such that $\Phi_t(\zeta) = 0$ and $\sigma\zeta = \zeta^n$ for some generator σ of $\mathrm{Gal}(L/\mathbf{Q})$. In Section 4 we will construct number fields L such that ϕ_n generates the Galois group; notice that then, if n is prime the existence of ζ is ensured by (2.3), (2.5) and (2.6).

In the sequel we will often encounter the above situation and therefore we introduce the following abbreviating definition.

(2.7) Definition. Let $n \in \mathbf{Z}_{\geq 2}$ and let $t \in \mathbf{Z}_{\geq 1}$ be coprime to n. A *t-th cyclotomic constellation for n* is a triple L, ζ, σ consisting of a Galois extension $L \supset \mathbf{Q}$ of degree $u = \operatorname{ord} n$, the order of n in $\left(\mathbf{Z}/t\mathbf{Z}\right)^*$, an element ζ in O_L/nO_L and an element $\sigma \in \mathrm{Gal}(L/\mathbf{Q})$, satisfying

$$\Phi_t(\zeta) = 0 \qquad \text{and} \qquad \sigma\zeta = \zeta^n.$$

(2.8) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$, $t \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,t) = 1$ and let u be the order of n in $\left(\mathbf{Z}/t\mathbf{Z}\right)^*$. Suppose that a t-th cyclotomic constellation L, ζ, σ for n exists. Then:*

$$O_L/nO_L = \mathbf{Z}/n\mathbf{Z}[\zeta] \text{ and } \gcd(\Delta_L, n) = 1.$$

Furthermore,

(2.9) *for every $r \mid n$ there exists $i \bmod u$ such that $r \equiv n^i \bmod t$.*

Finally, $L \supset \mathbf{Q}$ *is cyclic and* $\mathrm{Gal}(L/\mathbf{Q}) = \langle\sigma\rangle$, *while*

(2.10) *for every* $r \mid n$: $\phi_r = \sigma^i$ *with* i *as in* (2.9); *in particular* $\phi_n = \sigma$.

Finally, and

Proof. The element σ of $\mathrm{Gal}(L/\mathbf{Q})$ induces an automorphism of O_L/nO_L that we also indicate by σ. Since $\gcd(n,t) = 1$, the element ζ of O_L/nO_L has order t by (2.1). But $\sigma\zeta = \zeta^n$, and therefore the order of σ on O_L/nO_L is at least u, the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. This implies that $\langle\sigma\rangle = \mathrm{Gal}(L/\mathbf{Q})$.

Obviously $O_L/nO_L \supset \mathbf{Z}/n\mathbf{Z}[\zeta]$; we show that the elements $1, \zeta, \ldots, \zeta^{u-1}$ are linearly independent over $\mathbf{Z}/n\mathbf{Z}$, and so the cardinality of both rings must be n^u, which proves equality. To do so, look at the Vandermonde determinant

$$\det \begin{pmatrix} 1 & \zeta & \cdots & \zeta^{u-1} \\ \sigma 1 & \sigma\zeta & \cdots & \sigma\zeta^{u-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{u-1}1 & \sigma^{u-1}\zeta & \cdots & \sigma^{u-1}\zeta^{u-1} \end{pmatrix} = \prod_{\substack{i,j=0 \\ i>j}}^{u-1} \left(\sigma^i\zeta - \sigma^j\zeta\right) = \prod_{\substack{i,j=0 \\ i>j}}^{u-1} \left(\zeta^{n^i} - \zeta^{n^j}\right)$$

of which the value is a unit in $\mathbf{Z}/n\mathbf{Z}[\zeta]$ by (2.1). But a non-trivial $\mathbf{Z}/n\mathbf{Z}$-linear combination of the elements in the first row would give a non-trivial $\mathbf{Z}/n\mathbf{Z}$-linear combination of any other row as well, since σ leaves $\mathbf{Z}/n\mathbf{Z}$ invariant. Therefore the existence of a non-trivial relation between $1, \zeta, \ldots, \zeta^{u-1}$ would imply that the above determinant is a zero-divisor, and would thus lead to a contradiction.

By lifting ζ to an element in O_L, one proves also that $\gcd(\Delta_L, n) = 1$.

From the theorem of Kronecker-Weber it follows that $L \subset \mathbf{Q}(\zeta_m)$ for some m, and moreover such that m can be chosen coprime to n (see [58, Ch. V]).

Let r be a prime divisor of n; then $\phi_r \in \mathrm{Gal}(L/\mathbf{Q}) = \langle\sigma\rangle$, so $\phi_r = \sigma^i$ for some i with $0 \leq i < u$. As before, ϕ_r induces an automorphism on O_L/rO_L that we indicate by the same symbol, and which is by (2.3) just r-th powering on O_L/rO_L. From the commutative diagram

$$\begin{array}{ccc} O_L/nO_L & \xrightarrow{\mathrm{mod}\, r} & O_L/rO_L \\ \downarrow{\scriptstyle \sigma^i} & & \downarrow{\scriptstyle \phi_r} \\ O_L/nO_L & \xrightarrow{\mathrm{mod}\, r} & O_L/rO_L \end{array}$$

we see that $\zeta^{n^i} = \sigma^i(\zeta) \equiv \phi_r(\zeta) \equiv \zeta^r \bmod rO_L$. From Lemma (2.1) again, we see that the order of (the image of) ζ in O_L/rO_L is t. Therefore $\zeta^{n^i} \equiv \zeta^r \bmod rO_L$ implies $n^i \equiv r \bmod t$

and we find (2.9). Moreover, $n^i \equiv r \bmod t$ holds by multiplicativity for every divisor r of n, if we take i such that $\phi_r = \sigma^i$. In particular then, with $r = n$, we find that $n^i \equiv n \bmod t$ for i such that $\phi_n = \sigma^i$, and therefore $i \equiv 1 \bmod \operatorname{ord} n$, the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. Since this order equals u by definition and $u = \operatorname{ord}\sigma$ as well, we get $\phi_n = \sigma$. This proves (2.10).

That finishes the proof of (2.8).

(2.11) Proposition. *Let $n \in \mathbf{Z}_{\geq 2}$, and $t \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,t) = 1$. Suppose that a t-th cyclotomic constellation L, ζ, σ for n exists. Then for every divisor t' of t there exists a t'-th cyclotomic constellation L', ζ', σ' for n, with $L \supset L' \supset \mathbf{Q}$.*

Proof. Let $\mathrm{Gal}(L/\mathbf{Q}) = \langle\sigma\rangle$ and let L' be the intermediate field $\mathbf{Q} \subset L' \subset L$ of degree u' over $\mathbf{Q}$, where u' is the order of n in $(\mathbf{Z}/t'\mathbf{Z})^*$. Notice that L' is the invariant field of L under $H = \langle\sigma^{u'}\rangle$. Let $\zeta' = \zeta^{t/t'} \in O_L/nO_L$, then the basic properties of cyclotomic polynomials imply that $\Phi_{t'}(\zeta') = 0$. Since $\sigma(\zeta) = \zeta^n$ and $n^{u'} \equiv 1 \bmod t'$ we also have $\sigma^{u'}(\zeta') = \zeta'^{n^{u'}} = \zeta'$. Thus $\zeta' \in (O_L/nO_L)^H$. We want to prove that $\zeta' \in O_{L'}/nO_{L'}$. This is done by showing that in fact $O_{L'}/nO_{L'} = (O_L/nO_L)^H$, by a cardinality argument, as in (2.10).

Clearly, $O_{L'}/nO_{L'} \subset (O_L/nO_L)^H$, and since $\#O_{L'}/nO_{L'} = n^{u'}$, it suffices to show that $\#(O_L/nO_L)^H \leq n^{u'}$. Consider the value of the Vandermonde determinant

$$\det\begin{pmatrix} 1 & \zeta & \cdots & \zeta^{\frac{u}{u'}-1} \\ \sigma^{u'}1 & \sigma^{u'}\zeta & \cdots & \sigma^{u'}\zeta^{\frac{u}{u'}-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{(\frac{u}{u'}-1)u'}1 & \sigma^{(\frac{u}{u'}-1)u'}\zeta & \cdots & \sigma^{(\frac{u}{u'}-1)u'}\zeta^{\frac{u}{u'}-1} \end{pmatrix};$$

this is a unit, and hence $1, \zeta, \ldots, \zeta^{\frac{u}{u'}-1}$ are independent over $(O_L/nO_L)^H$. Therefore $n^u = \#(O_L/nO_L) \geq \#((O_L/nO_L)^H)^{u/u'}$, that is, $\#(O_L/nO_L)^H \leq n^{u'}$ as desired.

This concludes the proof of (2.11).

(2.12) Remarks. Let again L, ζ, σ be a t-th cyclotomic constellation for n, with $[L : \mathbf{Q}] = u$. Then the above shows in particular that for every u' dividing u, the degree u' subfield $L' \subset L$ forms together with $\zeta^{t/t'}$ and $\sigma^{u'}$ a t'-th cyclotomic constellation for n, where $t' = \gcd(t, n^{u'} - 1)$. It will later on be convenient to refer to this as the t'-th *cyclotomic sub-constellation.*

The definition of cyclotomic constellation L, ζ, σ above ensures that $O_L/nO_L, \zeta$, together with the induced automorphism σ on O_L/nO_L, form what is called a *cyclotomic*

extension of $\mathbf{Z}/n\mathbf{Z}$ in [87]. After we will have described a method for finding the field L and the element ζ (in Section 4), a cyclotomic constellation may be thought of as the construction of a cyclotomic extension for $\mathbf{Z}/n\mathbf{Z}$.

Proposition (2.11) merely states that the existence of t-th cyclotomic constellations guarantees the existence of t'-th cyclotomic constellations for divisors t' of t. Later on we will see that the nice property of Artin symbols in cyclotomic extensions expressed by (2.10) implies that the consequence expressed by (2.9), that every divisor of n is a power of n modulo t, even holds for many *multiples* of t. (In [87] it is also proved that for these multiples cyclotomic extensions do exist, but we will not use that fact.)

3. CHARACTERS AND GAUSS SUMS.

This section contains useful prerequisites about characters and Gauss sums. Most of these are well-known (and can for instance be found in [51], [92]), except that our characters take on their values in certain *rings*.

Throughout this section, let G denote a finite abelian group, and let $\exp G$ denote the exponent. That is, $\exp G$ is the smallest positive integer e for which $g^e = 1$ for every $g \in G$. Since our groups are finite, $\exp G$ is thus equal to the maximal order of the elements in G.

Also, throughout this section, A will be a commutative ring with 1, and $t \in \mathbf{Z}_{\geq 1}$ will be such that $t \cdot 1 \neq 0$ in A; furthermore $\zeta \in A$ will be such that $\Phi_t(\zeta) = 0$.

(3.1) Definitions. A *character* χ *on a finite abelian group* G *with values in* $\langle\zeta\rangle$ is a homomorphism

$$\chi: \quad G \to \langle\zeta\rangle$$

from G to the multiplicative subgroup $\langle\zeta\rangle$ of A^* generated by ζ. The *principal character* is the character χ with $\chi(g) = 1$ for every $g \in G$. The set of characters on G with values in $\langle\zeta\rangle$ forms a group under multiplication:

$$\chi_1 \cdot \chi_2(x) = \chi_1(x) \cdot \chi_2(x);$$

this group is denoted by $\mathrm{Hom}(G, \langle\zeta\rangle)$. Its unit element is the principal character on G.

(3.2) Lemma. *If* $\exp G$ *divides* $\mathrm{ord}\,\zeta$ *then* $\#\mathrm{Hom}(G, \langle\zeta\rangle) = \#G$.

Proof. To prove this assertion, use that any character on a quotient group G/H of G induces a character on G via the natural map $G \to G/H$; writing G as the direct product of cyclic subgroups, we thus see that it suffices to prove equality in case G is cyclic. A character on a cyclic group G is determined by its action on a generator g. But then it is immediately clear that every character on G is equal to one of the $\#G$ different powers of the character that sends g to an element of order $\#G$ in $\langle\zeta\rangle$; notice that such an element exists by our assumptions.

That proves (3.2).

(3.3) Lemma. *Let* $\chi \in \mathrm{Hom}(G, \langle\zeta\rangle)$ *be a non-principal character. Suppose that* $t \cdot 1 \in A^*$. *Then:*

$$\sum_{x \in G} \chi(x) = 0.$$

Proof. Let $a \in G$ be such that $\chi(a) \neq 1$. Let $1 \leq k < t$ be such that $\chi(a) = \zeta^k$. By Lemma (2.1), the element $\chi(a) - 1 = \zeta^k - 1$ is a unit in A. Now

$$\sum_{x \in G} \chi(x) = \sum_{x \in G} \chi(ax) = \chi(a) \sum_{x \in G} \chi(x)$$

which implies that $\sum \chi(x) = 0$ in A since $\chi(a) - 1$ is a unit.

That proves (3.3).

(3.4) Lemma. *Suppose that* $\exp G$ *divides* $\operatorname{ord} \zeta$. *Let* $x, y \in G$. *If*

$$\chi(x) = \chi(y) \qquad \textit{for every character} \quad \chi \in \operatorname{Hom}(G, \langle \zeta \rangle),$$

then $x = y$.

Proof. Denote $z = xy^{-1} \in G$. Suppose that $\chi(z) = 1$ for every character on G; if Z denotes the subgroup of G generated by z then all characters on G factor via G/Z. Thus

$$\#\operatorname{Hom}(G, \langle \zeta \rangle) \leq \#\operatorname{Hom}(G/Z, \langle \zeta \rangle) = \#G/Z \leq \#G;$$

by (3.2) this implies that Z is trivial, so $z = 1$ and $x = y$.

That ends the proof of (3.4).

We will mainly be interested in the case that G is the multiplicative group $(\mathbf{Z}/s\mathbf{Z})^*$ of integers modulo s; in the rest of this section $s \in \mathbf{Z}_{\geq 1}$.

(3.5) Definitions. Let $\chi \in \operatorname{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle \zeta \rangle)$. The *conductor* $\operatorname{cond} \chi$ *of* χ is the smallest divisor m of s for which the homomorphism χ factors as

$$\chi\colon \quad (\mathbf{Z}/s\mathbf{Z})^* \to (\mathbf{Z}/m\mathbf{Z})^* \to \langle \zeta \rangle$$

where the first map is the natural map. If $\operatorname{cond} \chi = s$ then the character is called *primitive*; otherwise it is *induced* from a primitive character in $\operatorname{Hom}((\mathbf{Z}/m\mathbf{Z})^*, \langle \zeta \rangle)$, for a proper divisor m of s. Note that for every χ there is a unique primitive character that induces χ. The unit element of $\operatorname{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle \zeta \rangle)$ is the *principal character* (of conductor 1) denoted by 1. The *inverse* of χ, denoted by χ^{-1}, is a character of conductor $\operatorname{cond} \chi^{-1} = \operatorname{cond} \chi$ that satisfies: $\chi^{-1}(x) = \chi(x)^{-1} = \chi(x^{-1})$ for every $x \in (\mathbf{Z}/\operatorname{cond} \chi \mathbf{Z})^*$. The *order* $\operatorname{ord} \chi$ of a character is its order as an element of $\operatorname{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle \zeta \rangle)$. Notice that by this definition $\operatorname{ord} \chi \mid \exp(\mathbf{Z}/\operatorname{cond} \chi \mathbf{Z})^*$.

A *character modulo* s is an element of $\mathrm{Hom}\big((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle\big)$, where $\exp(\mathbf{Z}/s\mathbf{Z})^* \mid \mathrm{ord}\,\zeta$.

Next we extend the definition of a character modulo s in such a way, that is defined on the integers. If χ is a character modulo s of conductor m, then for $x \in \mathbf{Z}/s\mathbf{Z}$ with $\gcd(x,m) > 1$ we define $\chi(x) = 0$. For any $z \in \mathbf{Z}$ we then define $\chi(z)$ by $\chi(z) = \chi(x)$, where $x \in \mathbf{Z}/m\mathbf{Z}$ and $x = (z \bmod m)$. It is important that m is the conductor here; as a consequence, if χ modulo s is any character, $\chi(x) \neq 0$ whenever x is coprime to the conductor of χ, even though $\gcd(x,s)$ may be non-trivial. For the the trivial character we have $1(z) = 1$ for every $z \in \mathbf{Z}$.

Let χ be a character modulo s; then χ defines a character χ_p modulo p^k, for each of the maximal prime powers $p^k \parallel s$, by projection onto the components in the decomposition:

$$(\mathbf{Z}/s\mathbf{Z})^* = \prod (\mathbf{Z}/p^k\mathbf{Z})^*.$$

Explicitly, for every $x \in (\mathbf{Z}/p^k\mathbf{Z})^*$ we can find by the Chinese remainder theorem an integer y such that

$$y \equiv x \bmod p^k \quad \text{and} \quad y \equiv 1 \bmod \frac{s}{p^k}.$$

The character χ_p modulo p^k is then defined by $\chi_p(x) = \chi(y)$ for every $x \in (\mathbf{Z}/p^k\mathbf{Z})^*$. The characters χ_p are called the *components* of χ. This gives the component decomposition $\chi = \prod \chi_p$ for any character. We have $\chi(x) = \prod \chi_p(x)$ for every $x \in (\mathbf{Z}/s\mathbf{Z})^*$. The components are completely determined by χ; conversely, a finite set of characters defined modulo mutually coprime moduli defines a character modulo the product of these moduli, so in particular χ is completely determined by its components χ_p. If χ is primitive, then so are all of its components, and vice versa.

Recall from (2.2) that we may view $A[\zeta_{k'}]$ as a subring of $A[\zeta_k]$, for any k' dividing k.

(3.6) Definition. Let χ be a primitive character modulo s and $a \in \mathbf{Z}$. Then we define the Gauss sums

$$\tau_a(\chi) = \sum_{x \in (\mathbf{Z}/s\mathbf{Z})^*} \chi(x)\zeta_s^{ax}$$

in $A[\zeta_s]$. The *Gauss sum* $\tau(\chi)$ associated to a character χ is by definition $\tau_1(\chi)$.

Notice that by our conventions we may as well let the summation in the definition of Gauss sums range over all $x \in \mathbf{Z}/s\mathbf{Z}$. Also note that $\tau(1) = 1$.

(3.7) Remark. It is important to note in the definition above that s is the conductor of χ. For every $k \in \mathbf{Z}_{\geq 1}$, the primitive character χ modulo s induces a character modulo ks. Hasse proves the following relation (cf [51, p. 449])

$$\sum_{x \in (\mathbf{Z}/ks\mathbf{Z})^*} \chi(x)\zeta_{ks}^{ax} = \frac{\phi(ks)}{\phi(s)}\mu(k)\chi(k)\tau_a(\chi);$$

here μ is Möbius's function, given by $\mu(p_1 p_2 \cdots p_l) = (-1)^l$ if all primes p_i are different, while $\mu(z) = 0$ if z is not squarefree.

It is somewhat easier to see that by our previous conventions

$$\sum_{x \in \mathbf{Z}/ks\mathbf{Z}} \chi(x)\zeta_{ks}^{ax} = 0$$

whenever $\zeta_k^a \neq 1$.

(3.8) Proposition. *Let χ be a character modulo s. Then:*

$$\tau_a(\chi) = \chi^{-1}(a)\tau(\chi) \qquad \text{for every} \quad a \in \mathbf{Z}.$$

Proof. Let m be the conductor of χ.

If $\gcd(a, m) = 1$ then $x \mapsto a \cdot x$ is injective on $\mathbf{Z}/m\mathbf{Z}$ so

$$\begin{aligned}\tau_a(\chi) &= \sum_{x \in (\mathbf{Z}/m\mathbf{Z})^*} \chi(x)\zeta_m^{ax} = \\ &= \chi(a^{-1}) \sum_{x \in (\mathbf{Z}/m\mathbf{Z})^*} \chi(ax)\zeta_m^{ax} = \\ &= \chi^{-1}(a) \sum_{ax \in (\mathbf{Z}/m\mathbf{Z})^*} \chi(ax)\zeta_m^{ax} = \chi^{-1}(a)\tau(\chi).\end{aligned}$$

If $\gcd(a, m) = d > 1$, let $I \subset \mathbf{Z}/m\mathbf{Z}$ be the ideal $I = \ker(x \mapsto a \cdot x)$. Then I is non-zero (it is for instance generated by m/d), and $J = (1 + I) \cap (\mathbf{Z}/m\mathbf{Z})^*$ is a subgroup of $(\mathbf{Z}/m\mathbf{Z})^*$. Also, J is equal to the kernel of the natural map $(\mathbf{Z}/m\mathbf{Z})^* \to ((\mathbf{Z}/m\mathbf{Z})/I)^*$; so if χ is trivial on J, it factors through I, which is impossible since m is the conductor of χ. But then with Y a set of representatives for $(\mathbf{Z}/m\mathbf{Z})^*/J$

$$\begin{aligned}\tau_a(\chi) &= \sum_{y \in Y} \sum_{x \in yJ} \chi(x)\zeta_m^{ax} = \\ &= \sum_{y \in Y} \chi(y)\zeta_m^{ay} \sum_{x \in J} \chi(x) = 0\end{aligned}$$

by (3.3). Because $\chi^{-1}(a) = 0$ as well in this case, this finishes the proof of (3.8).

(3.9) Lemma. *Let χ be a primitive character modulo s. Then for $a \in \mathbf{Z}$:*

$$\tau_a(\chi)\tau_a(\chi^{-1}) = \begin{cases} \tau(\chi)\tau(\chi^{-1}) & \text{if } \gcd(a,s) = 1; \\ 0 & \text{if } \gcd(a,s) \neq 1. \end{cases}$$

Proof. By (3.8) we find:

$$\tau_a(\chi)\tau_a(\chi^{-1}) = \chi^{-1}(a)\tau(\chi)\chi(a)\tau(\chi^{-1}).$$

If $\gcd(a,s) \neq 1$ this equals 0 because $\chi^{-1}(a) = 0$. If $\gcd(a,s) = 1$ it equals $\tau(\chi)\tau(\chi^{-1})$. This proves (3.9).

(3.10) Corollary. *For every character χ modulo s:*

$$\tau(\chi)\tau(\chi^{-1}) = \chi(-1)\operatorname{cond}\chi.$$

If $s \in A^*$, then for every $j \in \mathbf{Z}_{\geq 1}$:

$$\tau(\chi^j) \in A[\zeta_s]^*.$$

Proof. Let m be the conductor of χ. Consider the Gauss sums $\tau_a(\chi) \in (\mathbf{Z}[\zeta_t])[\zeta_s]$, where we let a range over a set of representatives of $\mathbf{Z}/m\mathbf{Z}$. On the one hand:

$$\sum_{a \in \mathbf{Z}/m\mathbf{Z}} \tau_a(\chi)\tau_a(\chi^{-1}) = \#(\mathbf{Z}/m\mathbf{Z})^*\tau(\chi)\tau(\chi^{-1})$$

by (3.9). On the other hand, the map $a \mapsto \zeta_m^{(x+y)a}$ is a character on $\mathbf{Z}/m\mathbf{Z}$, so:

$$\sum_{a \in \mathbf{Z}/m\mathbf{Z}} \tau_a(\chi)\tau_a(\chi^{-1}) = \sum_{x,y \in (\mathbf{Z}/m\mathbf{Z})^*} \Big(\chi(x)\chi^{-1}(y) \sum_{a \in \mathbf{Z}/m\mathbf{Z}} \zeta_m^{(x+y)a}\Big) = \chi(-1)\phi(m)m$$

since by (3.3) the inner sum is zero whenever $x \neq -y$. Since $\phi(m)$ is not a zero-divisor in $(\mathbf{Z}[\zeta_t])[\zeta_m]$, it follows that $\tau(\chi)\tau(\chi^{-1}) = \chi(-1)\cdot m \in \mathbf{Z}[\zeta_m]$; but then they are equal under the natural map $(\mathbf{Z}[\zeta_t])[\zeta_m] \to A[\zeta_m]$, sending ζ_t to ζ and ζ_m to ζ_m, as well. That yields the first part of the corollary.

The second part follows from this, since the conductor m of χ^j is a divisor of s. If m is a proper divisor of s, then $\tau(\chi^j) \in A[\zeta_m]$, which we have viewed as a proper subring of $A[\zeta_s]$; in both $\tau(\chi^j)$ is a unit.

(3.11) Lemma. *If n is prime, $\gcd(n,s) = 1$ and both χ and χ^n are primitive characters modulo s, then:*

$$\tau(\chi^n) \equiv \chi(n)^n \tau(\chi)^n \mod nA[\zeta_s].$$

Proof. In $A[\zeta_s]$ we have the following equalities:

$$\begin{aligned}
\tau(\chi^n) &= \sum_{x \in (\mathbf{Z}/s\mathbf{Z})^*} \chi(x)^n \zeta^x \\
&= \sum_{x \in (\mathbf{Z}/s\mathbf{Z})^*} \chi(nx)^n \zeta^{nx} \qquad \text{since } \gcd(n,s) = 1 \\
&= \chi(n)^n \sum_{x \in (\mathbf{Z}/s\mathbf{Z})^*} \chi(x)^n \zeta^{nx} \\
&\equiv \chi(n)^n \Big(\sum_{x \in (\mathbf{Z}/s\mathbf{Z})^*} \chi(x)\zeta^x \Big)^n \mod n \qquad \text{since } n \text{ is prime.}
\end{aligned}$$

But

$$\Big(\sum_{x \in (\mathbf{Z}/s\mathbf{Z})^*} \chi(x)\zeta^x \Big)^n = \tau(\chi)^n,$$

and the proof of (3.11) is finished.

4. CONSTRUCTING CYCLOTOMIC CONSTELLATIONS.

For the applications of Theorem (2.8) and some theorems in the next sections, we need to construct cyclotomic constellations L, ζ, σ of a given degree. We want to be able to compute efficiently in O_L/nO_L, in particular to find an element ζ with the desired properties. In this section we describe how to do this explicitly.

First we show that, to obtain suitable fields for our cyclotomic constellations, it suffices to construct extensions of prime power degree.

(4.1) Proposition. *Let $n \in \mathbf{Z}_{\geq 2}$, and suppose that for $i = 1, 2$ the number field $L_i \supset \mathbf{Q}$ is cyclic,* $\mathrm{Gal}(L_i/\mathbf{Q}) = \langle \sigma_i \rangle$, *of degree* $[L_i : \mathbf{Q}] = u_i$, *with* $\gcd(\Delta_{L_i}, n) = 1$. *If* $\gcd(u_1, u_2) = 1$ *then $L = L_1 \cdot L_2$ is cyclic over $\mathbf{Q}$ of degree $u_1 u_2$ with group* $\langle \sigma \rangle = \mathrm{Gal}(L/\mathbf{Q}) \cong \mathrm{Gal}(L_1/\mathbf{Q}) \times \mathrm{Gal}(L_2/\mathbf{Q})$, *and* $\gcd(\Delta_L, n) = 1$.

Proof. It is standard Galois theory that the composite L of two number fields L_1, L_2 that have cyclic Galois groups of coprime orders, is a cyclic number field with the direct product of these groups as Galois group. For the discriminants one has (cf. [139, p. 112]):

$$\Delta_L = \mathrm{N}_{L_1/\mathbf{Q}}(\Delta_{L/L_1}) \cdot \Delta_{L_1/\mathbf{Q}}^{[L:L_1]},$$

and as a consequence Δ_L is built up from primes in Δ_{L_1} and Δ_{L_2} only.

This proves (4.1).

The rest of this section is devoted to the construction of cyclotomic constellations of prime power degree. The idea is to give an explicit description of useful prime power degree fields as subfields of cyclotomic fields. For this description we use the correspondence, given in the next lemma, between intermediate fields $\mathbf{Q}(\zeta_m) \supset L \supset \mathbf{Q}$ and subgroups of characters in $\mathrm{Hom}((\mathbf{Z}/m\mathbf{Z})^*, \langle \zeta \rangle)$ for ζ a zero of a cyclotomic polynomial of large enough order. If X is a subgroup of $\mathrm{Hom}(G, \langle \zeta \rangle)$ then we denote

$$\ker \mathrm{X} = \bigcap_{\chi \in \mathrm{X}} \ker \chi = \bigcap_{\chi \in \mathrm{X}} \{x \in G: \ \chi(x) = 1\}.$$

As is customary in Galois theory, we denote for any subgroup H of $\mathrm{Gal}(L/\mathbf{Q})$ by L^H the invariant field of L under H.

Recall from the previous section that a character modulo m, with $m \in \mathbf{Z}_{\geq 1}$, is an element of $\mathrm{Hom}((\mathbf{Z}/m\mathbf{Z})^*, \langle \zeta \rangle)$; here ζ is an element of a commutative ring A with 1, and satisfies $\Phi_t(\zeta) = 0$, with $t \cdot 1 \neq 0$ and with $\exp(\mathbf{Z}/m\mathbf{Z})^* \mid \mathrm{ord}\, \zeta$.

(4.2) Proposition. *Let $m \in \mathbf{Z}_{\geq 2}$. There exists a bijection between the set of intermediate fields $\mathbf{Q}(\zeta_m) \supset L \supset \mathbf{Q}$ and the set of subgroups X of the group of characters modulo m, which is given by $L = \mathbf{Q}(\zeta_m)^{\ker \mathrm{X}}$, where $\ker \mathrm{X}$ acts via the usual identification of $\mathrm{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q})$ with $(\mathbf{Z}/m\mathbf{Z})^*$.*

Proof. Let $G = (\mathbf{Z}/m\mathbf{Z})^*$ and let X be a subgroup of characters on G. It is clear that $\ker \mathrm{X}$ is a subgroup of G. Conversely, to any subgroup H of G, we can assign a subgroup of characters on G, namely those characters that are trivial on H. This gives an inclusion reversing bijection between the set of subgroups X of characters on G and the set of subgroups H of G. By the main theorem of Galois theory the set of subgroups H of G corresponds bijectively to the set of intermediate fields $\mathbf{Q} \subset L \subset \mathbf{Q}(\zeta_m)$, via $L = \mathbf{Q}(\zeta_m)^H$.

That proves (4.2).

(4.3) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$. Let χ be a primitive character modulo m. Let $L = \mathbf{Q}(\zeta_m)^{\ker \chi}$ and let $\eta = \mathrm{Tr}_{\mathbf{Q}(\zeta_m)/L}\, \zeta_m \in O_L$. If $\gcd(n, m) = 1$, then $\sigma\eta \not\equiv \eta \bmod nO_L$ for any $\sigma \in \mathrm{Gal}(L/\mathbf{Q})$.*

Proof. Let G denote $\mathrm{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q})$ and let $H = \ker \chi \subset G$. Then G/H is cyclic, of order $u = \mathrm{ord}\, \chi$; let τ be a generator. The canonical map $G \to G/H$ induces a surjective map ϵ: $\mathbf{Z}[G] \to \mathbf{Z}[G/H]$ of group rings. We map any element $\mu \in \mathbf{Z}[G/H]$ to an element $\delta\mu \in \mathbf{Z}[G]$ by:

$$\mu = \sum_{\nu \in G/H} t_\nu \nu \mapsto \delta(\mu) = \sum_{\sigma \in G} s_\sigma \sigma, \text{ where } s_\sigma = t_{\epsilon(\sigma)}.$$

Note that δ is clearly injective. In this way, for every $\mu \in \mathbf{Z}[G/H]$ we have

$$\delta(\mu)x = \mu(\mathrm{Tr}\, x) \text{ for every } x \in \mathbf{Q}(\zeta_m),$$

where, as in the rest of this proof, $\mathrm{Tr} = \mathrm{Tr}_{\mathbf{Q}(\zeta_m)/L}$. Let $x \in \mathbf{Z}[\zeta_m]$ and suppose that $\rho x = x$ for some $\rho \in G \setminus H$. Then

$$\mathrm{Tr}\, x = \rho(\mathrm{Tr}\, x) = \epsilon(\rho)\, \mathrm{Tr}\, x.$$

Since $\epsilon(\rho) \neq 1$ in $G/H = \langle \tau \rangle$, we have $\tau^{u/p} \in \langle \epsilon(\rho) \rangle$ for some prime p dividing u, and therefore $(1 - \tau^{u/p})\, \mathrm{Tr}\, x = 0$.

Now choose

$$\psi = \prod_{p|u} (1 - \tau^{u/p}) \in \mathbf{Z}[G/H],$$

and let $\gamma = \delta(\psi) \in \mathbf{Z}[G]$. Note that $\gamma \neq 0$, since δ is injective. The above argument shows that for any $x \in \mathbf{Z}[\zeta_m]$

$$\rho x = x \text{ for some } \rho \notin H \quad \Rightarrow \quad \gamma x = \delta(\psi)x = 0;$$

the same is true for $x \in \mathbf{Z}/n\mathbf{Z}[\zeta_m]$.

Suppose that the image of $\mathbf{Z}/n\mathbf{Z}[\zeta_m]$ under γ is 0, so in particular $\gamma\zeta_m^i \equiv 0 \bmod n$ for every i; then

$$n \text{ divides } \det\left(\sigma\zeta_m^i\right)_{\substack{0 \le i < \phi(m) \\ \sigma \in G}}. \tag{4.4}$$

But

$$\Delta_{\mathbf{Q}(\zeta_m)} = \left(\det\left(\sigma\zeta_m^i\right)_{\substack{0 \le i < \phi(m) \\ \sigma \in G}}\right)^2$$

and $\Delta_{\mathbf{Q}(\zeta_m)}$ is divisible only by primes dividing m, so (4.4) contradicts the assumption that m is coprime to n.

Hence, $\gamma\zeta_m^i = \psi(\eta_i) \not\equiv 0 \bmod n$ for some i, where $\eta_i = \operatorname{Tr}\zeta_m^i$. Suppose $\gcd(i,m) > 1$ for this particular i, say the prime p divides $\gcd(i,m)$. Then η_i is contained in the invariant field of $\mathbf{Q}(\zeta_m)$ under $S = \ker\left(\left(\mathbf{Z}/m\mathbf{Z}\right)^* \to \left(\mathbf{Z}/\frac{m}{p}\mathbf{Z}\right)^*\right)$; from the definition of η it is also clear that η_i is invariant under H. Thus η_i is invariant under $H \cdot S \subset G$, a subgroup that is strictly larger than $H = \ker\chi$ since χ is primitive! Therefore η_i is contained in a proper subfield of L, which means that $(1 - \tau^{u/q})\eta_i \equiv 0 \bmod n$, for some prime q dividing u. Hence $\psi\eta_i \equiv 0 \bmod n$, which implies that $\delta(\psi)\eta_i = \gamma\eta_i \equiv 0 \bmod n$, contradicting the choice for i. That proves that i is relatively prime to m.

Now $\gamma\zeta_m^i \not\equiv 0 \bmod n$ implies that $\gamma\zeta_m \not\equiv 0 \bmod n$ and thus that $\psi\eta \not\equiv 0 \bmod n$, so by the definition of ψ we find that $(\tau^d - 1)\eta \not\equiv 0 \bmod n$ for every proper divisor d of u. Since τ generates the group, that proves the result.

(4.5) Corollary. *Let χ be a primitive character modulo m. Let $L = \mathbf{Q}(\zeta_m)^{\ker\chi}$ and let $\eta = \operatorname{Tr}_{\mathbf{Q}(\zeta_m)/L}\zeta_m \in O_L$. Then $L = \mathbf{Q}(\eta)$. Moreover, if n is prime, $n \nmid m$ and ϕ_n generates $\operatorname{Gal}(L/\mathbf{Q})$, then $O_L/nO_L \cong (\mathbf{Z}/n\mathbf{Z})[\eta]$.*

Proof. It is clear that $\mathbf{Q}(\eta) \subset L$. Also, the conjugates of η under $G = \operatorname{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q})$ are in L; from the previous theorem it follows that η is not contained in a proper subfield. That proves the first assertion.

If n is prime and ϕ_n generates $\mathrm{Gal}(L/\mathbf{Q})$, we know from (2.5) that O_L/nO_L is a field, containing $\mathbf{Z}/n\mathbf{Z}$ as its prime field. Now n does not divide $\Delta(f_{\mathbf{Q}}^{\eta})$, the discriminant of the minimal polynomial of η over $\mathbf{Q}$, if and only if the discriminant of $\overline{f_{\mathbf{Q}}^{\eta}}$ (the reduction modulo n) is unit in $\mathbf{Z}/n\mathbf{Z}$, that is, non-zero. By Vandermonde's formula for the discriminant this is equivalent to

$$\prod_{\substack{\sigma,\tau\in G\\ \sigma\neq\tau}} \overline{(\sigma\eta-\tau\eta)} = \prod_{\substack{\sigma,\tau\in G\\ \sigma\neq\tau}} (\overline{\sigma\eta}-\overline{\tau\eta}) \not\equiv 0 \bmod n,$$

where g abbreviates $\mathrm{Gal}(\mathbf{Q}(\zeta_m)/L)$. But by (4.3) the factors in the latter product are non-zero, whence the product in the field O_L/nO_L is non-zero. Since (cf. [139])

$$\Delta(f_{\mathbf{Q}}^{\eta}) = \mathrm{index}\big[O_L : \mathbf{Z}[\eta]\big]^2 \Delta_L,$$

this shows that $\mathrm{index}\big[O_L : \mathbf{Z}[\eta]\big]$ is not divisible by n, so it is a unit modulo n. Therefore $O_L/nO_L \cong (\mathbf{Z}/n\mathbf{Z})[\eta] \cong \mathbf{Z}/n\mathbf{Z}[X]/(f_{\mathbf{Q}}^{\eta})$, a field of $n^{[L:\mathbf{Q}]}$ elements.

That proves (4.5).

(4.6) Theorem. *Let l be prime and $k \in \mathbf{Z}_{\geq 1}$. Suppose that L is an intermediate field $\mathbf{Q} \subset L \subset \mathbf{Q}(\zeta_f)$ for some $f \in \mathbf{Z}_{\geq 1}$, with the property that it is cyclic of degree l^k over $\mathbf{Q}$. Then there exist a divisor m of f and a character χ on $(\mathbf{Z}/m\mathbf{Z})^*$ such that $\mathbf{Q}(\zeta_m)^{\ker\chi}$ is cyclic over $\mathbf{Q}$ of degree l^k and one of the following holds:*

(i) *m is prime and $l^k \mid m-1$;*

(ii) *l is odd and $m = l^{k+1}$;*

(iii) *$l^k = 2^k$ with $k \geq 2$ and $m = 2^{k+2}$. In this case χ may be any of the two different characters (up to conjugacy) of order l^k modulo m;*

(iv) *$l^k = 2$ and $m = 8$ or $m = 4$. Here χ may be any of the three quadratic characters modulo 8.*

Proof. Since $L \subset \mathbf{Q}(\zeta_f)$ is cyclic over $\mathbf{Q}$ of degree l^k, it corresponds by (4.2) to a character of order l^k on $(\mathbf{Z}/f\mathbf{Z})^*$. Let $\chi = \prod \chi_p$ be the component decomposition of χ. Then χ_p must be of order l^k for one of the components χ_p. Let the conductor of this component χ_p be p^d, then $\mathrm{ord}\,\chi_p \mid \exp(\mathbf{Z}/p^d\mathbf{Z})^*$.

If p is odd, $\exp(\mathbf{Z}/p^d\mathbf{Z})^* = (p-1)p^{d-1}$; if $p = 2$ then $\exp(\mathbf{Z}/p^d\mathbf{Z})^*$ is 2^{d-1} for $d \leq 2$ and 2^{d-2} for $d \geq 3$.

Therefore either $l^k \mid p-1$ or $l = p$. In the former case take $m = p$ and in the latter take $m = l^{k+1}$ (if l is odd) and $m = 2^{k+2}$ (if $l = 2$), and let χ be the character of order l^k modulo m induced by χ_p. Then the field $\mathbf{Q}(\zeta_m)^{\ker \chi}$ is cyclic of degree l^k by (4.2).

If $m = 2^{k+2}$ with $k \geq 1$ then $(\mathbf{Z}/m\mathbf{Z})^*$ is the product of a group of order 2 and a cyclic group of order 2^k. Therefore, if we fix a primitive 2^k-th root of unity, there exist precisely 2 different characters of order 2^k, unless $k = 1$; in the latter case there exist 3 different characters, two of conductor 8 and one of conductor 4.

This proves (4.6).

(4.7) Remarks. Since every abelian number field is contained in some cyclotomic field by Kronecker-Weber's theorem (cf. [58, p. 165]), every cyclic extension of $\mathbf{Q}$ is contained in a cyclotomic field. Thus the condition $L \subset \mathbf{Q}(\zeta_f)$ imposes no restriction at all.

The smallest m for which a given abelian field L is contained in $\mathbf{Q}(\zeta_m)$ is called the *conductor* of L; after the above discussion it will be clear that the conductor of the cyclic field L equals the conductor m of the character χ for which $L \cong \mathbf{Q}(\zeta_m)^{\ker \chi}$.

Let χ be a primitive character modulo m of prime power order l^e. Let $L = \mathbf{Q}(\zeta_m)^{\ker \chi}$ and let $\eta = \mathrm{Tr}_{\mathbf{Q}(\zeta_m)/L}\, \zeta_m \in O_L$; then $L = \mathbf{Q}(\eta)$ by (4.5). For use in Chapter IV, we make the following observations concerning bases for L over $\mathbf{Q}$ (as vector space). For the constructions we choose m either prime or a power of l, which is not a genuine restriction by (4.6). Suppose that σ generates the group $\mathrm{Gal}(L/\mathbf{Q})$. If m is prime, we can always choose $\eta = \sigma^0\eta, \sigma^1\eta, \ldots, \sigma^{u-1}\eta$ as a basis for L over $\mathbf{Q}$. In case m is a power of l however, we have to choose the basis differently; so assume for the rest of (4.7) that m is a power of l. Let $\eta_1 = 1$ and $\eta_{l^e} = \eta$, and consider the chain

$$\mathbf{Q} = \mathbf{Q}(\eta_{l^0}) \subset \mathbf{Q}(\eta_{l^1}) \subset \cdots \subset \mathbf{Q}(\eta_{l^e}) = \mathbf{Q}(\eta),$$

where $[\mathbf{Q}(\eta_{l^j}) : \mathbf{Q}] = l^j$. We claim that for $0 \leq j \leq e$ the set

$$\{1\} \cup \left\{ \sigma^{d-l^{i-1}}(\eta_{l^i}) : \ 1 \leq i \leq j, \ \ l^{i-1} \leq d < l^i \right\}$$

forms a basis for the ring of integers of $\mathbf{Q}(\eta_{l^j})$ over $\mathbf{Z}$, and hence for $\mathbf{Q}(\eta_{l^j})$ over $\mathbf{Q}$.

The claim can be proved as follows. Let $A \subset (\mathbf{Z}/l^k\mathbf{Z})^*$ be such that A maps bijectively to $(\mathbf{Z}/l^{k-1}\mathbf{Z})^*$ under the natural homomorphism $(\mathbf{Z}/l^k\mathbf{Z})^* \to (\mathbf{Z}/l^{k-1}\mathbf{Z})^*$. Since $\mathrm{Tr}_{\mathbf{Q}(\zeta_{l^k})/\mathbf{Q}(\zeta_{l^{k-1}})}(\zeta_{l^k}) = 0$, we find that for every $a \in A$:

$$\sum_{\substack{x \equiv a \bmod l^{k-1} \\ x \in (\mathbf{Z}/l^k\mathbf{Z})^*}} \zeta_{l^k}^x = 0.$$

Using this, and by comparing ranks, it follows that

$$\mathbf{Z}[\zeta_{l^k}] = \left(\bigoplus_{b \notin A} \mathbf{Z} \cdot \zeta_{l^k}^b\right) \bigoplus \mathbf{Z}[\zeta_{l^{k-1}}]$$

as $\mathbf{Z}$-modules. Let H be the subgroup $\ker \chi$ of $(\mathbf{Z}/l^k\mathbf{Z})^*$, which is of order $l-1$ if l is odd, and equals $\langle -1 \rangle$ if $l = 2$. Since H maps injectively into $(\mathbf{Z}/l^{k-1}\mathbf{Z})^*$, it follows that A can be chosen in such a way that it is a union of cosets of H in $(\mathbf{Z}/l^k\mathbf{Z})^*$; the same then applies to $B = (\mathbf{Z}/l^k\mathbf{Z})^* \setminus A$. Now $\sum_{h \in H} \zeta_{l^k} = \eta_{l^j}$, with $j = k-1$ for l odd and $j = k-2$ for $l = 2$ and therefore

$$\mathbf{Z}[\zeta_{l^k}]^H = \left(\bigoplus_{b \notin A} \mathbf{Z} \cdot \zeta_{l^k}^b\right)^H \bigoplus \mathbf{Z}[\zeta_{l^{k-1}}]^H = \left(\bigoplus_{b \in B/H} \mathbf{Z} \cdot \eta_{l^j}\right) \bigoplus \mathbf{Z}[\zeta_{l^{k-1}}]^H.$$

The claim follows by induction, if we choose A such that $B/H = \{g^0, g^1, \ldots, g^{l^j - l^{j-1}}\}$, for a generator g of $(\mathbf{Z}/l^k\mathbf{Z})^*/H$, and define σ by $\sigma(\zeta_{l^k}) = \zeta_{l^k}^g$ and restriction.

From (4.6) we now know where to look for our cyclic extensions L. For our primality testing purposes we will have to find an element $\zeta \in O_L/nO_L$ satisfying certain conditions; if we succeed, we will know among others that the cyclic Galois group $\mathrm{Gal}(L/\mathbf{Q})$ is generated by ϕ_n. The following lemma gives us a necessary and sufficient condition for this in terms of just n, l and m. Thus we will look for our cyclic extension only in those cyclotomic fields where this condition is met.

(4.8) Lemma. *Let l be a prime number, $k \in \mathbf{Z}_{\geq 1}$ and $n \in \mathbf{Z}_{\geq 1}$. Let χ be a character of conductor m and order l^k, where either m is prime or l is odd and $m = l^{k+1}$. Let $L = \mathbf{Q}(\zeta_m)^{\ker \chi}$. If $\gcd(n, m) = 1$ then:*

$$\mathrm{Gal}(L/\mathbf{Q}) = \langle \phi_n \rangle \iff \mathrm{ord}\, \chi(n) = l^k \iff n^{\frac{\phi(m)}{l}} \not\equiv 1 \bmod m.$$

Proof. Under the isomorphism $G = \mathrm{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q}) \cong \left(\mathbf{Z}/m\mathbf{Z}\right)^*$ the Artin symbol ϕ_n corresponds to $n \bmod m$, for n coprime to m. By Galois theory, $\mathrm{Gal}(L/\mathbf{Q}) \cong G/\ker \chi$. Therefore ϕ_n generates the Galois group $\mathrm{Gal}(L/\mathbf{Q})$ if and only if the image of $n \bmod m$ generates $G/\ker \chi$, which is the case if and only if $\chi(n)$ generates the image of χ. This is true if and only if n is not an l-th power modulo m, which is equivalent to the condition on the right hand side, since (under the hypotheses) $\left(\mathbf{Z}/m\mathbf{Z}\right)^*$ is cyclic, of order $\phi(m)$.

That ends the proof.

(4.9) Remarks. Note that by the previous theorem for every cyclic L of degree l^k either we may choose m as in Lemma (4.8), or $l = 2$ and we may choose $m = 2^{k+2}$; the latter case will be covered by the proposition below.

Notice that if n is prime, the existence of the element ζ in O_L/nO_L with the property that $\Phi_t(\zeta) = 0$ and $\phi(\zeta) = \zeta^n$ is guaranteed by (2.3), (2.5) and (2.6), if L is constructed in such a way that the condition in (4.8) is satisfied.

In the next corollary the fields having both degree and conductor a power of 2 are explicitly described, and criteria are given for these to have the desired property that ϕ_n generates their Galois group.

(4.10) Proposition. *Let $k \in \mathbf{Z}_{\geq 1}$ and let $L_1^{(k)}$, $L_2^{(k)}$ and $L_3^{(k)}$ be the subfields of $\mathbf{Q}(\zeta_{2^{k+2}})$ defined by:*

$$\begin{aligned} L_1^{(k)} &= \mathbf{Q}(\eta_1), \quad \text{where } \eta_1 = \zeta_{2^{k+2}} + \zeta_{2^{k+2}}^{-1}, \\ L_2^{(k)} &= \mathbf{Q}(\eta_2), \quad \text{where } \eta_2 = \zeta_{2^{k+2}} - \zeta_{2^{k+2}}^{-1} \\ L_3^{(k)} &= \mathbf{Q}(\eta_3), \quad \text{where } \eta_3 = \zeta_{2^{k+1}}. \end{aligned}$$

A subfield $L \subset \mathbf{Q}(\zeta_{2^{k+2}})$ is quadratic over $\mathbf{Q}$ if and only if $L = L_i^{(1)}$, for some $i \in \{1,2,3\}$; it is cyclic of degree 2^j (with $j > 1$) over $\mathbf{Q}$ if and only if $L = L_i^{(j)}$, for some $i \in \{1,2\}$. Furthermore:

$$\begin{aligned} \mathrm{Gal}(L_1^{(k)}/\mathbf{Q}) = \langle \phi_n \rangle &\iff k \geq 1 \quad \text{and} \quad n \equiv \pm 3 \bmod 8, \\ \mathrm{Gal}(L_2^{(k)}/\mathbf{Q}) = \langle \phi_n \rangle &\iff \begin{cases} k = 1 & \text{and} \quad n \equiv -1 \text{ or } -3 \bmod 8, \\ k \geq 2 & \text{and} \quad n \equiv \pm 3 \bmod 8, \end{cases} \\ \mathrm{Gal}(L_3^{(k)}/\mathbf{Q}) = \langle \phi_n \rangle &\iff k = 1 \quad \text{and} \quad n \equiv 3 \bmod 4. \end{aligned}$$

Proof. We prove that the fields $L_i^{(k)}$ (with $k = 1$ if $i = 3$) are the cyclic fields of degree 2^k over $\mathbf{Q}$ indicated by (4.6) (iii) and (iv). That these are the only ones, is then an immediate consequence of (4.6).

First let $k \geq 2$. We will use the fact that $\left(\mathbf{Z}/2^{k+2}\mathbf{Z}\right)^*$ is generated by the elements -1 (of order 2) and 5 (of order 2^k); furthermore, $a \in \left(\mathbf{Z}/2^{k+2}\mathbf{Z}\right)^*$ has order 2^k if and only if $a \equiv \pm 3 \bmod 8$ (since exactly half of the elements have this order and since $a^{2^{k-1}} \equiv 1 \bmod 2^{k+2}$ if $a \equiv \pm 1 \bmod 8$). To find the subfields of $\mathbf{Q}(\zeta_{2^{k+2}})$ that are cyclic of degree 2^k over $\mathbf{Q}$, we have to find the invariant fields under characters of order 2^k. There are 2 such

characters (up to choice of primitive root of unity), given by $\chi_1(-1) = 1$ and $\chi_1(5) = \zeta_{2^k}$, respectively $\chi_2(-1) = -1$ and $\chi_2(5) = \zeta_{2^k}$; the kernel of χ_1 is generated by -1 and that of χ_2 by $2^{k+1}-1 \equiv -5^{2^{k-1}}$. According to corollary (4.5)(i), their invariant fields are generated by $\zeta_{2^{k+2}} + \zeta_{2^{k+2}}^{-1}$ and $\zeta_{2^{k+2}} + \zeta_{2^{k+2}}^{-5^{2^{k-1}}}$ respectively. Since $5^{2^{k-1}} \equiv 2^{k+1}+1 \bmod 2^{k+2}$ (being "another" square root of 1), and since $\zeta_{2^{k+2}}^{2^{k+1}} = -1$ we find that these fields are $L_1^{(k)}$ and $L_2^{(k)}$ respectively. Their Galois groups are generated by ϕ_n if and only if the order of n in $\left(\mathbf{Z}/2^{k+2}\mathbf{Z}\right)^*$ equals 2^k; these are exactly the elements congruent to ± 3 modulo 8 as we mentioned above.

The only remaining cases are when $k = 1$: here again we find the characters χ_1 and χ_2, but in addition there is the character $\chi_3 = \chi_1\chi_2$ of order 2. Just as above, the invariant fields of χ_1 and χ_2 are $L_1^{(1)}$ and $L_2^{(1)}$; now ϕ_n generates $\mathrm{Gal}(L_1^{(1)}/\mathbf{Q})$ if and only if $\chi_1(n) = -1$, which is exactly when $n \equiv \pm 3 \bmod 8$. But ϕ_n generates $\mathrm{Gal}(L_2^{(1)}/\mathbf{Q})$ if and only if $\chi_2(n) = -1$, which means $n \equiv -1$ or $n \equiv -3$ modulo 8. Finally, the character χ_3 has kernel generated by 5; here however the invariant field is not given by η as in (4.5) because the conductor of χ_3 is 4 instead of 8. Here the invariant field is $\mathbf{Q}(\zeta_8^2) = \mathbf{Q}(i)$ and ϕ_n generates the group if and only if $\chi_3(n) = -1$, that is when $n \equiv 3 \bmod 4$.

This proves (4.10).

(4.11) Remarks. Altogether, for given u (and n), we now have the following explicit construction for a suitable ring O_L/nO_L.

First decompose u into its maximal prime power factors $u = \prod l^k$. For every l find a conductor m that is either prime and 1 mod l^k or (if l is odd) equal to l^{k+1}, satisfying

$$n^{\frac{\phi(m)}{l}} \not\equiv 1 \bmod m.$$

Choose a primitive root g modulo m and put

$$\eta_l = \sum_{i=0}^{\frac{\phi(m)}{l^k}-1} \zeta_m^{g^{il^k}}.$$

For $l = 2$ and $k \geq 2$, one may also choose $m = 2^{k+2}$, and $\eta_2 = \zeta_{2^k} \pm \zeta_{2^k}^{-1}$, provided that $n \equiv \pm 3 \bmod 8$; if $l = 2$ and $k = 1$, one may choose $m = 8$ and $\eta_2 = \zeta_{2^k} + \zeta_{2^k}^{-1}$ if $n \equiv \pm 3 \bmod 8$, or $m = 4$ and $\eta_2 = \zeta_4$ in case $n \equiv 3 \bmod 4$. By (4.5), (4.8) and (4.10) the field $\mathbf{Q}(\eta_l)$ is cyclic of degree l^k with group $\langle \phi_n \rangle$, and moreover, if n is prime then $O_L/nO_L \cong \mathbf{Z}/n\mathbf{Z}[\eta_l]$.

For the composite field $L = \mathbf{Q}(\{\eta_l : l \mid u\})$ containing all η_l, we find, taking tensor products over $\mathbf{Z}/n\mathbf{Z}$

$$\begin{aligned} O_L/nO_L &\cong \bigotimes_{l|u} \mathbf{Z}/n\mathbf{Z}[\eta_l] \cong \bigotimes_{l|u} \mathbf{Z}/n\mathbf{Z}[X]/(f_{\mathbf{Q}}^{\eta_l}(X)) \\ &\cong \mathbf{Z}/n\mathbf{Z}[X_1, X_2, \ldots, X_h]/(f_1(X_1), f_2(X_2), \ldots, f_h(X_h)), \end{aligned}$$

where h is the number of distinct prime divisors of u, and the f_i are the minimal polynomials of the η_l over $\mathbf{Q}$.

Now that we have an explicit construction for our ring, we construct the element ζ.

(4.12) Proposition. *Let $n \in \mathbf{Z}_{\geq 2}, t \in \mathbf{Z}_{\geq 1}$ and let $u = \operatorname{ord} n$ in $(\mathbf{Z}/t\mathbf{Z})^*$. Let $L \supset \mathbf{Q}$ be cyclic of degree u, with $\gcd(\Delta_L, n) = 1$ and suppose that ϕ_n generates $\operatorname{Gal}(L/\mathbf{Q})$. Suppose also that for every prime divisor p of t there exists an element $\alpha_p \in O_L/nO_L$ satisfying:*

$$\phi_n(\alpha_p) = \alpha_p^n. \tag{4.13}$$

Define

$$\beta_p = \alpha_p^{\frac{n^u-1}{p}} - 1 \quad \text{and} \quad \gamma_p = \alpha_p^{\frac{n^u-1}{p^k}},$$

where $p^k \parallel n^u - 1$. Suppose finally that

$$\operatorname{ord} \alpha_p = \operatorname{ord} \beta_p = n \quad \text{in the additive group of} \quad O_L/nO_L. \tag{4.14}$$

Then:

$$\Phi_t(\zeta) = 0 \quad \text{where} \quad \zeta = \prod_{p|t} \gamma_p.$$

Proof. Let $\sigma = \phi_n$ be a generator for $\operatorname{Gal}(L/\mathbf{Q})$. Let I be the ideal in O_L/nO_L that is generated by α_p; since $\sigma(\alpha_p) = \phi_n(\alpha_p) = \alpha_p^n$ we find $\sigma I \subset I$. But then $I = \sigma^u I \subset \sigma^{u-1} I \subset \cdots \subset \sigma I \subset I$ and so $\sigma I = I$. Writing O_L on a basis $\{b_1, b_2, \ldots, b_u\}$ over $\mathbf{Z}$, we know that $\Delta_L = \left(\det(\sigma^i b_j)\right)^2$ is coprime to n and therefore a unit in O_L/nO_L. Taking everything modulo n, we get a basis $\{\overline{b_1}, \overline{b_2}, \ldots, \overline{b_u}\}$ of O_L/nO_L over $\mathbf{Z}/n\mathbf{Z}$. Suppose now that $x \in I$, where $x = \sum x_j \overline{b_j}$ with coordinates $x_j \in \mathbf{Z}/n\mathbf{Z}$. Then $\sigma^i x = \sum x_j \sigma^i(\overline{b_j})$ and $\sigma^i x \in I$ for $i = 1, 2, \ldots, u$. So $\sum_j x_j \sigma^i(\overline{b}_j) \equiv 0 \bmod I$ for every i, which implies that $x_j \equiv 0 \bmod I$

since $\det(\sigma^i(\overline{b_j})) \in (O_L/nO_L)^*$. In other words, every $x \in I$ has coordinates in the ideal $I \cap \mathbf{Z}/n\mathbf{Z}$ of $\mathbf{Z}/n\mathbf{Z}$ over O_L/nO_L; this holds in particular for α_p itself. Since the additive order of α_p in O_L/nO_L is by assumption equal to n, the ideal $I \cap \mathbf{Z}/n\mathbf{Z}$ must be $\mathbf{Z}/n\mathbf{Z}$. This implies that $I = O_L/nO_L$ and therefore α_p is a unit in O_L/nO_L.

Since $\phi_n(\beta_p) \in \beta_p O_L/nO_L$, a similar argument shows that β_p is a unit.

Now $\alpha_p = \sigma^u \alpha_p = \alpha_p^{n^u}$ so $\alpha_p^{n^u-1} = 1$. Therefore the element γ_p has the property that $\gamma_p^{p^k} - 1 = 0$ while $\gamma_p^{p^{k-1}} - 1 = \beta_p$ is a unit. Thus γ_p is a zero of Φ_{p^k} in O_L/nO_L. Using the multiplicative properties of roots of unity, it is shown that ζ is a zero of Φ_t.

Thereby the proof of (4.12) is finished.

(4.15) Remark. If we have constructed the ring O_L/nO_L as in (4.11), the construction of the element ζ we need in our primality proof is now easy by (4.12): for every prime p in t we have to find an element α_p satisfying (4.13). This can be done by just taking a random choice for (the coordinates over $\mathbf{Z}/n\mathbf{Z}$ of) α_p and by checking (4.13) and (4.14). The latter is easy, since (4.14) is equivalent to

$$\text{(4.16)} \qquad \text{for every divisor } r \text{ of } n: \quad \alpha_p \notin r \cdot O_L/nO_L \quad \text{and} \quad \beta_p \notin r \cdot O_L/nO_L,$$

so one just checks that $\gcd(a_p b_p, n) = 1$ for some non-zero coefficient a_p of α_p and some non-zero coefficient b_p of β_p; either this gcd equals 1 and (4.14) holds, or one finds a factor of n. Verifying (4.13) takes one n-th powering in O_L/nO_L.

If n is prime, then (4.13) is trivially satisfied for any choice of α_p, while (4.14) holds for non-zero α_p with probability $1 - \frac{1}{p}$ (it only fails if α_p is a p-th power).

Note that the checks on the root β_p of Φ_p imply the existence of the root γ_p of Φ_{p^k}.

5. LUCAS-LEHMER TYPE TESTS.

As a first application in primality proving we show in this section how to use Theorem (2.8). The computationally explicit constructions of the previous section enable one to exhibit certain cyclotomic constellations; the mere existence of such a cyclotomic constellation may be enough to prove primality.

(5.1) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$; let $t \in \mathbf{Z}_{\geq 1}$, coprime to n and let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. Suppose that for every prime divisor l of u there exists a number field L_l that is cyclic of degree l^k, where $l^k \parallel u$, and such that $\mathrm{Gal}(L_l/\mathbf{Q}) = \langle \phi_n \rangle$. Let L be the composite of all L_l.*

Suppose moreover, that for every prime divisor p of t the element $\alpha_p \in O_L/nO_L$ satisfies:

$$\phi_n(\alpha_p) = \alpha_p^n$$

and that for the order in the additive group of O_L/nO_L:

$$\operatorname{ord} \alpha_p = \operatorname{ord}(\alpha_p^{\frac{n^u-1}{p}} - 1) = n.$$

Then:

$$\text{for every } r \mid n \text{ there exists } i \bmod u \text{ such that } r \equiv n^i \bmod t.$$

In particular, n is prime if $t > \sqrt{n}$ and none of $r_1, r_2, \ldots, r_{u-1}$ is a proper divisor of n, where r_i is the least positive integer satisfying $r_i \equiv n^i \bmod t$.

Proof. By Proposition (4.1) the field L is cyclic over $\mathbf{Q}$ of degree u, and ϕ_n is clearly a generator of the group $\mathrm{Gal}(L/\mathbf{Q})$. The element

$$\zeta = \prod_{\substack{p^k \parallel t \\ p \text{prime}}} \alpha_p^{\frac{n^u-1}{p^k}}$$

has the property that $\phi_n(\zeta) = \zeta^n$, by the hypotheses on α_p, while $\Phi_t(\zeta) = 0$ by Proposition (4.12). Thus L, ζ and ϕ_n satisfy the conditions of t-th cyclotomic constellations for n, so Theorem (2.8) proves the first assertion. The second holds obviously.

That proves (5.1).

(5.2) Remarks. For the construction of the fields L_l, one may use (4.8) and (4.10).

Theorem (5.1) leads to primality proofs for primes n for which we can find a small u and a large enough, completely factored divisor t of $n^u - 1$. Testing $\Phi_t(\zeta) = 0$ seems impossible without knowing the prime factorization of t, which makes it necessary that t is completely factored.

Arithmetic in O_L/nO_L is easy, using (4.11); as remarked in (4.15), the conditions on $\operatorname{ord} \alpha_p$ and $\operatorname{ord}(\alpha_p^{(n^u-1)/p} - 1)$ are checked by showing that one of the coordinates of each of these elements on a basis of O_L/nO_L over $\mathbf{Z}/n\mathbf{Z}$ is coprime to n.

The primality test implied by (5.1) is a generalization of well-known primality tests, that are classical for numbers n of a special form. By way of example we show to retrieve the classical formulation for a few of these tests; compare Chapter I.

(5.3) Corollary. *Let $n \in \mathbf{Z}_{\geq 2}$; let $t \mid n-1$. Suppose that for every prime divisor p of t the element $\alpha_p \in \mathbf{Z}/n\mathbf{Z}$ satisfies:*

$$\alpha_p = \alpha_p^n \quad \text{and} \quad \gcd(\alpha_p, n) = \gcd(\alpha_p^{\frac{n-1}{p}} - 1, n) = 1.$$

Then every divisor r of n satisfies:

$$r \equiv 1 \bmod t.$$

In particular: n is prime if $t > \sqrt{n}$.

Proof. This is the case that $u = 1$ in (5.1); then $L = \mathbf{Q}$, $\phi_n = \mathrm{id}$ and $\alpha_p \in \mathbf{Z}/n\mathbf{Z}$. For the condition on α_p, see (5.2) and use that $O_L = \mathbf{Z}$ now.

That proves (5.3).

This is the same as Pocklington's Theorem I.(6.14), combined for all p^k dividing t. A particular case is the following special purpose test for numbers of the form $hp^k + 1$ (with h small), in which finding a non-p-th power modulo n suffices. Note that this includes (the non-trivial implications in) Proth's Theorem I.(6.12) and Pepin's test I.(6.9).

(5.4) Corollary. *Let $n = hp^k + 1$ with p prime and $p^k > h$. If $\alpha \in \mathbf{Z}/n\mathbf{Z}$ satisfies:*

$$\alpha^n = \alpha \quad \text{and} \quad \gcd(\alpha, n) = \gcd(\alpha^{\frac{n-1}{p}} - 1, n) = 1,$$

then n is prime.

Proof. Take $t = p^k$ in (5.3).

As a final example, we prove the correctness of Lucas-Lehmer test for Mersenne numbers (see I.(7.9)) once more, using (5.1) with $u = 2$.

(5.5) Lucas-Lehmer test. *Let $n = 2^k - 1$ with $k > 2$ and define $e_i \in \mathbf{Z}$ for $i \geq 1$ by $e_1 = 4$ and $e_{i+1} = e_i^2 - 2$. Then:*

$$n \text{ is prime} \quad \Longleftrightarrow \quad e_{k-1} \equiv 0 \bmod n.$$

Proof. Suppose that k is even. Then $3 \mid n$ so n is composite. On the other hand $e_i \equiv -1 \bmod 3$ for $i > 1$ so n cannot divide e_{k-1}. In the rest of the proof we therefore assume that k is odd.

Let $L = L_2 = \mathbf{Q}(\sqrt{3}) \subset \mathbf{Q}(\zeta_{12})$, let σ the non-trivial automorphism of L and let $\zeta = 2^{\frac{k-1}{2}}(1 + \sqrt{3}) \in O_L/nO_L$; notice that $\zeta^{-1} = 2^{\frac{k-1}{2}}(-1 + \sqrt{3})$. Also, observe that

$$e_i \equiv \zeta^{2^i} + \zeta^{-2^i} \bmod n \text{ for } i \geq 1,$$

since

$$\zeta^2 + \zeta^{-2} = 2^{k+2} \equiv 1 \bmod n \text{ and } \zeta^{2^{i+1}} + \zeta^{-2^{i+1}} = (\zeta^{2^i} + \zeta^{-2^i})^2 - 2$$

in $\mathbf{Z}/n\mathbf{Z} \subset O_L/nO_L$.

By its definition, $n \equiv 3 \bmod 4$. Also, $k > 2$ is odd so $n \not\equiv 0 \bmod 3$; and since $n + 1 = 2^k \not\equiv 0 \bmod 3$ we have $n \equiv 1 \bmod 3$. Therefore $n \equiv 7 \bmod 12$ and

$$\phi_n(\sqrt{3}) = \phi_n(\zeta_{12} + \zeta_{12}^{-1}) = \zeta_{12}^7 + \zeta_{12}^{-7} = -\zeta_{12} - \zeta_{12}^{-1} = -\sqrt{3}$$

so

$$\phi_n(\zeta) = \phi_n(2^{\frac{k-1}{2}}(1 + \sqrt{3})) = 2^{\frac{k-1}{2}}(1 - \sqrt{3}) = -\zeta^{-1}.$$

Note in particular that ϕ_n generates the Galois group $\mathrm{Gal}(L/\mathbf{Q})$.

Suppose that n is prime. Then $\phi_n(\zeta) = \zeta^n$ by (2.3), and we get

$$0 = \zeta^n + \zeta^{-1} = \zeta^{2^{k-1}-1}(\zeta^{2^{k-1}} + \zeta^{-2^{k-1}})$$

so $\zeta^{2^{k-1}} + \zeta^{-2^{k-1}} = 0$, in other words: $e_{k-1} \equiv 0 \bmod n$.

For the converse, let $t = 2^{k+1}$. The order of n in $\left(\mathbf{Z}/t\mathbf{Z}\right)^*$ is $u = 2$. As we saw above, ϕ_n generates $\mathrm{Gal}(L/\mathbf{Q})$. Furthermore,

$$\phi_n(\zeta) = \zeta^n \iff \zeta^{2^{k-1}} + \zeta^{-2^{k-1}} = 0 \iff e_{k-1} \equiv 0 \bmod n$$

since $\phi_n(\zeta) = \phi_n(2^{\frac{k-1}{2}}(1+\sqrt{3})) = 2^{\frac{k-1}{2}}(1-\sqrt{3}) = -\zeta^{-1}$. Finally, the additive order of ζ is n, since $2 \nmid n$ (cf. (4.16)), and the same applies to $\zeta^{\frac{n^2-1}{2}} - 1$, since in O_L/nO_L

$$\zeta^{\frac{n^2-1}{2}} - 1 = \left(\zeta^{n+1}\right)^{\frac{n-1}{2}} - 1 = \left(\zeta\phi_n(\zeta)\right)^{\frac{n-1}{2}} - 1 = (-1)^{\frac{n-1}{2}} - 1 = -2,$$

because $n \equiv 3 \bmod 4$. Thus, if $e_{k-1} \equiv 0 \bmod n$ the conditions in Theorem (5.1) are verified (with $\alpha_p = \zeta$), and every divisor must be either 1 or n modulo t, with $t > n$.

This proves (5.5).

6. THE JACOBI SUM TEST.

In this section we continue our investigation of cyclotomic constellations that started in Section 2. In Theorem (2.8) we saw that the existence of t-th cyclotomic constellations for n implies that every divisor of n is a power of n modulo t. The first proposition below will be used to show that, under some mild extra conditions, the same conclusion will even hold modulo any t' that is built up from primes in t only.

It will be convenient to use the informal notation t^∞ for an integer that is the product of "large enough" powers of all primes dividing t; more specifically, we will use this to write $t' \mid t^\infty$ as an abbreviation for the statement that every prime divisor of t' divides t, and to use $\gcd(k, t^\infty)$ for the largest divisor of k built up from primes in t only.

(6.1) Proposition. *Let $n \in \mathbf{Z}_{\geq 2}$, $t \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,t) = 1$ and let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. Suppose that:*

(i) $\gcd(t, \frac{n^u-1}{t}) = 1$,

(ii) *for every $r \mid n$ there exists $i \bmod u$ such that $r \equiv n^i \bmod t$.*

Let $t' \in \mathbf{Z}_{\geq 1}$ and let u' be $\operatorname{ord} n$ in $(\mathbf{Z}/t'\mathbf{Z})^*$. Suppose moreover that:

(iii) $t' \mid t^\infty$,

(iv) $t' \not\equiv 0 \bmod 8$ *if* $t \equiv 2 \bmod 4$.

Then:

$$\text{for every } r \mid n \text{ there exists } i \bmod u' \text{ such that } r \equiv n^i \bmod t'.$$

Also, if $t \mid t'$, then $\gcd(t', \frac{n^{u'}-1}{t'}) = 1$.

Proof. Let r be a divisor of n and let i be such that $r \equiv n^i \bmod t$, as in (ii). Let p be a prime divisor of t'. If $o_p(t') \leq o_p(t)$ then $r \equiv n^i \bmod p^{o_p(t')}$. If $o_p(t') > o_p(t)$ then by (iii) and (i) we have $n^u \equiv 1 \bmod p^{o_p(t)}$ but $n^u \not\equiv 1 \bmod p^{o_p(t)+1}$. The cyclic subgroup $1+p^{o_p(t)} \subset (\mathbf{Z}/p^{o_p(t')}\mathbf{Z})^*$ of order $p^{o_p(t')-o_p(t)}$ is then generated by $n^u \bmod p^{o_p(t')}$, provided that for $p = 2$ we require that $8 \nmid t'$ in case $t \equiv 2 \bmod 4$. Therefore we can find j such that $rn^{-i} \equiv (n^u)^j \bmod p^{o_p(t')}$; here j is determined modulo $p^{o_p(t')-o_p(t)}$. But then we can combine these congruences for all p dividing t to get $rn^{-i} \equiv (n^u)^j \bmod t'$ simultaneously.

Since the observations above imply that $o_p(n^{up} - 1) = o_p(n^u - 1) + 1$ the second assertion follows.

That proves (6.1).

(6.2) Corollary. *Let $n \in \mathbf{Z}_{\geq 2}$, $t \in \mathbf{Z}_{\geq 1}$ with $t \equiv 0 \bmod 4$ and $\gcd(n,t) = 1$. Let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. Suppose that:*

(i) $\gcd(t, \frac{n^u-1}{t}) = 1$,

(ii) *for every $r \mid n$ there exists $i \bmod u$ such that $r \equiv n^i \bmod t$.*

Then for every divisor r of n, for every prime number l dividing t and every $d \in \mathbf{Z}_{\geq 1}$ there exists a unique $j \bmod l^d$ with the following property.

If $t' \in \mathbf{Z}_{\geq 1}$ satisfies

(iii) $t' \mid t^\infty$,

(iv) $l^d \mid u'$, *where u' is the order of n in $(\mathbf{Z}/t'\mathbf{Z})^*$,*

then the integer i' such that $r \equiv n^{i'} \bmod t'$ satisfies $i' \equiv j \bmod l^d$.

If moreover a t'-th cyclotomic constellation L, ζ, σ exists, then the integer k such that $\phi_r = \phi_n^k$ in $\mathrm{Gal}(L/\mathbf{Q})$ satisfies $k \equiv j \bmod l^d$.

Proof. First of all note that there exist positive integers t' satisfying (iii) and (iv): for k large enough, l^d divides the order of n modulo $l^k t$.

Fix a divisor r of n. Suppose that both t'_1 and t'_2 satisfy the conditions for t' in the statement of the corollary. Then so does $\mathrm{lcm}(t'_1, t'_2)$, and by (6.1) there exists i' such that $r \equiv n^{i'} \bmod \mathrm{lcm}(t'_1, t'_2)$. Clearly $i' \equiv i'_1 \bmod u'_1$ and $i' \equiv i'_2 \bmod u'_2$, where u'_1 and u'_2 are the orders of n modulo t'_1 and t'_2 and where $r \equiv n^{i'_1} \bmod t'_1$ while $r \equiv n^{i'_2} \bmod t'_2$. In particular $i'_1 \equiv i' \equiv i'_2 \bmod l^d$. That proves the first assertion.

The other assertion is now immediate from (2.8).

That proves (6.2).

As we saw in Section 5, the scope of the applications of Theorem (2.8) to primality testing is rather limited. We get much more powerful tools if besides the existence of a t-th cyclotomic constellation L, ζ, σ, we have additional information on the character group $\mathrm{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle)$. The kind of information we want to obtain, is that in O_L/nO_L

$$\frac{\tau(\chi)^n}{\phi_n \tau(\chi)} \in \langle\zeta\rangle, \tag{6.3}$$

for a set of primitive characters χ generating $\mathrm{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle)$; here the Gauss sum $\tau(\chi)$ is considered as an element of $O_L/nO_L[\zeta_s]$, cf. (3.6) and the discussion preceding that. The action of ϕ_n is extended to $O_L/nO_L[\zeta_s]$ by defining $\phi_n(\zeta_s) = 1$.

It will be convenient to write the action of ϕ_n exponentially, so

$$\tau(\chi)^{n-\phi_n} = \frac{\tau(\chi)^n}{\phi_n \tau(\chi)}.$$

Note that ϕ_n is only defined if n is coprime to the order of χ, and that for the division by $\phi_n\tau(\chi)$ we require s coprime to n (see (3.10)). In this case

$$\tau(\chi)^{n-\phi_n} = \frac{\tau(\chi)^n}{\tau(\chi^n)}$$

if n is coprime to the order of χ; by (3.11) we then know that (6.3) holds if n is prime.

(6.4) Proposition. *Let $n \in \mathbf{Z}_{\geq 2}$, $t \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,t) = 1$ and let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. Suppose that:*

(i) $\gcd(t, \frac{n^u-1}{t}) = 1$,

(ii) *L, ζ, σ forms a t-th cyclotomic constellation for n.*

Let $t' \in \mathbf{Z}_{\geq 1}$ and let u' be the order of n in $(\mathbf{Z}/t'\mathbf{Z})^*$. Suppose also that:

(iii) $t' \mid t^\infty$,

(iv) $t' \not\equiv 0 \bmod 8$ *if* $t \equiv 2 \bmod 4$.

Let $s \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,s) = 1$ and let $\chi \in \mathrm{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle)$ be a primitive character modulo s. Suppose that for some $z \in (O_L/nO_L)^*$:

(v) $\mu_\chi = (z\tau(\chi))^{n-\phi_n} \in \langle\zeta\rangle$

in the ring $(O_L/nO_L)[\zeta_s]$.

Then for every divisor r of n there exists $i \bmod \mathrm{lcm}(u',t)$ such that:

$$r \equiv n^i \bmod t' \tag{6.5}$$

and

$$\chi(r) = \chi(n)^i. \tag{6.6}$$

Furthermore $\mu_\chi = \chi(n)^{-n}$.

Proof. It suffices to consider the case that t' is a multiple of t^2; if necessary, replace t' by $\mathrm{lcm}(t', t^2)$. So let in the rest of the proof $t^2 \mid t'$; then $t \mid u'$.

The hypotheses of Theorem (2.8) are satisfied, and therefore $\phi_n = \sigma$, and for every divisor r of n there exists j such that both $r \equiv n^j \bmod t$ and $\phi_r = \phi_n^j$. But then the hypotheses for (6.1) are satisfied, and so there exists $k \bmod u'$ such that $r \equiv n^k \bmod t'$.

Note that both $\tau(\chi^n)$ and $\tau(\chi)$ are units in $O_L/nO_L[\zeta_s]^*$ by (3.10), so it makes sense to consider $\mu_\chi = \big(z\tau(\chi)\big)^{n-\phi_n}$. Since μ_χ is a power of ζ by hypothesis, $\phi_n(\mu_\chi) = \mu_\chi^n$. Hence we find for every $j \in \mathbf{Z}_{\geq 1}$:

$$\big(z\tau(\chi)\big)^{n^j-\phi_n^j} = \mu_\chi^{j\cdot n^{j-1}}.$$

From $\operatorname{ord}\chi \mid t$ and $t \mid n^u - 1$, we see that $\phi_n^u = \mathrm{id}$. By taking $j = u$ in the above

$$\big(z\tau(\chi)\big)^{n^u-1} = \big(z\tau(\chi)\big)^{n^u-\phi_n^u} \in \langle\zeta\rangle.$$

Thus $\operatorname{ord}(z\tau(\chi))$ divides $t(n^u - 1) = t^2 y$, if we abbreviate $y = \frac{n^u-1}{t}$ for the rest of this proof.

Let r be a prime divisor of n; by our hypotheses and the previous proposition there exists i such that $r \equiv n^i \bmod t^2$; notice that i is determined modulo u' and that $i \equiv k \bmod u'$, where k is such that $r \equiv n^k \bmod t'$. Also, $i \equiv j \bmod u$, where j is such that $r \equiv n^j \bmod t$, so $\phi_r = \phi_n^i$ and

$$ry \equiv n^i y \bmod t^2 y;$$

therefore ry and $n^i y$ are certainly congruent modulo $\operatorname{ord}(z\tau(\chi))$. Hence

$$\big(z\tau(\chi)\big)^{ry-\phi_r y} = \big(z\tau(\chi)\big)^{n^i y-\phi_n^i y} = \mu_\chi^{in^{i-1}y}.$$

Now r is prime, so $z^{r-\phi_r} \equiv 1 \bmod r$ by (2.3); moreover r is not a divisor of $\operatorname{ord}\chi$, so by Lemma (3.11)

$$\big(z\tau(\chi)\big)^{r-\phi_r} = \tau(\chi)^{r-\phi_r} = \chi(r)^{-r} \in O_L/rO_L$$

which implies by the above that

$$\chi(r)^{-ry} \equiv \big(z\tau(\chi)\big)^{(r-\phi_r)y} = \mu_\chi^{in^{i-1}y} \bmod r.$$

Since distinct powers of ζ are distinct modulo r by (2.1), and $\gcd(t, y) = \gcd(t, \frac{n^u-1}{t}) = 1$ by (i), we get

$$\chi(r)^{-r} = \mu_\chi^{in^{i-1}} \qquad \text{for every prime divisor } r \text{ of } n.$$

But $n^{i-1} \equiv rn^{-1} \bmod t$ by definition of i, hence the equality $\chi(r)^{-n} = \mu_\chi^i$; by multiplicativity this equality holds for *every* divisor of n. In particular, for $r = n$ we find $i = 1$ so $\chi(n)^{-n} = \mu_\chi$. Thus:

$$\chi(r) = \chi(n)^i, \qquad \text{for every divisor } r \text{ of } n.$$

This ends the proof for (6.4).

(6.7) Remarks. The use of the unit $z \in (O_L/nO_L)^*$ will become clear later on. You may think of it as an extra degree of freedom that will be exploited for practical purposes; for the time being one might as well put $z = 1$.

Now we are ready for the principal theorem of this section. It tells us what we know if we require that (6.4)(v) is satisfied for a set of generators for the characters modulo s.

(6.8) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$, $t \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,t) = 1$ and let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$; let $t_1 = \gcd(n^u - 1, t^\infty)$. Suppose that:*

(i) *L, ζ, σ forms a t_1-th cyclotomic constellation for n.*

Let $t' \in \mathbf{Z}_{\geq 1}$ and let u' be $\operatorname{ord} n$ in $(\mathbf{Z}/t'\mathbf{Z})^*$. Suppose that:

(ii) *$t' \mid t^\infty$,*

(iii) *$t' \not\equiv 0 \bmod 8$ if $t_1 \equiv 2 \bmod 4$,*

Let $s \in \mathbf{Z}_{\geq 1}$ such that $\gcd(nt, s) = 1$; suppose also that:

(iv) *$\exp(\mathbf{Z}/s\mathbf{Z})^* \mid t_1$.*

Suppose moreover that for some set Y generating the group $\operatorname{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle)$:

(v) *for all $\chi \in Y$ there exist $\mu_\chi \in \langle\zeta\rangle$ and $z \in (O_L/nO_L)^*$ such that*

$$(z\tau(\chi))^{n-\phi_n} = \mu_\chi.$$

Then for every divisor r of n there exists $i \bmod \operatorname{lcm}(u', t_1)$ such that:

$$r \equiv n^i \bmod st'.$$

Furthermore

(6.9) $$\mu_\chi = \chi(n)^{-n} \quad \text{for all} \quad \chi \in Y.$$

Proof. Fix a divisor r of n. As in the proof of Proposition (6.4) we may restrict to the case that t_1 divides u'. By (6.4) there exists $j \bmod u'$ such that:

(6.10) $$r \equiv n^j \bmod t'$$

and for all $\chi \in Y$:

(6.11) $$\chi(r) = \chi(n)^j;$$

note that j does not depend on χ since it is determined modulo the order of χ, a divisor of t_1, by (6.10). By the hypothesis on Y and multiplicativity, (6.11) holds for every character modulo s. The order of ζ is t_1 by (2.1) and $\exp(\mathbf{Z}/s\mathbf{Z})^* \mid t_1$ by hypothesis, so by (3.4):

$$r \equiv n^j \bmod s,$$

with j as in (6.10) and (6.11). Since s and t' are relatively prime this combines with (6.10) to $r \equiv n^j \bmod st'$; note that j only matters modulo u^*, where $u^* = \operatorname{ord} n$ in $(\mathbf{Z}/st'\mathbf{Z})^*$.

Assertion (6.9) is a consequence of the final statement in Proposition (6.4).

That proves (6.8).

One of the conditions in Theorem (6.8) is that $\exp(\mathbf{Z}/s\mathbf{Z})^*$ divides t, (if we require that $\gcd(t, \frac{n^*-1}{t}) = 1$). For primality testing purposes one wishes that s becomes large, while t should remain small. The following lemma shows that it is advisable to choose a highly composite, even value for t in order to get s large.

(6.12) Lemma. *Let $s \in \mathbf{Z}_{\geq 1}$. Then:*

$$\exp(\mathbf{Z}/s\mathbf{Z})^* \mid t \iff s \mid e(t)$$

where

$$e(t) = \begin{cases} 2 \cdot \prod_{\substack{q \text{ prime} \\ q-1 \mid t}} q^{o_q(t)+1}, & \text{if } t \text{ is even;} \\ 2, & \text{if } t \text{ is odd.} \end{cases} \tag{6.13}$$

Proof. For odd primes q, the group $(\mathbf{Z}/q^k\mathbf{Z})^*$ is annihilated by t if and only if $(q-1)q^{k-1}$ divides t. If $q = 2$, then $(\mathbf{Z}/q^k\mathbf{Z})^*$ is annihilated by t if and only if either $k \leq 2$ and t is even, or $k > 2$ and 2^{k-2} divides t. The result is now immediate from the Chinese remainder theorem.

The following corollary shows the use of Theorem (6.8) in primality testing. We use the notation $e(t)$ as in (6.13).

(6.14) Corollary. *Let $n \in \mathbf{Z}_{\geq 2}$, $s, t \in \mathbf{Z}_{\geq 1}$ with $4 \mid t$ and $\gcd(n, st) = 1$; let $u = \operatorname{ord} n$ in $(\mathbf{Z}/t\mathbf{Z})^*$ and let $t_1 = \gcd(n^u - 1, t^\infty)$ and $s_1 = s/\gcd(s, t^\infty)$. Suppose that*

(i) *L, ζ, σ forms a t_1-th cyclotomic constellation for n;*

(ii) $s_1 \mid e(t_1)$.

Suppose moreover that for some set Y generating the group $\mathrm{Hom}((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle)$:

(iii) *for all $\chi \in Y$ there exist $\mu_\chi \in \langle\zeta\rangle$ and $z \in (O_L/nO_L)^*$ such that*

$$(z\tau(\chi))^{n-\phi_n} = \mu_\chi.$$

Then for every divisor r of n there exists i modulo ord n *such that $r \equiv n^i \bmod s$, where* ord n *is the order of n in $(\mathbf{Z}/s\mathbf{Z})^*$.*

Proof. Take $t' = \dfrac{s}{s_1}$ in Theorem (6.8).

The Jacobi sum primality testing algorithm based on this corollary may be described as follows.

(6.15) Jacobi sum test. *Choose a positive multiple t of 4 such that $e(t) > \sqrt{n}$, for instance by using a table. Let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. Let s be a divisor of $e(t)$ such that $s > \sqrt{n}$ and let s_1 be the largest factor of s that is coprime to t. Construct a t_1-th cyclotomic constellation L, σ, ζ for n, where $t_1 = \gcd(n^u - 1, t^\infty)$. Finally verify that $\tau(\chi)^{n-\phi_n} \in \langle\zeta\rangle$ for every character in $\bigcup_{q \mid s_1} Y_q$, where q is prime and Y_q consists of characters of conductor q and order $p^k \parallel q-1$, one for each prime $p \mid q-1$. If these steps have been performed and $\gcd(n, st\Delta_L) = 1$, check for $i = 1, 2, \ldots,$ ord $n - 1$ whether r_i divides n, in case $1 < r_i \leq \sqrt{n}$; here $r_i \equiv n^i \bmod s$ and* ord n *is the order of n in $(\mathbf{Z}/s\mathbf{Z})^*$.*

(6.16) Remarks. If all conditions are satisfied and if in the final step no proper divisor of n is found, n must be prime as a consequence of Corollary (6.14).

Notice that s_1 is squarefree: suppose that q^k divides s_1 for some prime q and some $k \geq 2$, then q divides $\exp(\mathbf{Z}/s_1\mathbf{Z})^*$ and therefore t; by definition s_1 contains no factors q, a contradiction. Therefore $\bigcup_{q \mid s_1} Y_q$ generates $\mathrm{Hom}((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle)$. In Section 4 a construction for L was described that is guaranteed to work if n is not a (non-trivial) power in $\mathbf{Z}$, as well as a construction for ζ that is likely to work if n is indeed prime.

The verifications $\tau(\chi)^{n-\phi_n} \in \langle\zeta\rangle$ would take place in the ring $O_L/nO_L[\zeta_m]$ for a character of conductor m; for large m this ring is much too large for practical purposes. In Section 8 we will see how this can be overcome by using Jacobi sums instead of Gauss sums. There it will also be explained what the use of the mysterious unit z that first appeared in (6.4) will be in practice.

(6.17) Remark. The condition $4 \mid t$ is imposed on t in (6.14) and (6.15) to ensure that t' satisfies (6.4)(iv), or (6.8)(iii). Alternatively, one may do the following.

First of all notice that (6.4)(iv) is only restricting in the (rather special) case that $o_2(n^u - 1) = 1$ and $o_2(t') \geq 3$, where o_2 denotes the number of factors 2. In the notation of (6.15) that means that $t_1 \equiv 2 \bmod 4$, that $n \equiv 3 \bmod 4$, that u is odd, and that $e(t) \equiv s s_1^{-1} \equiv 8 \bmod 16$. This means that s/s_1 does not satisfy the condition for t', but $s/(2s_1)$ does; therefore in this case we find:

$$r \equiv n^j \bmod \frac{s}{2} \qquad \text{for every divisor } r \text{ of } n.$$

If one is not satisfied with this weaker conclusion, for instance because $s < 2\sqrt{n}$, one more Gauss sum test can help: let χ_0 be a quadratic character of conductor 8, then

$$\tau(\chi_0)^{n-1} \in \langle -1 \rangle \tag{6.18}$$

implies by (6.4), with $s = 8$, $u = 1$, $t = 2$, and $\zeta = -1$, that $\chi_0(r) = \chi_0(n)^j$, which gives together with $r \equiv n^j \bmod \frac{s}{2}$ that indeed $r \equiv n^j \bmod s$.

Notice that (6.18) is equivalent to

$$8^{\frac{n-1}{2}} \equiv \pm 1 \bmod n \tag{6.19}$$

since $\tau(\chi_0)^2 = \pm 8$ and since $n \equiv 3 \bmod 4$ in this case, so $\tau(\chi_0^n) = \tau(\chi_0)$.

(6.20) Remarks. Again, we would like to point out that in choosing t there are conflicting considerations. In order to get s larger than $\sqrt{n}$ one wishes t (and thus $e(t)$) to be large; on the other hand, in the final stage one has to perform usually about t trial divisions, and therefore t should not be too large. Below we will quote a theorem due to Odlyzko, that shows how fast t grows asymptotically (cf. [2], [30]).

Also, we refer to Section 9, for the description of Lenstra's algorithm with which one finds the divisors of n in a given residue class every modulo s efficiently, provided that $s > \sqrt[3]{n}$. At the cost of performing this algorithm, the condition that $e(t) > \sqrt{n}$ can be relaxed to $e(t) > \sqrt[3]{n}$.

Finally we present a table of nice values for $t > 1$ in the Jacobi sum test; here nice means by definition that the value for $e(t)$ is larger than any value $e(t')$ for a smaller divisor t' of the number $t_0 = 6983776800 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$.

(6.21) Theorem. *There exists an effectively computable positive constant c such that for every $n > 15$ there exists $t \in \mathbf{Z}_{\geq 1}$ with:*

$$t < (\log n)^{c \log\log\log n} \text{ and } e(t) > \sqrt{n}.$$

t	$\log_{10} e(t)$	t	$\log_{10} e(t)$
$2 = 2$	1.380	$128520 = 2^3 \cdot 3^3 \cdot 5 \cdot 7 \cdot 17$	145.431
$4 = 2^2$	2.380	$131040 = 2^5 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13$	151.897
$6 = 2 \cdot 3$	2.702	$166320 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11$	156.844
$12 = 2^2 \cdot 3$	4.816	$196560 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 13$	169.327
$24 = 2^3 \cdot 3$	5.117	$257040 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 17$	188.309
$30 = 2 \cdot 3 \cdot 5$	5.235	$332640 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11$	206.979
$36 = 2^2 \cdot 3^2$	8.140	$393120 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 13$	215.405
$60 = 2^2 \cdot 3 \cdot 5$	9.833	$514080 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 17$	223.283
$72 = 2^3 \cdot 3^2$	10.304	$655200 = 2^5 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 13$	232.767
$108 = 2^2 \cdot 3^3$	10.654	$720720 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13$	237.414
$120 = 2^3 \cdot 3 \cdot 5$	11.747	$831600 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11$	251.010
$144 = 2^4 \cdot 3^2$	11.836	$942480 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 17$	251.021
$180 = 2^2 \cdot 3^2 \cdot 5$	15.415	$982800 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 13$	260.117
$240 = 2^4 \cdot 3 \cdot 5$	15.660	$1081080 = 2^3 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$	263.037
$360 = 2^3 \cdot 3^2 \cdot 5$	19.192	$1285200 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 17$	272.555
$420 = 2^2 \cdot 3 \cdot 5 \cdot 7$	20.574	$1413720 = 2^3 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 17$	283.806
$540 = 2^2 \cdot 3^3 \cdot 5$	23.095	$1441440 = 2^5 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13$	301.222
$720 = 2^4 \cdot 3^2 \cdot 5$	23.105	$1663200 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11$	315.558
$840 = 2^3 \cdot 3 \cdot 5 \cdot 7$	24.936	$1965600 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 13$	326.018
$1008 = 2^4 \cdot 3^2 \cdot 7$	25.465	$2162160 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$	349.475
$1080 = 2^3 \cdot 3^3 \cdot 5$	26.872	$2827440 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 17$	357.833
$1200 = 2^4 \cdot 3 \cdot 5^2$	29.004	$3341520 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 13 \cdot 17$	389.642
$1260 = 2^2 \cdot 3^2 \cdot 5 \cdot 7$	31.059	$3603600 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$	396.884
$1680 = 2^4 \cdot 3 \cdot 5 \cdot 7$	33.430	$4324320 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$	455.899
$2016 = 2^5 \cdot 3^2 \cdot 7$	33.886	$5654880 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 17$	458.434
$2160 = 2^4 \cdot 3^3 \cdot 5$	36.757	$6683040 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 13 \cdot 17$	469.891
$2520 = 2^3 \cdot 3^2 \cdot 5 \cdot 7$	40.687	$7207200 = 2^5 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$	494.198
$3360 = 2^5 \cdot 3 \cdot 5 \cdot 7$	42.073	$10810800 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$	560.776
$3780 = 2^2 \cdot 3^3 \cdot 5 \cdot 7$	44.198	$16707600 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 13 \cdot 17$	575.923
$5040 = 2^4 \cdot 3^2 \cdot 5 \cdot 7$	52.185	$18378360 = 2^3 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	599.160
$7560 = 2^3 \cdot 3^3 \cdot 5 \cdot 7$	57.704	$21621600 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$	716.709
$8400 = 2^4 \cdot 3 \cdot 5^2 \cdot 7$	59.712	$36756720 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	762.754
$10080 = 2^5 \cdot 3^2 \cdot 5 \cdot 7$	64.132	$61261200 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	819.989
$12600 = 2^3 \cdot 3^2 \cdot 5^2 \cdot 7$	68.994	$73513440 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	966.850
$15120 = 2^4 \cdot 3^3 \cdot 5 \cdot 7$	79.352	$122522400 = 2^5 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	1038.433
$25200 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7$	89.622	$183783600 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	1171.776
$30240 = 2^5 \cdot 3^3 \cdot 5 \cdot 7$	95.780	$367567200 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	1501.792
$42840 = 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 17$	101.235	$698377680 = 2^4 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	1532.790
$50400 = 2^5 \cdot 3^2 \cdot 5^2 \cdot 7$	101.569	$1163962800 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	1650.980
$55440 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11$	106.691	$1396755360 = 2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	1913.604
$65520 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13$	115.895	$2327925600 = 2^5 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	2082.848
$75600 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7$	116.790	$3491888400 = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	2388.470
$85680 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 17$	129.398	$6983776800 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	3010.872
$110880 = 2^5 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11$	137.324		

7. COMBINING JACOBI SUM AND LUCAS-LEHMER TYPE TESTS.

The aim of this section is to give the proof of Theorem (7.1) below, which combines Theorem (2.8) and Theorem (6.8). The primality testing algorithm outlined briefly in (7.7) and described in detail in Chapter IV is the main application of this theorem. The important new feature here is, that special properties of a particular n are used (as in Lucas-Lehmer type tests based on (2.8)) as well as the power of the Jacobi sum test, which works almost irrespective of the properties of n (other than its size).

After the proof of Theorem (7.1) we will comment upon its complicated hypotheses.

(7.1) Theorem. *Let $n \in \mathbf{Z}_{\geq 2}$, $s,t,v \in \mathbf{Z}_{\geq 1}$ with $\gcd(n,stv) = 1$, and let u,w be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$ and in $(\mathbf{Z}/v\mathbf{Z})^*$ respectively. Let $t_1 = \gcd(n^u - 1, t^\infty)$ and let $s_1 = s/\gcd(s,t^\infty)$.*

Suppose that:

(i) $t_1 \equiv 0 \bmod 4$;

(ii) *L, ζ, σ forms an $\operatorname{lcm}(t_1,v)$-th cyclotomic constellation for n;*

(iii) $\exp(\mathbf{Z}/s_1\mathbf{Z})^* \mid t_1$;

(iv) *for every prime l dividing t:*

$$0 < o_l(u) < o_l(\gcd(t_1,w)) \quad \Rightarrow \quad o_l(v) = o_l(n^f - 1),$$

where $f = \operatorname{lcm}(u,w) = [L : \mathbf{Q}]$;

(v) *$X \subset \operatorname{Hom}((\mathbf{Z}/\operatorname{lcm}(s_1,v)\mathbf{Z})^*, \langle\zeta\rangle)$ is a set of characters that contains generators for the group $\operatorname{Hom}((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle)$ and that also contains for every prime power $l^d > 1$ for which $l^d \parallel \gcd(t_1,w)$, but $l \nmid u$, a character χ of order l^d and conductor m, with the property that $L_{l^d} \cong \mathbf{Q}(\zeta_m)^{\ker\chi}$, where $L_{l^d} \subset L$ is the subfield of degree l^d;*

(vi) *for every $\chi \in X$ there exist $\mu_\chi \in \langle\zeta\rangle$ and $z \in (O_L/nO_L)^*$ such that*

$$(z\tau(\chi))^{n-\phi_n} = \mu_\chi.$$

Then for every divisor r of n there exists $i \bmod \operatorname{lcm}(t_1,u,w)$ such that:

$$r \equiv n^i \bmod \operatorname{lcm}(s_1 t, v).$$

Proof. The field L in the cyclotomic constellation L, ζ, σ of (ii) has degree $f = \operatorname{lcm}(u,w)$ over $\mathbf{Q}$. By (2.11) therefore, L, ζ, σ contains a t_1-th cyclotomic sub-constellation L_1, ζ_1, σ_1 of degree u and a v-th cyclotomic sub-constellation L_2, ζ_2, σ_2 of degree w.

Let r be a fixed divisor of n, and let M be a sufficiently large integer (in fact, it will suffice that $M \geq 2$, and that $M \geq o_l(v)$ for each prime divisor l of t). By (6.8), with $t' = t_1^M$, we see that there exists i such that

$$r \equiv n^i \bmod s_1 t_1^M. \tag{7.2}$$

Then also $r \equiv n^i \bmod t_1$, so by (2.8) we have $\phi_r = \phi_n^i \in \mathrm{Gal}(L_1/\mathbf{Q})$. Likewise, there exists $j \pmod w$ such that $\phi_r = \phi_n^j \in \mathrm{Gal}(L_2/\mathbf{Q})$, and

$$r \equiv n^j \bmod v \tag{7.3}$$

by (2.8).

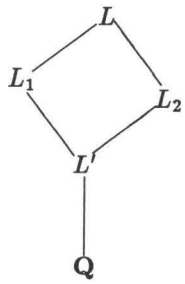
To obtain the statement of the theorem from (7.2) and (7.3), it suffices to prove that

$$i \equiv j \bmod l^d, \tag{7.4}$$

for every prime $l \mid \gcd(w, \mathrm{lcm}(t_1, u))$, where d is such that $l^d \parallel \gcd(w, \mathrm{lcm}(t_1, u))$.

In the rest of the proof l will be a prime divisor of w. We distinguish three cases.

(a) First suppose that $l^d \mid u$. In this case l^d divides both u and w; therefore L has a subfield L' of degree l^d over $\mathbf{Q}$, that is contained in L_1 as well as in L_2.



Since ϕ_n generates $\mathrm{Gal}(L_1/\mathbf{Q})$, we find by restriction to L' that $\phi_r = \phi_n^{i'} \in \mathrm{Gal}(L'/\mathbf{Q})$ for some i'. But $\mathrm{Gal}(L'/\mathbf{Q}) \cong \mathrm{Gal}(L_1/\mathbf{Q})/\langle \phi_n^{l^d} \rangle$, and $\phi_r = \phi_n^i \in \mathrm{Gal}(L_1/\mathbf{Q})$, so $i \equiv i' \bmod l^d$, with i as in (7.2). Using L_2 instead of L_1 we get similarly that $j \equiv i' \bmod l^d$, with j as in (7.3). Thus $i \equiv j \bmod l^d$ in this case.

(b) Next assume that $l \mid u$, but $l^d \nmid u$. In this case l^d divides both w and t_1. Let $h = o_l(v)$; then $h \leq M$ by the choice of M, so from (7.2) and (7.3) we see that $n^i \equiv r \equiv n^j \bmod l^h$. Therefore $i \equiv j \bmod e$, where e is the order of n modulo l^h. To prove (7.4) it therefore suffices to show that l^d divides e.

By (iv) we have $h = o_l(n^f - 1)$. Since l divides t it divides $n^u - 1$; and u divides f/l, since we are in case (b), so l divides $n^{f/l} - 1$. But then $h = o_l(n^f - 1) > o_l(n^{f/l} - 1)$, which means that e divides f but does not divide f/l. Therefore $o_l(e) = o_l(f)$, and since f is divisible by l^d this proves what we want.

(c) Finally assume that $l \nmid u$. Then $l^d \mid t_1$. Let $f = \operatorname{lcm}(u, w)$ as before; then $o_l(f) = o_l(w) \geq d$. By assumption (v) there exists a character χ in X of order l^d and conductor m such that $L_{l^d} \cong \mathbf{Q}(\zeta_m)^{\ker \chi}$, and for which by (vi) we have $(z\tau(\chi))^{n-\phi_n} \in \langle\zeta\rangle$, for some unit z in O_L/nO_L.

Apply Proposition (6.4), with t_1, u, t_1^2, ut_1, L_1, m in the roles of t, u, t', u', L, s, respectively. Then it follows that there exists an integer k modulo ut_1 for which

$$r \equiv n^k \bmod t_1^2, \quad \text{and} \quad \chi(r) = \chi(n)^k. \tag{7.5}$$

The latter equality means that $(r \bmod m) \equiv (n \bmod m)^k \bmod \ker\chi$ in the group $(\mathbf{Z}/m\mathbf{Z})^*$, and so

$$\phi_r \equiv \phi_n^k \bmod \ker\chi \tag{7.6}$$

in $\operatorname{Gal}(\mathbf{Q}(\zeta_m)/\mathbf{Q})$. Hence, $\phi_r = \phi_n^k$ in $\operatorname{Gal}(\mathbf{Q}(\zeta_m)^{\ker\chi}/\mathbf{Q})$, where $\mathbf{Q}(\zeta_m)^{\ker\chi} \cong L_{l^d}$ by assumption (v). But also $\phi_r = \phi_n^j$ in $\operatorname{Gal}(L_{l^d}/\mathbf{Q})$, because $L_{l^d} \subset L_2$. Since $\operatorname{Gal}(L_{l^d}/\mathbf{Q})$ is generated by ϕ_n this implies that $k \equiv j \bmod l^d$.

On the other hand, from (7.2) and (7.5) we see that $k \equiv i \bmod t_1$, since t_1 divides the order of n modulo t_1^2. In particular $k \equiv i \bmod l^d$. So $i \equiv k \equiv j \bmod l^d$, and we have (7.4) again.

That completes the proof of (7.1).

Informally, the primality test based on Theorem (7.1) may be described as follows.

(7.7) Cyclotomy test. *Find a completely factored integer v, and let w be the order of n in $(\mathbf{Z}/v\mathbf{Z})^*$. Choose a positive multiple t of 4 such that $\operatorname{lcm}(e(t), v) > \sqrt{n}$, for instance by using a table. Let u be the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$. For every prime divisor l of t for which $0 < o_l(u) < o_l(\gcd(t_1, w))$, make sure that v contains as many factors l as $n^f - 1$ does, where $f = \operatorname{lcm}(u, w)$. Let s be a divisor of $e(t)$ such that $\operatorname{lcm}(s, v) > \sqrt{n}$ and let s_1 be the largest factor of s that is coprime to t. Construct an $\operatorname{lcm}(t_1, v)$-th cyclotomic constellation L, σ, ζ for n, where $t_1 = \gcd(n^u - 1, t^\infty)$. Verify that $\tau(\chi)^{n-\phi_n} \in \langle\zeta\rangle$ for every character in $\bigcup_{q \mid s_1} Y_q$, where q is prime and Y_q consists of characters of conductor*

q and order $p^k \parallel q-1$, one for each prime $p \mid q-1$, as well as for a character of order l^d in $\mathrm{Hom}\big((\mathbf{Z}/\mathrm{lcm}(s_1,v)\mathbf{Z})^*, \langle\zeta\rangle\big)$, for every $l^d \parallel \gcd(t_1,w)$ with $l \nmid u$, as in (7.1)(v). If these steps have been performed and $\gcd(n, stuvw\Delta_L) = 1$, check for $i = 1,2,\ldots,\mathrm{ord}\, n-1$ whether r_i divides n, in case $1 < r_i \leq \sqrt{n}$; here $r_i \equiv n^i \bmod \mathrm{lcm}(s,v)$ and $\mathrm{ord}\, n$ is the multiplicative order of n modulo $\mathrm{lcm}(s,v)$.

(7.8) Remarks. If all conditions in (7.7) are satisfied, and if in the final step no proper divisor of n is found, n must be prime.

We briefly comment upon the conditions in Theorem (7.1).

Concerning the condition $4 \mid t_1$ imposed on t in (i), the remark made in (6.17) carries over. The conditions (ii) and (iii) are also found in Theorem (6.8). In the next section we will explain how the verification of $\tau(\chi)^{n-\phi_n} \in \langle\zeta\rangle$ can be done within O_L/nO_L, and also how z enables us to combine the verifications for several characters (cf. (6.16)).

The mechanism that ensures the compatibility of the Lucas-Lehmer part of the above theorem (the congruence (7.3), derived from (2.8)) and the Jacobi sum part (congruence (7.2), obtained from Theorem (6.8)), is that of choosing the same extensions: whenever for both parts an extension of l-th power degrees is needed, we use for the smaller one the subextension inside the larger. In certain cases we need to impose some extra conditions for this compatibility; these are contained in (iv) and (v).

Condition (iv) comes down to the following. If both the degree of the extension necessary for the Lucas-Lehmer part and the degree of the extension needed for the Jacobi sum part contain an l-power, but that power is larger in the former, that is, if $0 < o_l(u) < o_l(w)$, then we require that v contains all h factors l that occur in $n^f - 1$ (where f is the total degree $\mathrm{lcm}(u,w)$). This means that in the Lucas-Lehmer part of the test the existence of an l^h-th root of unity in O_L/nO_L must be shown. This is used in part (b) of the proof.

In case there is no common l-th power degree extension for the Lucas-Lehmer and the Jacobi sum parts because l does not divide the degree u used in the latter, we need a "special" character in X for which (vi) is checked. The ordinary characters in (v), that is, those necessary for generating $\mathrm{Hom}\big((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle\big)$, serve the same purpose as in Theorem (6.8). Note that the special characters in X may have some special properties. Although $l^d \mid \lambda(m)$ will hold necessarily (where l^d is the order and m the conductor), we did in particular not require that $l^d \parallel \lambda(m)$; in addition it may very well be that such a special character χ is the only character of conductor m that is in X. On the other hand, if one

is fortunate, it may happen that χ is already in X as a generator of $\mathrm{Hom}((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle)$, in which case no extra Jacobi sum test is necessary.

As in Theorem (6.8), it is possible to replace t in Theorem (7.1) by t', an integer built up from primes in t only. For $t' > t$ it usually implies in the final step of (7.7) that more potential divisors r_i have to be tested.

The primality test (7.7) that results from (7.1), consists of the choice of parameters s, t, u, v, and w, and a set of characters X, satisfying (i)–(vi), and such that $\mathrm{lcm}(s_1 t, v) > \sqrt{n}$. The problem of course is, *how* to choose the parameters in such a way that the necessary verifications can be done efficiently. That is the subject of the next chapter.

8. JACOBI SUMS.

In this section we present some results on Jacobi sums that are used in the primality testing algorithm. Jacobi sums will be used for two purposes, both aimed at reducing the amount of work to be done for the Jacobi sum test. Firstly, Jacobi sums will make it possible to verify identities of the type

$$\frac{\tau(\chi)^n}{\tau(\chi^n)} \in \langle \zeta \rangle$$

without leaving the ring O_L/nO_L, cf. (6.16). Secondly, Jacobi sums will enable us to do these verifications for several characters at the same time.

We recall from Section 3 that characters modulo s take on their values in the cyclic subgroup $\langle \zeta \rangle$ of the unit group of some commutative ring A with 1; here ζ is a zero of a cyclotomic polynomial Φ_t with $t \cdot 1 \neq 0$ in A, and $\exp(\mathbf{Z}/s\mathbf{Z})^* \mid \operatorname{ord} \zeta$.

(8.1) Definition. Let χ_1 and χ_2 be characters of conductors s_1 and s_2. Define $s = \operatorname{lcm}(s_1, s_2)$. Let A be such that $s \in A^*$. The *Jacobi sum* $J(\chi_1, \chi_2)$ is the element of $A[\zeta_s]$ defined by

$$J(\chi_1, \chi_2) = \frac{\tau(\chi_1)\tau(\chi_2)}{\tau(\chi_1\chi_2)}.$$

(8.2) Remarks. Jacobi sums are well-defined this way as a consequence of (3.10), and the fact that by (2.2) $\tau(\chi_1)$, $\tau(\chi_2)$ and $\tau(\chi_1\chi_2)$ may all be regarded as elements of $A[\zeta_s]$. Note that by (3.10) in fact $J(\chi_1, \chi_2) \in A[\zeta_s]^*$.

(8.3) Lemma.

(i) $J(\chi, 1) = 1$ *for any character* χ.

(ii) $J(\chi, \chi^{-1}) = \chi(-1)s$, *where* s *is the conductor of* χ.

(iii) *If* s *is prime and* χ_1, χ_2 *and* $\chi_1\chi_2$ *are primitive characters modulo* s, *then:*

$$J(\chi_1, \chi_2) = \sum_{x=0}^{s-1} \chi_1(x)\chi_2(1-x).$$

Proof.

(i) Immediate from the definition.

(ii) This is (3.10), since $\tau(1) = 1$.

(iii) This follows from:

$$\begin{aligned}\tau(\chi_1)\tau(\chi_2) &= \Big(\sum_{x=0}^{q-1}\chi_1(x)\zeta_q^x\Big)\Big(\sum_{y=0}^{q-1}\chi_2(y)\zeta_q^y\Big)\\ &= \sum_{x,y=0}^{q-1}\chi_1(x)\chi_2(y)\zeta_q^{x+y} = \sum_{z=0}^{q-1}\Big(\sum_{x=0}^{q-1}\chi_1(x)\chi_2(z-x)\Big)\zeta_q^z\\ &= \sum_{z=1}^{q-1}\Big(\sum_{x=0}^{q-1}\chi_1(zx)\chi_2(z-zx)\Big)\zeta_q^z + \sum_{x=0}^{q-1}\chi_1(x)\chi_2(-x)\\ &= \sum_{z=1}^{q-1}\Big(\chi_1\chi_2(z)\sum_{x=0}^{q-1}\chi_1(x)\chi_2(1-x)\Big)\zeta_q^z + \sum_{x=0}^{q-1}\chi_1(x)\chi_2(-x)\\ &= \sum_{x=0}^{q-1}\chi_1(x)\chi_2(1-x)\tau(\chi_1\chi_2) + 0,\end{aligned}$$

by (3.3) since $\chi_1\chi_2 \neq 1$.

(8.4) Remarks. For characters of the same prime conductor whose product is non-principal, (8.3)(iii) is often used as definition for the Jacobi symbol. Notice that in this case we can define $J(\chi_1,\chi_2)$ even if not $q \in A^*$. Also remark that here clearly $J(\chi_1,\chi_2) \in A$; below we will see that this is more generally true.

(8.5) Lemma. *Let χ_1 and χ_2 be characters of conductors s_1 and s_2 respectively, and let $s = \operatorname{lcm}(s_1,s_2)$. If $s \in A^*$, then $J(\chi_1,\chi_2) \in A^*$.*

Proof. Define the automorphism σ_h on $A[\zeta_s]$, for h coprime to s, by $\sigma_h(\zeta_s) = \zeta_s^h$ and $\sigma_h(a) = a$ for $a \in A$. Then obviously $A \subset B$, if we define:

$$B = \{z \in A[\zeta_s]: \quad \sigma_h(z) = z \text{ for every } h \text{ coprime to } s\}.$$

Using a Vandermonde determinant and Lemma (2.1) (as in the proof of (2.10)) one shows that $\zeta_s^0, \zeta_s^1, \ldots, \zeta_s^{\phi(s)-1}$ are linearly independent over B, hence $B[\zeta_s] = A[\zeta_s]$, so $A = B$. Since

$$\sigma_h\Big(\frac{\tau(\chi_1)\tau(\chi_2)}{\tau(\chi_1\chi_2)}\Big) = \frac{\tau_h(\chi_1)\tau_h(\chi_2)}{\tau_h(\chi_1\chi_2)} = \frac{\chi_1^{-1}(h)\chi_2^{-1}(h)}{(\chi_1\chi_2)^{-1}(h)}\frac{\tau(\chi_1)\tau(\chi_2)}{\tau(\chi_1\chi_2)} = \frac{\tau(\chi_1)\tau(\chi_2)}{\tau(\chi_1\chi_2)}$$

we find that $J(\chi_1,\chi_2) = \tau(\chi_1)\tau(\chi_2)/\tau(\chi_1\chi_2) \in A$. The result now follows from the remark made in (8.2) that $J(\chi_1,\chi_2) \in A[\zeta_s]^*$.

(8.6) Remarks. Lemma (8.5) will be used for the second purpose mentioned in the introduction of this section, as follows. In applying (7.1) one has to check that for some unit z in the relevant ring, $(z\tau(\chi))^{n-\phi_n} \in \langle\zeta\rangle$, for a set of characters including a set of generators for $\mathrm{Hom}((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle)$. In (8.9) below we will see that $\tau(\chi)^{n-\phi_n}$ can be calculated as a small product times an enormous exponentiation (roughly an n-th powering) of a product of Jacobi sums. Of course one would like to perform this n-th powering as few times as possible; (8.5) enables us to combine the calculations for two characters. For, checking that $\big(\tau(\chi_1)\tau(\chi_2)\big)^{n-\phi_n} \in \langle\zeta\rangle$ proves that $\big(J(\chi_1,\chi_2)\tau(\chi_1\chi_2)\big)^{n-\phi_n}$ has the same property, and since $J(\chi_1,\chi_2)$ is a unit, the condition for the character $\chi_1\chi_2$ has been verified. If we make sure that $\langle\chi_1\chi_2\rangle$ is the same subgroup as $\langle\chi_1,\chi_2\rangle$, which we do by ensuring that the orders of χ_1 and χ_2 are coprime, we obtain the same information by essentially halving the amount of work.

This introduces an interesting optimization problem: how to combine tests for characters in such a way that the amount of work to be done is minimized? We will answer this question in Chapter III. We will choose the set of characters generating $\mathrm{Hom}((\mathbf{Z}/s_1\mathbf{Z})^*, \langle\zeta\rangle)$ to consist of characters of conductor q, a prime divisor of s_1, and order p^k, a prime power exactly dividing $q-1$. One has to keep in mind here that the computations for a character of order a prime power p^k will be done in an extension ring O_L/nO_L of degree u, the order of n in $(\mathbf{Z}/p^k\mathbf{Z})^*$.

This choice implies that for the first application mentioned in the introduction to this section, we are only interested in the case of prime conductor.

(8.7) Lemma. *Let χ be a character of conductor s and suppose that $s \in A^*$. For every integer $h \geq 0$:*

$$\frac{\tau(\chi)^h}{\tau(\chi^h)} = \prod_{j=1}^{h-1} J(\chi,\chi^j).$$

Proof. For $h = 0, 1$ the right hand side is the empty product, which equals 1 by definition; the left hand side equals 1 as well in both cases (χ^0 is the principal character and $\tau(1) = 1$).

For $h > 1$ we see from (3.10) that $\tau(\chi^j) \in A[\zeta_s]^*$. By (8.1) we then have in $A[\zeta_s]^*$:

$$\prod_{j=1}^{h-1} J(\chi,\chi^j) = \frac{\tau(\chi)\tau(\chi)}{\tau(\chi^2)} \cdot \frac{\tau(\chi)\tau(\chi^2)}{\tau(\chi^3)} \cdots \frac{\tau(\chi)\tau(\chi^{h-1})}{\tau(\chi^h)} = \frac{\tau(\chi)^h}{\tau(\chi^h)}.$$

That proves (8.7).

(8.8) Corollary. *Let χ be a character of conductor s and suppose that $s \in A^*$. Then:*

$$\tau(\chi)^{\operatorname{ord}\chi} = \chi(-1)s \prod_{j=1}^{\operatorname{ord}\chi-2} J(\chi,\chi^j) = \chi(-1)s\frac{\tau(\chi)^{\operatorname{ord}\chi-1}}{\tau(\chi^{\operatorname{ord}\chi-1})}.$$

Proof. Applying (3.10) and (8.7) we find

$$\begin{aligned}\tau(\chi)^{\operatorname{ord}\chi} &= \tau(\chi)\tau(\chi^{\operatorname{ord}\chi-1})\frac{\tau(\chi)^{\operatorname{ord}\chi-1}}{\tau(\chi^{\operatorname{ord}\chi-1})}\\ &= \chi(-1)s\,\frac{\tau(\chi)^{\operatorname{ord}\chi-1}}{\tau(\chi^{\operatorname{ord}\chi-1})}\\ &= \chi(-1)s\prod_{j=1}^{\operatorname{ord}\chi-2} J(\chi,\chi^j),\end{aligned}$$

which proves (8.8).

(8.9) Corollary. *Let χ be a character of conductor s and suppose that $s \in A^*$. Then for every $h \in \mathbf{Z}_{\geq 0}$:*

$$\frac{\tau(\chi)^h}{\tau(\chi^h)} = \Big(\prod_{j=1}^{h^*-1} J(\chi,\chi^j)\Big)\Big(\chi(-1)s\prod_{j=1}^{\operatorname{ord}\chi-2} J(\chi,\chi^j)\Big)^l$$

where $0 \leq h^* < \operatorname{ord}\chi$ with $h^* \equiv h \bmod \operatorname{ord}\chi$, and $h = h^* + l\operatorname{ord}\chi$. In particular:

$$\frac{\tau(\chi)^h}{\tau(\chi^h)} \in A^*.$$

Proof. With h^* and l as defined in the statement:

$$\frac{\tau(\chi)^h}{\tau(\chi^h)} = \frac{\tau(\chi)^{h^*}\tau(\chi)^{l\operatorname{ord}\chi}}{\tau(\chi^{h^*})}.$$

The identity follows immediately from (8.7) and (8.8). By (3.10) the element $\tau(\chi)^h/\tau(\chi^h)$ is a unit in $A[\zeta_s]$; but $J(\chi,\chi^j) \in A^*$ and the result follows. That proves (8.9).

(8.10) Remarks. The main use of (8.7)–(8.9) is that it shows that $\tau(\chi)^h/\tau(\chi^h)$ can always be expressed as a product of Jacobi sums in A^*. That means that we can check $\tau(\chi)^n/\tau(\chi^n) \in \langle\zeta\rangle$, as necessary in the Jacobi sum test, in A^*. In practice we will want to

express $\tau(\chi)^h/\tau(\chi^h)$ as a product of as few Jacobi sums as possible, and to avoid inverting elements in A we would like to have all exponents positive. In IV.(2.4) an algorithm is given that yields good results (these results can be found in the Appendix). To explain some of the ideas behind this algorithm, we give an example below. But first there is an obvious but very useful identity that we state for future reference.

(8.11) Lemma. *Let χ be a character of prime conductor q and suppose that $q \in A^*$. If $\chi^h \neq 1 \neq \chi^l$, then for every $i, j \in \mathbf{Z}$:*

$$\frac{\tau(\chi^h)}{\tau(\chi^l)} = \frac{\chi^h(-1)\tau(\chi^{i\cdot \operatorname{ord}\chi - l})}{\chi^l(-1)\tau(\chi^{j\cdot \operatorname{ord}\chi - h})}.$$

Proof. This is a consequence of the fact that $\tau(\chi^{j\cdot \operatorname{ord}\chi - h}) = \tau(\chi^{-h})$, and of (3.10), implying that

$$\frac{\tau(\chi^h)}{\tau(\chi^l)} = \frac{\chi^l(-1)q\tau(\chi^{j\cdot \operatorname{ord}\chi - h})}{\chi^h(-1)q\tau(\chi^{i\cdot \operatorname{ord}\chi - l})}.$$

(8.12) Example. We propose to generate expressions for $\tau(\chi)^h/\tau(\chi^h)$, with $h = 1, 2, \ldots, 17$ for a character of order 17 with values in a ring in which the conductor of χ is a unit.

We use the exponential notation again, so $\tau(\chi)^{\sigma_h} = \sigma_h\tau(\chi)$, where $\sigma_h\zeta = \zeta^h$, so $\tau(\chi)^{\sigma_h} = \tau(\chi^h)$. We are after expressions

$$\tau(\chi)^{i-\sigma_i} = \prod J(\chi^a, \chi^b)^{e_J}$$

for $i = 1, 2, \ldots, 17$; here the product ranges over the Jacobi sums $J(\chi^a, \chi^b)$, so in fact over the pairs (a, b) with $1 \leq a, b \leq 17$. We want the exponents e_J to be of the form $\sum_{j\geq 0} z_j\sigma_j$ with $z_j \geq 0$, and we would like to use only a small set of different Jacobi sums (pairs (a, b)).

The first steps are easy: $\tau(\chi)^{1-\sigma_1} = 1$ and for $\tau(\chi)^{2-\sigma_2}$ we have not much choice but to write it as $\tau(\chi)^{2-\sigma_2} = J(\chi, \chi) = J_2$. (Here the index of J will denote the sum $a + b$ in $J(\chi^a, \chi^b)$.) But this means that we are able to find expressions for all $\tau(\chi)^{i-\sigma_i}$ for all i that are powers of 2, since

$$\tau(\chi)^{4-\sigma_4} = \left(\tau(\chi)^{2-\sigma_2}\right)^{2+\sigma_2} = J_2^{2+\sigma_2},$$

and similarly

$$\tau(\chi)^{8-\sigma_8} = J_2^{4+2\sigma_2+\sigma_4}$$
$$\tau(\chi)^{16-\sigma_{16}} = J_2^{8+4\sigma_4+2\sigma_4+\sigma_8}.$$

It turns out that for $\tau(\chi)^{3-\sigma_3}$ we need a new Jacobi sum, namely $J_3 = \tau(\chi)\tau(\chi^2)/\tau(\chi^3)$; then $\tau(\chi)^{3-\sigma_3} = J_2 J_3$. Next we can generate $\tau(\chi)^{i-\sigma_i}$ for every i that is built up from powers of 2 and 3. For instance

$$\tau(\chi)^{6-\sigma_6} = \Big(\frac{\tau(\chi)^3}{\tau(\chi^3)}\Big)^2 \frac{\tau(\chi^3)^2}{\tau(\chi^6)} = \Big(\tau(\chi)^{3-\sigma_3}\Big)^2 J_2^{\sigma_3} = J_2^{2+\sigma_3} J_3^2;$$

similarly

$$\tau(\chi)^{9-\sigma_9} = \tau(\chi)^{(3-\sigma_3)(3+\sigma_3)} = J_2^{3+\sigma_3} J_3^{3+\sigma_3},$$
$$\tau(\chi)^{12-\sigma_{12}} = \tau(\chi)^{(3-\sigma_3)4}\tau(\chi)^{(2-\sigma_2)(2\sigma_3+\sigma_6)} = J_2^{4+2\sigma_3+\sigma_6} J_3^4.$$

Actually, for $\tau(\chi)^{9-\sigma_9}$ we could have done without J_3, since by (8.11)

$$\tau(\chi)^{9-\sigma_9} = \tau(\chi)^8 \frac{\tau(\chi)}{\tau(\chi^9)} = \tau(\chi)^8 \frac{\tau(\chi^8)}{\tau(\chi^{16})} = J_2^{4+2\sigma_2+\sigma_4+\sigma_8}.$$

Note that $\chi(-1) = 1$ for this character.

From this it may be clear that the problem really is to generate $\tau(\chi)^{i-\sigma_i}$ for prime i. It turns out that for $\tau(\chi)^{5-\sigma_5}$ we need again a new Jacobi sum, but to illustrate that this not always the case and how one should try to find alternatives, we turn to $\tau(\chi)^{7-\sigma_7}$ first. We will attempt to use (8.11) again; suppose we can find integers x, y such that $x + y = 7$ (with $1 \le x, y < 7$), such that both $\tau(\chi)^{x-\sigma_x}$ and $\tau(\chi)^{y-\sigma_y}$ have been generated before, with the additional property that for some $j \ge 1$ and $h \ge 1$ we have

$$\frac{\tau(\chi^x)\tau(\chi^{j\cdot 17-7})}{\tau(\chi^{17-y})} = J_2^{\sigma_h} \quad \text{or} \quad = J_3^{\sigma_h}.$$

Then we are done, since by (8.11):

$$\frac{\tau(\chi^x)\tau(\chi^{j\cdot 17-7})}{\tau(\chi^{17-y})} = \frac{\tau(\chi^x)\tau(\chi^y)}{\tau(\chi^7)}$$

so we find $\tau(\chi)^{7-\sigma_7}$ on multiplying this by the expression we found previously (by assumption) for $\tau(\chi)^{x-\sigma_x}$ and $\tau(\chi)^{y-\sigma_y}$.

Such x, y, j, h do indeed exist here; take $x = 3,\ y = 4,\ j = 1,\ h = 10$:

$$\frac{\tau(\chi^3)\tau(\chi^{1\cdot 17-7})}{\tau(\chi^{17-4})} = \frac{\tau(\chi^{20})\tau(\chi^{10})}{\tau(\chi^{30})} = J_3^{\sigma_{10}}.$$

With $\tau(\chi)^{3-\sigma_3}$ and $\tau(\chi)^{4-\sigma_4}$ as above we thus find

$$\tau(\chi)^{7-\sigma_7} = J_2^{3+\sigma_2} J_3^{1+\sigma_{10}}.$$

This immediately gives

$$\tau(\chi)^{14-\sigma_{14}} = J_2^{6+2\sigma_2+\sigma_7} J_3^{2+2\sigma_{10}}.$$

Exactly the same procedure will also work for $i = 11$, using $x = 3,\ y = 8,\ j = 1,\ h = 3$ (with J_3), to obtain:

$$\tau(\chi)^{11-\sigma_{11}} = \frac{\tau(\chi)^3}{\tau(\chi^3)} \frac{\tau(\chi)^8}{\tau(\chi^8)} \frac{\tau(\chi^3)\tau(\chi^6)}{\tau(\chi^9)} = J_2^{5+2\sigma_2+\sigma_4} J_3^{1+\sigma_3}$$

using the expressions for $\tau(\chi)^{3-\sigma_3}$ and $\tau(\chi)^{8-\sigma_8}$ given above.

It will work for $i = 13$ as well, using $x = 4,\ y = 9,\ j = 1,\ h = 4$ (with J_2), to obtain:

$$\tau(\chi)^{13-\sigma_{13}} = \frac{\tau(\chi)^4}{\tau(\chi^4)} \frac{\tau(\chi)^9}{\tau(\chi^9)} \frac{\tau(\chi^4)\tau(\chi^4)}{\tau(\chi^8)} = J_2^{6+3\sigma_2+2\sigma_4+\sigma_8}$$

using the second alternative for $\tau(\chi)^{9-\sigma_9}$ given above.

That leaves only $i = 5, 10$ and 15. It is clear that an expression for $\tau(\chi)^{5-\sigma_5}$ will also solve the problem for $\tau(\chi)^{10-\sigma_{10}}$ and $\tau(\chi)^{15-\sigma_{15}}$; it is interesting however, that both $\tau(\chi)^{10-\sigma_{10}}$ and $\tau(\chi)^{15-\sigma_{15}}$ can be done by the above method as well, using only J_2 and J_3. Namely, for $\tau(\chi)^{10-\sigma_{10}}$ we may take $x = 7,\ y = 3,\ j = 1,\ h = 7$ (with J_2), to arrive at:

$$\tau(\chi)^{10-\sigma_{10}} = \frac{\tau(\chi)^3}{\tau(\chi^3)} \frac{\tau(\chi)^7}{\tau(\chi^7)} \frac{\tau(\chi^7)\tau(\chi^7)}{\tau(\chi^{14})} = J_2^{4+\sigma_2+\sigma_7} J_3^{2+\sigma_{10}}$$

using $\tau(\chi)^{7-\sigma_7}$.

For $\tau(\chi)^{15-\sigma_{15}}$ we may take $x = 2,\ y = 13,\ j = 1,\ h = 2$ (with J_2), to get:

$$\tau(\chi)^{15-\sigma_{15}} = \frac{\tau(\chi)^2}{\tau(\chi^2)} \frac{\tau(\chi)^{13}}{\tau(\chi^{13})} \frac{\tau(\chi^2)\tau(\chi^2)}{\tau(\chi^4)} = J_2^{7+4\sigma_2+2\sigma_4+\sigma_8}$$

using $\tau(\chi)^{13-\sigma_{13}}$ above.

However, if we try to use the same method for $i = 5$, we do not succeed; all we have to do is checking the following. If

$$\frac{\tau(\chi^x)\tau(\chi^{j\cdot 17-5})}{\tau(\chi^{17-y})} = J_2^{\sigma_h}$$

holds with $j = 1$, then $x = h = 1 \cdot 17 - 5 = 12$ and $y = 10$, in which case clearly not $x, y < 5$; taking $j \geq 2$ will not help here since y mod 17 will be unaffected. If

$$\frac{\tau(\chi^x)\tau(\chi^{j\cdot 17-5})}{\tau(\chi^{17-y})} = J_3^{\sigma_h}$$

holds with $j = 1$, then either $x = h(1 \cdot 17 - 5) \equiv 7 \bmod 17$ and $y = 15$, or $2x = h = 1 \cdot 17 - 5 = 12$ and $y = 16$; in both cases neither x nor y is smaller than 5. Again, for y modulo 17 nothing changes for larger j.

So our method fails and we introduce a new Jacobi sum J_5; here a choice has to be made: either we take $J_5 = J(\chi^2, \chi^3)$ or $J_5 = J(\chi, \chi^4)$. Both will do, but the resulting exponents will differ. It turns out that in making exponents like this for orders other than 17, one will have a preference for $J(\chi^2, \chi^3)$. Then: $\tau(\chi)^{5-\sigma_5} = J_2^2 J_3 J_5$.

For reference in Chapter IV we state here the result that was used several times in the previous example.

(8.13) Lemma. *Let χ be a character of prime conductor q, with $q \in A^*$, of order p^k, with p prime. Let $0 < i < p^k$, and i not divisible by p.*

Suppose that positive integers a and j exist, as well as a prime $\pi \neq p$, such that the following hold:

$$a \mid j \cdot p^k - i \text{ and } (\pi - a)\Big(\frac{j \cdot p^k - i}{a}\Big) \equiv m \bmod p^k \text{ for some } m \text{ with } 0 < m < i. \tag{8.14}$$

Then there exist positive integers b, $c < i$, $d < i$ and e, such that

$$J(\chi^a, \chi^b)^{\sigma_e} = \pm\frac{\tau(\chi^c)\tau(\chi^d)}{\tau(\chi^i)}. \tag{8.15}$$

Explicitly, this is true for

$$b = \pi - a, \quad c = i - (\pi - a)\Big(\frac{j \cdot p^k - i}{a}\Big), \quad d = (\pi - a)\Big(\frac{j \cdot p^k - i}{a}, \text{ and } e = \frac{j \cdot p^k - i}{a},$$

and the sign in (8.15) equals $\chi^d(-1)$.

Proof. Apply (8.11), for the given values of b, c, d and e:

$$J(\chi^a, \chi^{\pi-a})^{\sigma_e} = \frac{\tau(\chi^{jp^k-i})}{\tau(\chi^{jp^k-i+d})}\tau(\chi^d) = \frac{\chi^{-i}(-1)}{\chi^{-i+d}(-1)}\frac{\tau(\chi^{i-d})\tau(\chi^d)}{\tau(\chi^i)},$$

and the result is immediate.

9. THE FINAL STAGE.

In the final stage of the algorithm some trial divisions will have to be performed. The outcome of the previous parts of the algorithm will be that every possible divisor of n is congruent to a power of n modulo an auxiliary number, that we shall call m in this section. We describe an algorithm due to H.W. Lenstra Jr., for finding all divisors of n quickly in this situation (see [88]).

At this stage we know (cf. Theorem (7.1)) that every divisor r of n satisfies

$$r \equiv n^i \bmod m \tag{9.1}$$

for an auxiliary number m.

If $m > \sqrt{n}$, then it is clear what the final step will consist of: for every i, find the representative

$$r_i \equiv n^i \bmod m, \quad \text{with} \quad 0 \leq r_i < m;$$

if $1 < r_i \leq \sqrt{n}$, then do a trial division of n by r_i.

As we pointed out before, making m large is expensive, and therefore it would be desirable to allow $m \leq \sqrt{n}$. In that case however, it is not obvious any more that there exists an efficient way of finding all divisors of n; checking all possibilities leads to an exponential algorithm. Here we invoke the following result of [88].

(9.2) Theorem. *Let r, m and n be integers satisfying*

$$0 \leq r < m < n, \quad \textit{with} \quad m > \sqrt[3]{n} \quad \textit{and} \quad \gcd(r, m) = 1. \tag{9.3}$$

Then there is an algorithm for determining all positive divisors of n that are congruent to r modulo n, which requires $O((\log n)^3)$ bit operations.

It has even been proved that in the above situation there are at most 11 divisors in any given residue class r modulo m. We will present the algorithm for finding them below.

(9.4) Algorithm. Let r, m, and n satisfy (9.3).

Use the Euclidean algorithm to find the inverse r^{-1} of r modulo m, and determine the integer

$$r' \equiv r^{-1} n \bmod m, \quad \text{with } 0 \leq r' < m.$$

Do the following for $j = 0, 1, \ldots$ in succession. Calculate the triple a_j, b_j, c_j of integers, defined by:

$$a_0 = m \qquad a_1 \equiv r'r^{-1} \bmod m,\ 0 < a_1 \leq m, \qquad a_j = a_{j-2} - q_j a_{j-1},\ \text{for } j \geq 2,$$

$$b_0 = 0 \qquad b_1 = 1, \qquad b_j = b_{j-2} - q_j b_{j-1},\ \text{for } j \geq 2,$$

$$c_0 = 0 \qquad c_1 \equiv \frac{(n - rr')}{m} r^{-1} \bmod m, \qquad c_j = c_{j-2} - q_j c_{j-1},\ \text{for } j \geq 2,$$

where q_j is the unique integer for which

$$0 \leq a_j < a_{j-1} \quad \text{if } j \text{ is even},$$
$$0 < a_j \leq a_{j-1} \quad \text{if } j \text{ is odd}.$$

Next, solve the quadratic equation

$$T^2 - (cs + a_j r + b_j r')T + a_j b_j n = 0, \tag{9.5}$$

for every integer c satisfying

$$c \equiv c_j \bmod m \quad \text{and} \begin{cases} -m < c < m & \text{if } j \text{ is even}, \\ 2a_j b_j \leq c \leq \dfrac{n}{m^2} + a_j b_j & \text{if } j \text{ is odd}. \end{cases}$$

If the solutions to (9.5) are integers, and if there exist non-negative integers x and y such that these solutions equal

$$t_1 = a_j(xm + r) \quad \text{and} \quad t_2 = b_j(ym + r'),$$

then $xm + r$ is a divisor of n congruent to r modulo m. If $a_j = 0$, the algorithm terminates, otherwise we continue with the next j.

(9.6) Remarks. We refer to [88] for a proof of the fact that the algorithm in (9.4) has the properties claimed in Theorem (9.2).

Note that the number of residue classes r that need to be checked in the primality test is at most the order of n in $(\mathbf{Z}/m\mathbf{Z})^*$ by (9.1), which is relatively small by our construction of m, as in Sections 6 and 7. In fact, (9.4) need to be applied at most half that number of times, since in solving (9.5) one detects divisors congruent to $r_i \equiv n^i \bmod m$ and congruent to $r' \equiv n^{1-i} \bmod m$ simultaneously.

III. OPTIMIZATION.

1. INTRODUCTION.

In this chapter we describe the strategy used in the optimization stage of the algorithm, described in detail in Section IV.4.

During the optimization stage of the algorithm several choices have to be made. In the first place it should be decided how much time will be spent on looking for additional "Lucas-Lehmer factors"; secondly one has to choose the values of the auxiliary integers s, t, u, v and w, and finally the Jacobi sum tests that have to be performed are to be combined. The aim of the optimization stage is to make these choices in such a way that the running time of the primality test (including this optimization stage!) for a given number n is minimal.

Solving this minimization problem seems to be a difficult and complex problem however. As a result, it will in general not be possible to find the minimal solution, but we will have to be satisfied with an approximation.

Since for small numbers the primality proofs will be very short, the optimization stage cannot take too much time, since otherwise the total running time to complete the proof would be influenced considerably by the time needed to perform the optimization stage.

On the other hand, in the proofs of primality for large numbers, it will be worthwhile to invest some time in the optimization stage, since it can speed up the various stages of the primality test considerably.

These considerations show that before starting the optimization routine an estimate should be made of the time that will be invested in optimizing. This estimate may depend on two rough estimates; firstly for the time necessary to complete the primality proof without further optimizing, and secondly for the expected profit of optimizing. The first could for instance consist of the calculation of the time necessary to complete the proof using only Jacobi sum tests, depending on the *size* of n only. The second could be based on previous experience. This also implies that the steps of the optimization stage, as described below, may be repeated, as long as it is expected to be profitable to do so.

The rest of this introductory section is devoted to the description of an overview of the optimization stage, explaining the constraints in the optimization problem and the meaning of the symbols used, while referring to the previous chapter. The following five steps each correspond, in reverse order, to one of the next five sections.

(1.1) Factoring time. The primality proofs are based on Theorem II.(7.1), in which

Jacobi sum tests and Lucas-Lehmer type tests are combined. The parameters s, t and u refer to the Jacobi sum test, v and w refer to the Lucas-Lehmer type tests.

The basic constraint on the auxiliary integers is that

$$\text{lcm}(s,v) > \sqrt{n}, \tag{1.2}$$

(or $\text{lcm}(s,v) > \sqrt[3]{n}$, applying the techniques of II.9).

As explained in II.5, a Lucas-Lehmer type test consists of constructing a v_i-th cyclotomic constellation L, ζ, σ, where L is cyclic field of degree w_i, the order of n in $(\mathbf{Z}/v_i\mathbf{Z})^*$. Here the prime factorization of v_i must be known. Roughly speaking, the test comes down to performing a n^{w_i}-th powering in the ring O_L/nO_L, an extension ring of degree w_i of $\mathbf{Z}/n\mathbf{Z}$. The integer v will be the product of the v_i for which one performs such Lucas-Lehmer tests, and $w = \text{lcm}(w_i)$. Thus v is (a divisor of) the completely factored part of $n^w - 1$.

It should be remarked that, although w is the total degree of the extension in which all Lucas-Lehmer type tests can be performed, the actual tests will all take place in proper subextensions (unless a *primitive* factor of $n^w - 1$ is used).

The contribution to v may come from different sources. It may be that someone has found factors of $n-1, n^2-1, \ldots$ in the past. To find additional factors one should apply some factorization algorithm. The optimization step first determines how much time should be spent on this. This decision is based on the calculation of the expected size of the product of the factors that are to be found and balancing the costs for the factor search and the Lucas-Lehmer test with the costs of doing additional Jacobi sum tests instead. This strategy is discussed in Section 6.

(1.3) Choosing v. The second choice that is made during the optimization step, is that of the value of v, and of w at the same time. That means that one decides which of the factors v_i (either given beforehand or found from factoring) are to be used. Using a small factor v_i for large w_i may be much too expensive. One is particularly interested in factors of $n^{w_i} - 1$ for values of w_i that divide some $u_{p,k}$, since these may be regarded as "free" (see below). The strategy for choosing v and w is described in Section 5; basically one first limits the set of values for w taken into consideration drastically and then performs an exhaustive search over the remaining, useful values.

It may be that some small prime factors of v will in fact be used as factors for t.

(1.4) Choosing t. The integer s will be a divisor $e(t)$, as in II.(6.13), possibly multiplied by a few extra small prime factors dividing t. Thus it will mainly be built up from primes q with the property that $q-1 \mid t$. Since we want s in general to be large, the integer t will be built up from small primes, cf. II.(6.13).

The third choice made during the optimization, is that of t. For each prime power dividing t and for each prime q dividing s we have to do a few initialization steps, which can be done in advance, i.e., without knowledge about the number n. Therefore we require that the number t will be a divisor of an initially chosen value t_0.

Again, one basically performs an exhaustive search, after discarding "bad" values for t. Here "bad" either means that $\operatorname{lcm}(e(t), v) < \sqrt{n}$ so that (1.2) will never hold, or that using this t is obviously more expensive than using other values. This can often easily be seen by calculating a rough underestimate of the costs of the principal steps.

The integer u will be equal to $u = \operatorname{ord} n$ in $(\mathbf{Z}/t\mathbf{Z})^*$, the total degree of the extensions in which the Jacobi sum tests will take place. Therefore u is determined by the choice for t.

The first part of the Jacobi sum test consists of showing the existence of t_1-th cyclotomic constellations; here t_1 will be the largest divisor of $n^u - 1$ that is built up from primes in t only. This comes down to finding p^k-th roots of unity, for all prime power divisors $p^k \parallel t_1$, in extensions of $\mathbf{Z}/n\mathbf{Z}$ of degree $u_{p,k}$, the order of n in $(\mathbf{Z}/p^k\mathbf{Z})^*$. This means roughly that an n-th powering in that extension has to be done. If also a Lucas-Lehmer test has to be done in an extension of degree $w_i \mid u_{p,k}$, both can be done by the same exponentiation. This explains why it was remarked above that such a Lucas-Lehmer test is gotten (almost) for free. For a description see Section 4.

(1.5) Choosing s. Next one chooses the divisor s of $e(t)$ that will be used. For every pair (p^k, q), with q a prime divisor of s and $p^k \parallel q-1$, a Jacobi sum test must be performed, in an extension of degree $u_{p,k}$, the order of n in $(\mathbf{Z}/p^k\mathbf{Z})^*$. But since two such tests can sometimes be combined into one cheaper test, the costs of using different q's are not independent. An exhaustive search over all s satisfying (1.2) soon becomes too time-consuming, and therefore one has to find a reasonable estimate for the cost of using q, taking the possibility of combining characters into account. See Section 3 for this.

(1.6) Finding an optimal matching. Finally, once s is chosen, one has to find an optimal combination of the Jacobi sum tests that have to be performed. An efficient

algorithm for this is described in Section 2.

Thus the optimization can be divided into five consecutive steps, each of which can be regarded as a sub-step of the preceding one. Since several steps consist of a search over possible values for the parameters that are to be chosen, it seems natural to discuss the steps in reverse order. In the first section to come, we will discuss the innermost stage of the optimization problem: suppose that all parameters have been fixed, except for the matching of the Jacobi sums. Subsequently we will work our way out from this stage to the outermost stage. In this way we will complete the description of all the strategies used by the optimization stage of the algorithm.

2. FINDING AN OPTIMAL MATCHING.

(2.1) Assumptions. In this section we consider the problem of combining the Jacobi sum tests in an optimal way. Throughout this section we assume that the values for the auxiliary integers s, t, u, v, and w are fixed. Moreover we assume that a set X of characters is given for which Jacobi sum tests have to be performed. Characters $\chi \in X$ will also be represented here as pairs (p^k, q), consisting of the order and the conductor of the character; here p and q will be prime and $p^k \mid q-1$. Finally it is assumed that for every p^k for which there exists q such that $(p^k, q) \in X$, some p^k-th cyclotomic constellation L, ζ_{p^k}, σ has been constructed.

(2.2) Remark. Most of the characters (p^k, q) in X have the property that q is a prime divisor of s and that $p^k \parallel q-1$. However, X may contain some "special" characters as well; for these q need not divide s, and p^k need not be the largest power of p dividing $q-1$, see II.(7.8).

(2.3) Performing Jacobi sum tests. Performing a Jacobi sum test for a character χ of order p^k and conductor q consists, in principle, of checking the condition (cf. II.(6.4)(v))

$$(z\tau(\chi))^{n-\phi_n} \in \langle \zeta_{p^k} \rangle \tag{2.4}$$

for χ. However, in practice one does not check (2.4), but one employs Jacobi sums. As pointed out in II.(8.10), $\tau(\chi)^{n-\phi_n}$ is rewritten as a product of Jacobi sums, which means that (2.4) can be checked without leaving the ring O_L/nO_L, an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree $u_{p,k}$, the order of n in $(\mathbf{Z}/p^k\mathbf{Z})^*$. Checking (2.4) essentially consists then of taking the n-th power of an element consisting of $u_{p,k}$ coordinates over $\mathbf{Z}/n\mathbf{Z}$. This has to be done for every character, that is pair (p^k, q), in X.

The aim is to find an algorithm for minimizing the cost of performing the Jacobi sum tests for a given set X. The reason that there is a minimization problem at all lies in the possibility of *combining* tests, for which again Jacobi sums are utilized.

Instead of checking (2.4) for a character χ_1 in an extension of degree u_1 and for a character χ_2 in an extension of degree u_2, one may perform one combined test

$$(J(\chi_1, \chi_2)\tau(\chi_1\chi_2))^{n-\phi_n} \in \langle \zeta \rangle \tag{2.5}$$

in an extension of degree $\mathrm{lcm}(u_1, u_2)$, as was pointed out in II.(8.6). So two (or more) n-th powerings can be replaced by one, but possibly on an element with more coordinates.

This means that whether or not it is profitable to combine these tests depends on the degrees u_1 and u_2, as well as on the function $G(u)$ that expresses how expensive it is to multiply two elements of u coordinates (over $\mathbf{Z}/n\mathbf{Z}$). If for instance $G(u) = u$, so the cost of multiplication is linear in the number of coordinates, then it is profitable to combine tests in extensions of degree 6, 10 and 15 into one in degree $\text{lcm}(6,10,15) = 30$ since $30 < 6+10+15$; but if $G(u) = u^2$, quadratic in the number of coordinates, which is certainly more realistic, then this combination is not profitable at all: $30^2 > 6^2+10^2+15^2$.

There is one more rule that is to be obeyed, namely: two Jacobi sum tests as in (2.4) will only be combined to one test as in (2.5) if the orders of the characters involved (the p^k above) are relatively prime, see II.(8.6).

Now we describe the resulting optimization problem in a slightly more general form. The reason for this is that we will show that the computational complexity of the problem changes dramatically – from having a polynomial time solution (e.g. in the case we need) to being NP-complete – if the imposed conditions are slightly modified.

(2.6) The problem. *Given the disjoint union $X = \coprod_{i\in I} X_i$ of a finite number of finite sets and a function*

$$d : X \ \to \ \mathbf{Z}_{\geq 1}$$

assigning a positive integer to every element of X, as well as a function

$$G : \ \mathbf{Z}_{\geq 1} \ \to \ \mathbf{R}_{\geq 0}.$$

Find a disjoint covering $\coprod_{j\in J} W_j = X$ such that:

$$\text{for every } i \in I \text{ and } j \in J : \ \ \#\big(X_i \cap W_j\big) \leq 1$$

and

$$\sum_{j\in J} G\big(\text{lcm}\{d(w) : w \in W_j\}\big) \text{ is minimized.}$$

(2.7) Comments. We translate this problem in terms of the description given above. The collection X in the above formulation can be thought of as the collection of characters

(p^k, q) for which one has to do a Jacobi sum test. These are grouped in sets X_i of characters of order a power of the same prime p. Then d may be thought of as the function that assigns to a character the degree of the extension in which the Jacobi sum for that character exists. The function G reflects the cost of a multiplication: it gives the cost $G(k)$ for the multiplication of two elements having k coordinates, that is, in an extension of degree k. Every W_j is a combination for which the tests are done simultaneously; since characters within the same X_i may not be combined, the total cost should be minimized under the given restriction.

In fact the problem described above is a *matching* problem.

(2.8) Matchings. Let $X_1, X_2, \ldots, X_n$ be finite sets. Let

$$M \subset X_1 \times X_2 \times \cdots \times X_n;$$

a *matching* M' is a subset $M' \subset M$ such that no two elements of M' agree in any coordinate.

The *2-dimensional matching* problem is: given a subset $M \subset X_1 \times X_2$, with $\#X_1 = \#X_2 = k$, does M contain a maximal matching, that is, a matching of cardinality k?

The 2-dimensional matching problem can be interpreted in terms of graphs; the sets X_1 and X_2 form the disjoint sets of nodes of a bipartite graph, and edges between $x_1 \in X_1$ and $x_2 \in X_2$ exist precisely if $(x_1, x_2) \in M$. A matching consists of a subset of edges such that no pair is incident with the same vertex.

If we assign (positive) weights to the edges of the bipartite graph, we arrive at the *weighted bipartite matching* problem: given a bipartite graph with positive weights attached to every edge, find a matching for which the sum of the weights is maximal.

Another way of looking at weighted bipartite matching leads to the *assignment* problem: given a $n \times n$-matrix W_{ij} with non-negative entries, find a subset of the entries containing precisely one element in every row and in every column, for which the sum of the values is minimal. One may think of this as minimizing the cost of assigning one task to each of n people. This is equivalent with the weighted bipartite matching problem.

The *3-dimensional matching* problem reads as follows: given a subset $M \subset X_1 \times X_2 \times X_3$, with $\#X_1 = \#X_2 = \#X_3 = k$, does M contain a maximal matching, that is, a matching of cardinality k?

For both the 2-dimensional matching problem and the weighted bipartite matching problem polynomial time solutions exist (cf. [67, Ch. 5]). The 3-dimensional matching problem is known to be NP-complete, cf. [43].

Below we will deal with several special cases of problem (2.6), either by imposing conditions on the function G or by restricting the number of sets X_i.

We start with the bipartite case.

(2.9) Theorem. *Let notations be as in* (2.6). *If* $\#I = 2$, *there is a polynomial time algorithm to find the minimal covering in problem* (2.6).

Proof. In this case our problem is equivalent to the assignment problem, as follows.

Let $\#X = n$, and write $X = \{x_1, \ldots, x_n\}$. Let W be the $n \times n$-matrix with entries w_{ij} (for $1 \leq i, j \leq n$) obtained as follows. If $x_i \in X_1$ and $x_j \in X_2$, or the other way around, put $w_{ij} = \frac{1}{2}G\big(\mathrm{lcm}\{d(x_i), d(x_j)\}\big)$; if x_i and x_j are both in X_1 or both in X_2, put $w_{ij} = \infty$, unless $x_i = x_j$, in which case $w_{ij} = w_{ii} = G(d(x_i))$. One verifies easily that solving (2.6) is the same as solving the assignment problem for the above matrix W.

That proves (2.9).

Next we consider the "free multiplication case", that is the case where $G \equiv 1$.

(2.10) Theorem. *Let notations be as in* (2.6). *If* $G(u) = 1$ *for every* u, *there is a polynomial time algorithm to find the minimal covering in problem* (2.6).

Proof. If $G(u) = 1$ for every u, then

$$\sum_{j \in J} G\big(\mathrm{lcm}\{d(w) : w \in W_j\}\big)$$

is minimized exactly when $\#J$ is minimized. From $\#\big(W_j \cap X_i\big) \leq 1$ we see that $\#J \geq \max\{\#X_i : i \in I\}$. But it is easy to realize a covering by $\#J = \max\{\#X_i : i \in I\}$ sets W_j as follows: for every $i \in I$ put each of the elements of X_i in one of the W_j, different elements in different sets. Then all requirements are met and this gives certainly a polynomial time algorithm.

(2.11) Remark. We could replace 1 in Theorem (2.10) by any constant $C > 0$; the reason we have taken $C = 1$ and call this the free multiplication case is that it corresponds to the

case of multiplication exponent 0: the cost of multiplication of elements on u coordinates is independent of u. Compare this to the realistic case below.

(2.12) Theorem. *Let notations be as in* (2.6). *Suppose that G satisfies*

$$G(u) \geq \sum_{\substack{p \mid u \\ p \text{ prime}}} G(\frac{u}{p}) \qquad \text{for every } u \tag{2.13}$$

and suppose moreover that d satisfies

$$\text{for every } i \in I \text{ and every } y, y' \in X_i\text{:} \quad d(y) \mid d(y') \text{ or } d(y') \mid d(y). \tag{2.14}$$

Then there is a polynomial time algorithm to find the minimal covering in problem (2.6).

We prove this theorem by describing a greedy algorithm that yields a solution, and show afterwards that it is optimal.

(2.15) Matching algorithm. *Let notations be as in* (2.6). *Find a covering $X = \coprod W_j$ by repeating the following steps until the set X is empty.*

(i) *Suppose that the sets $W_1, \ldots, W_{j-1}$ have been found; find an element in X, say $y_0 \in X_{i_0}$, such that $d(y_0)$ is maximal, put it in W_j and remove it from X.*

(ii) *Next find for every $i \neq i_0$ an element $y \in X_i$ (if it exists) such that $d(y)$ is maximal under the restriction that $d(y) \mid d(y_0)$; add these elements to W_j and remove them from X. This finishes the description of W_j.*

Proof of (2.12). First we observe the following. Suppose that $X' \subset X$ and $X'' \subset X$; let $\coprod_{j \in J'} W'_j$ be an optimal covering of X' and let $\coprod_{j \in J''} W''_j$ be an optimal covering of X''. If we let $G(\coprod_{j \in J} W_j)$ abbreviate

$$\sum_{j \in J} G\big(\operatorname{lcm}\{d(w) : w \in W_j\}\big),$$

then $G(\coprod_{j \in J'} W_j) \leq G(\coprod_{j \in J''} W_j)$ provided that X' "injects piecewise" into X'' as follows. Denote $X'_i = X' \cap X_i$ and $X''_i = X'' \cap X_i$; suppose that for every $i \in I$ there exists an injection

$$\psi_i : X'_i \hookrightarrow X''_i$$

combining to

$$\psi : X' \hookrightarrow X''$$

such that

$$\text{for every } x' \in X' \text{:} \quad d(x') \leq d(\psi(x'))$$

then by (2.14) in fact $d(x') \mid d(\psi(x'))$. Under this condition

$$\sum_{j\in J'} G\big(\mathrm{lcm}\{d(w) : w \in W'_j\}\big) \leq \sum_{j\in J''} G\big(\mathrm{lcm}\{d(w) : w \in X',\, \psi(w) \in W''_j\}\big),$$

since the W'_j cover X' optimally; also $d(w) \mid d(\psi(w))$, and therefore

$$\sum_{j\in J''} G\big(\mathrm{lcm}\{d(w) : w \in X',\, \psi(w) \in W''_j\}\big) \leq \sum_{j\in J''} G\big(\mathrm{lcm}\{d(w) : w \in W''_j\}\big).$$

Combining these we find indeed that $G(\coprod_{j\in J'} W_j) \leq G(\coprod_{j\in J''} W_j)$.

Let $\coprod_{j\in J} W_j$ be the covering found by applying the algorithm in (2.15) to X and let $\coprod_{k\in K} W^*_k$ be any other disjoint covering. We show that

$$\sum_{j\in J} G\big(\mathrm{lcm}\{d(w) : w \in W_j\}\big) \leq \sum_{k\in K} G\big(\mathrm{lcm}\{d(w) : w \in W^*_k\}\big).$$

Let $x \in X$ be an element for which $d(x)$ is maximal, and let j_0 and k_0 be such that $x \in W_{j_0}$ and $x \in W^*_{k_0}$. Let $L = \mathrm{lcm}\{d(w) : w \in W^*_{k_0}\}$. Suppose that $W^*_{k_0}$ contains an element w such that $d(w) \nmid d(x)$; by maximality $d(x) \mid d(w)$ is impossible now, so $L \geq \mathrm{lcm}(d(w), d(x)) > \max(d(x), d(w)) = d(x)$. Therefore, the value L is not equal to $d(w)$ for any $w \in W^*_{k_0}$ and that means that by (2.13) we will find a solution that is at least as good as the one provided by the covering $\coprod W^*_k$ if we split $W^*_{k_0}$ into sets for which

$$W^*_{k_0,p} \subset \{w \in W^*_{k_0} : \; d(w) \mid \frac{L}{p}\},$$

for primes p dividing L. Making choices if necessary, we can obtain such sets forming a disjoint covering of $W^*_{k_0}$; one of the $W^*_{k_0,p}$ contains x, and we can repeat the above reasoning, replacing $W^*_{k_0}$ by $W^*_{k_0,p}$. But L will now be replaced by L/p, and after finitely many steps in which the solution can only have changed for the better, we arrive at the situation in which every w in the subset containing x satisfies $d(w) \mid d(x)$.

We may as well assume rightaway that for every $w \in W^*_{k_0}$ we have $d(w) \mid d(x)$; thus $G\big(\mathrm{lcm}\{d(w) : w \in W_{j_0}\}\big) = G\big(\mathrm{lcm}\{d(w) : w \in W^*_{k_0}\}\big) = G(d(x))$. Next let $X' = X \setminus W_{j_0}$ and likewise $X'' = X \setminus W^*_{k_0}$. Let $z \in W_{j_0}$ with $z \neq x$; then $z \in X_i$ for some $i \neq k_0$, and z was chosen to be that element of X_i for which $d(z)$ is the maximal divisor of $d(x)$, according to algorithm (2.15). Now either $W^*_{k_0} \cap X_i$ is empty, or it contains one element, z^*, for which $d(z^*) \leq d(z)$. As a consequence, for every i there exists an injection

$$\psi_i : X' \cap X_i \hookrightarrow X'' \cap X_i$$

with the property that

$$\text{for every } x'_i \in X' \cap X_i\text{:}\quad d(x'_i) \leq d(\psi_i(x'_i)).$$

In other words, the observation at the beginning of this proof applies, and we find

$$\begin{aligned}\sum_{j\in J} G\big(\mathrm{lcm}\{d(w) : w \in W_j\}\big) &= G(d(x)) + \sum_{j_0 \neq j \in J} G\big(\mathrm{lcm}\{d(w) : w \in W_j\}\big) \\ &\leq G(d(x)) + \sum_{k_0 \neq k \in K} G\big(\mathrm{lcm}\{d(w) : w \in W^*_k\}\big) \\ &= \sum_{k\in K} G\big(\mathrm{lcm}\{d(w) : w \in W^*_k\}\big).\end{aligned}$$

That completes the proof of (2.12).

(2.16) Corollary. *Let the notation be as in* (2.6). *If* $G(u) = u^\rho$ *for some* $\rho \geq \rho_0 = 1.41\cdots$, *where the constant* ρ_0 *satisfies*

$$\sum_{p \text{ prime}} \frac{1}{p^{\rho_0}} = 1,$$

and if d *satisfies* (2.14), *then there is a polynomial time algorithm to find the minimal covering in problem* (2.6).

Proof. If $G(u) = u^\rho$ with $\rho \geq \rho_0$, then

$$\sum_{\substack{p \mid u \\ p \text{ prime}}} G(\frac{u}{p}) = \sum_{\substack{p \mid u \\ p \text{ prime}}} (\frac{u}{p})^\rho < u^\rho \sum_{p \text{ prime}} \frac{1}{p^\rho} \leq G(u),$$

and the result follows immediately from Theorem (2.12).

(2.17) Remarks. We call the above case the realistic case, because in our primality testing application both conditions (2.13) and (2.14) are satisfied. Condition (2.14) holds because the order of an integer modulo p^k will divide the order of that integer modulo p^l if $l \geq k$. Condition (2.13) is satisfied if we use naïve multiplication (requiring u^2 multiplications and some additions to multiply two elements with u coordinates), or if we make at least the realistic assumption that we cannot do better than something with exponent of multiplication ρ_0.

Moreover, we work in practice in extensions of degrees that are small powers of 2, 3 or 5 only, and therefore even every exponent of multiplication ρ larger than $1.033\cdots$ leads to a polynomial solution.

Finally we show that if condition (2.13) on G is slightly relaxed in the realistic case, we get an NP-complete problem.

(2.18) Theorem. *Problem* (2.6) *is NP-complete.*

This remains the case if we assume that G satisfies

(2.19) $\quad G(u) \geq G(d)$ *for every divisor d of u*

and that d satisfies

(2.20) $\quad$ *for every $i \in I$ and every $y, y' \in X_i$:* $\quad d(y) \mid d(y')$ *or* $d(y') \mid d(y)$.

Proof. We construct a subclass, for which we can show that it is NP-complete by transforming 3-dimensional matching into it. Let $X = U \cup V \cup W$ where $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$, and $W = \{w_1, \ldots, w_n\}$. Define

$$d(u_i) = 2^i, \quad d(v_i) = 3^i, \quad d(w_i) = 5^i, \quad \text{for } i = 1, \ldots, n\ .$$

If we now define

$$G(2^i 3^j 5^k) = \begin{cases} 2(i+j+k) - 1 & \text{if } \min(i,j,k) > 0 \text{ and } (i,j,k) \in M; \\ 2(i+j+k) & \text{else,} \end{cases}$$

then it is clear that finding an optimal solution comes down to finding a matching of maximal cardinality. Since d and G can easily be seen to satisfy (2.19) and (2.20), this proves the theorem.

(2.21) Remark. It would of course be interesting to know how much of an improvement combining tests gives. Some experiments for this have been done, and it seems that a speed-up of no more than 20% will be achieved.

3. CHOOSING s.

(3.1) Assumptions. Throughout this section we assume that the values for t, u, v and w are fixed. It is also assumed that for every prime power $p^k \parallel \operatorname{lcm}(t_1, v)$ the p^k-th cyclotomic constellation L, ζ_{p^k}, σ has been constructed; here $t_1 = \gcd(n^u - 1, t^\infty)$.

The aim of this section is to show how a set X of characters is chosen; in the previous section we discussed the problem of finding optimal combinations of characters for which Jacobi sum tests will have to be performed (but see Remark (3.3)). Again, characters will often be represented by the pair (p^k, q) of its order and its conductor.

The constraints on X will be the following.

Every character (p^k, q) has prime conductor q, and $q - 1$ must divide t_1. By $\prod_X q$ we will denote the product over the set of all different conductors occurring in X; note that this product divides $e(t_1)$, with $e(t_1)$ as in II.(6.14). By s_1 we denote the largest factor $\prod_X q$ that is coprime to t.

Furthermore, we want that X contains generators for all characters modulo s_1, and therefore we impose the following condition on X.

(3.2) *If X contains a character of conductor q, it shall for every prime p dividing $q - 1$ contain a character (p^k, q) with $p^k \parallel q - 1$.*

(3.3) Remark. Notice that the difference between the set of characters X chosen here and that of X in the previous section is that in the previous section X may contain a few more, the "special" characters, cf. (2.2). These do not depend on s, cf. II.(7.8).

Finally, we want to be able to complete the primality test by applying Theorem II.(7.1), using the set X. In particular we want that $s = s_1 t_1 = \operatorname{lcm}(\prod_X q, t_1)$ satisfies the basic inequality, given in (1.2), that $\operatorname{lcm}(s, v) > \sqrt{n}$. Therefore we require that every conductor q is coprime to both t and v, to arrive at a *uniform* lower bound B (i.e. not depending on X) given by the following equation:

$$\prod_X q > B, \tag{3.4}$$

with $B = \sqrt{n}/t_1 v$; as usual the square root may be replaced by a cube root using II.9.

(3.5) Remark. From now on we will assume in this section that X is chosen in such a way that $\prod_X q$ is coprime to tv, so

$$\prod_X q \qquad \text{divides} \qquad e(t_1)t^{-\infty}v^{-\infty};$$

here we have extended the definition of the symbol ∞ as follows: $kr^{-\infty}$ will be the integer obtained by removing all primes from k that occur in r. Then choosing X satisfying (3.2) is equivalent to making a choice for $s = t_1 \prod_X q$, whence the title of this section.

Note that we did not use the extra freedom of multiplying more prime factors of t into s, at the cost of having to do more trial divisions in the end (this is the choice of t' in II.(6.8), see also II.(7.8)).

Before stating our general problem, we introduce another famous combinatorial optimization problem.

(3.6) Knapsacks. One way to describe the knapsack problem is as follows.

Given a finite set U, two functions $s, v : U \to \mathbf{Z}_{\geq 1}$ and a positive integer S. Find a subset $U' \subset U$ such that

$$V = \sum_{u \in U'} v(u) \quad \text{is maximized, under the restriction} \quad \sum_{u \in U'} s(u) \leq S.$$

One may think of this as the problem of maximizing the total value V of the objects chosen out of the finite set, under the restriction that the sum of their sizes does not exceed the size S of the knapsack. Equivalently, but more suited for the subsequent discussion, one has the following formulation (that can be obtained by taking complements).

Given a finite set U, two functions $s, v : U \to \mathbf{Z}_{\geq 1}$ and an integer S. Find a subset $U' \subset U$ such that

$$\sum_{u \in U'} v(u) \quad \text{is minimized, under the restriction} \quad \sum_{u \in U'} s(u) \geq S.$$

This knapsack problem is known to be NP-complete [43], [67], even if one knows the cardinality $\#U'$ beforehand.

Using the same generality for the degree function d and the cost of multiplication function G as used in the previous section, our problem may be stated as follows.

(3.7) The general problem. *Given $B \in \mathbf{R}_{\geq 1}$, finite sets $\bar{I} \subset \mathbf{Z}_{\geq 1}$ and $\bar{J} \subset \mathbf{Z}_{\geq 1}$ and a finite non-empty set $\bar{X}$, with $\bar{X} \subset \bar{I} \times \bar{J}$; also given, functions*

$$d : \bar{X} \to \mathbf{Z}_{\geq 1} \quad \text{and} \quad e : \bar{J} \to \mathbf{Z}_{\geq 1} \quad \text{and} \quad G : \mathbf{Z}_{>0} \to \mathbf{R}_{\geq 0}.$$

Find a subset J of $\bar{J}$ and a disjoint covering $\coprod_{h \in H} W_h = X$, where $X = \bar{X} \cap (\bar{I} \times J)$, with the following properties:

(i) $\prod_{j \in J} e(j) > B$;

(ii) *for every $h \in H$: if $x_1 = (i_1, j_1) \in W_h$ and $x_2 = (i_2, j_2) \in W_h$ with $x_1 \neq x_2$, then $i_1 \neq i_2$;*

(iii) $\sum_{h \in H} G\big(\mathrm{lcm}\{d(x) : x \in W_h\}\big)$ *is minimal.*

(3.8) Comments. The following description shows that (3.7) generalizes the problem of choosing Jacobi sum tests.

Think of $\bar{X}$ as a set of characters to choose from; it consists of pairs (p, q), representing the prime of which the order is a power, and the conductor of the character. The function d assigns to a character the degree of the extension in which the test for that character has to be performed. The function e serves two purposes. It allows different $j \in \bar{J}$ to have the same value $e(j)$; in the present context that is superfluous, but it is useful in the proof of (3.9), where it avoids the awkwardness of the *set* $\bar{J}$ containing like elements. Also, it allows us to value the contribution of the "special characters" towards reaching (3.4) differently (compare II.7). The function G determines the cost of multiplication as a function of the degree of the extension. The problem is to find a subset X of $\bar{X}$ that satisfies (3.2) and (3.4), and a collection H of combinations W_h with minimal costs.

(3.9) Theorem. *Problem* (3.7) *is NP-complete.*

This remains the case if we assume that G satisfies

$$G(u) \geq \sum_{\substack{p \mid u \\ p \text{ prime}}} G\left(\frac{u}{p}\right) \qquad \text{for every } u \tag{3.10}$$

and that d satisfies

$$\text{for every } i \in \bar{I} \text{ and every } j, j' \in \bar{J}: \quad d((i,j)) \mid d((i,j')) \text{ or } d((i,j')) \mid d((i,j)). \tag{3.11}$$

Proof. We prove this by reducing the knapsack problem to the problem described in (3.7), with the restrictions (3.10) and (3.11) imposed.

Let the set U of cardinality $\#U = k$, the functions $s, v : U \to \mathbf{Z}_{\geq 1}$ and the positive integer S be given. Suppose that $b = \#U'$ is the number of items in the solution of the knapsack. Then we choose the parameters in problem (3.7) as follows.

First we enumerate the elements of U in such a way that $U = \{u_1, u_2, \ldots, u_k\}$ with $v(u_1) \leq v(u_2) \leq \cdots \leq v(u_k)$. Choose $B = 2^S - 1$, so $B \in \mathbf{Z}_{\geq 1}$. Choose $I \subset \mathbf{Z}_{\geq 1}$ such that $\#\bar{I} = 1$; then we may as well identify $\bar{X}$ and $\bar{J}$. Without loss of generality we may assume that U consists of a finite number of integers; we choose $\bar{J} = \bar{X} = U$ and we define $e(j) = M + s(j)$ for every $j \in \bar{J}$, and a sufficiently large integer M (it suffices to take $M = (2V)^b$, with $V = \max(v(u_i))$. Choose $B = M^b + M^{b-1}S \in \mathbf{Z}_{\geq 1}$. Furthermore, let $d : U = \bar{X} \to \mathbf{Z}_{\geq 1}$ be defined by $d(u_i) = 2^i$. Next choose $r \geq 2$ large enough (it suffices that $2^{ir} > v(u_i)$ for $1 \leq i \leq k$), and define G as follows:

$$G(m) = \begin{cases} v(u_f), & \text{if } m = 2^f \text{ and } 1 \leq f \leq k \\ m^r, & \text{otherwise.} \end{cases}$$

Clearly $G(d(u)) = v(u)$, for $u \in U$. It is easy to see (3.10) is now satisfied, using (2.16). Also (3.11) holds, by our choice of d.

A solution of (3.7) with these parameters, will consist of a subset $J = X$ of $\bar{J} = \bar{X} = U$ of cardinality $\#J = b$, and a disjoint covering $\coprod_{h \in H} W_h = J = X$, satisfying (i), (ii) and (iii) in (3.7). But

$$\prod_{j \in J} e(j) = \prod_{j \in J} (M + s(j)) > B = M^b + M^{b-1}S \quad \Longleftrightarrow \quad \sum_{j \in J} s(j) \geq S,$$

by the definition of b and our choice of M. Furthermore, (ii) means in this case (since $\#\bar{I} = 1$) that every non-empty W_h shall consist of precisely 1 element. Finally, by (iii), the value of

$$\sum_{h \in H} G\big(\mathrm{lcm}\{d(x) : x \in W_h\}\big) = \sum_{x \in X} G(d(x)) = \sum_{x \in X} v(x)$$

will be minimized.

Choosing $U' = X = J \subset U$, the solution to (3.7) thus solves the knapsack problem.

That proves (3.9).

Even though (3.10) and (3.11) already impose "realistic" restrictions, as we explained in Remark (2.17), we are interested in practice in the following specific problem.

(3.12) The specific problem. *Given* $n, t_1, v \in \mathbf{Z}_{\geq 1}$ *with* $\gcd(n, t_1 v) = 1$, *finite sets*

$$\bar{J} = \{q : q \text{ prime and } q \text{ divides } e(t_1) t_1^{-\infty} v^{-\infty}\},$$
$$\bar{I} = \{p^k : p \text{ prime}, p^k \parallel q-1 \text{ with } q \in \bar{J}\},$$
$$\bar{X} = \{(p^k, q) : p \text{ and } q \text{ prime}, p^k \parallel q-1\} \subset \bar{I} \times \bar{J},$$

and the function $d : \bar{X} \to \mathbf{Z}_{\geq 1}$ *defined by*

$$d : (p^k, q) \mapsto \operatorname{ord} n, \quad \text{the order of } n \text{ in } \left(\mathbf{Z}/p^k\mathbf{Z}\right)^*,$$

as well as the function $G(u) = u^2$ *for* $u \in \mathbf{Z}_{\geq 1}$.
Find a subset J *of* $\bar{J}$ *and a disjoint covering* $\coprod_{h \in H} W_h = X$, *where* $X = \bar{X} \cap (\bar{I} \times J)$, *with the following properties:*

(i) $\prod_{q \in J} q > B$, *where* $B = \frac{\sqrt{n}}{t_1 v}$

(ii) *for every* $h \in H$: *if* $x_1 = (i_1, j_1) \in W_h$ *and* $x_2 = (i_2, j_2) \in W_h$ *with* $x_1 \neq x_2$, *then* $i_1 \neq i_2$;

(iii) $\sum_{h \in H} G\big(\operatorname{lcm}\{d(w) : w \in W_h\}\big)$ *is minimal.*

(3.13) Remark. An even more special (but probably still intractable) case is obtained if we insist that $\#W_h \leq 1$ for every $h \in H$, which means that every combination consists of one test. In other words, this describes the situation in which no combination of tests takes place. This description applies to earlier versions of the Jacobi sum test (cf. [29]).

In the rest of this section we discuss the specific problem (3.12). Since solving (3.12) in general seems to be hard, the description of the optimization will from now on focus on strategies that may not be guaranteed to give the best solution, but, while being efficient, seem to yield a reasonable approximation to the optimum.

(3.14) Cost per prime. The restriction on the subset X of $\bar{X}$ imposed by (3.2) means that if we remove a pair (p^k, q) from $\bar{X}$, we will at the same time remove all pairs with the same q; in other words, in removing tests from $\bar{X}$ we make sure to remove all tests for characters with the same conductor at the same time. Thus $\prod_X q$ will be decreased, and it seems sensible to remove the most expensive q's first. This leads to the notion of the cost $c(q)$ for the prime conductor q, a measure for the time it will take to complete all the tests for the characters with conductor q.

(3.15) Cost per bit. Two problems arise in introducing the cost $c(q)$ per prime q in s. The first is, that this cost should be made *relative*; for larger q we are willing to spend more time, since it will help more in terms of achieving our goal $\prod_X q > B$. Therefore we make the assumption that in comparing different primes q it is reasonable to apply a *cost per bit* criterion; that is, the costs of completing all tests for the characters with conductor q ought to be divided by $\log q$.

(3.16) Dependency of tests. The second problem for the cost function lies in the combinability of tests, as described in the previous section. It may very well be that (some of) the tests that have to be performed for a particular q can be combined with tests for other q, implying that the cost of performing these tests becomes small (or even zero) once the other tests must be done. In other words, the costs for q depend on the rest of X.

There are various ways of treating this difficulty: one may either ignore it (as is done in the first option below), or try to find a reasonable way of taking this dependency into account (the second option).

(3.17) The first cost function. If we ignore the possibility of combining tests, the costs of q per bit are given by

$$c_1(q) = \frac{1}{\log q} \sum_{\substack{p^k \| q-1 \\ p \text{ prime}}} G(u_{p,k}),$$

where $u_{p,k}$ is the order of n in $\left(\mathbf{Z}/p^k\mathbf{Z}\right)^*$.

(3.18) The first algorithm. *Perform step* (i); *put* $J = \bar{J}$ *and repeat step* (ii) *until termination.*

(i) *Compute* $c_1(q)$ *for all* q *in* $\bar{J}$.

(ii) *If* $J' = \{q' \in J : \frac{1}{q'} \prod_J q > B\}$ *is non-empty, let* $q_0 \in J$ *be such that* $c_1(q_0) = \max\{c_1(q') : q' \in J'\}$, *and replace* J *by* $J \setminus \{q_0\}$. *If* J' *is empty, the algorithm is terminated.*

(3.19) The second cost function. Taking combinability into account for the cost function, means letting the cost of q depend on (the rest of) J. One way of doing this is by letting the cost for q be equal to the difference between the minimal cost of performing the tests in J including the tests for q, and the minimal cost of performing the tests in J

excluding those for q. Here the minimal cost $C(J)$ for a given set J can be computed by applying matching algorithm (2.14) and putting

$$C(J) = \sum_{j \in J} G(\mathrm{lcm}\{d(w) : w \in W_j\}),$$

with now d and G as above and the covering W_j as supplied by the matching algorithm. Applying the cost per bit principle we find

$$c_2(q) = c_2(q, J) = \frac{C(J) - C(J \setminus \{q\})}{\log q}.$$

(3.20) The second algorithm. *Put $J = \bar{J}$ and repeat the following two steps until termination.*

(i) *For all q in $\bar{J}$ compute $c_2(q)$ as in* (3.19) *by applying the matching algorithm* (2.15) *to both J and $J \setminus \{q\}$.*

(ii) *If $J' = \{q' \in J : \frac{1}{q'} \prod_J q > B\}$ is non-empty, let $q_0 \in J$ be such that $c_2(q_0) = \max\{c_2(q') : q' \in J'\}$, and replace J by $J \setminus \{q_0\}$. If J' is empty, the algorithm is terminated.*

(3.21) Remarks. Several variants of the above algorithms may be considered. First of all, in both algorithms the strategy of constructing J by deleting primes q from $\bar{J}$ can be replaced by the strategy of building up J from the empty set.

Secondly, in the second algorithm, instead of recalculating the costs for q after every change made to J, one can do such recalculation for instance after five changes have been made. In particular if the set J is very large, this does not make a big difference to the resulting solution, but it speeds up the algorithm considerably.

Finally, it appears to be beneficial in practice to use a small off-set factor: remove q_0 from J only if the product of the remaining q's exceeds B by that factor.

(3.22) Comparing the algorithms. There are two striking differences between the two algorithms for deleting q's. One is that in the first algorithm calculating the cost is much easier; in the second algorithm the matching algorithm has to be applied for every evaluation of the cost function. The second important difference is that in the first algorithm the cost for given q can be calculated once and for all; the second cost function depends on J and must therefore be recalculated for every q after every change made in J.

One would expect that the second algorithm, in which combinability is taken into account, generates a solution closer to optimal. In practice this is not always the case. Usually, the second algorithm gives a slightly better result, but we never found a case in which any of the two algorithms produced a solution with costs not within 2 % of that for the optimal solution.

The fact that the second algorithm does not exhibit the expected superior performance, is perhaps partly due to the limited effect of combining tests anyhow, as pointed out in the previous section.

The optimization step described in the next section involves a search over sets $\bar{X}$. The following strategy, based on the considerations above, has been adopted for constructing J from $\bar{J}$, and hence X from $\bar{X}$.

(3.23) Strategy. Given $\bar{J}$, construct J as follows. Let C' be the cost of the best solution found so far, or, in case no solution has been found yet, put $C' = \infty$. Apply the first algorithm (3.18) to $\bar{J}$ and next apply matching algorithm (2.15) to the resulting set $X = \bar{X} \cap (\bar{I} \times J)$. If the cost $C(X)$ of the solution found this way satisfies $C(X) < bC'$, where b is a blow-up factor, also apply the second algorithm (3.20) to $\bar{J}$.

The blow-up factor has to be chosen in advance, it could for instance be 1.05; its significance is, that every new solution with cost within 5% of the present optimum (for $b = 1.05$) is subjected to the closer scrutiny of the second algorithm.

4. CHOOSING t.

(4.1) Assumptions. Throughout this section the values for v and w are fixed. Moreover we assume that for every prime power $p^k \parallel \operatorname{lcm}(t_0, v)$ we have found a field L for the p^k-th cyclotomic extension for n, of which the degree equals the order of n in $(\mathbf{Z}/p^k\mathbf{Z})^*$. For the choice of the initial value t_0, see (1.4) and II.(6.22).

Now making a choice for t will be described. Since by definition $u = \operatorname{ord} n$, the order of n in $(\mathbf{Z}/t\mathbf{Z})^*$, this will determine u as well. We have to deal with the following constraints. In fact the integer $t_1 = \gcd(n^u - 1, t^\infty)$ will be more important than t.

Since for every maximal prime power divisor of t we need the result of several precomputations (see Section IV.2), we require that t is a divisor of the value t_0, that is chosen once and for all. This implies that u divides u_0, the exponent of $(\mathbf{Z}/t_0\mathbf{Z})^*$. We do not require that t_1 divides t_0. If $t > 1$, we will for simplicity assume that $4 \mid t$, cf. II.(6.17).

Let $e(t_1)$ be as in II.(6.13). Since we want to use Theorem II.(7.1), we have to require by II.(6.14) that

$$\operatorname{lcm}(e(t_1), v) > \sqrt{n}, \tag{4.2}$$

(or $\sqrt[3]{n}$ using II.(9.4)).

(4.3) Exhaustive search. The basic strategy for finding the best value for t is to do an exhaustive search over all divisors of t_0. Although the search space is usually rather restricted by (4.2), a good strategy reduces the work considerably.

(4.4) Small n. For two reasons it is worthwhile to consider adapting the strategy for "small" n. Firstly because the search space quickly expands with decreasing n, and secondly because any reasonable first choice for t will lead to a primality proof that is fast enough anyway.

Therefore one might skip the rest of the optimization step if for the first choice for t the running time for the rest of the algorithm does not exceed a certain threshold value $\bar{T}$. An approach that is a bit more sophisticated, is to decide that the time spent on searching for better t is bounded by some function of the running time for the first choice.

Another suggestion is to apply the algorithm with given t_0 for a somewhat restricted range of primes. It is advisable to choose a smaller t_0 for testing primes of say up to 100

digits than the one in Chapter IV; this will reduce the search space considerably, while hardly affecting the minimal time needed to complete the actual primality proof.

This ties in with the suggestion of building up t from below, instead of constructing it by removing factors from t_0.

(4.5) First estimate. In doing our search, we would like to reject most values for t that are too expensive by comparing a rough lower bound for the cost of using it to the cost of using the best value found so far. That makes the choice of a good first value for t paramount.

Roughly speaking, the amount of work grows with t, and therefore an obvious candidate is the smallest t for which (4.2) holds, which can for instance be found by using a table. In practice, better results are obtained for values of t that exceed the bound in (4.2) by a small offset factor (of about 5%). To such a value t one applies the strategy of finding an optimal factor s_1 of $e(t_1)$ as described in the previous section. Next one calculates C, the minimal costs found so far of executing the primality test, by analyzing the various stages in detail; this is done in Chapter VI, see also below.

(4.6) Rejecting too expensive values. The next step is to reject most other values of t satisfying (4.2) as being too expensive. For this we need a reasonable lower bound for the amount of work that is required for given t. So suppose for the moment that t, u and t_1 are chosen; in the next three paragraphs we give an approximation for the costs of the three most time-consuming parts of the actual primality test. These concern generating the necessary "roots of unity" (the elements ζ_{p^k} in the cyclotomic constellations), performing the Jacobi sum tests, and the final trial division.

(4.7) Roots of unity. For every maximal prime power $p^k \parallel \mathrm{lcm}(t_1, v)$, we have to find, for II.(7.1)(ii), an element ζ_{p^k} that is a zero of the p^k-th cyclotomic polynomial in the ring O_L/nO_L, where L is the field of degree $u_{p,k}$ associated to p^k as in the assumptions (4.1). The way to do this was described in II.(4.15). Basically, it means that a random element of O_L/nO_L must be raised to the power $(n^{u_{p,k}} - 1)/p$. This takes time $O((u_{p,k} \log n)^3)$, using straightforward multiplication. It should be noted that, starting with a random element, this method fails with probability $\frac{1}{p}$, in which case it is repeated for another random element.

In total that would imply that this part requires time $O(\sum_U (u_{p,k} \log n)^3)$, summing over the set U consisting of all primes p dividing $\mathrm{lcm}(t_1, v)$. We can do a little bit bet-

ter though, since we can combine the construction for several roots of unity, as follows. Generating a ${p_1}^{k_1}$-th root of unity in an extension of degree u_{p_1}, and a ${p_2}^{k_2}$-th root of unity in an extension of degree u_{p_2} can be done simultaneously in an extension of degree $\mathrm{lcm}(u_{p_1}, u_{p_2})$. It is only beneficial to do so however, if either $u_{p_1} \mid u_{p_2}$ or vice versa. Thus one gets that the construction of all roots of unity can be done in time $O(\sum_W (u_{p,k} \log n)^3)$, summing over a minimal subset W of U with the property that for every $p \in U$ there exists $p' \in W$ such that $u_{p,k} \mid u_{p'}$. Such a subset W can be found efficiently by a simple greedy algorithm.

For more on this, see V.(4.3).

(4.8) Jacobi sum tests. As was pointed out in (2.3), performing a Jacobi sum test for a character (p^k, m) means taking the n-th power of an element of $u_{p,k}$ coordinates over $\mathbf{Z}/n\mathbf{Z}$; this requires time $O(u_{p,k}^2 (\log n)^3)$. If we want to know what this amounts to in total, we first have to decide what s to use (applying the ideas from Section 3) and next we should combine the tests in an optimal way (using (2.15)). This leads to a contribution $O(\sum_J u_{p,k}^2 (\log n)^3)$, where the index set J is given by the matching algorithm (2.15).

For more on this, see V.(4.4).

(4.9) Final trial division. In the final trial division stage of the algorithm, one checks all different residue classes $n^i \bmod \mathrm{lcm}(e(t_1), v)$ for possible divisors of n. There are at most $\mathrm{lcm}(t_1, u, w)$ of these, and thus this stage requires $O(\mathrm{lcm}(t_1, u, w)(\log n)^2)$.

For more on this, see V.(4.5).

(4.10) Strategy. All this leads to the following strategy for treating the values for t satisfying (4.2), once initial values t^* and C^* have been found (but see (4.4)).

Calculate the time C_t needed to perform the final trial division stage with t as in (4.9). If C_t exceeds C^*, proceed to the next value of t; otherwise add the cost of generating the necessary roots of unity to C_t, as in (4.7). If now C_t exceeds C^*, proceed to the next value of t.

If C_t does still not exceed C^*, this t needs more attention. Apply the strategies from the previous two sections to find good values for s and the set of characters X, and use these to determine the time needed to complete the Jacobi sum tests, as in (4.8). Add this to C_t; if now $C_t < C^*$, we have improved upon our best solution, and we replace it. Next we proceed to the next value of t.

5. CHOOSING v.

(5.1) Assumptions. Throughout this section we assume that a finite set $\bar{\Omega}$ of positive integers has been found, and for every $\omega \in \bar{\Omega}$ a completely factored divisor $v_\omega > 1$ of $n^\omega - 1$.

The aim of this section is to explain how to decide which of the factors v_ω will be used in the Lucas-Lehmer part of the primality test. That means that we want to choose a subset Ω of $\bar{\Omega}$. Since $v = \operatorname{lcm}_{\omega\in\Omega} v_\omega$, this is the same as choosing v.

(5.2) The relation between w and Ω. It is clear that making a choice for Ω determines w, since by definition, w is the smallest degree of the ring extension of $\mathbf{Z}/n\mathbf{Z}$ in which the Lucas-Lehmer test may take place, that is,

$$w = \operatorname{lcm}\{\omega \in \Omega\}.$$

But the converse is not true, w does not generally determine Ω. So we do not just want to choose w, we also decide for which divisors of ω of w we use v_ω. For instance, if we choose $w = 6$, it makes a big difference whether we only use factors of $n-1$, n^2-1 and n^3-1, or we use primitive factors of $n^6 - 1$ as well. The reason is, that in the former case the Lucas-Lehmer tests will all take place in quadratic or cubic extensions, while in the latter we are forced to work in an extension of degree 6. (We will call a prime power factor of $n^\omega - 1$ *primitive* if it is not a divisor of $n^\nu - 1$ for any proper divisor ν of ω.)

(5.3) Performing Lucas-Lehmer type tests. We emphasize that performing a Lucas-Lehmer test for a primitive factor v_ω of $n^\omega - 1$ means that a v_ω-th cyclotomic constellation must be constructed. Essentially, this comes down to finding a v_ω-th root of unity in an extension of degree ω, as in (4.7). This takes time $O((\omega \log n)^3)$ (if we do not combine it with other roots), see V.(4.3).

(5.4) Free factors. Let ω divide $u_{p,k}$, the order of n in $(\mathbf{Z}/p^k\mathbf{Z})^*$, for some maximal prime power divisor p^k of the number t_1 we will use. The necessary v_ω-th root of unity, and the p^k-th root of unity that must be constructed for the Jacobi sum part, can be generated in the same ring, and usually in one stroke. Thus we get the factor v_ω in v almost for free — almost, because of the small possibility of failure. Therefore we would certainly incorporate in Ω all $u_{p,k}$ that appear in $\bar{\Omega}$; note that in particular the factor v_1 is

always "free". The problem however, is that the choice of t (that is as yet unknown), and thus of the $u_{p,k}$, depends on the size of v. The larger v, the smaller t will usually be and thus the fewer "free" factors. Of course, even including factors that are not "free" may be cheaper than having to use a larger t.

Our strategy consists of trying all reasonable subsets Ω of $\bar{\Omega}$ (in a sense that is to be explained), and comparing the costs of each with the minimal costs so far.

(5.5) First estimate. Too find a first upper bound for the costs of completing the primality test, take $w = 1$; these factors are "free" in any case (unless no Jacobi sum test will have to be done at all, but even then one will utilize v_1). Then apply the strategies of the previous sections to find a first approximation C^* of the minimal costs.

(5.6) Upper bound. It only makes sense to include ω in Ω for which the cost of constructing a v_ω-th root of unity alone does not exceed the minimal costs C^* found so far. By (5.3) that means that we have an upper bound $O(\sqrt[3]{C^*}/\log n)$ on the elements of Ω. We may as well replace $\bar{\Omega}$ by its subset of elements not exceeding that bound. Every time we improve C^* the upper bound for $\bar{\Omega}$ decreases.

(5.7) Cost per bit. To further reduce the number of cases to be considered, one may introduce a cost per bit criterion for a set Ω. A way of measuring whether Ω contains "expensive degrees" ω with small contribution v_ω, is to consider the cost per bit function $C(\Omega)/\log v$, where $v = \prod v_\omega$ over Ω and where $C(\Omega)$ is the cost of using Ω, given by

$$C(\Omega) = \sum_{i \in I} (\omega_i \log n)^3;$$

here the summation is over a minimal set such that for every $\omega \in \Omega$ a multiple ω_i is included (compare (4.7)). The cost per bit criterion is that only those Ω would be taken into consideration for which the cost per bit is smaller than the cost per bit in the optimal solution found so far, given by $C^*/\log \sqrt{n}$.

One could, alternatively, apply the cost per bit criterion to individual values of ω.

Some care should be taken in applying this principle though; it may very well be that using a large v with a somewhat larger cost per bit value leads to a dramatic drop in the Jacobi sum costs and thus to an improved solution.

(5.8) Strategy. This leads to the following strategy for choosing the subset Ω of $\bar{\Omega}$. Do the following for the divisors $w = 1, 2, \ldots, \mathrm{lcm}\{\omega \in \bar{\Omega}\}$ of $\mathrm{lcm}\{\omega \in \bar{\Omega}\}$ in succession.

Determine all subsets Ω of $\bar{\Omega}$ with the property that $\mathrm{lcm}\{\omega \in \Omega\} = w$. Apply some form of the cost per bit criterion to decide which of these require further investigation. For the sets thus selected apply the techniques of the previous section to find an approximately optimal choice for the other parameters and calculate the cost of using these. Whenever the costs are smaller than the minimal costs so far, replace the optimal solution, replace $\bar{\Omega}$ by a subset if possible, using an upper bound as in (5.6), and continue.

6. FACTORING TIME.

This section deals with the problem of deciding how much time should be spent on the search for more Lucas-Lehmer factors. Since Lucas-Lehmer type tests are cheaper than Jacobi sum tests, increasing v and decreasing s usually leads to a cheaper primality proof; however, the cost of finding extra factors has to be taken into account, and thus another optimization problem arises.

The starting point will be the situation in which an approximation C^* for the costs of the primality test has been found, using the present set $\bar{\Omega}$ and the factors v_ω for $\omega \in \bar{\Omega}$. By s^*, t^*, u^* and v^*, w^* we will denote the values of the parameters for which the minimal cost C^* found so far, is realized.

(6.1) Finding factors. Finding additional factors in $n^i - 1$ for small i will be done by trial division, using a table of primes. This is done as follows. Choose a value W; we will search for factors in $n^i - 1$ for $1 \leq i \leq W$ simultaneously. Next choose a lower bound A and an upper bound B, and check for all primes p with $A \leq p \leq B$ in succession whether any of $\bar{n}, \bar{n}^2, \ldots, \bar{n}^W$ is congruent to 1 modulo p, where $\bar{n}$ is the reduction of n modulo p. This can be done in time $O(\log n + W)$. Since there are approximately $B/\log B$ primes up to B, checking the range from A to be can be done in time

$$O\Big((\log n + W)\big(\frac{B}{\log B} - \frac{A}{\log A}\big)\Big).$$

(We should remark that this accounts only time $O(1)$ to calculate $\bar{n}$, which is motivated by the fact that in practice B will be single-precision, and hence reduction modulo p is cheap.)

(6.2) Other factorization methods. For huge n it may very well be that one is willing to spend more time on factoring once the prime table is exhausted. If the total time for completing the test would be in terms of weeks or even months, one might try one's luck in applying a Pollard method or an elliptic curve method, on $n - 1$ and $n + 1$, for a few days. Using heuristics for these methods and the considerations below, an estimate for the probability of improvement by these methods can be obtained similarly as for trial division.

(6.3) Randomness. The basic assumption underlying the following analysis, is that with respect to the distribution of their prime divisors, the numbers $n^i - 1$ behave as random

numbers. That means that we expect to find roughly $\log\log B - \log\log A$ prime factors of $n^i - 1$ between A and B, each of size roughly $(A+B)/2$. Searching this range leads under the randomness hypothesis thus to an extra factor in v_i of size roughly:

$$\left(\frac{A+B}{2}\right)^{\log\log B - \log\log A}.$$

(6.4) Search bound. It is easy to find an upper B bound for the factor search in terms of the best solution found so far: suppose that we spend all of the currently found minimal time C^* on finding more factors, then by (6.1)

$$C^* = (\log n + W)\left(\frac{\log B}{B} - \frac{\log A}{A}\right),$$

where A is the previous search bound and where we choose W as follows. Recall that W will be the maximal value for i for which factors of $n^i - 1$ will be taken into consideration; since we are particularly interested in "free" factors, see (5.4), we will choose

$$W = \max\{u_{p,k} : u_{p,k} = \operatorname{ord} n, \text{ the order of } n \text{ in } (\mathbf{Z}/p^k\mathbf{Z})^*\},$$

the maximum taken over all maximal prime power divisors $p^k \mid t^*$, with t^* the value of t in the currently found minimal solution.

(6.5) Range. In the best known solution so far, the value for s^* will be approximately equal to $\sqrt{n}/v^*$. (Approximately, because there may be an overshoot, and because we ignored common factors.)

Suppose that we conduct a search for factors over primes in the interval from A to B as in (6.4). Then we expect to increase v^* by the factor

$$\left(\frac{A+B}{2}\right)^{\log\log B - \log\log A},$$

and thus s^* may be decreased by the same factor to a value we denote by s_*. The optimization now comes down to scanning the interval $[s_*, s^*]$ for s for an optimal solution, which corresponds to varying the time spent on factoring from maximal to zero.

(6.6) Linearity hypothesis. It turns out that the costs of performing the Jacobi sum primality test are almost linear in $\log s$ as long as the value for t is fixed. This means that

the function describing these costs is almost piecewise linear as a function of $\log s$, with discontinuities whenever one is forced to use a larger t. The slope of this linear function depends on specific properties of n, particularly its residue classes modulo small prime powers (those in t_0).

(6.7) Minimizing. The piecewise linearity of the cost function makes it easy to perform the minimization over $[s_*, s^*]$ as proposed in (6.5); first apply the techniques of the previous sections to find an approximately optimal solution with s of size s_* (by which we mean that the lower bound for s will be $s_* = \sqrt{n}/v_*$). Two cases have to be distinguished now.

If in this solution the same value t^* for t is used, we may be reasonably confident that the cost for the Jacobi sum part of the test is given by a linearly increasing function on $[s_*, s^*]$, which we can write down explicitly since we know its value in the end points. Optimizing means minimizing the sum of this linear function and the function expressing the costs of the Lucas-Lehmer part. Both can be written down explicitly (see below) and thus we are left with an easy minimization problem of a function in one variable (s). This yields an optimal value $\tilde{s} \in [s_*, s^*]$.

If, on the other hand, another value than t^* appears in the minimal solution at the end point s_*, there will probably be a discontinuity (and usually just one) in the cost function for the Jacobi sum part on the interval. This discontinuity can be approximated as follows. Minimize as in the previous case, i.e., as if the linearity hypothesis holds to find $\tilde{s}$; next perform an optimization with a Jacobi sum test of size $\tilde{s}$ (as we did before for size s_*). The t used in the solution tells on which side of the discontinuity $\tilde{s}$ is. Repeating this step a few times, we find an approximation to the discontinuity. Usually the best value for s is just to the left of this continuity; anyway an optimal value $\tilde{s} \in [s_*, s^*]$ can again be found.

(6.8) Cost function. Now we give the cost function f explicitly as a function of s that is to be minimized under the assumption of linearity (6.6). Firstly, $f = f' + f''$, the cost of the Jacobi sum and the cost of the Lucas-Lehmer part respectively.

Now f' is just the function

$$f'(s) = f'(s_*)\left(\frac{f'(s^*) - f'(s_*)}{\log s^* - \log s_*}\right)(\log s - \log s_*),$$

linear in $\log s$.

The function f'' consists of two parts: the costs of factoring and the costs of performing the Lucas-Lehmer tests for the non-free factors. The costs for factoring are calculated as follows. To complete the test with s, we must have that

$$sv^*F > \sqrt{n},$$

where F is the product over the factors to be found. By (6.3) a search bound B_F is expected to be given by

$$F = \left(\frac{A + B_F}{2}\right)^{\log\log B_F - \log\log A}$$

and by (6.1) the time needed for factoring will be

$$(\log n + W)\left(\frac{B_F}{\log B_F} - \frac{A}{\log A}\right).$$

But after factoring we will have found factors of $n^\omega - 1$ for all $1 \leq \omega \leq W$, including those not occurring in the set

$$\{u_{p,k} : \quad u_{p,k} = \operatorname{ord} n, \text{ the order of } n \text{ in } \left(\mathbf{Z}/p^k\mathbf{Z}\right)^*, \text{ with } p^k \parallel t_1^*\}.$$

But these are not free: we will have to generate cyclotomic constellations of degree ω for them. By (4.7) this gives to f'' a contribution

$$\sum_\omega (\omega \log n)^3,$$

summing over the $\omega \leq W$ not occurring as $u_{p,k}$.

All in all, we have now found a value $\tilde{s} \in [s_*, s^*]$ which we expect to be optimal. Just as we explained in (6.8), this value determines the size of the factor that has to be found by factoring and hence a search bound B.

Finally it will be time to transform expectations into facts: the trial division now should be executed. Once this has been done, v will be chosen as in the previous section, etcetera.

IV. ALGORITHM.

1. OUTLINE OF THE ALGORITHM.

(1.1) Remark. This section contains a rough outline of the primality testing algorithm that is described in the following sections in detail. We will also comment upon the differences with its predecessor, the Cohen-Lenstra version of the Adleman-Pomerance-Rumely primality proving algorithm, which we will refer to as *the old algorithm*, (cf. [2], [29], [30]).

The detailed description in the next five sections is interspersed with comments (in small print), which do not form part of the algorthm. They are meant to elucidate the steps of the algorithm, and provide references to the other chapters. The following five steps correspond to these five sections.

(1.2) Preparation of tables. This pre-calculation step is done once and for all. The integers t_0 and s_0 are chosen; the auxiliary numbers t and s, to be chosen in step (1.4), will be divisors of t_0 and s_0 respectively. The choice of s_0 gives an upper bound for the size of the integers that can be dealt with by the Jacobi sum test alone: n may not exceed s_0^3 (cf. II.9). Furthermore, all tables that will be needed, and that can be created irrespective of the arithmetic properties of n, are generated. They include: a list of primes, a list of data for the extension rings, a list of Jacobi sums and a table of exponents to express certain quotients of Gauss sums as products of Jacobi sums.

(1.3) Initializations. This is the preliminary step for testing n. It is checked whether n is not obviously composite, by subjecting it to a compositeness test and some trial divisions by small primes; during the latter at the same time factors of $n-1$ and $n+1$ are found. Here also known factors for $n^w - 1$ for small values of w, for instance found before by someone else, can be read into the program.

(1.4) Optimization. Integers s, t, u, v and w have to be chosen; roughly speaking these have the following meaning. The completely factored part of $n^w - 1$ is denoted by v; it will consist of the factors found in the previous steps and new factors found here by trial division of $n^w - 1$ for small w. The integer s will be a product $\prod q$ of primes q with the property that $q - 1 \mid t$; also we want s to be large, in particular we want $s \cdot v > n^\mu$, with $\mu = \frac{1}{2}$ or $\mu = \frac{1}{3}$. Therefore t can best be built up from powers of small primes; we require $t \mid t_0$. The integer u is the total degree of the ring extension in which the Jacobi sum test will take place.

An effort is made in this step, to determine the auxiliary integers in such a way that the total running time of the algorithm will be minimal. The main contributions to the running time are the time spent on looking for more factors of $n^w - 1$, the time needed to construct the necessary roots of unity in the extension rings, the time necessary for doing the Jacobi sum tests, which is influenced by how well they combine, and the time necessary to do the final trial divisions.

(1.5) Lucas-Lehmer and Jacobi sum tests. A Jacobi sum test has to be performed for every pair (p^k, q) consisting of a prime power p^k dividing $q-1$ and a prime factor q of s. Basically, this comes down to raising a product of Jacobi sums to a power which is of the same magnitude as n and checking that the result equals some p^k-th root of unity; all this is done in a ring extension of $\mathbf{Z}/n\mathbf{Z}$ of degree equal to the order of n in $\left(\mathbf{Z}/p^k\mathbf{Z}\right)^*$, which divides u. Several of these tests will be combined to one test in common ring extension of $\mathbf{Z}/n\mathbf{Z}$, of which the degree is equal to the maximum of, and divisible by each of, the degrees of the individual tests in the combination. Performing the Lucas-Lehmer test comes down to finding v-th roots of unity in the proper w-th degree ring extension of $\mathbf{Z}/n\mathbf{Z}$.

(1.6) Final trial divisions. If the preceding tests have been performed successfully, then every divisor of n must be a power of n modulo $s \cdot v$. Thus the final step consists of finding out whether there exist integers r dividing n in the residue class $r_i \equiv n^i \bmod (s \cdot v)$, with $1 \leq i < t \cdot w$.

(1.7) Remarks. There are three important differences between this algorithm and the old algorithm, which have far-reaching consequences, especially in step (1.4) above.

In the first place there is the combination of the Jacobi sum test with the Lucas-Lehmer type tests. Every factor found in $n^w - 1$ contributes to v above. This makes it possible to decrease s by the same factor. Of course one has to balance the cost of finding extra factors and performing the Lucas-Lehmer tests against the expected gain of having less Jacobi sum tests to perform.

Secondly, an observation made in [30] is used, namely that the Jacobi sum tests can be done in a ring extension of $\mathbf{Z}/n\mathbf{Z}$ of degree equal to the order of n in $\left(\mathbf{Z}/p^k\mathbf{Z}\right)^*$ instead of the ring $\mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$, which is of degree $\phi(p^k)$. Usually the degree used is much smaller than $\phi(p^k)$; it may be equal to 1, in which case this observation was also used in the implementation of the old algorithm.

As an aside it should be remarked that in fact the fixed extension $\mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$ has been replaced by a list of extensions of $\mathbf{Z}/n\mathbf{Z}$, all having a non-zero probability of at most $\frac{1}{2}$ of not being suitable for a given n. That makes it more cumbersome to find efficient multiplication algorithms in each of the extension rings, than in the old algorithm.

Thirdly, advantageous use is made of the possibility to combine several Jacobi sum tests in rings of degree u_i into one large test in a ring of degree $\mathrm{lcm}(u_i)$ (compare II.(8.6)). Of course this only makes sense if the resulting amount of work is less than it would have been without combining. The Jacobi sum tests consist roughly of n-th powerings of elements that are represented as tensor products over $\mathbf{Z}/n\mathbf{Z}$ modulo polynomials of prime power degree $l^e \parallel u_i$. Assuming that the time to perform one multiplication is quadratic in the number of coordinates, combining a set of tests in rings of degree u_i into one test only makes sense if $\sum u_i^2 > (\mathrm{lcm}(u_i))^2$, where the sum is taken over different values of u_i; tests in rings having the same degree can be combined without any extra costs. Using that the u_i are built up from small prime powers, this is easily seen to be true only in case $\max\{u_i\} = \mathrm{lcm}(u_i)$ (see also III.2); that is, if all u_i divide the maximal u in the combination. It is part of the optimization step to determine the optimal combination of tests for given s, t and u. There is an easy, efficient procedure for doing that. On the other hand it is hard to find out for what choice of s and t the optimal choice will be minimal (for more on this, see Sections III.3 and III.4).

All this makes the optimization step much more complicated than in the old algorithm; but especially for large n the investment made pays off tremendously (see Chapter VI).

It should be remarked that the first and second of the differences pointed out above, and the use of $s \cdot v \geq \sqrt[3]{n}$ instead of $s \cdot v \geq \sqrt[2]{n}$, are the main contributions to the improvement of the primality test over the old Jacobi sum test.

(1.8) Conventional notation. In the next sections we will use the phrase "$a = b \bmod c$" when a is defined to be the unique integer in $\{0, 1, \ldots, c-1\}$ congruent to b modulo c.

2. PREPARATION OF TABLES.

Perform steps (2.1) through (2.5).

In this section a description is given of the preparation of the tables needed in the primality test described in Sections 3 through 6. It should be emphasized that the work done here is done once and for all, and is independent of properties, other than the size, of the integers n that are to be tested. The choice of the parameters determines the size of the integers that can be tested. With the choices made below, every n of up to 6000 digits (and some larger n) can be dealt with; the optimization routines work best however for integers n up to about 1500 decimal digits.

(2.1) Creation of a file to generate prime numbers.

Select a positive integer B_0, and create a file containing the differences between the consecutive primes up to B_0.

To find small divisors of a large number (e.g. in steps (3.2) and (4.6)), one can use a table of prime numbers up to a certain bound B_0. This table is made here. The bound B_0 could for instance be 10^6.

(2.2) Selection of t_0.

Perform steps (a) through (h).

In the Jacobi sum test one needs auxiliary integers t and s with the property that t is small, while s is large and built up from primes q such that $q-1 \mid t$ (cf. II.6). In this step a number t_0 and sets $\mathcal{P}_0$ and $\mathcal{Q}_0$ are chosen. The value of t, to be selected for a specific n during the optimization step, will divide t_0 and is built up from powers of small primes in a subset $\mathcal{P}$ of $\mathcal{P}_0$, and s will be built up from primes in a subset $\mathcal{Q}$ of $\mathcal{Q}_0$.

(a) Select a positive integer t_0 with $t_0 \equiv 0 \bmod 4$.

The integer t_0 should be "rich" in the sense that it is small but $q-1$ divides t_0 for many primes q. For a table of nice values see II.6. In II.(6.17) it is explained why preferably $t_0 \equiv 0 \bmod 4$. A possible choice is $t_0 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 6983776800$.

(b) Let $\mathcal{P}_0 = \{p : p \text{ prime}, p \mid t_0\}$, and let k_p be given by $\prod_{p \in \mathcal{P}_0} p^{k_p} = t_0$.

(c) Calculate $\phi(p^k) = (p-1)p^{k-1}$ for $p \in \mathcal{P}_0$ and $1 \leq k \leq k_p$.

(d) Put $\lambda(t_0) = \operatorname{lcm}\{2^k,\ (p-1)p^{k_p-1} :\ p \in \mathcal{P}_0, p \text{ odd}\}$ with $k = \max\{1, k_2 - 2\}$, let $\mathcal{L}_0 = \{l : l \text{ prime}, l \mid \lambda(t_0)\}$, and let e_l be given by $\prod_{l \in \mathcal{L}_0} l^{e_l} = \lambda(t_0)$.

The integer $\lambda(t_0)$ is the exponent of $(\mathbf{Z}/t_0\mathbf{Z})^*$, i.e., the maximal order of the elements in the group $(\mathbf{Z}/t_0\mathbf{Z})^*$. For the choice of t_0 suggested above, one gets $\lambda(t_0) = \operatorname{lcm}\{8, 18, 20, 6, 10, 12, 16, 18\} = 2^4 \cdot 3^2 \cdot 5$.

(e) Determine $\mathcal{Q}_0 = \{q : q \text{ prime}, q-1 \mid t_0\}$.

For the above choice of t_0 one gets $\#\mathcal{Q}_0 = 618$, and $\prod_{q \in \mathcal{Q}_0} q$ just exceeds $5.3 \cdot 10^{3000}$. This product determines the maximal size of integers that can be taken care of by the Jacobi sum test alone; one can deal with numbers of size (= logarithm) up to twice (or even thrice, see II.9) the size of this product times $2t_0$.

(f) For all $q \in \mathcal{Q}_0$ and $p \in \mathcal{P}_0$ determine $o_p(q-1)$, the number of times that p occurs in the prime factorization of $q-1$.

(g) For every divisor t of t_0 such that $t \equiv 0 \bmod 4$, calculate

$$i(t) = \sum_{p \in \mathcal{P}_0} o'_p(t) \prod_{\substack{p' > p \\ p' \in \mathcal{P}_0}} (k_{p'} + 1) \quad \text{and} \quad e'_{i(t)} = \prod_{\substack{q-1 \mid t \\ q \text{ prime}}} q \ ;$$

here we define $o'_p(t)$ by $o'_2(t) = o_2(t) - 2$ and $o'_p(t) = o_p(t)$ for other primes p, with $o_p(t)$ the number of times that p occurs in the prime factorization of t.

In the optimization step, one uses a divisor t of t_0 for which the primality test can be finished; i.e., for which $e(t)$ is large enough. Note that $e(t)$ as in II.(6.12) is the product of $2t$ and the value $e'_{i(t)}$ given here. In order to be able to retrieve these values quickly for all divisors t of a given t', they are indexed by the number $i(t)$, running from 0 to one less than the total number of divisors congruent to 0 mod 4. The numbers k_p are defined in (2.2)(b).

(h) Tabulate t_0, $\mathcal{P}_0$, k_p for $p \in \mathcal{P}_0$, $\phi(p^k)$ for $p \in \mathcal{P}_0$ and $1 \le k \le k_p$, $\lambda(t_0)$, $\mathcal{L}_0$, e_l for $l \in \mathcal{L}_0$, $\mathcal{Q}_0$, and $o_p(q-1)$ for $q \in \mathcal{Q}_0$, $p \in \mathcal{P}_0$. For $i = 0, \ldots, \#\{t : t \mid t_0,\ 4 \mid t\} - 1$, also tabulate the pair $(t,\ \log e'_{i(t)})$, where t is such that $i(t) = i$ (cf. (2.2)(g)).

(2.3) Generation of Galois extensions.

Perform steps (a) through (c) for all prime powers $u = l^e$ dividing $\lambda(t_0)$, with $e > 0$.

In the Jacobi sum test one will compute in certain extension rings of $\mathbf{Z}/n\mathbf{Z}$; as explained in II.(4.11) these rings can be constructed from rings of integers of cyclic subfields (of prime power degrees dividing $\lambda(t_0)$) of cyclotomic fields. Here some preliminary steps are performed without knowledge of the integer n; they will facilitate arithmetic in the extension rings later on. Since a given cyclic field of degree l^e has for random n a probability of $l-1$ out of l to provide a "good" ring (useful in the test of n), one pre-calculates a list of such extensions for every l^e.

(a) Put $\mathcal{M}(u)$ equal to the empty set.

The set $\mathcal{M}(u)$ will in the end contain the conductors m of the field extensions of degree u from which the ring extensions will be constructed.

(b) Select a constant C.

If $u = 2$ perform steps (b1), (b2), (b3), and (b10) for every $m \in \{4, 8\}$ and steps (b1), (b2), (b4), (b5) and (b10) for every $m = k \cdot u + 1$ for which m is prime and $m \le C \cdot l^{e_l}$.

If $u = 2^e$ with $e > 1$, perform steps (b1), and (b5) through (b10) for every $m = k \cdot u + 1$ for which m is prime and $m \le C \cdot l^{e_l}$, as well as for $m = 2^{e+2}$.

If $u = l^e$ with l odd and $e \ge 1$, perform steps (b1), and (b5) through (b10) for every $m = k \cdot u + 1$ for which m is prime and $m \le C \cdot l^{e_l}$, as well as for $m = l^{e+1}$.

As was pointed out in II.4, every prime conductor m for which $u \mid m-1$ provides a cyclic field of degree u; moreover, for $u=2$ there are 3 quadratic fields inside $\mathbf{Q}(\zeta_8)$ that can be used, corresponding to conductors 4, 8 and 8. From II.(4.10) it follows that if the field of conductor 8 generated by $\zeta_8 - \zeta_8^{-1}$ can be used in the test for n, then at least one of the other fields might also be used. Therefore the field of conductor 8 with generator $\zeta_8 + \zeta_8^{-1}$ and the field of conductor 4 with generator ζ_4 suffice.

For $u = 2^e$ with $e > 1$ and $u = l^e$ with l odd there are additional useful extensions of conductor 2^{e+2} and l^{e+1} respectively, cf II.(4.6). The constant C gives an arbitrary bound on the size – and thus on the number – of conductors in the tables. One could for instance use $C = 10$.

(b1) Replace the set $\mathcal{M}(u)$ by $\mathcal{M}(u) \cup \{m\}$. Put $r = 1$ if m is prime, and put $r = 0$ if m is not prime.

(b2) Put $D = 1$. If m is prime put

$$S = \begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix} \text{ and } S^* = \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}.$$

If $m \in \{4, 8\}$ put S and S^* both equal to the 2×2 unit matrix.

In general, the matrices S and S^* will be the transition matrices between the bases consisting of powers of the generator η of the ring (see II.(4.11)) and that consisting of the elements $\varsigma_{g,i,u}$ for $i = 0, \ldots, u-1$. For $u = 2$ and m is prime we have that $\varsigma_{g,0,u} = \sigma_g^0(\eta_u) = \eta_u = \eta$ and $\varsigma_{g,1,u} = \sigma_g^1(\eta_u) = \sigma_g(\eta)$; for $u = 2$ and $m \in \{4, 8\}$ we have that $\varsigma_{g,0,u} = 1$ and $\varsigma_{g,1,u} = \eta_u = \eta$. The isomorphism σ_g is defined by $\sigma_g(\zeta_m) = \zeta_m^g$, for some element g of order $\lambda(m)$ modulo m (cf. (2.2)(d)). For $u \neq 2$ the elements $\varsigma_{g,i,u}$ for $i = 0, \ldots, u-1$ will be defined below. The number $D \in \mathbf{Z}_{>0}$ will be the denominator of S^{-1}, i.e., the smallest integer such that $S^* = D \cdot S^{-1}$ is an integral matrix. The matrix S will express an element which is represented in the basis $\{\eta^0, \eta^1, \ldots, \eta^{u-1}\}$ in terms of the elements $\varsigma_{g,i,u}$ for $i = 0, \ldots, u-1$. The matrix $D^{-1} \cdot S^*$ performs the inverse operation.

(b3) If $m = 4$ put $g = 3$ and $f = X^2 + 1$; if $m = 8$ put $g = 5$ and $f = X^2 - 2$.

In the sequel $g \in (\mathbf{Z}/m\mathbf{Z})^*$ will be such that the restriction of $\zeta_m \mapsto \zeta_m^g$ generates the Galois group of a u-th degree cyclic subextension of $\mathbf{Q}(\zeta_m)$, where m is the conductor. Furthermore, f will be the minimal polynomial for η, the generator of the cyclic field, which equals here ζ_4, respectively $\zeta_8 + \zeta_8^{-1}$, see II.(4.10).

(b4) Put $f = X^2 + X + (4 \cdot \lfloor \frac{m+1}{4} \rfloor - m) \cdot \lfloor \frac{m+1}{4} \rfloor$.

This is the minimal polynomial $X^2 + X + (1 \mp m)/4$ for the generator

$$\eta = \sum_{i=0}^{(m-3)/2} \zeta_m^{g^{2i}}$$

of our ring in the quadratic subfield $\mathbf{Q}(\sqrt{\pm m})$ of $\mathbf{Q}(\zeta_m)$, where g is a primitive root modulo m. The sign under the square root equals the Legendre symbol $(\frac{-4}{m})$, so the square root is $\sqrt{-m}$ for $m \equiv 3 \bmod 4$ and $\sqrt{m}$ for $m \equiv 1 \bmod 4$.

(b5) If m is odd, find a primitive root g modulo m, for instance by trying all $g \geq 2$ with $\gcd(m, g) = 1$ in succession. If $m = 2^{e+2}$, put $g = 5$.

It would be more efficient to determine a primitive root g modulo m for each m only once, but since this table is made only once, and since the time to determine a primitive root is

relatively small in comparison to the time needed to generate other elements in the table, this improvement will not be described.

(b6) Let $b_{i,j} = 0$ for $0 \leq i < u$ and $0 \leq j < m$. Put $j = 1$. If m is odd, for $i = 0, 1, \ldots, \phi(m) - 1$ in succession replace $b_{i \bmod u, j}$ by 1 and j by $j \cdot g \bmod m$. If $m = 2^{e+2}$, for $i = 0, 1, \ldots, 2^e - 1$ in succession replace $b_{i \bmod u, j}$ and $b_{i \bmod u, m-j}$ by 1 and j by $j \cdot g \bmod m$. Define $\eta = \sum_{j=0}^{m-1} b_{0,j} \cdot \zeta_m^j$ and, for $i = 0, 1, \ldots, u-1$, its conjugates $\sigma_g^i(\eta) = \sum_{j=0}^{m-1} b_{i,j} \cdot \zeta_m^j$. The $\sigma_g^i(\eta)$ will be represented by the vectors $(b_{i,j})_{j=0}^{m-1}$.

The element $\eta = \eta_u$ generates a cyclic subfield of degree u inside $\mathbf{Q}(\zeta_m)$ by II.(4.8). The conjugates $\sigma_g^i(\eta)$ are also computed, where σ_g acts via $\sigma_g(\zeta_m) = \zeta_m^g$. Notice that $\sigma_g^0(\eta) = \eta$; also notice that the representation given is not unique (since the powers $\zeta_m^0, \zeta_m^1, \ldots, \zeta_m^{m-1}$ are dependent) and that $b_{i,j} = 0$ for $0 \leq i < u$, $0 \leq j < m$ with $\gcd(j, m) \neq 1$.

(b7) Determine the polynomial $f = \prod_{i=0}^{u-1}(X - \sigma_g^i(\eta)) \in \mathbf{Z}[X]$ by calculating sufficiently close approximations $c_i \in \mathbf{C}$ to $\sum_{j=1}^{m-1} b_{i,j} \cdot \zeta_m^j$ for $0 \leq i < u$, with $\zeta_m = e^{2\pi\sqrt{-1}/m}$, and by rounding the coefficients of the polynomial $\prod_{i=0}^{u-1}(X - c_i)$ to the nearest integers.

Here f is again the minimal polynomial of η over $\mathbf{Q}$; writing $f = \sum f_i X^i$, one should notice that $f_u = 1$ and $f_{u-1} = r$, with r as in (b1).

Since all coefficients $|b_i| \leq 1$ the value c_i approximates $\sum_{j=1}^{m-1} b_{i,j} \cdot \zeta_m^j$ within $m \cdot \epsilon$, where ϵ is the absolute error made in the calculation of $\zeta_m = e^{2\pi\sqrt{-1}/m}$. Since $|\zeta_m| \leq 1$, this is at most equal to the machine-precision. For the machines we used the machine-precision was at most 2^{-24}.

(b8) If m is prime, define $\varsigma_{g,k,u}$ for $0 \leq k < u$ by

$$\varsigma_{g,k,u} = \sigma_g^k(\eta_u) = \sigma_g^k(\eta).$$

If m is not prime, define $\varsigma_{g,k,u}$ for $0 \leq k < u$ by $\varsigma_{g,0,u} = 1$, and

$$\varsigma_{g,k,u} = \sigma_g^{(k-l^{i-1})}(\eta_{l^i}),$$

with i such that $1 \leq i \leq e$ and $l^{i-1} \leq k < l^i$. Compute a $u \times u$ dimensional integer matrix S by performing steps (b8a) through (b8c).

The matrix S is to convert an element expressed in the basis of $\eta^0, \ldots, \eta^{u-1}$ to its representation in the basis $\varsigma_{g,0,u}, \varsigma_{g,1,u}, \ldots, \varsigma_{g,u-1,u}$. For any element $x \in \mathbf{Z}[\eta]$ which is represented as a u-dimensional column vector over $\mathbf{Z}$, such that $x = \sum_{i=0}^{u-1} x_i \cdot \eta^i$, the u-dimensional column vector $y = S \cdot x$ represents the same element with respect to the basis $\varsigma_{g,0,u}, \varsigma_{g,1,u}, \ldots, \varsigma_{g,u-1,u}$:

$$\sum_{i=0}^{u-1} y_i \cdot \varsigma_{g,i,u} = \sum_{i=0}^{u-1} x_i \cdot \eta^i.$$

The element η_{l^i} generates the cyclic subfield of degree l^i inside $\mathbf{Q}(\zeta_m)$ for $i = 1, \ldots, e$, see II.(4.7). In case m is not prime, the basis $\varsigma_{g,0,u}, \varsigma_{g,1,u}, \ldots, \varsigma_{g,u-1,u}$ consists of the basis

of the cyclic subfield of degree u/l with generator $\eta_{u/l}$ extended with the conjugates $\sigma_g^k(\eta)$, for $0 \leq k < u - u/l$; the basis of the cyclic subfield of degree u/l equals 1 if $e = 1$ and equals $\varsigma_{g,0,u/l}, \varsigma_{g,1,u/l}, \ldots, \varsigma_{g,u/l-1,u/l}$ if $e > 1$.

(b8a) Determine $d_{i,j}$ such that $\eta^i = \sum_{j=0}^{m-1} d_{i,j} \cdot \zeta_m^j$ for $i = 2, 3, \ldots, u-1$, by computing the consecutive powers of the polynomial $\sum_{j=0}^{m-1} b_{i,j} \cdot T^j$ modulo the polynomial $T^m - 1$.

The powers of η are expressed as linear combinations of $\zeta_m^0, \zeta_m^1, \ldots, \zeta_m^{m-1}$. Note again that this is not a unique representation.

(b8b) Perform step (b8b1) if m is prime and perform step (b8b2) for $k = 1, \ldots, e$ in succession if m is not a prime.

(b8b1) First put $h = 1$ and for $j = 0, 1, \ldots, u-1$ in succession first put $s_{j,i} = d_{i,h} - d_{i,0}$ for $0 \leq i < u$ and next replace h by $h \cdot g \bmod m$.

(b8b2) Put $h = 1$ and for $j = l^{k-1}, \ldots, l^k - 1$ in succession first put $h' = (hm/l^k) \bmod m$, next put $s_{j,i} = d_{i,h'}$ for $0 \leq i < u$ and finally replace h by $h \cdot g \bmod m$. For $j = l^k, \ldots, l^k + l^{k-1} - 1$ in succession first put $h' = (hm/l^k) \bmod m$, next put $s_{(j-zl^{k-1}),i} = s_{(j-zl^{k-1}),i} - d_{i,h'}$ for $z = 1, \ldots, l-1$ and $0 \leq i < u$ and finally replace h by $h \cdot g \bmod m$.

(b8c) Put $s_{0,i} = d_{i,0}$ for $0 \leq i < u$. Let S be the matrix having the $(s_{j,i})_{j=0}^{u-1}$ as columns, for $0 \leq i < u$.

(b9) Compute the $u \times u$ dimensional integer matrix S^* and $D \in \mathbf{Z}_{>0}$ such that $D^{-1} \cdot S^* = S^{-1}$ and D is minimal. This can efficiently be done using a variant of the Gaussian elimination method. See V.6 for more details.

(b10) Tabulate u, $m_u = m$, $r_{u,m} = r$, $g_{u,m} = g$, $f_{u,m} = f$, $S_{u,m} = S$, $S^*_{u,m} = S^*$, and $D_{u,m} = D$.

(c) Tabulate $\mathcal{M}(u)$.

(2.4) Replacing Gauss sums by Jacobi sums.

Perform steps (a) through (f).

In this step a method is described to determine a set of prime numbers $\mathcal{J}$, and for every prime $\pi \in \mathcal{J}$ one pair of positive integers (a, b) with $a + b = \pi$, as well as a set of expressions $e = e_{\pi,p^k,i}$ of the form $e = \sum z_j \sigma_j$ with $z_j \in \mathbf{Z}_{\geq 0}$ and j ranging over $(\mathbf{Z}/p^k\mathbf{Z})^*$, for every $\pi \in \mathcal{J}$, every $p^k \mid t_0$ and $1 \leq i \leq p^k$ with $p \nmid i$. One should think of the prime $\pi \in \mathcal{J}$ and the pair (a, b) as a representation of the Jacobi sum $J_\pi = J(\chi^a, \chi^b) = \tau(\chi^a)\tau(\chi^b)/\tau(\chi^\pi)$, and the exponents $e_{\pi,p^k,i}$ will be chosen in such a way that for every character χ of order p^k

$$\frac{\tau(\chi)^i}{\tau(\chi^i)} = \prod_{\pi \in \mathcal{J}} J(\chi^a, \chi^b)^{e_{\pi,p^k,i}};$$

here $J(\chi^a,\chi^b)^{z_j\sigma_j} = \sigma_j J(\chi^a,\chi^b)^{z_j} = J(\chi^{aj},\chi^{bj})^{z_j}$. Note that $\sigma_1 = 1$ and $\sigma_j\sigma_l = \sigma_{jl \bmod p^k}$. It turns out that for this purpose it suffices to consider only Jacobi sums $J(\chi^a,\chi^b)$ with $a+b$ prime.

Since the calculation of Jacobi sums (see (2.5)) is cumbersome for large conductor and the memory needed to store them expands fast for growing conductor, an effort is made here to find both a small set $\mathcal{J}$ and for given p^k a small subset of $\mathcal{J}$ such that for every π outside this subset $e_{\pi,p^k,i} = 0$ for every i. For more information, see II.(8.7) – (8.13) and V.(2.1).

(a) Let $\Pi = \max\{\pi : \pi \text{ prime, there exists a prime power } p^k \mid t_0 \text{ such that } p^k > \pi\}$. Put $\mathcal{J} = \emptyset$, put $\rho_{p^k,i} = \rho'_{p^k,i} = 0$ and put $\alpha_{p^k,i,a} = 0$ for every $p^k \mid t_0$, every $0 \le i \le p^k$, and $0 \le a \le \lfloor \frac{\Pi}{2} \rfloor$. Perform step (a1) successively for every prime power p^k dividing t_0.

The primes π in $\mathcal{J}$ will be at most equal to Π. While constructing the set $\mathcal{J}$ below, one will sometimes be forced to add primes π to $\mathcal{J}$, while some choice in splitting π into $\pi = a+b$ remains; the set $\mathcal{A}_\pi$ will for $\pi \in \mathcal{J}$ consist of the "breaking points" that are still allowed, i.e. those values for $0 < a \le \lfloor \frac{\pi}{2} \rfloor$ that one is free to choose from in (a,b). In the end choices are made such that $\#\mathcal{A}_\pi = 1$.

The exponents will be built up recursively, using the values for ρ and ρ', indicating which prime from $\mathcal{J}$ is used in the final step of the construction of $\tau(\chi)^i/\tau(\chi^i)$, and the values for α, indicating the operation that we have to perform on it. So far ρ and α were initialized only.

(a1) For $i = 2,\dots,p^k-1$ with $p \nmid i$ in succession, do the following. Put $\kappa_{p^k,i} = 1$. If there exist primes π dividing i such that $\pi \in \mathcal{J}$ or $\pi < i$, let π be minimal among these; put $\rho_{p^k,i} = \pi$ and put $\alpha_{p^k,i,0} = \sigma_{i/\pi}$.

We are trying to find an expression for $\tau(\chi)^{i-\sigma_i} = \tau(\chi)^i/\tau(\chi^i)$ that holds for any character of the present order p^k ; generating the proper denominator is the main problem. In the present case we can use the identity: $\sigma_{i/\pi}J(\chi^a,\chi^b) = \tau(\chi^{ai/\pi})\tau(\chi^{bi/\pi})/\tau(\chi^i)$, if $a+b=\pi$, which is particularly useful because both $ai/\pi < i$ and $bi/\pi < i$; this means that the numerator can be cancelled using previous instances of this step yielding $\tau(\chi^{ai/\pi})$ and $\tau(\chi^{bi/\pi})$ in the denominator. Therefore we let ρ point to the prime π and α to $\sigma_{i/\pi}$; since the above relation holds independently of the value of a, we use $\alpha_{p^k,i,0}$. The use of κ will become clear in (c).

If there exist $\pi \in \mathcal{J}$ such that for every $a \in A_\pi$ there exists j with $1 \le j \le a$ having the property that either

(*) $$a \mid j\cdot p^k - i \quad \text{and} \quad 0 < d < i, \quad \text{for some} \quad d \equiv (\pi-a)\left(\frac{j\cdot p^k - i}{a}\right) \bmod p^k$$

or

(**) $$\pi - a \mid j\cdot p^k - i \quad \text{and} \quad 0 < d < i, \quad \text{for some} \quad d \equiv a\left(\frac{j\cdot p^k - i}{\pi - a}\right) \bmod p^k,$$

then let π be the smallest of these; for this π put $\rho'_{p^k,i} = \pi$, choose j such that (*) or (**) holds and for every $a \in \mathcal{A}_\pi$ replace $\alpha_{p^k,i,a}$ by $\sigma_{(j\cdot p^k-i)/a}$ if (*) holds and else by $\sigma_{(j\cdot p^k-i)/(\pi-a)}$.

In this case we can utilize the rule $J(\chi^a, \chi^{\pi-a})^\sigma = \tau(\chi^c)\tau(\chi^d)/\tau(\chi^i)$ for certain $c, d < i$ and the indicated value for σ, as was pointed out in II.(8.13). For step (c) we have to distinguish between the use of II.(8.13) and the rule applied in the previous step, and for that purpose we introduce ρ'. Since for the chosen value of π the "breaking point" has not been established yet and σ depends on it, we have to record α now for every value for a that is still available.

If there exist $\pi \in \mathcal{J}$ with the property that there exists at least one $a \in \mathcal{A}_\pi$ for which there is $1 \leq j \leq a$ such that either $(*)$ or $(**)$ above holds, let π be the smallest of these; if $\rho_{p^k,i} = \rho'_{p^k,i} = 0$ remove all a from $\mathcal{A}_\pi$ for which neither $(*)$ nor $(**)$ can be met for any j. Also in this case, if either $\rho_{p^k,i} = \rho'_{p^k,i} = 0$ or $\pi < \rho'_{p^k,i}$, replace $\rho'_{p^k,i}$ by π, choose j such that $(*)$ or $(**)$ holds, and replace $\alpha_{p^k,i,a}$ for $a \in \mathcal{A}_\pi$ by $\sigma_{(j\cdot p^k-i)/a}$ if $(*)$ holds and by $\sigma_{(j\cdot p^k-i)/(\pi-a)}$ if $(**)$ holds.

Here the same rule is used as in the previous case, but now it does not work for every a any more. To use the rule we thus have to make restrictions concerning the "breaking points" a; we are only willing to do so in case we found nothing before (for the present i) or in case we can use a smaller π than before. The latter is often advantageous, since smaller π are more likely to be needed later on anyhow.

If now still $\rho_{p^k,i} = \rho'_{p^k,i} = 0$ (in which case i must be prime), put i in $\mathcal{J}$, put $1, \ldots, \lfloor \frac{i}{2} \rfloor$ in $\mathcal{A}_\pi$, put $\rho_{p^k,i} = \pi$ and put $\alpha_{p^k,i,a} = 1$ for every $a \in \mathcal{A}_\pi$.

If we have not yet been able to succeed for the current i so far, we will have to introduce a new prime. That is done here; every "breaking point" is allowed.

(b) For every π with $\#\mathcal{A}_\pi > 1$, let a be the smallest of the elements of $\mathcal{A}_\pi$, and replace $\mathcal{A}_\pi$ by $\{a\}$.

For every prime π for which there is a choice left, we choose the "breaking point" a in $\pi = a + b$.

(c) Put $e_{\pi,p^k,1} = 0$ for every $\pi \in \mathcal{J}$ and every $p^k \mid t_0$. For every prime power $p^k \mid t_0$ do the following for $i = 2, \ldots, p^k - 1$ (with $p \nmid i$) in succession to find the exponents $e_{\pi,p^k,i}$ for $i < p^k$. Put $e_{\pi,p^k,i} = 0$ for every $\pi \in \mathcal{J}$. If $\rho_{p^k,i} \neq 0$ and either $\rho_{p^k,i} \leq \rho'_{p^k,i}$ or $\rho'_{p^k,i} = 0$, then perform step (c1); in all other cases replace $\alpha_{p^k,i,0}$ by $\alpha_{p^k,i,a}$ with $a \in \mathcal{A}_\pi$ and perform step (c2).

Using the values for ρ and α, we will now assemble the exponents $e_{\pi,p^k,i}$ recursively.

(c1) If $\rho = \rho_{p^k,i} \in \mathcal{J}$, let a, b be such that $a + b = \rho$ and $a \in \mathcal{A}_\rho$. In this case put $e_{\rho,p^k,\rho} = \sigma_{i/\rho}$ and next replace $e_{\pi,p^k,i}$ by $e_{\pi,p^k,i} + \sigma_{i/\rho} \cdot (e_{\pi,p^k,a} + e_{\pi,p^k,b}) + \rho \cdot e_{\pi,p^k,i/\rho}$ for every $\pi \in \mathcal{J}$. If $\rho = \rho_{p^k,i} \notin \mathcal{J}$, put $e_{\pi,p^k,i} = \sigma_{i/\rho} \cdot e_{\pi,p^k,\rho} + \rho \cdot e_{\pi,p^k,i/\rho}$ for every $\pi \in \mathcal{J}$.

(c2) Let $\rho' = \rho'_{p^k,i}$. Replace $e_{\rho',p^k,i}$ by $\alpha_{p^k,i} \cdot e_{\rho',p^k,\rho'}$. If $\alpha_{p^k,i} = \sigma_{(j\cdot p^k-i)/a}$ (that is, if $(*)$ held above) put $d = (\pi - a) \cdot (j \cdot p^k - i)/a \bmod p^k$ and else (if $(**)$ was true above) put $d = a \cdot (j \cdot p^k - i)/(\pi - a) \bmod p^k$. Put $c = i - d$ and replace $e_{\pi,p^k,i}$

by $e_{\pi,p^k,i} + e_{\pi,p^k,c} + e_{\pi,p^k,d}$ for every $\pi \in \mathcal{J}$. Finally, if $p = 2$, replace $\kappa_{p^k,i}$ by $\chi^d(-1)\kappa_{p^k,i}\kappa_{p^k,c}\kappa_{p^k,d}$.

We use $\kappa_{p^k,i}$, for $i < p^k$, to keep track of sign changes, due to an appeal to II.(8.11). A factor -1 can only arise if $p = 2$.

(d) For every prime power $p^k \mid t_0$ put $e_{\pi,p^k,p^k} = e_{\pi,p^k,p^k-1}$ for every $\pi \in \mathcal{J}$. Put $\kappa_{p^k,p^k} = \chi(-1)q\kappa_{p^k,p^k-1}$.

The relation $\tau(\chi)^{p^k} = \chi(-1)q \cdot \prod_{\pi\in\mathcal{J}} J^{e_{\pi,p^k,p^k-1}}$ of II.(8.8) is used. In this particular case we multiply κ by the non-Gauss-sum factor $\chi(-1)q$.

(e) Let $\mathcal{J}_{p^k} \subset \mathcal{J}$ consist of those $\pi \in \mathcal{J}$ such that $e_{\pi,p^k,i} \neq 0$ for some $i \leq p^k$.

The set $\mathcal{J}_{p^k}$ indicates which Jacobi sums we need for any character of order p^k to express $\tau(\chi)^{i-\sigma_i}$ for every $i \leq p^k$.

(f) Tabulate $\mathcal{J}$, and also for every prime $\pi \in \mathcal{J}$ the pair (a,b) with $a \in \mathcal{A}_\pi$ and $a + b = \pi$, the subsets $\mathcal{J}_{p^k}$ for every $p^k \mid t_0$, and for every $\pi \in \mathcal{J}$ the exponents $e_{\pi,p^k,i}$ for every $p^k \mid t_0$ and every $0 < i \leq p^k$; also tabulate $\kappa_{p^k,i}$, for all pairs p^k, i.

(2.5) Calculation of Jacobi sums.

Create a direct access file. Perform steps (a) through (d) for all odd $q \in \mathcal{Q}_0$.

A direct access file is a file which can be read and written in random access order, i.e., without sequentially reading the complete file to read one entry from or write one entry in the file (as in a sequential file). Each entry in the direct access file has the same fixed size. Using a direct access file is beneficial here, since only a few Jacobi sums will be needed for the primality test of a particular n.

For every pair p^k, q consisting of a prime $q \in \mathcal{Q}_0$ (as determined in (2.2)(e)) and a prime power p^k such that $p^k \parallel q-1$, in this step the Jacobi sums $J(\chi^a,\chi^b)$ with $a+b=\pi$ as in (2.4)(f) are computed, for $\pi \in \mathcal{J}_{p^k}$, for a character of conductor q and order p^k; notice that $p \in \mathcal{P}_0$ (cf. (2.2)(b)). This is done by using the definition, cf. II.(8.1) :

$$J(\chi^a,\chi^b) = \sum_{x=0}^{q-1} \chi^a(x)\chi^b(1-x) \quad \text{that can be expressed as} \quad \sum_{i=0}^{\phi(p^k)-1} c_{\pi,q,p,i} \cdot \zeta_{p^k}^i;$$

so we represent the Jacobi sum $J(\chi^a,\chi^b)$ by the vector $(c_{\pi,q,p,i})_{0\leq i<\phi(p^k)}$.

No use is made of direct formulae for $J(\chi^a,\chi^b)$ like for instance if $\mathrm{ord}(\chi) = 2$, because the calculations of all Jacobi sums for a fixed conductor q are performed in parallel. Omitting the calculation of a single Jacobi sum from this parallel computation hardly influences the total computing time. For more information, see II.(8.3) and V.(3.6).

(a) Find a primitive root g modulo q.

Among others, we will use this primitive root to make a choice for a character of order p^k and conductor q; we will choose $\chi(g) = \zeta_{p^k}$, where ζ_{p^k} is a primitive p^k-th root of unity.

(b) For all prime powers $p^k \parallel q-1$, put $c_{\pi,q,p,i} = 0$ for $0 \leq i < p^k$ and for every $\pi \in \mathcal{J}_{p^k}$, as determined in step (2.4).

Let M_0 be the maximal number of integers less than q that can be stored in the available memory. Let $M = \min(M_0, q-2)$, and let $c(q)$ be the number of distinct prime divisors of $q-1$. If $M \geq \min\{q-2, 2q/((c(q)+2)\log_2 q)\}$, then perform step (b1), otherwise perform step (b2).

In (b1) and (b2) two methods are given for computing J. Comparing the number of operations required for each of these (see below), one arrives at the crossover point mentioned; in practice however, the crossover point between (b1) and (b2) will depend on the implementation and is therefore best determined empirically.

(b1) Perform step (b1a) for $0 \leq m \leq \lfloor (q-3)/M \rfloor$.

In the first method (for each of $\lfloor (q-3)/M \rfloor + 1$ blocks) a table is made of pairs $(x, f(x))$ such that $1 - g^x = g^{f(x)}$ for a block of length M out of the $q-2$ different powers of g modulo q. This requires about q^2/M multiplications modulo q.

(b1a) Put $F_i = 0$ for $i = m \cdot M + 2, m \cdot M + 3, \ldots, (m+1) \cdot M + 1$. For $x = 1, 2, \ldots, q-2$ in succession, do the following.

Compute $\gamma \equiv g^x \bmod q$ and $\delta \equiv 1 - g^x \bmod q$ with $0 \leq \gamma, \delta \leq q-1$. If $m \cdot M + 2 \leq \gamma \leq (m+1) \cdot M + 1$ and $F_\gamma \neq 0$, then increase $c_{\pi,q,p,ax+bF_\gamma \bmod p^k}$ by 1, for all prime powers $p^k \parallel q-1$ and all $\pi \in \mathcal{J}_{p^k}$; here $a + b = \pi$. Similarly, if $m \cdot M + 2 \leq \gamma \leq (m+1) \cdot M + 1$ and $F_\gamma = 0$ put $F_\gamma = x$. If $m \cdot M + 2 \leq \delta \leq (m+1) \cdot M + 1$ and $F_\delta \neq 0$, then increase $c_{\pi,q,p,aF_\delta+bx \bmod p^k}$ by 1, for all prime powers $p^k \parallel q-1$ and all $\pi \in \mathcal{J}_{p^k}$; here $a+b=\pi$. If $m \cdot M + 2 \leq \delta \leq (m+1) \cdot M + 1$ and $F_\delta = 0$ put $F_\delta = x$.

For a block of length at most M we find all pairs $(x, f(x))$ such that $g^x \bmod q$ or $g^{f(x)} \bmod q$ is in the current block and $g^x + g^{f(x)} \equiv 1 \bmod q$. Each pair $(x, f(x))$ gives a contribution $\zeta^{ax}\zeta^{bf(x)}$ to the Jacobi sum $J(\chi^a, \chi^b)$ for the choice of the character as in (a). Finding all pairs $(x, f(x))$ is done as follows. If $\gamma \equiv g^x \bmod q$ is in the block, we store x in F_γ, unless $F_\gamma \neq 0$; that can only happen if $\gamma = 1 - g^y$, for some y that we have dealt with before, in which case $y = F_\gamma$ and $f(y) = x$. Similarly, if $\delta \equiv 1 - g^x \bmod q$ is in the block, we store x in F_δ, unless $F_\delta \neq 0$; that can only happen if $\delta = g^y$, for some y that we have dealt with before, in which case $y = F_\delta$ and $f(y) = x$.

(b2) For all prime powers $p^k \parallel q-1$ put $g_{\bar{p},i} = g^{i(q-1)/p^k} \bmod q$ for $i = 0, 1, \ldots, p^k - 1$ in succession. Perform step (b2a) for $x = 1, 2, \ldots, q-2$ in succession.

For the second method one first computes the $c(q)$ powers $g^{(q-1)/p^k}$; this can be done in roughly $((c(q)/2)+1)\log_2 q$ multiplications modulo q, if one does $\log_2 q$ squarings of g followed by assembling $g^{(q-1)/p^k}$ (requiring approximately $(\log_2 q)/2$ multiplications on the average) for each of the p^k. The same is later done for each of $q-2$ different values for $\bar{g}_x$. That adds up to $(q-1)((c(q)/2)+1)\log_2 q$ multiplications; at most q more are needed to find the p^k different powers $(g^{(q-1)/p^k})^i$ for all p^k.

(b2a) Put $\bar{g}_x = 1 - g^x \bmod q$. For all prime powers $p^k \parallel q-1$ put $\bar{g}_{p,x} = \bar{g}_x^{(q-1)/p^k} \bmod q$. Next for all prime powers $p^k \parallel q-1$, find i such that $g_{p,i} = \bar{g}_{p,x}$ and increase

$c_{\pi,q,p,ax+bi \bmod p^k}$, by 1 for every $\pi \in \mathcal{J}_{p^k}$, where $a + b = \pi$. Finding i can for instance be done using hashing.

A character χ of order p^k is chosen as in (a). The $g_{p,i}$ are the powers of $g_{p,1}$, a generator for the p-Sylow subgroup (of order p^k) inside $\left(\mathbf{Z}/q\mathbf{Z}\right)^*$. To evaluate $\chi(1-g^x)$, one raises $1-g^x$ in the power $\frac{q-1}{p^k}$ (to kill the non-p part) and finds from the list the $g_{p,i}$ to which the result is equal. The contribution to the Jacobi sum $J(\chi^a, \chi^b)$ is $\zeta^{ax}\zeta^{bi}$.

(c) For every prime power $p^k \parallel q-1$, every i such that $\phi(p^k) \leq i < p^k$ and every $1 \leq j < p$, decrease $c_{\pi,q,p,i-jp^{k-1}}$ by $c_{\pi,q,p,i}$ for all $\pi \in \mathcal{J}_{p^k}$.

Here we transform to a basis for $\mathbf{Z}[\zeta_{p^k}]$ by using the relation $\zeta^0+\zeta^{p^{k-1}}+\ldots+\zeta^{(p-1)p^{k-1}} = 0$.

(d) Add the vector $(c_{\pi,q,p,i})_{0 \leq i < \phi(p^k)}$ representing the Jacobi sum corresponding to π and (a,b) to the content of the direct access file for every $\pi \in \mathcal{J}_{p^k}$ and every $p^k \parallel q-1$.

3. INITIALIZATIONS.

Let $n > 1$ be an odd integer to be tested for primality. Suppose that files and tables are prepared according to Section 2.

(3.1) Initialization.

Put $G = \prod_{p \in \mathcal{P}_0 \cup \mathcal{Q}_0} p \bmod n$ (cf. (2.2)(b), (e)). If $G = 0$ the complete factorization of n can easily be derived, and the primality test is terminated.

> As part of the primality proof for n, one has to verify that several integers are relatively prime to n. Instead of calculating the various gcd's, one calculates the product G modulo n of these integers, and one checks if the product equals 0 mod n. If $G \equiv 0 \bmod n$, a factor of n is easily derived, by taking the gcd of n and the last integer G has been multiplied with; the algorithm will be aborted then. If $G \equiv 0 \bmod n$ never occurs, $\gcd(G, n)$ will be computed once at the end of the algorithm (in step (6)).

(3.2) Trial division.

Put Ω equal to the empty set and perform steps (a) and (b).

> In this step the numbers n, $n-1$, and $n+1$ are checked on divisibility by all small primes up to some bound B. In this way composite n with a small factor are easily detected, while the algorithm determines at the same time the small prime factors of $n^2 - 1$, which may be used in the Lucas-Lehmer part of the algorithm (cf. (3.5), (4.5), (5.3)). This step makes use of the table of prime numbers created in step (2.1). If no divisors of n are found, the sets $\mathcal{F}_1$ and $\mathcal{F}_2$ will contain all prime divisors up to B of $n-1$ and $n+1$ respectively; v_1 and v_2 will be equal to the product of all prime divisors up to B of $n-1$ and $n+1$ respectively, with their multiplicities. The value of B is highly dependent of $\log_2 n$, and should be determined empirically (cf. Chapter VI). Finally, Ω will contain all positive integers ω for which a factor of $n^\omega - 1$ is known, and w will be equal to $\mathrm{lcm}\{\omega : \omega \in \Omega\}$. Both Ω and w may be changed in steps (3.5) and (4.5).

(a) Set r_1 and r_2 equal to the largest odd factors of $n-1$ and $n+1$, respectively, and set $\mathcal{F}_1$ and $\mathcal{F}_2$ equal to $\{2\}$. Furthermore put $v_1 = 2^{o_2(n-1)}$ and $v_2 = 2^{o_2(n+1)}$. Select a trial division bound $1 \leq B \leq \min(B_0, \sqrt{n})$ (cf. (2.1)). Perform step (a1) for all odd primes $p \leq B$, where the primes p are generated using the file created in (2.1).

(a1) Let n_p be the (smallest positive) remainder of the division of $n+1$ by p. If $n_p = 1$, then p is a divisor of n, and the primality test is terminated because n is composite. Otherwise, if $n_p = 0$, replace $\mathcal{F}_2$ by $\mathcal{F}_2 \cup \{p\}$, replace v_2 by $v_2 \cdot p^{o_p(r_2)}$ and r_2 by $r_2 / p^{o_p(r_2)}$. Finally, if $n_p = 2$, replace $\mathcal{F}_1$ by $\mathcal{F}_1 \cup \{p\}$, replace v_1 by $v_1 \cdot p^{o_p(r_1)}$ and r_1 by $r_1 / p^{o_p(r_1)}$.

(b) If $B = \lfloor \sqrt{n} \rfloor$, then n is proved to be prime and the test is terminated. Otherwise, put $v = v_1 \cdot v_2$, put $w = 2$ and put $\Omega = \{1, 2\}$.

(3.3) Compositeness test.

Let $n-1 = r \cdot 2^k$ with r odd and $k \geq 1$. Select a small positive integer C, and perform step (a) at most C times.

> In this step a compositeness test is performed, see II.1. If some witness a to the compositeness of n is found, the number n is proved to be composite. Recall from II.1.3 that if n is a composite number, the probability that a random a in $\{2, 3, \ldots, n-1\}$ is a witness is at least 3/4. The number of attempts C to find a witness, could for instance be taken equal to 4.

(a) Randomly select an integer a from $\{2, 3, \ldots, n-2\}$, and compute $a_r = a^r \bmod n$. If $a_r \not\equiv \pm 1 \bmod n$, then check by repeated squaring that there is an i in $\{1, 2, \ldots, k-1\}$ such that $a_r^{2^i} \equiv -1 \bmod n$; if such an i does not exist the primality test is terminated because n is composite.

(3.4) Preliminary calculations.

Perform steps (a), (b), and (c) for all primes p dividing t_0 (cf. (2.2)(a)).

> In this step $u_{p,k} = \mathrm{ord}(n \bmod p^k)$ and $o^*_{p,k} = o_p(n^{u_{p,k}} - 1)$ for all prime powers p^k dividing t_0 are calculated. The $u_{p,k}$ will be the degrees of certain rings in which the calculations of Section 5 will be done. The number $o^*_{p,k}$ of factors p in $n^{u_{p,k}} - 1$ may be used in the Lucas-Lehmer step of the algorithm.

(a) For $k = 1, \ldots, k_p$ (cf. (2.2)(c)), perform step (a1).

(a1) If $p^k = 2$, then put $i = 1$. Otherwise, if $p^k \neq 2$, let $n_{p,k} = n \bmod p^k$, and find by repeated multiplication the minimal i in $\{1, 2, \ldots, \lambda(p^k)/2\}$ such that $n^i_{p,k} \equiv \pm 1 \bmod p^k$ (cf. (2.2)(d)). If $n^i_{p,k} \equiv 1 \bmod p^k$, then put $u_{p,k} = i$. Otherwise, if $n^i_{p,k} \equiv -1 \bmod p^k$, then put $u_{p,k} = 2i$.

(b) If p is odd, or $n \equiv 1 \bmod 4$, put $\bar{k} = 1$ and $\bar{u} = u_{p,\bar{k}} = u_{p,1}$. If $p = 2$ and $n \equiv 3 \bmod 4$, put $\bar{k} = 2$ and $\bar{u} = u_{p,\bar{k}} = 2$. Put $c = 0$ and calculate $o^*_{p,\bar{k}}$ by performing step (b1) only once and step (b2) as long as $c = 0$.

> In this step $o^*_{p,k}$, the number of factors p in $n^{u_{p,k}} - 1$, is determined. Since $n^{u_{p,k}} \equiv 1 \bmod p^k$, it follows that the number of factors p in $n^{u_{p,k}} - 1$ is at least k. Also, $o^*_{p,k} = \max\{o^*_{p,\bar{k}}, k\}$ for $\bar{k} \leq k \leq k_p$, and $o^*_{p,k} = 1$ for $1 \leq k < \bar{k}$, so $o^*_{p,k}$ can be calculated from $o^*_{p,\bar{k}}$. To determine $o^*_{p,\bar{k}}$ one first divides $n^{u_{p,\bar{k}}} - 1 = n^{\bar{u}} - 1$ by $p^{\bar{k}}$. This is done by first writing $n^{\bar{u}} - 1$ in base n, i.e., $n^{\bar{u}} - 1 = \sum_{i=0}^{\bar{u}-1} b_i \cdot n^i$, and by next sequentially dividing all coefficients b_i by $p^{\bar{k}}$ while keeping track of a carry c. The b_i now satisfy $\sum_{i=0}^{\bar{u}-1} b_i \cdot n^i = (n^{\bar{u}} - 1)/p^{\bar{k}}$.

(b1) First put $o^*_{p,\bar{k}} = \bar{k}$. Next put $b_i = n - 1$ for $i = 0, 1, \ldots, \bar{u} - 1$. After this, for $i = \bar{u} - 1, \ldots, 0$ in succession first put $c' = (c \cdot n + b_i) \bmod p^{\bar{k}}$, next put $b_i = \lfloor (c \cdot n + b_i)/p^{\bar{k}} \rfloor$, and finally replace c by c'.

(b2) For $i = \bar{u}-1, \ldots, 0$ in succession first put $c' = (c \cdot n + b_i) \bmod p$ and $b_i = \lfloor (c \cdot n + b_i)/p \rfloor$ and next replace c by c'. If $c = 0$ replace $o^*_{p,\bar{k}}$ by $o^*_{p,\bar{k}} + 1$.

Using the same method as in step (b1) the number $\sum_{i=0}^{\bar{u}-1} b_i \cdot n^i$ is divided by p as long as c, the remainder of the division, is equal to zero. The resulting $o^*_{p,\bar{k}}$ is the number of factors p of $n^{\bar{u}} - 1$.

(c) Put $o^*_{p,k} = 1$ for $1 \leq k < \bar{k}$, and put $o^*_{p,k} = \max\{o^*_{p,\bar{k}}, k\}$ for $\bar{k} \leq k \leq k_p$. Put $u_0 = \operatorname{lcm}\{u_{p,k} : p^k \parallel t_0\} = \operatorname{lcm}\{u_{p,k_p}\}$ (cf. (2.2)(c)).

(3.5) Utilization of known factors.

Perform steps (a) and (b) for every known prime factor f of $n^\omega - 1$ with $\omega \in \mathbf{Z}_{>0}$ and either $\omega \notin \Omega$ or $f \nmid v_\omega$.

In the algorithm any prime factor f of $n^\omega - 1$ that is known is useful, since such a factor may be used in the Lucas-Lehmer part of the algorithm (cf. (5.3)). Therefore any known factor f will be stored, together with its multiplicity and ω. In the optimization step (Section 4) it will be decided which of these factors will be used. The set Ω will contain all values ω for which a factor is known (cf. (3.2)), w will be equal to $\operatorname{lcm}\{\omega : \omega \in \Omega\}$ and v_ω denotes the completely factored part of $\Phi_\omega(n)$, for $\omega \geq 1$, with $\omega \in \Omega$, where Φ_ω denotes the ω-th cyclotomic polynomial; that is, v_ω consists of the known primitive prime factors of $n^\omega - 1$: those that are not already in $n^i - 1$ for $i \mid \omega$ and $i \neq \omega$. Furthermore, v will be the product of v_ω for $\omega \in \Omega$ and finally, $\mathcal{F}_\omega$ denotes the set of prime factors found in $n^\omega - 1$. Note that in (3.2)(a1) we have found values for v_1 and v_2. All values and sets may be changed in step (4.5).

(a) Let $k = o_f(n^\omega - 1)$, and $c = o_f(\omega)$. Put $\bar{k} = 1$ if f is odd or $n \equiv 1 \bmod 4$, and $\bar{k} = 2$ if f is 2 and $n \equiv 3 \bmod 4$. Put $o^*_{f,\bar{k}} = k - c + \bar{k} - 1$, $o^*_{f,i} = 1$ for $1 \leq i < \bar{k}$ and $o^*_{f,i} = \max\{o^*_{f,\bar{k}}, i\}$ if $\bar{k} \leq i \leq k$. Calculate $\bar{\omega} = \min\{i : i \mid \omega,\ f \mid (n^i - 1)\}$.

Calculating $o^*_{f,i}$ from $o^*_{p,\bar{k}}$ as well as calculating $\bar{\omega}$ is done in the same way as in (3.4). When calculating the values $v_{\omega'}$, one only has to update those values $v_{\omega'}$, with $\omega' = \bar{\omega} \cdot f^i$, with $i \geq 0$, since the number of factors f in $n^{\omega'} - 1$ only changes for these values of ω'.

(b) For $i = 0, \ldots, c$ in succession, put $\omega' = \bar{\omega} \cdot f^i$ and perform step (b1) if $\omega' \notin \Omega$, and step (b2) if $\omega' \in \Omega$.

Although $f^{k-c+i} \mid n^{\omega'} - 1$, one should not multiply v by f^{k-c+i} because then factors f would be counted more than once in v. The number of factors f in $\Phi_{\bar{\omega} \cdot f^{j_1}}(n)$, not in $\Phi_{\bar{\omega} \cdot f^{j_2}}(n)$ for $0 \leq j_2 \leq j_1$ is equal to $j_1 - j_2$. Only these factors f may be used by the Lucas-Lehmer part of the algorithm. Therefore one has to keep track of factors f which are due to $\Phi_{\omega'}(n)$ itself, and factors f which are due to $\Phi_{\omega''}(n)$ for some divisor ω'' of ω'.

(b1) Replace Ω by $\Omega \cup \{\omega'\}$, put $\mathcal{F}_{\omega'}$ equal to $\{f\}$, replace w by $\operatorname{lcm}(w, \omega')$, put $v_{\omega'} = f^j$, replace v by $v \cdot f^j$, where $j = \bar{k}$ if $i = 0$ and $j = 1$ otherwise.

If $\omega' \notin \Omega$ the set $\mathcal{F}_{\omega'}$ and the $v_{\omega'}$ have to be initialized. Furthermore the set Ω, and the integers w and v are updated.

(b2) If $f \nmid v_{\omega'}$ then replace $\mathcal{F}_{\omega'}$ by $\mathcal{F}_{\omega'} \cup \{f\}$, replace $v_{\omega'}$ by $v_{\omega'} \cdot f^j$, replace v by $v \cdot f^j$, where $j = \bar{k}$ if $i = 0$ and $j = 1$ otherwise.

4. OPTIMIZATION.

In this section we give a description of the optimization step. It should be noted that steps (4.2)–(4.6) are not performed sequentially, but are used while executing step (4.1).

(4.1) Main optimization step.

Select a value for $\bar{T}$ and perform steps (a) through (d).

The value $\bar{T}$ is the time one is prepared to spend on proving the primality of the number n.

In this step one tries to choose the values of $\mathcal{P}$, $\mathcal{Q}$, s, t, u, v, and w in such a way that the time needed to perform the algorithm is (close to) minimal; if this time exceeds $\bar{T}$, the algorithm will be aborted, since one cannot prove the primality within a reasonable amount of time. One should keep in mind that, roughly speaking, $\mathrm{lcm}(s,v)$ should exceed $\sqrt{n}$; here s forms the Jacobi sum contribution and v the Lucas-Lehmer contribution.

(a) Put $B_2 = 1$. Set $\mathcal{L}^+$ and $\mathcal{L}^-$ equal to the empty set. For all primes l dividing $\mathrm{lcm}(u_0, w)$ put $a_l = 0$, and define $\hat{e}_l$ by $\mathrm{lcm}(u_0, w) = \prod_{l \text{ prime}} l^{\hat{e}_l}$, (cf. (3.4) and (3.5) for the definitions of u_0 and w). Perform step (4.2) for every prime power $l^{\hat{e}_l}$. Perform steps (a1) and (a2).

The variable B_2 indicates up to which bound factors have been searched for in step (4.6). It will be changed in step (4.6).

The sets $\mathcal{L}^+$ and $\mathcal{L}^-$ will be used to indicate which extensions have been found. If at least one extension of degree l^e that can be used in the algorithm for testing n has been found, then l^e will be put in $\mathcal{L}^+$. If no suitable extension of degree l^e has been found, then l^e will be put in $\mathcal{L}^-$.

The variable a_l is used to indicate whether or not step (4.2) has been performed for any power of the prime l. If $a_l \neq 0$ then certain conditions in step (4.2) have been checked for some power of l. For more information, we refer to step (4.2).

(a1) For all $t \mid t_0$ put $\mathcal{Q}_t$ equal to $\{q \in \mathcal{Q}_0 : q-1 \mid t\}$ and put $\mathcal{T}_t$ equal to $\{(q,p,h) : h = o_p(u_{p,o_p(q-1)})$ with $q \in \mathcal{Q}_t$ and prime $p \mid q-1\}$.

The set $\mathcal{Q}_t$ contains the prime factors of $e'_{i(t)}$, (cf. (2.2)(g)). We get a contribution q to s, if we perform Jacobi sum tests for all triples (q,p,h) in $\mathcal{T}_t$, where q is the conductor and $p^{o_p(q-1)}$ is the order of the character involved. This test can be performed in an extension of degree $u_{p,o_p(q-1)} = p^h \cdot u_{p,1}$. These sets are calculated in advance, since in step (4.3) one needs to compare the costs of using t and (subsets of) $\mathcal{Q}_t$ and $\mathcal{T}_t$ quickly. For all prime powers p^k dividing t_0 the $u_{p,k}$ have been defined as $\mathrm{ord}(n \bmod p^k)$ (cf. (3.4)).

(a2) For all $q \in \mathcal{Q}_0$ calculate $c_1(q) = (\sum_{p \mid q-1} u^2_{p,o_p(q-1)})/\log_2(q)$. Order the elements $q \in \mathcal{Q}_0$ in such a way that $c_1(q)$ is decreasing (cf. III.(3.17)).

Given a value for t, a quick and reliable method is needed to delete the most expensive q's from $\mathcal{Q}_t$ (and all corresponding triples (q,p,h) from $\mathcal{T}_t$). The ordering of q's according to c_1 assumes that the costs for using q's are independent; in fact this is not true, since Jacobi sum tests for different conductor may be combined. Therefore c_1 is not guaranteed to give the optimal ordering, but experiments have shown that it hardly differs from this. This ordering is used in step (4.3), to reject most values for t as being too expensive. Possibly better, but more expensive ways of ordering are used in step (4.3).

(b) Let v and w be as found in (3.5). Find initial values $\mathcal{P}'$, $\mathcal{Q}'$, $\mathcal{T}'$, s', t', u', v', w', μ', and an initial estimate C' for the running time of the steps of the algorithm described in Sections 5 and 6, by performing step (4.3) with $C^* = -1$, $B^* = \lceil \sqrt{n}/v_1 \rceil$, $z^* = 0$, and $\mu^* = \frac{1}{2}$. Next put $\mathcal{P}' = \mathcal{P}^*$, $\mathcal{Q}' = \mathcal{Q}^*$, $\mathcal{T}' = \mathcal{T}^*$, $s' = s^*$, $t' = t^*$, $u' = u^*$, $v' = v_1$, $w' = 1$, and $\mu' = \frac{1}{2}$. Finally put $C' = C^*$.

In this step initial values for $\mathcal{P}'$, $\mathcal{Q}'$, $\mathcal{T}'$, s', t', u', v', w', and μ' are chosen in such a way, that it hardly takes any time to find them, and that the time to perform the steps of Sections 5 and 6 with these values is close to minimal. These values will be improved in the next steps of (4.1). Throughout the rest of step (4.1), the $\mathcal{P}'$, $\mathcal{Q}'$, $\mathcal{T}'$, s', t', u', v', w', and μ' will denote the values that give the minimal cost C' found so far.

Initially, the value C^* is negative to indicate that step (4.3) has not yet been performed.

The flag z^* is used to indicate which kind of optimization should be performed in (4.3). For further comments on this subject we refer to that step. The value μ' indicates which final trial division will be used in Section 6. During the final trial division it will be checked whether there exist divisors of n in the residue classes $r \equiv n^i \bmod \operatorname{lcm}(s', v')$ for $i = 1, \ldots, \operatorname{lcm}(t', w')$, where $\operatorname{lcm}(s', v') > n^{\mu'}$.

(c) Put $T = \min(\bar{T}, C')$ and put $P = T - T_0$, where T_0 is the minimal time needed to perform the rest of step (c). If $P > 0$, repeat for $\mu = \frac{1}{2}$ as well as for $\mu = \frac{1}{3}$ steps (c1) through (c6) until either $P < 0$ or $T < 0$ in step (c4); as soon as this happens, we jump to step (d). Otherwise, if $P \leq 0$, perform step (4.3) with $C^* = C'$, $B^* = \lceil n^{\mu}/v_1 \rceil$, $z^* = 2$, and $\mu^* = \mu$.

In this step a further effort is made to optimize C'. It may be that if more factors are found in step (4.3), the total running time (including the time needed to find these factors) is less than the minimal running time found up till now.

The time left to be spent is denoted by T; if T_0 exceeds T, then optimization only consists of performing step (4.3)(b) for the current value of t'. The value for T_0 should be determined empirically.

(c1) First put $\hat{x} = \lceil n^{\mu}/v_1 \rceil$, and perform step (4.3) with $C^* = C'$, $B^* = \hat{x}$, $z^* = 1$, and $\mu^* = \mu$. Next put $f'(\hat{x}) = C^*$, $f''(\hat{x}) = 0$, $f(\hat{x}) = f'(\hat{x}) + f''(\hat{x})$. If $f(\hat{x}) < C'$, then put $C' = f(\hat{x})$ and put $\mathcal{P}' = \mathcal{P}^*$, $\mathcal{Q}' = \mathcal{Q}^*$, $\mathcal{T}' = \mathcal{T}^*$, $s' = s^*$, $t' = t^*$, $u' = u^*$, $v' = v_1$, $w' = 1$, and $\mu' = \mu^*$.

In this step (and steps (c2) through (c6)) an attempt is made to find a value $\tilde{x}$, for which the sum $f(\tilde{x})$ of the cost $f'(\tilde{x})$ of performing a Jacobi sum test with $s \approx \tilde{x}$, and the cost $f''(\tilde{x})$ of performing a Lucas-Lehmer test with $v \approx n^{\mu}/\tilde{x}$, is minimal. This is done by varying the amount of time spent on looking for more factors for the Lucas-Lehmer part and estimating the consequences. As soon as $f(\tilde{x}) < C'$, it is expected to be profitable to search for more factors in order to change $\mathcal{P}'$, $\mathcal{Q}'$, $\mathcal{T}'$, s', t', u', v', w', μ', and C'.

(c2) Put $W^{\#} = \max\{\operatorname{ord}(n \bmod p^k) : p \text{ prime}, p^k \parallel \operatorname{lcm}(t', v')\}$. Let A_1 be an approximate solution for A in the equation

$$\frac{A}{\log A} = \frac{C'}{(\log_2 n + W^{\#})} + \frac{B_2}{\log B_2}.$$

Let $\check{x}$ be an approximate solution for x in the equation

$$x \cdot v' \cdot \left(\frac{A_1 + B_2}{2}\right)^{W^{\#}(\log\log A_1 - \log\log B_2)} = n^{\mu}.$$

Perform (4.3) with $C^* = C'$, $B^* = \check{x}$, $z^* = 0$, and $\mu^* = \mu$, and next put $f'(\check{x}) = C^*$.

If we spend time equal to C' in looking for additional Lucas-Lehmer factors, we will find all prime factors up to the expected bound A_1 of $n^i - 1$, where $1 \leq i \leq W^{\#}$. If all these factors are used in a Lucas-Lehmer test, then a Jacobi sum test with $s \approx \check{x}$ would be sufficient to complete the primality proof. The time necessary to perform the Jacobi sum test is about the value of f' calculated here. The $\check{x}$ serves as lower bound for values of x that will be taken into consideration.

(c3) Select a value for ϵ. The machine dependent functions c_d and c_n must be known (cf. (4.4)). Find an approximate minimum $\tilde{x}$ between $\check{x}$ and $\hat{x}$ to the function

$$f''(x) + \left(\frac{f'(\hat{x}) - f'(\check{x})}{\log_2 \hat{x} - \log_2 \check{x}}\right) \cdot (\log_2 x - \log_2 \check{x}) + f'(\check{x}).$$

The function $f''(x)$ is approximated by

$$C_3(w') + \sum_{\substack{\omega \leq W^{\#} \\ \omega \nmid \mathrm{lcm}(u^*, w')}} \omega^2 \log_2 n \cdot c_n(\log_2(n)) + \left(\frac{A_2}{\log A_2} - \frac{B_2}{\log B_2}\right) \cdot (W^{\#} + c_d(\log_2(n), \log_2(M))),$$

where A_2 is an approximate solution for A in

$$x \cdot v' \cdot \left(\frac{A + B_2}{2}\right)^{W^{\#}(\log\log A - \log\log B_2)} = n^{\mu},$$

M is the largest integer representable in single precision, and C_3 is a function defined in (4.4). Perform step (c3a) until $\check{x} = \hat{x}$, but at most 5 times. Put $\tilde{x} = \check{x}$.

We minimize the sum of the cost $f''(x)$ of searching for Lucas-Lehmer factors (which is decreasing with increasing x) and the cost $f'(x)$ of finishing the primality proof using a Jacobi sum test with $s \approx x$, under the assumption that the latter is approximately linear in $\log x$.

For fixed t, the cost $f'(x)$ of the Jacobi sum test grows almost linearly as a function of $\log s$; changing to another value of t however, introduces a discontinuity. As a result the function f' will be almost piecewise linear in $\log x$. Here the assumption is made that no discontinuities on the interval $(\check{x}, \hat{x})$ exist; in (c3a) this assumption is verified.

The approximation for $f''(x)$ consists of a contribution $C_3(w')$ for generating the proper roots of unity (as in (4.4)(c)) for the known Lucas-Lehmer factors, an approximation for the costs of generating roots of unity for the Lucas-Lehmer factors expected to be found, and finally the costs of searching for these factors up to the bound A_2.

Notice that in the approximation of $f''(x)$ above, the first two terms are independent of x, so minimization of f in fact only includes minimization of the sum of the third term of $f''(x)$ and the linear approximation of the costs of the Jacobi sum test.

The value ϵ is the maximal relative error that is tolerated in f'. A suitable value for

ϵ is for instance 0.5. The machine dependent constant c_0^* is given by the ratio of the costs of division and multiplication of single precision integers. Minimization is done using the methods described in [15].

(c3a) Calculate $f'(\tilde{x})$ by performing (4.3) with $C^* = C'$, $B^* = \tilde{x}$, $z^* = 0$, and $\mu^* = \mu$, and by putting $f'(\tilde{x}) = C^*$. If

$$\left| \frac{f'(\tilde{x}) - f'(\check{x})}{\log_2(\tilde{x}) - \log_2(\check{x})} - \frac{f'(\tilde{x}) - f'(\hat{x})}{\log_2(\tilde{x}) - \log_2(\hat{x})} \right| < \epsilon$$

then put $\check{x} = \tilde{x}$ and $\hat{x} = \tilde{x}$. Otherwise, if

$$\frac{f'(\tilde{x}) - f'(\check{x})}{\log_2(\tilde{x}) - \log_2(\check{x})} > \frac{f'(\tilde{x}) - f'(\hat{x})}{\log_2(\tilde{x}) - \log_2(\hat{x})}$$

then put $\hat{x} = \tilde{x}$ and $\tilde{x} = (\hat{x} + \check{x})/2$. Otherwise, if

$$\frac{f'(\tilde{x}) - f'(\check{x})}{\log_2(\tilde{x}) - \log_2(\check{x})} < \frac{f'(\tilde{x}) - f'(\hat{x})}{\log_2(\tilde{x}) - \log_2(\hat{x})}$$

then put $\check{x} = \tilde{x}$ and $\tilde{x} = (\hat{x} + \check{x})/2$.

If f' is approximately linear in $\log x$ on the interval $(\check{x}, \hat{x})$, then f assumes its minimum at the initial value of $\tilde{x}$; if f' contains a discontinuity on the interval $(\check{x}, \hat{x})$, then we apply a few bisection steps in order to approximate the abscissa of this discontinuity from below.

(c4) First approximate $f''(\tilde{x})$ by calculating

$$C_3(w') + c_3^* \cdot \sum_{\substack{\omega \le W^\# \\ \omega \nmid \mathrm{lcm}(u^*, w')}} (\omega \log_2 n)^3 + c_0^* \cdot \left(\frac{A_2}{\log A_2} - \frac{B_2}{\log B_2} \right) \cdot (W^\# + \log_2 n),$$

where A_2 is an approximate solution for A in

$$\tilde{x} \cdot v' \cdot \left(\frac{A + B_2}{2} \right)^{W^\# (\log\log A - \log\log B_2)} = n^\mu,$$

and where c_0^*, c_3^* are machine dependent constants (cf. c3)). Next put $f(\tilde{x}) = f'(\tilde{x}) + f''(\tilde{x})$.

Put P equal to $C' - f(\tilde{x})$. If $P > 0$ then first perform step (4.6) with $W^\# = \max\{\mathrm{ord}(n \bmod p^k) : p \text{ prime}, p^k \parallel \mathrm{lcm}(t', v')\}$, and with $B^\# = A_2$.

Next replace T by

$$T - c_0^* \cdot \left(\frac{A_2}{\log A_2} - \frac{B_2}{\log B_2} \right) \cdot (\log_2 n + W^\#).$$

If either $P \le 0$ or $T \le 0$, proceed with step (d).

The expected profit P of performing steps (c2)–(c4) is determined by calculating $P(\mathcal{P}', \mathcal{Q}', s', t', u', v', w', \mu')$, which gives the difference between the running time of the rest of the algorithm (calculated in step (4.4)) with the present values of $\mathcal{P}'$, $\mathcal{Q}'$, s', t', u', v', w', μ', and the sum of the expected time necessary to find better values $\mathcal{P}^*$, $\mathcal{Q}^*$, s^*, t^*, u^*, v^*, w^*, μ^*, and the expected time C^* to run the rest of the algorithm with these. If the expected profit is positive, it is expected to be beneficial to search for additional factors.

If the cost to use additional factors is larger than the cost to use the optimal values found up till now, the complete interval where profit could be found has been examined, and the optimization is terminated.

(c5) Perform step (4.3) with $C^* = C'$, $B^* = \tilde{x}$, $z^* = 1$, and $\mu^* = \mu$, and next put $\mathcal{P}' = \mathcal{P}^*$, $\mathcal{Q}' = \mathcal{Q}^*$, $\mathcal{T}' = \mathcal{T}^*$, $s' = s^*$, $t' = t^*$, $u' = u^*$, $\mu' = \mu^*$, and $f'(\tilde{x}) = C^*$. Next put $C' = f'(\tilde{x}) + f''(\tilde{x})$.

Using the new factors found in the previous step, the values of $\mathcal{P}'$, $\mathcal{Q}'$, $\mathcal{T}'$ s', t', u', v', w', and μ' are updated, as well as C'.

(c6) For all primes l dividing $\mathrm{lcm}(u', w')$, let $\hat{e}_l$ be such that $\mathrm{lcm}(u', w') = \prod_{l \text{ prime}} l^{\hat{e}_l}$. Put $a_l = 0$ for all primes $l \mid \mathrm{lcm}(u', w')$ with $l \notin \mathcal{L}^- \cup \mathcal{L}^+$. Next perform step (4.2) for every $l^{\hat{e}_l} \notin \mathcal{L}^- \cup \mathcal{L}^+$.

For $l^{\hat{e}_l} \notin \mathcal{L}^- \cup \mathcal{L}^+$ one has to find extensions that can be used in Sections 5 and 6. This can only happen for those $l^{\hat{e}_l} \mid w'$ with $l^{\hat{e}_l} \nmid u_0$, (cf. (4.1)(a)).

If $l \notin \mathcal{L}^- \cup \mathcal{L}^+$ then step (4.2) has not yet been performed for any power of l; this is indicated by putting $a_l = 0$.

(d) If $C' \leq \bar{T}$, then put $\mathcal{P} = \mathcal{P}'$, $\mathcal{Q} = \mathcal{Q}'$, $\mathcal{T} = \mathcal{T}'$, $s = s'$, $t = t'$, $u = u'$, $v = v'$, $w = w'$, and $\mu = \mu'$, and proceed with Section 5; if $C' > \bar{T}$ the algorithm is aborted.

If the running time C' for the rest of the algorithm with the present values of $\mathcal{P}'$, $\mathcal{Q}'$, s', t', u', v', w', and μ' is "acceptable", i.e., at most $\bar{T}$, then the algorithm will be finished with these values. Otherwise the algorithm is aborted since the primality of n cannot be proved within a reasonable amount of time.

(4.2) Finding good extensions.

A value for $l^{\hat{e}_l}$ must have been specified. Perform steps (a) through (c).

For all prime powers $l^e \mid l^{\hat{e}_l}$ all entries in the table made in (2.3) that are useful in the test of n are retrieved, and the conductors are put in $\mathcal{M}^+(l^e)$; if there is at least one, l^e is put in the set $\mathcal{L}^+$, and else it is put in the set $\mathcal{L}^-$. If $l^e \in \mathcal{L}^-$, an extension of degree l^e can only be used in Sections 5 and 6 if it is created in step (5.1).

(a) If $a_l = 0$, put $e_0 = 0$ and perform step (a1). Otherwise, let $e_0 < \hat{e}_l$ be the largest e for which $l^e \in \mathcal{L}^- \cup \mathcal{L}^+$ and perform step (a2).

Initially, the sets $\mathcal{L}^+$ and $\mathcal{L}^-$ are empty. Step (a1) is performed only once per l.

(a1) Put $\mathcal{M}^+(l^e)$ equal to the empty set for $1 \leq e \leq \hat{e}_l$. For every prime m such that $m \in \mathcal{M}(l)$ (cf. (2.3)) and $n^{(m-1)/l} \not\equiv 1 \bmod m$ replace $\mathcal{M}^+(l)$ by $\mathcal{M}^+(l) \cup \{m\}$. Next, for every $e = 2, 3, \ldots, \hat{e}_l$ in succession, replace $\mathcal{M}^+(l^e)$ by $\{m \in \mathcal{M}^+(l^{e-1}) : l^e \mid m - 1\}$. Finally, for $1 \leq e \leq \hat{e}_l$ put l^e in $\mathcal{L}^+$ if $\mathcal{M}^+(l^e)$ is non-empty.

One checks whether the extensions that were precomputed in step (2.3) can be used for testing n; an extension of degree l^e and prime conductor m can be used if $n^{(m-1)/l} \not\equiv 1 \bmod m$ (cf. II.(4.8)). Note that this condition does not involve e.

(a2) Put $\mathcal{M}^+(l^e)$ equal to the empty set for $e_0+1 \leq e \leq \hat{e}_l$. For $e = e_0+1, e_0+2, \ldots, \hat{e}_l$ in succession, do the following. Replace $\mathcal{M}^+(l^e)$ by $\{m \in \mathcal{M}^+(l^{e-1}) : l^e \mid m-1\}$; if $\mathcal{M}^+(l^e)$ is non-empty put l^e in $\mathcal{L}^+$.

If the extension of degree l^{e_0} and prime conductor m can be used, then one can also use the extension of degree l^e and conductor m provided that $l^e \mid m-1$.

(b) If $l = 2$ perform step (b1), if l is odd perform step (b2).

In this step the case that the conductor is a power of the prime l is considered; one has to distinguish between the case that $l = 2$ and the case that l is odd.

(b1) If $a_2 = 0$ then perform step (b1a) through (b1c). Otherwise, if $a_2 > 0$, perform step (b1c).

If $a_2 = 0$ then step (4.2) has not yet been performed for any power of 2. Otherwise, if $a_2 > 0$ then step (4.2) has been performed for powers of 2, with a_2 as maximal value for $\hat{e}_l$. Finally, if $a_2 < 0$ then it has been checked that no extension of degree 2^e with prime power conductor can be used, since $n \not\equiv \pm 3 \bmod 8$.

(b1a) If $n \equiv \pm 3 \bmod 8$ put $a_2 = 1$, replace $\mathcal{M}^+(2)$ by $\mathcal{M}^+(2) \cup \{8\}$, and replace $\mathcal{L}^+$ by $\mathcal{L}^+ \cup \{2\}$. If $n \not\equiv \pm 3 \bmod 8$ put $a_2 = -1$.

In the case that $l = 2$ and $e \geq 1$, one can use an extension of degree 2^e with conductor $m = 2^{e+2}$ if $n \equiv \pm 3 \bmod 8$ holds.

One has to check this condition only once; if it holds one can add 2^{e+2} to $\mathcal{M}^+(2^e)$ for every e (cf. (2.4)(b3) and II.(4.10)).

(b1b) Check if $n \equiv 3 \bmod 4$. If this holds, replace $\mathcal{M}^+(2)$ by $\mathcal{M}^+(2) \cup \{4\}$, and replace $\mathcal{L}^+$ by $\mathcal{L}^+ \cup \{2\}$.

If $n \equiv 3 \bmod 4$ holds, one can use the extension $\mathbf{Q}(\zeta_4)$ with conductor $m = 4$ (cf. (2.4)(b3) and II.(4.10)).

(b1c) If $a_2 > 0$, then replace $\mathcal{M}^+(2^e)$ by $\mathcal{M}^+(2^e) \cup \{2^{e+2}\}$ and $\mathcal{L}^+$ by $\mathcal{L}^+ \cup \{2^e\}$ for $e = a_2+1, a_2+2, \ldots, \hat{e}_l$, and next put $a_2 = \hat{e}_l$.

(b2) If $a_l = 0$ then perform step (b2a). Otherwise, if $a_l > 0$ then perform step (b2b).

In the case that l is odd, one can use an extension of degree l^e with conductor $m = l^{e+1}$ if $n^{l-1} \not\equiv 1 \bmod l^2$ holds.

If $a_l = 0$ then step (4.2) has not yet been performed for any power of l. Otherwise, if $a_l > 0$ then step (4.2) has been performed for powers of l, with a_l as maximal value for $\hat{e}_l$. Finally, if $a_l < 0$ then it has been checked that no extension of degree l^e with conductor l^{e+1} can be used, since $n^{l-1} \equiv 1 \bmod l^2$.

(b2a) Check whether $n^{l-1} \not\equiv 1 \bmod l^2$; if this holds, replace $\mathcal{M}^+(l^e)$ by $\mathcal{M}^+(l^e) \cup \{l^{e+1}\}$ and $\mathcal{L}^+$ by $\mathcal{L}^+ \cup \{l^e\}$ for $e = 1, \ldots, \hat{e}_l$. Next put $a_l = \hat{e}_l$. Otherwise, if $n^{l-1} \equiv 1 \bmod l^2$, put $a_l = -1$.

One has to check this condition only once; if it holds one can add l^{e+1} to $\mathcal{M}^+(l^e)$ for every e.

(b2b) For $e = a_l+1, \ldots, \hat{e}_l$ replace $\mathcal{M}^+(l^e)$ by $\mathcal{M}^+(l^e) \cup \{l^{e+1}\}$, and next put $a_l = \hat{e}_l$.

If $a_l > 0$ then $n^{l-1} \not\equiv 1 \bmod l^2$, so one can add l^{e+1} to $\mathcal{M}^+(l^e)$.

(c) For every e such that $1 \le e \le \hat{e}_l$ and $\mathcal{M}^+(l^e)$ is empty, put l^e in $\mathcal{L}^-$ and do the following. Let e^* be the smallest e for which $\mathcal{M}^+(l^e)$ is empty. If $e^* = 1$ check whether n is an l^{th} power; if this is the case the primality test is terminated. If n is not an l^{th} power, for $e = e^*, \ldots, \hat{e}_l$ in succession find the smallest prime m such that both $l^e \mid m-1$ and $n^{(m-1)/l} \not\equiv 1 \bmod m$, and replace $\mathcal{M}^+(l^{e'})$ by $\mathcal{M}^+(l^{e'}) \cup \{m\}$ for $1 \le e' \le e$.

If n is an l-th power, then one will never succeed in finding a conductor with the correct properties. If this is not the case, then there exist prime conductors m with $n^{(m-1)/l} \not\equiv 1 \bmod m$. In practice, these m are not very hard to find.

(4.3) Finding good sets $\mathcal{P}^*$, $\mathcal{Q}^*$, and $\mathcal{T}^*$ and good values for s^*, t^*, and u^*.

Values for B^*, C^*, z^*, and μ^* must have been specified. Select values for b_1 and b_2.

If $C^* < 0$ and $z^* = 0$ then perform step (a). If $z^* = 2$, then perform step (b) with $\tilde{t} = t'$. In all other cases perform step (b), first for $\tilde{t} = t'$ and next for all other divisors $\tilde{t}$ of t_0 satisfying $e'_{i(\tilde{t})} \ge B^*$ and $c_5(e'_{i(\tilde{t})}, \tilde{t}, \mathrm{ord}(n \bmod \tilde{t}), 1, 1, \mu^*) < C^*$ (cf. (4.4)(e)).

Given values for B^*, C^* (representing the minimal costs found up till now), and z^*, values for $\mathcal{P}^*$, $\mathcal{Q}^*$, $\mathcal{T}^*$, s^*, t^*, u^* are chosen in this step in such a way, that the costs C^* to perform Jacobi sum tests for these particular values is (close to) minimal and such that $s^* \ge B^*$. The flag z^* is used to indicate if the (expensive) step (b2) should be performed after performing step (b1). The values b_1 and b_2 are used in steps (a) and (b2). Possible values are $b_1 = 1.05$ and $b_2 = 1.05$. For further comments we refer to steps (a) and (b2).

(a) Put t^* equal to the value t in the table made in step (2.2)(g) for which $e'_{i(t^*)}$ is equal to $\min\{e'_{i(t)} : e'_{i(t)} > b_1 \cdot B^*\}$. Next put $\mathcal{P}^* = \{p : p \mid t^*\}$, $\mathcal{Q}^* = \mathcal{Q}_{t^*}$, $s^* = \prod_{q \in \mathcal{Q}^*} q$, $u^* = \mathrm{lcm}\{u_{p,k} : p^k \parallel t^*\}$ (cf. (2.4)), and $\mathcal{T}^* = \mathcal{T}_{t^*}$ (cf. (4.1)(a1)). Finally, calculate $C^* = C(\mathcal{P}^*, \mathcal{Q}^*, \mathcal{T}^*, s^*, t^*, u^*, v_1, 1, \mu^*)$ by performing step (4.4) with $\tilde{\mathcal{P}} = \mathcal{P}^*$, $\tilde{\mathcal{Q}} = \mathcal{Q}^*$, $\tilde{\mathcal{T}} = \mathcal{T}^*$, $\tilde{s} = s^*$, $\tilde{t} = t^*$, $\tilde{u} = u^*$, $\tilde{v} = v_1$, $\tilde{w} = 1$, and $\tilde{\mu} = \mu^*$.

If $C^* < 0$, a first estimate for the running time, C^*, and initial values for s^*, t^*, u^*, $\mathcal{P}^*$, $\mathcal{Q}^*$ will be found in this step. It is known that the time needed to perform the steps of Sections 5 and 6 with this particular choice for s^*, t^*, u^*, $\mathcal{P}^*$, and $\mathcal{Q}^*$ may happen to be not so very close to the minimum. Its main purpose is to find very quickly an upper bound for the running time needed to perform the steps of Sections 5 and 6. In this way we can speed up the other optimization steps of (4.3). This is the case because some parts of the steps of Sections 5 and 6 may be even too expensive for particular values of $\tilde{t}$. These values can then be rejected, without doing any expensive optimization steps.

For fairly small n however, the choices made in this step may be the final choices, since any choice which gives approximately the minimum will be fine.

Experiments have shown that values of $\tilde{t}$ with $e'_{i(\tilde{t})} \approx B^*$ do not give a very good approximation for the minimum time needed for the steps of Sections 5 and 6. Better values of $\tilde{t}$ have values of $e'_{i(\tilde{t})}$ which seem to be at least some offset times the specified value of B^*. The value b_1 is equal to the offset needed to get a better value for $\tilde{t}$.

(b) First put $\tilde{\mathcal{Q}} = \mathcal{Q}_{\tilde{t}}$, $\tilde{\mathcal{P}} = \{p \text{ prime} : p \mid \tilde{t}\}$, $\tilde{\mathcal{T}} = \mathcal{T}_{\tilde{t}}$ (cf. (3.4)), $\tilde{s} = \prod_{q\in\tilde{\mathcal{Q}}} q$, $\tilde{u} = \text{ord}(n \bmod \tilde{t})$. If $z^* \geq 1$ put $C^{**} = C^*$ and perform steps (b1) and (b2). Otherwise, if $z^* = 0$, perform step (b1).

In this step one tries to find the best values for $\mathcal{P}^*$, $\mathcal{Q}^*$, $\mathcal{T}^*$, s^*, t^*, and u^*. This is done by checking for all values $\tilde{t} \mid t_0$ if a subset of $\mathcal{Q}_{\tilde{t}}$ containing the least expensive q's results in a better value for C^*. Since it is easy to estimate the time needed to perform the final trial division, this is used as criterion to reject too expensive values (cf. (6.1), (4.4)(e)).

(b1) Put $\hat{\mathcal{Q}} = \tilde{\mathcal{Q}}$, $\hat{\mathcal{P}} = \tilde{\mathcal{P}}$, $\hat{\mathcal{T}} = \tilde{\mathcal{T}}$, $\hat{s} = \tilde{s}$, $\hat{t} = \tilde{t}$, and $\hat{u} = \tilde{u}$, $\hat{\mu} = \tilde{\mu}$, and perform steps (b1a) and (b1b).

Given initial sets $\tilde{\mathcal{P}}$, $\tilde{\mathcal{Q}}$, and $\tilde{\mathcal{T}}$, and initial values for $\tilde{s}$, $\tilde{t}$, $\tilde{u}$, $\tilde{v}$, and $\tilde{w}$, one tries to find subsets $\hat{\mathcal{Q}}$ and $\hat{\mathcal{T}}$ of $\mathcal{Q}$ and $\mathcal{T}$ respectively, and $\hat{s} \mid s$, such that the running time to perform the Jacobi-sum tests and the final trial division for these values is (possibly) smaller than the running time for the present values, while $\hat{s}$ is still large enough.

Since one might perform two optimization steps, a copy has to be made of all values in order to start the second optimization step with the same values. In the first optimization step all $q \in \mathcal{Q}$ are weighted according to $c_1(q) = (\sum_{p\mid q-1} u^2_{p,o_p(q-1)})/\log_2(q)$ (cf. (4.1)(a2)). So the q with $c_1(q) = \max\{c_1(q)\}$ is regarded as being the most expensive q.

(b1a) For q_i, with $q_i \in \tilde{\mathcal{Q}}$, $i = 1, 2, \ldots$ in succession, if $s/q_i \geq B^*$, then replace $\tilde{\mathcal{Q}}$ by $\tilde{\mathcal{Q}}\backslash\{q_i\}$, $\tilde{\mathcal{T}}$ by $\tilde{\mathcal{T}}\backslash\{(q_i, p, h) : p \mid q_i - 1, h = o_p(u_{p,o_p(q_i-1)})\}$, and $\tilde{s}$ by $\tilde{s}/q_i$.

In this step the q_i's were ordered according to step (4.1)(a2).

(b1b) Calculate $\tilde{C} = C(\tilde{\mathcal{P}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{T}}, \tilde{s}, \tilde{t}, \tilde{u}, 1, 1, \tilde{\mu})$ by performing step (4.4). If $\tilde{C} < C^*$ or $\tilde{C} = C^*$ and $s^* < \tilde{s}$, replace $\mathcal{Q}^*$, $\mathcal{P}^*$, $\mathcal{T}^*$, s^*, t^*, u^*, μ^*, and C^* by $\tilde{\mathcal{Q}}$, $\tilde{\mathcal{P}}$, $\tilde{\mathcal{T}}$, $\tilde{s}$, $\tilde{t}$, $\tilde{u}$, $\tilde{\mu}$, and $\tilde{C}$ respectively.

(b2) If $\tilde{C} < b_2 \cdot C^{**}$ or $C^{**} < 0$ then put $\tilde{\mathcal{Q}} = \hat{\mathcal{Q}}$, $\tilde{\mathcal{P}} = \hat{\mathcal{P}}$, $\tilde{\mathcal{T}} = \hat{\mathcal{T}}$, $\tilde{s} = \hat{s}$, $\tilde{t} = \hat{t}$, and $\tilde{u} = \hat{u}$, $\tilde{\mu} = \hat{\mu}$ and perform steps (b2a) and (b2b).

In the second optimization step all $q \in \hat{\mathcal{Q}}$ are weighted according to $c_2(q) = C_q/\log_2(q)$, were

$$C_q = C(\tilde{\mathcal{Q}}, \tilde{\mathcal{P}}, \tilde{\mathcal{T}}, \tilde{s}, \tilde{t}, \tilde{u}, 1, 1, \tilde{\mu}) - C(\tilde{\mathcal{Q}}\backslash\{q\}, \tilde{\mathcal{P}}, \tilde{\mathcal{T}}\backslash\{(q, \tilde{p}, \tilde{h}) \in \tilde{\mathcal{T}}\}, \tilde{s}/q, \tilde{t}, \tilde{u}, 1, 1, \tilde{\mu}).$$

That is, C_q is the additional cost of performing the necessary Jacobi sum test for conductor q, if all other tests in $\tilde{\mathcal{T}}$ have been done.

This optimization step is more expensive than the one described in step (b1), and is only performed when indicated by the flag z^*. If the time needed to perform the steps of Sections 5 and 6 with the present values for $\tilde{\mathcal{P}}$, $\tilde{\mathcal{Q}}$, $\tilde{\mathcal{T}}$, $\tilde{s}$, $\tilde{t}$, and $\tilde{u}$, found in step (b1), is close to the minimum up till now (within a margin determined by a blow-up factor b_2), or if $C^{**} < 0$ (indicating that no initial values of s^*, t^*, u^*, $\mathcal{P}^*$, $\mathcal{Q}^*$, and $\mathcal{T}^*$ have been found) one tries this possibly better (but more expensive) optimizing procedure to find the minimum.

(b2a) For all $q \in \tilde{\mathcal{Q}}$ let C_q be defined as the difference between $C(\tilde{\mathcal{Q}}, \tilde{\mathcal{P}}, \tilde{\mathcal{T}}, \tilde{s}, \tilde{t}, \tilde{u}, 1, 1, \tilde{\mu})$ and $C(\tilde{\mathcal{Q}}\backslash\{q\}, \tilde{\mathcal{P}}, \tilde{\mathcal{T}}\backslash\{(q, \tilde{p}, \tilde{h}) \in \tilde{\mathcal{T}}\}, \tilde{s}/q, \tilde{t}, \tilde{u}, 1, 1, \tilde{\mu})$. Perform steps (b2a1) through (b2a3) as long as there exists a $q \in \tilde{\mathcal{Q}}$ with $C_q > 0$ and $\tilde{s}/q \geq B^*$.

(b2a1) For all $q \in \tilde{\mathcal{Q}}$ calculate C_q and $c_2(q) = C_q/\log_2(q)$ by performing step (4.4).

(b2a2) Find $\bar{q}$ such that $c_2(\bar{q}) = \max\{c_2(q) : q \in \tilde{\mathcal{Q}}, C_q \neq 0, \mathrm{lcm}(\tilde{s}/q, \tilde{v}) \geq B^*\}$.

(b2a3) Replace $\tilde{\mathcal{Q}}$ by $\tilde{\mathcal{Q}}\backslash\{\bar{q}\}$, $\tilde{\mathcal{T}}$ by $\tilde{\mathcal{T}}\backslash\{(\bar{q}, p, h) : p \mid \bar{q} - 1, h = o_p(u_{p,o_p(\bar{q}-1)})\}$, and $\tilde{s}$ by $\tilde{s}/\bar{q}$. Finally replace $\tilde{C}$ by $\tilde{C} - C_{\bar{q}}$.

(b2b) If $\tilde{C} < C^*$ or $\tilde{C} = C^*$ and $s < \tilde{s}$, replace $\mathcal{Q}^*$, $\mathcal{P}^*$, $\mathcal{T}^*$, s^*, t^*, u^*, μ^*, and C^* by $\tilde{\mathcal{Q}}$, $\tilde{\mathcal{P}}$, $\tilde{\mathcal{T}}$, $\tilde{s}$, $\tilde{t}$, $\tilde{u}$, $\tilde{\mu}$, and $\tilde{C}$ respectively.

(4.4) Running times.

Values for $\tilde{\mathcal{P}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{T}}, \tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}$, $\tilde{w}$, and $\tilde{\mu}$ must have been specified. The machine dependent functions c_m, c_d, c_n and c_f must be known. Perform steps (a) through (f).

On input $\tilde{\mathcal{P}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{T}}, \tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}, \tilde{w}$, in this step an estimate of the running time $C(\tilde{\mathcal{P}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{T}}, \tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}, \tilde{w})$ of the steps of Sections 5 and 6 is computed.

The function $c_m(a)$ denotes the time needed to perform a multiplication of two integers of size (binary logarithm) a. The function $c_d(a, b)$ denotes the time needed to perform a division of an integer of size a by an integer of size b. The function $c_n(a)$ denotes the time needed to perform a multiplication of two integers modulo n, where $\lceil \log_2(n) \rceil = a$ and finally the function $c_f(a)$ denotes the time needed to perform the final trial division for n with $\log_2(n) = a$ if $\mu = \frac{1}{3}$ for one residue class. Each function will be an approximation of the time needed to perform the operation averaged over several cases of n; it should be determined empirically and will be specified in Chapter VI for some machines.

(a) Put

$$C_1 = \sum_{l^{\hat{e}_l} \in \mathcal{L}^-} c_1(n, l^{\hat{e}_l}, m_l),$$

with $c_1(n, l^{\hat{e}_l}, m_l) = (l^{3\hat{e}_l} + m_l^2 \cdot l^{\hat{e}_l}) \cdot c_n(\log_2(n))$, where m_l is the smallest $m \in \mathcal{M}^+(l^{\hat{e}_l})$.

This is the time needed to perform step (5.1), i.e., the time needed to compute additional Galois extensions. These extensions only have to be computed for prime powers $l^{\hat{e}_l} \in \mathcal{L}^-$, since for all $l^{\hat{e}_l} \in \mathcal{L}^+$, the extensions have been generated in advance in step (2.3). The time needed for this step is dominated by the time needed for the operations in step (5.1)(b8) and (5.1)(b9).

(b) Put

$$C_2 = \sum_{l^{\hat{e}_l} \| \mathrm{lcm}(\tilde{u}, \tilde{w})} c_2(n, l^{\hat{e}_l}),$$

where $c_2(n, l^{\hat{e}_l}) = (l^{\hat{e}_l})^3 \cdot c_n(\log_2(n))$.

This is the time needed to perform step (5.2), i.e., the time needed to select cyclic rings and to create the transition matrices. The time needed for this step is dominated by the time needed for the operations in step (5.2)(b4).

(c) Put $c = 0$ and for each $\omega \in \Omega$ with $\omega \mid \tilde{w}$ put T_ω equal to $\max\{\frac{p}{p-1} : p \mid v_\omega\}$. First perform step (c1) as long as there exists an $\omega \in \Omega$ with $\omega \mid \tilde{w}$ such that $T_\omega > 0$. Next put $C_3 = c$.

This is an estimate of the time needed to perform step (5.3), i.e., the time needed to calculate cyclotomic extensions. The time needed for this step is dominated by the time needed for the operations in steps (5.3)(d3) and (5.3)(d4). A more accurate estimate could be calculated by dynamic programming techniques.

(c1) Put $\bar{\omega} = \max\{\omega \in \Omega,\ \omega \mid \tilde{w} : T_\omega > 0\}$. Replace c by $c + T_{\bar{\omega}} \cdot (2\bar{\omega}^3 \cdot \log_2 v_{\bar{\omega}} + \bar{\omega}^2 \cdot \log_2 n) \cdot c_n(\log_2(n))$, and replace for all $\omega \in \Omega$ with $\omega \mid \bar{\omega}$ the value T_ω by $T_\omega - T_{\bar{\omega}}$ (cf. V.(4.3)).

(d) Perform (4.5) to compute $c_4(\tilde{T})$, and set $C_4 = c_4(\tilde{T}) \cdot \log_2 n \cdot c_n(\log_2(n))$.

This is the time needed to perform step (5.4), i.e., the time needed to perform the Jacobi sum tests. The time needed for this step is dominated by the time needed for the operations in step (5.4)(d).

(e) Put $C_5 = c_5(\tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}, \tilde{w}, \tilde{\mu})$, where

$$c_5(\tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}, \tilde{w}, \tilde{\mu}) = t_f \cdot (1 + \sqrt{n}/s_f) \cdot (c_m(\log_2(s_f)) + c_d(2\log_2(s_f), \log_2(s_f))),$$

if $\tilde{\mu} = \frac{1}{2}$, and

$$c_5(\tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}, \tilde{w}, \tilde{\mu}) = t_f/2 \cdot c_f(\log_2(n))$$

if $\tilde{\mu} = \frac{1}{3}$, with $t_f = \mathrm{lcm}(\tilde{t}, \tilde{u}, \tilde{w})$ and $s_f = \mathrm{lcm}(\tilde{v}, \tilde{s} \cdot \prod_{p \mid \tilde{t}} p^{o_p(n^{\tilde{u}}-1)-o_p(\tilde{s})})$.

This is the time needed to perform step (6.1), i.e., the time needed to perform the final trial divisions.

(f) Put $C(\tilde{\mathcal{P}}, \tilde{\mathcal{Q}}, \tilde{T}, \tilde{s}, \tilde{t}, \tilde{u}, \tilde{v}, \tilde{w}, \tilde{\mu}) = C_1 + C_2 + C_3 + C_4 + C_5$.

(4.5) Finding an optimal set of combined Jacobi sum tests.

The set $\tilde{T}$ must have been specified.

Let $\bar{\mathcal{Q}} = \{q : \exists p, h \text{ with } (q,p,h) \in \tilde{T}\}$, $\bar{\mathcal{P}} = \{p : \exists q, h \text{ with } (q,p,h) \in \tilde{T}\}$, and $\bar{\mathcal{H}}_p = \{h : \exists q \text{ with } (q,p,h) \in \tilde{T}\}$ for all $p \in \bar{\mathcal{P}}$.

Assume that the set $\tilde{T}$ of triples (q,p,h) is given, where each triple (q,p,h) represents a Jacobi sum test (cf. (5.4)) for a character χ with $\mathrm{cond}(\chi) = q$, $\mathrm{ord}(\chi) = p^{o_p(q-1)}$, and $h = o_p(\mathrm{ord}(n \bmod p^{o_p(q-1)}))$. In this step a method is described that determines how these tests can be combined in an optimal way. A combination consists of a subset $\mathcal{S} \subset \tilde{T}$ of triples, $\mathcal{S} = \cup_i \{(q_i, p_i, h_i)\}$ such that $p_i \neq p_j$ for $(q_i, p_i, h_i), (q_j, p_j, h_j) \in \mathcal{S}$ with $i \neq j$ (cf. II.(8.9) and III.(3.12)). The optimal set $\mathbf{S}_{\tilde{T}}$ of such combinations $\mathcal{S}$ is found, i.e., the set of combinations for which the cost of performing step (5.4) with this $\tilde{T}$ is minimal. This cost will be calculated as well, and is denoted by $c_4(\tilde{T})$.

This is done as follows. Each Jacobi sum test, represented by a triple (q,p,h), involves an exponentiation in an extension of degree $u_{p,1} \cdot p^h$, where the exponent is roughly of the same magnitude as n. The time needed to do such an exponentiation is proportional to $(u_{p,1} \cdot p^h)^2 \cdot \log_2^3(n)$. With hardly any extra work, one is able to perform a number of Jacobi sum tests at the same time, thereby saving much time. Two (or more) Jacobi sum tests can be done together in an extension of degree which is the least common multiple of the original degrees. If the degree for one test divides that for another test, both can be done

in the extension of the largest degree, in about the same time as performing the Jacobi sum test in the largest extension. Thus one can perform the Jacobi sum tests in extensions of degrees dividing u together in (almost) the same time as that for the single test in degree u itself. Performing tests together can however only be done if the orders of the characters involved are relatively prime, i.e., the primes p of the triples representing these tests should be relatively prime. These observations suggests a greedy strategy: find the largest degree, and next find all degrees dividing this degree (and relatively prime orders), and perform the tests together. As shown in III.3 this will give (under reasonable assumptions, cf. III.(2.17) and III.(3.1)) an optimal solution.

(a) For all $p \in \bar{\mathcal{P}}$ and all $h \in \bar{\mathcal{H}}_p$, let $d_{p,h} = \#\{q : q \in \bar{\mathcal{Q}}, (q,p,h) \in \tilde{\mathcal{T}}\}$. Put $\mathbf{S}_{\tilde{\mathcal{T}}}$ equal to the empty set, and $c_4(\tilde{\mathcal{T}}) = 0$. Put $b = 0$. Perform steps (a1) and (a2) as long as $d_{p,h} \neq 0$ for some $p \in \bar{\mathcal{P}}$ and some $h \in \bar{\mathcal{H}}_p$.

Each Jacobi sum test is represented by a triple (q,p,h), and consists of an exponentiation in an extension of degree $u_{p,1} \cdot p^h$, where the exponent is roughly of the same magnitude as n (and with $u_{p,1}$ as in (3.4)(b)). Two such tests will be combined only if the degree in which one extension has to be performed divides the degree of the other extension. The value $d_{p,h}$ gives the number of Jacobi sum tests left for a character of order p^k (for some k) to be done in an extension of degree $u_{p,1} \cdot p^h$.

(a1) Let $\bar{u} = \max\{u_{p,1} \cdot p^h : d_{p,h} \neq 0,\ p \in \bar{\mathcal{P}},\ h \in \bar{\mathcal{H}}_p\}$. Find $\bar{p}$ and $\bar{h}$ such that $\bar{u} = u_{\bar{p},1} \cdot \bar{p}^{\bar{h}}$. If more than one choice is possible, any choice with $u_{\bar{p},1} \cdot \bar{p}^{\bar{h}} = \bar{u}$ will do; to increase the speed of this step one can take a choice with $d_{p,h}$ maximal. Put $\bar{d} = d_{\bar{p},\bar{h}}$. Let $\bar{\mathcal{Q}}' = \{q : (q,\bar{p},\bar{h}) \in \tilde{\mathcal{T}}\}$, and order the elements $q_i \in \bar{\mathcal{Q}}'$ such that $\{q_i\}_{i=1}^{\bar{d}}$ is increasing. Put $\mathcal{S}_{b+i} = \{(q_i,\bar{p},\bar{h}) : (q_i,\bar{p},\bar{h}) \in \tilde{\mathcal{T}},\ (q_i,\bar{p},\bar{h}) \notin \mathcal{S}_f,\ 0 \leq f < b+i\}$, for $1 \leq i \leq \bar{d}$. For all $p \in \bar{\mathcal{P}}$ for which $p \neq \bar{p}$, first put $j = 0$ and next perform step (a1a).

In this step one finds all triples $(q,\bar{p},\bar{h})$ such that the degree $u_{\bar{p},1} \cdot \bar{p}^{\bar{h}}$ in which the Jacobi sum test has to be performed is maximal. All of these $\bar{d}$ tests have to be put in different combinations $\mathcal{S}_{b+i}$ because the p's are the same.

(a1a) Find $\tilde{h} \in \bar{\mathcal{H}}_p$ such that $\tilde{h} = \max\{h : d_{p,h} \neq 0 \text{ and } u_{p,1} \cdot p^h \mid \bar{u}\}$. If such a $\tilde{h}$ can be found let $d = \min\{d_{p,\tilde{h}},\ \bar{d} - j\}$ and let $\bar{\mathcal{Q}}' = \{q : (q,p,\tilde{h}) \in \tilde{\mathcal{T}}\}$, and order the elements $q_i \in \bar{\mathcal{Q}}'$ such that $\{q_i\}_{i=1}^{d}$ is increasing. For $1 \leq i \leq d$, replace $\mathcal{S}_{b+j+i}$ by $\mathcal{S}_{b+j+i} \cup \{(q_i,p,\tilde{h}) : (q_i,p,\tilde{h}) \in \tilde{\mathcal{T}}\}$, replace $d_{p,\tilde{h}}$ by $d_{p,\tilde{h}} - d$ and j by $j + d$. Repeat this step if $j < \bar{d}$ and $\{(q,p,h) : (q,p,h) \in \tilde{\mathcal{T}},\ d_{p,h} \neq 0,\ h < \tilde{h} \text{ and } u_{p,1} \cdot p^h \mid \bar{u}\}$ is not empty.

In this step as many tests as possible with degrees dividing $u_{\bar{p},1} \cdot \bar{p}^{\bar{h}}$ are added to the combinations $\mathcal{S}_{b+i}$ created in step (a1).

(a2) Replace $\mathbf{S}_{\tilde{\mathcal{T}}}$ by $\mathbf{S}_{\tilde{\mathcal{T}}} \cup \{\mathcal{S}_{b+i}\}$ for $1 \leq i \leq \bar{d}$, and increase $c_4(\tilde{\mathcal{T}})$ by $\bar{d} \cdot \bar{u}^2$. Replace b by $b + \bar{d}$ and replace $\bar{d}$ by zero.

At this point, no triple (q,p,h) with degree dividing $u_{\bar{p},1} \cdot \bar{p}^{\bar{h}}$ can be found. Therefore the combinations $\mathcal{S}_{b+i}$ are added to $\mathbf{S}_{\tilde{\mathcal{T}}}$. The time needed to perform a combination of tests is

dominated by the time to do the exponentiation, which takes place in an extension of the maximal degree $(u_{p,1}p^h)$, and takes about $(u_{p,1} \cdot p^h)^2 \cdot (\log_2 n)^3$ operations.

(4.6) Trial division for Lucas-Lehmer step.

The positive integers $W^\#$ and $B^\#$ must have been specified.

Set a new trial division bound $B_1 = \min(B_0, \sqrt{n}, B^\#)$ (cf. (2.1)). First perform steps (a) through (d), and next put $B_2 = B_1$.

In this step an attempt is made to increase the value of v (cf. (3.2) and (3.5)) by finding factors of $n^\omega - 1$ for small ω up to the bound B_1. The choice made in step (2.1) implies an upper bound for B_1. So far all prime factors up to B_2 of $n^\omega - 1$ have been found for $\omega \leq W^\#$; initially we have $B_2 = 1$. By v_ω is denoted the product of the prime power factors of $n^\omega - 1$, not dividing $n^{\omega'} - 1$ for any $\omega' \mid \omega$ with $\omega' < \omega$. Furthermore, $o^*_{p,k} = o_p(n^{\mathrm{ord}(n \bmod p^k)} - 1)$ (cf. (3.4) and (3.5)).

(a) For all $\omega \leq W^\#$ and for $i = 0, 1, \ldots, \omega - 1$ put $r_{\omega,i}$ equal to $n - 1$.

In this step one writes $n^\omega - 1$ in base n, i.e., $n^\omega - 1 = \sum_{i=0}^{\omega-1} r_{\omega,i} \cdot n^i$.

(b) Perform step (b1) for all $\omega \leq W^\#$.

(b1) If $\omega \in \Omega$ then divide $\sum_{i=0}^{\omega-1} r_{\omega,i} \cdot n^i$, which is initially $n^\omega - 1$, by v_ω. This can be done as follows. Put $c = 0$, and for $i = \omega - 1, \ldots, 0$ in succession put $c' = (c \cdot n + r_{\omega,i}) \bmod v_\omega$, put $r_{\omega,i} = \lfloor (c \cdot n + r_{\omega,i})/v_\omega \rfloor$, and replace c by c'. If $\omega \notin \Omega$ then put $\mathcal{F}_\omega$ equal to the empty set.

If $\omega \in \Omega$, then factors of $n^\omega - 1$ have already been found. The product of all the factors of $n^\omega - 1$, which is v_ω, will be divided out of $\sum_{i=0}^{\omega-1} r_{\omega,i} \cdot n^i$ This is done in this step, and is basically the same as step (3.4)(b). For further comments, we refer to this step.

(c) If $2 \leq B_2$ continue with step (d). Otherwise, if $B_2 < 2 \leq B_1$, let $o_2(n-1) = a$, let $o_2(n+1) = b$, and do the following. If $2^a \nmid v_1$ then replace v by $v \cdot 2^a$, v_1 by $v_1 \cdot 2^a$, and put $o^*_{2,1} = a$. If $2^b \nmid v_2$ then replace v by $v \cdot 2^b$, v_2 by $v_2 \cdot 2^b$, and put $o^*_{2,2} = a + b$. Perform step (c1) for all $\omega = 2^k \leq W^\#$, with $k \geq 2$.

(c1) If $\omega \notin \Omega$ then replace Ω by $\Omega \cup \{\omega\}$. If $2^{a+b+k-1} \nmid v_\omega$ then divide $\sum_{i=0}^{\omega-1} r_{\omega,i} \cdot n^i$ by $2^{a+b+k-1}$ using the same method as in (b1), replace $\mathcal{F}_\omega$ by $\mathcal{F}_\omega \cup \{2\}$, w by $\mathrm{lcm}(w, \omega)$, v by $v \cdot 2$, and v_ω by $v_\omega \cdot 2$, and put $o^*_{2,o_2(\omega)} = a + b + k - 1$.

(d) Perform step (d1) for all primes p with $B_2 < p \leq B_1$ (where the primes p are generated using the file created in (2.1)), but only as long as $v \leq \sqrt{n}$.

In this step it is checked if $n^\omega - 1$ is divisible by p, for some $\omega \leq W^\#$. If such an ω can be found, then $\sum_{i=0}^{\omega-1} r_{\omega,i} \cdot n^i$ is actually divided by p to find the multiplicity of p in $n^\omega - 1$.

(d1) Let $n_p = n \bmod p$. Calculate $n_p^\omega \bmod p$ for $\omega = 1, 2, \ldots$ in succession until $n_p^\omega \equiv 1 \bmod p$ or $\omega = W^\#$. If $n_p^\omega \equiv 1 \bmod p$ for some $\omega \leq W^\#$, then perform steps (d1a) through (d1d).

If $n_p^\omega \equiv 1 \bmod p$, then p is a divisor of $n^\omega - 1$ and $o_p(n^\omega - 1)$ should be calculated.

(d1a) Put $e = 0$, $m = 0$, and perform (d1a1) until $m \neq 0$.

(d1a1) Put $e = e + 1$ and let $c = 0$. For $i = \omega - 1, \omega - 2, \ldots, 0$ in succession, first put $c' = (c \cdot n + r_{\omega,i}) \bmod p$ and $r_{\omega,i} = \lfloor (c \cdot n + r_{\omega,i})/p \rfloor$, and next replace m by $(m \cdot n_p + r_{\omega,i}) \bmod p$ and c by c'.

In this step $n^\omega - 1$ is divided $o_p(n^\omega - 1)$ times by p. This is done differently from other multi-divisions (such as in (3.4)(b)), since one not only keeps track of a carry c, but also of a variable m, which has the value $(n^\omega - 1)/p^{e+1} \bmod p$. So m is used to check whether $e = o_p(n^\omega - 1)$. After every call to step (d1a) the $r_{\omega,i}$ satisfy $\sum_{i=0}^{\omega-1} r_{\omega,i} \cdot n^i = (n^\omega - 1)/(v_\omega \cdot p^e)$ and m is equal to $n^\omega - 1 \bmod p^{e+1}$.

(d1b) If $\omega \notin \Omega$ then replace Ω by $\Omega \cup \{\omega\}$ and w by $\mathrm{lcm}(w, \omega)$. If $p \nmid v_\omega$ then replace v_ω by $v_\omega \cdot p^e$ and v by $v \cdot p^e$, and put $o^*_{p,e} = e$.

(d1c) For all $\omega' = \omega \cdot p^k \leq W^\#$, $k \geq 1$, do the following. If $\omega' \notin \Omega$ then replace Ω by $\Omega \cup \{\omega'\}$ and w by $\mathrm{lcm}(w, \omega')$. If $p \nmid v_{\omega'}$ then replace $v_{\omega'}$ by $v_{\omega'} \cdot p$ and put $o^*_{p,e+k} = e + k$.

(d1d) For all $\omega' \leq W^\#$, with $\omega \cdot p^k \mid \omega'$ and $k \geq 1$, divide $\sum_{i=0}^{\omega'-1} r_{\omega',i} \cdot n^i$ by p^{e+k} using the same method as in (b1).

5. LUCAS-LEHMER AND JACOBI SUM TESTS.

In this section the actual Lucas-Lehmer tests and Jacobi sum tests are described.

(5.1) Generation of additional Galois extensions.

Perform steps (a) and (b) for every prime power $u = l^e \in \mathcal{L}^-$.

As explained in step (2.3), one will have to compute in certain extension rings of $\mathbf{Z}/n\mathbf{Z}$, which can be constructed from rings of integers of cyclic subfields of cyclotomic extensions. For all prime power degrees l^e dividing $\lambda(t_0)$, a list of extensions is precalculated in step (2.3). As explained in Section 4, an extension can only be used, if the conductor m of the corresponding field extension satisfies the correct condition (cf. (4.2) and II.(4.6)). It may be that none of the extensions on the list for l^e satisfies this condition.

Secondly, for ω with $l^e \mid \omega$ and $l^e \nmid \lambda(t_0)$, it may be decided during the optimization step in Section 4, that using a factor of $n^\omega - 1$ (and therefore using an extension of degree ω), would reduce the running time of the algorithm.

In both cases, the degree l^e was added to the set $\mathcal{L}^-$ during the optimization step. In this step one generates an additional Galois extension for every prime power $l^e \in \mathcal{L}^-$. This step is basically the performance of step (2.3)(b) for all prime powers l^e in $\mathcal{L}^-$. Explanation can be found there. Here we will only comment on the differences.

(a) Find the smallest $m \in \mathcal{M}^+(u)$ such that $m \mid s$ (cf. (4.2)); if none such m exists take the smallest $m \in \mathcal{M}^+(u)$.

(b) If $u = 2$ and $m \in \{4, 8\}$ perform steps (b1), (b2), (b3), and (b10). If $u = 2$ and $u \mid m - 1$, perform steps (b1), (b2), (b4), (b5) and (b10). If $u = 2^e$ with $e > 1$, or $u = l^e$ with l odd and $e \geq 1$ perform steps (b1), and (b5) through (b10).

(b1) Put $r = 1$ if m is prime, and put $r = 0$ if m is not prime.

(b2) Put $D = 1$. If m is prime put

$$S = \begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix} \text{ and } S^* = \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}.$$

If $m \in \{4, 8\}$ put S and S^* both equal to the 2×2 unit matrix.

(b3) If $m = 4$ put $g = 3$ and $f = X^2 + 1$; if $m = 8$ put $g = 5$ and $f = X^2 - 2$.

(b4) Put $f = X^2 + X + (4 \cdot \lfloor \frac{m+1}{4} \rfloor - m) \cdot \lfloor \frac{m+1}{4} \rfloor$.

(b5) If m is odd, find a primitive root g modulo m, for instance by trying all $g \geq 2$ with $\gcd(m, g) = 1$ in succession. If $m = 2^{e+2}$, put $g = 5$.

(b6) Let $b_{i,j} = 0$ for $0 \leq i < u$ and $0 \leq j < m$. Put $j = 1$. If m is odd, for $i = 0, 1, \ldots, \phi(m) - 1$ in succession replace $b_{i \bmod u, j}$ by 1 and j by $j \cdot g \bmod m$. If $m = 2^{e+2}$, for $i = 0, 1, \ldots, 2^e - 1$ in succession replace $b_{i \bmod u, j}$ and $b_{i \bmod u, m-j}$ by 1 and j by $j \cdot g \bmod m$. Define $\eta = \sum_{j=0}^{m-1} b_{0,j} \cdot \zeta_m^j$ and, for $i = 0, 1, \ldots, u - 1$, its conjugates $\sigma_g^i(\eta) = \sum_{j=0}^{m-1} b_{i,j} \cdot \zeta_m^j$. The $\sigma_g^i(\eta)$ will be represented by the vectors $(b_{i,j})_{j=0}^{m-1}$.

(b7) Determine the polynomial $f = \prod_{i=0}^{u-1}(X - \sigma_g^i(\eta)) \in \mathbf{Z}[X]$ by calculating sufficiently close approximations $c_i \in \mathbf{C}$ to $\sum_{j=1}^{m-1} b_{i,j} \cdot \zeta_m^j$ for $0 \leq i < u$, with $\zeta_m = e^{2\pi\sqrt{-1}/m}$,and by rounding the coefficients of the polynomial $\prod_{i=0}^{u-1}(X - c_i)$ to the nearest integers.

(b8) If m is prime, define $\varsigma_{g,k,u}$ for $0 \leq k < u$ by

$$\varsigma_{g,k,u} = \sigma_g^k(\eta_u) = \sigma_g^k(\eta).$$

If m is not prime, define $\varsigma_{g,k,u}$ for $0 \leq k < u$ by $\varsigma_{g,0,u} = 1$, and

$$\varsigma_{g,k,u} = \sigma_g^{(k-l^{i-1})}(\eta_{l^i}),$$

with i such that $1 \leq i \leq e$ and $l^{i-1} \leq k < l^i$. Compute a $u \times u$ dimensional integer matrix S by performing steps (b8a) through (b8c).

(b8a) Determine $d_{i,j}$ such that $\eta^i = \sum_{j=0}^{m-1} d_{i,j} \cdot \zeta_m^j$ for $i = 2, 3, \ldots, u-1$, by computing the consecutive powers of the polynomial $\sum_{j=0}^{m-1} b_{i,j} \cdot T^j$ modulo the polynomial $T^m - 1$.

(b8b) Perform step (b8b1) if m is prime and perform step (b8b2) for $k = 1, \ldots, e$ in succession if m is not a prime.

(b8b1) First put $h = 1$ and for $j = 0, 1, \ldots, u-1$ in succession first put $s_{j,i} = d_{i,h} - d_{i,0}$ for $0 \leq i < u$ and next replace h by $h \cdot g \bmod m$.

(b8b2) Put $h = 1$ and for $j = l^{k-1}, \ldots, l^k - 1$ in succession first put $h' = (hm/l^k) \bmod m$, next put $s_{j,i} = d_{i,h'}$ for $0 \leq i < u$ and finally replace h by $h \cdot g \bmod m$. For $j = l^k, \ldots, l^k + l^{k-1} - 1$ in succession first put $h' = (hm/l^k) \bmod m$, next put $s_{(j-zl^{k-1}),i} = s_{(j-zl^{k-1}),i} - d_{i,h'}$ for $z = 1, \ldots, l-1$ and $0 \leq i < u$ and finally replace h by $h \cdot g \bmod m$.

(b8c) Put $s_{0,i} = d_{i,0}$ for $0 \leq i < u$. Let S be the matrix having the $(s_{j,i})_{j=0}^{u-1}$ as columns, for $0 \leq i < u$.

(b9) Compute the $u \times u$ dimensional integer matrix S^* and $D \in \mathbf{Z}_{>0}$ such that $D^{-1} \cdot S^* = S^{-1}$ and D is minimal. This can efficiently be done using a variant of the Gaussian elimination method. See V.6 for more details.

Since n is known, one can perform these operations in $\mathbf{Z}/n\mathbf{Z}$, thereby reducing the length of the integers involved.

(b10) Tabulate u, $m_u = m$, $r_{u,m} = r$, $g_{u,m} = g$, $f_{u,m} = f$, $S_{u,m} = S$, $S^*_{u,m} = S^*$, and $D_{u,m} = D$.

(5.2) Selection of cyclic rings and creation of transition matrices.

Let t and v be as determined in step (4.1), and let $\operatorname{lcm}\{u_{p,\hat{k}} : p \text{ prime}, p^{\hat{k}} \parallel \operatorname{lcm}(t,v)\} = \prod_{l \text{ prime}} l^{\hat{e}_l}$. Perform steps (a) and (b) for all prime powers $\hat{u} = l^{\hat{e}_l}$.

In steps (1.3) and (5.1) cyclic extensions $\mathbf{Z}[\eta_{\hat{u}}]/n\mathbf{Z}[\eta_{\hat{u}}]$ of degree $\hat{u}$ over $\mathbf{Z}/n\mathbf{Z}$ were found; here $\eta = \eta_{\hat{u}}$ is a zero of $f_{\hat{u},m}$ and equals $\sum \zeta_m^j$ where j ranges over the powers of $g^{\hat{u}}$ modulo m, where g is a primitive root modulo m. The matrix $S_{\hat{u},m}$ converts an element expressed on the basis $(\eta^0, \eta^1, \ldots, \eta^{\hat{u}-1})$ to its representation on the basis $(\varsigma_{g,0,\hat{u}}, \varsigma_{g,1,\hat{u}}, \cdots, \varsigma_{g,\hat{u}-1,\hat{u}})$. If m_l is prime then $\varsigma_{g,k,\hat{u}}$ is defined by

$$\varsigma_{g,k,\hat{u}} = \sigma_g^k(\eta_{\hat{u}}) = \sigma_g^k(\eta),$$

for $0 \le k < \hat{u}$. If m_l is not prime then $\varsigma_{g,k\hat{u}}$ is defined by $\varsigma_{g,0,\hat{u}} = 1$ and

$$\varsigma_{g,k,\hat{u}} = \sigma_g^{(k-l^{i-1})}(\eta_{l^i})$$

for $l^{i-1} \le k < l^i$ and $i = 1, \ldots, \hat{e}_l$. Here the map σ_g acts via $\sigma_g(\zeta_m) = \zeta_m^g$. Instead of the automorphism σ_g, one needs σ_n in steps (5.3) and (5.4), where σ_n acts via $\sigma_n(\zeta_m) = \zeta_m^n$. Therefore, if $n \not\equiv g \bmod \hat{u}$, the matrix $S_{\hat{u},m}$ and its inverse are transformed in such a way that they perform the transformations with respect to the basis $(\varsigma_{n,0,\hat{u}}, \varsigma_{n,1,\hat{u}}, \cdots, \varsigma_{n,\hat{u}-1,\hat{u}})$. Finally a matrix $S^{\#}_{\hat{u},m}$ is calculated, such that for any element $x \in \mathbf{Z}[\eta]/n\mathbf{Z}[\eta]$ which is represented as a $\hat{u}$-dimensional column vector over $\mathbf{Z}/n\mathbf{Z}$, the element $\sigma_n(x)$ equals $S^{\#}_{\hat{u},m} \cdot x$.

(a) Retrieve the smallest $m_{\hat{u}} \in \mathcal{M}^+(\hat{u})$ with $m_{\hat{u}} \mid s$ from the list made in (4.2). If no such $m_{\hat{u}}$ exists, take the smallest $m_{\hat{u}}$ from the list. Put $m_l = m_{\hat{u}}$.

The set $\mathcal{M}^+(\hat{u})$ as constructed in step (4.2) contains those conductors m that satisfy the correct condition (cf. (4.2) and II.(4.8)). In step (4.1) and step (5.1) the sets $\mathcal{M}^+(\hat{u})$ are constructed in such a way that they will not be empty. By choosing m one fixes the extension of degree l^e which will be used in steps (5.3) and (5.4).

(b) Put $m = m_l$ and replace G by $(G \cdot m) \bmod n$. If $G = 0$ then n is composite and the primality test terminates. For $1 \le e \le \hat{e}_l$ put $u = l^e$ and perform steps (b1) through (b4).

If a cyclic extensions of degree l^e, $1 \le e < \hat{e}_l$ is needed, the l^e-th degree subextension of the extension of degree $l^{\hat{e}_l}$ is used for this; here $l^{\hat{e}_l}$ is the largest l-th power degree that will be used. This means in particular that the same conductor m is taken for all extensions of degree l^e with $1 \le e \le \hat{e}_l$. By construction the extension of degree u_1 is now contained in the extension of degree u_2 whenever $u_1 \mid u_2$. Furthermore it has to be checked that n and m are relatively prime.

(b1) Retrieve $r_{u,m}$, $g_{u,m}$, $f_{u,m}$, $S_{u,m}$, $S^*_{u,m}$, and $D_{u,m}$ from the table made in (1.3) and (5.1).

(b2) Replace $S^*_{u,m}$ by $D^{-1} \cdot S^*_{u,m} \bmod n$; if $D^{-1} \bmod n$ does not exist, then n is composite and the primality test is terminated.

After reducing the matrix S^* modulo n and multiplying it by $D^{-1} \bmod n$, the matrix S^* converts an element expressed in terms of $(\varsigma_{g,0,u}, \varsigma_{g,1,u} \cdots, \varsigma_{g,u-1,u})$, to its representation in terms of η^j with $0 \le j < u$.

(b3) Find $i^* \in \{1, \ldots, m-2\}$ such that $g_{u,m}^{i^*} \equiv n \bmod m$ by trying $i^* = 1, 2, \ldots$ in succession. Perform steps (b3a) and (b3b) if $i^* \neq 1$.

If $i^* \neq 1$, the matrices $S_{u,m}$ and $S_{u,m}^*$ as calculated in steps (1.3) or (5.1) are different from the matrices needed in the algorithm. In this case the rows of $S_{u,m}$ and the columns of $S_{u,m}^*$ have to be changed. This is much faster then recalculating them by using for instance step (5.1).

(b3a) Introduce a matrix $S'_{u,m} = (s'_{i,j})_{i,j=0}^{u-1} \in \mathbf{Z}^{u \times u}$ and initially put $s'_{i,j} = 0$ for $0 \leq i < u$ and $0 \leq j < u$. Perform step (b3a1) if m_l is prime and for $k = 1, \ldots, e$ perform step (b3a2) if m_l is not a prime. Finally replace $S_{u,m}$ by $S'_{u,m}$.

(b3a1) Put the matrix $S'_{u,m}$ equal to the whose j-th row equals the $(i^* \cdot j \bmod u)$-th row of $S_{u,m}$, for $0 \leq j < u$.

The matrix $S_{u,m}$, as constructed in step (1.3) or (5.1) converts an element expressed in terms of η^j, $0 \leq j < u$, to its representation in terms of $(\varsigma_{g,0,u}, \varsigma_{g,1,u}, \ldots, \varsigma_{g,u-1,u}) = (\sigma_g^0(\eta), \sigma_g^1(\eta), \ldots, \sigma_g^{u-1}(\eta))$, where the map σ_g acts via $\sigma_g(\zeta_m) = \zeta_m^g$. The map σ_n, which will be used in the sequel, acts via $\sigma_n(\zeta_m) = \zeta_m^n = \sigma_g^{i^*}(\zeta_m)$. This implies that if the matrix $S_{u,m}$ is to convert an element expressed in terms of η^j, $0 \leq j < u$, to its representation in terms of $(\varsigma_{n,0,u}, \varsigma_{n,1,u}, \ldots, \varsigma_{n,u-1,u}) = (\sigma_n^0(\eta), \sigma_n^1(\eta), \ldots, \sigma_n^{u-1}(\eta))$, which is in fact $(\sigma_g^{0 \cdot i^*}(\eta), \sigma_g^{1 \cdot i^*}(\eta), \ldots, \sigma_g^{(u-1) \cdot i^*}(\eta))$, one has to replace the matrix $S_g = S_{u,m}$ by a matrix S_n. This is done by permuting the 0-th through the $(u-1)$-nd row of S_g.

(b3a2) For $j = l^{k-1}, \ldots, l^k - 1$ first put $j' = (i^* \cdot (j - l^{k-1}) \bmod l^k) + l^{k-1}$ and next if $j' < l^k$ put the j-th row of $S'_{u,m}$ equal to the j'-th row of $S_{u,m}$; if $j' \geq l^k$ then put the j-th row of $S'_{u,m}$ equal to -1 times the sum of the $(j' - z \cdot l^{k-1})$-th rows of $S_{u,m}$ with $z = 1, \ldots, l-1$.

The matrix $S_{u,m}$ converts an element expressed in terms of η^j, $0 \leq j < u$, to its representation in terms of $(\varsigma_{g,0,u}, \varsigma_{g,1,u}, \ldots, \varsigma_{g,u-1,u})$. By changing from g to $n \bmod m_l$ the elements $\varsigma_{g,l^i-1,u}, \ldots, \varsigma_{g,l^i-1,u}$ are permuted such that $\varsigma_{n,j,u} = \varsigma_{g,j',u}$, where j' is defined above, for $j = l^{k-1}, \ldots, l^k - 1$ and $k = 1, \ldots, e$. During the permutation elements $\varsigma_{g,j',u}$ with $j' \geq l^k$ may be introduced in the basis. These can be represented by $-\sum_{z=1}^{l-1} \varsigma_{g,j'-z \cdot l^{k-1},u}$.

(b3b) Similarly as in step (b3a), the matrix $S_{u,m}^*$ will be replaced by a permuted version. Introduce a matrix $S'_{u,m} = (s'_{i,j})_{i,j=0}^{u-1} \in \mathbf{Z}^{u \times u}$ and initially put $s'_{i,j} = 0$ for $0 \leq i < u$ and $0 \leq j < u$. Perform step (b3b1) if m_l is prime and for $k = 1, \ldots, e$ perform step (b3b2) if m_l is not a prime. Finally replace $S_{u,m}^*$ by $S'_{u,m}$.

The matrix $S_{u,m}^*$, as constructed in step (5.2)(b2) converts an element expressed in terms of $(\varsigma_{g,0,u}, \varsigma_{g,1,u}, \ldots, \varsigma_{g,u-1,u})$, to its representation in terms of η^j, $0 \leq j < u$, where the map σ_g (used in the definition of $\varsigma_{g,k,u}$) acts via $\sigma_g(\zeta_m) = \zeta_m^g$. Since the matrix $S_{u,m}^*$ is to convert an element expressed in terms of $(\varsigma_{n,0,u}, \varsigma_{n,1,u}, \ldots, \varsigma_{n,u-1,u})$, to its representation in terms of η^j, $0 \leq j < u$, one has to replace the matrix $S_g^* = S_{u,m}^*$ by a matrix S_n^*. This is done in the same way as in step (b3a) by replacing the row operations by the corresponding column operations.

(b3b1) Put the matrix $S'_{u,m}$ equal to the whose j-th row equals the $(i^* \cdot j \bmod u)$-th row of $S_{u,m}$, for $0 \leq j < u$.

(b3b2) For $j = l^{k-1}, \ldots, l^k - 1$ first put $j' = (i^* \cdot (j - l^{k-1}) \bmod l^k) + l^{k-1}$ and next if $j' < l^k$ put the j-th column of $S'_{u,m}$ equal to the j'-th column of $S_{u,m}$; if $j' \geq l^k$ then put the j-th column of $S'_{u,m}$ equal to -1 times the sum of the $(j' - z \cdot l^{k-1})$-th columns of $S_{u,m}$ with $z = 1, \ldots, l-1$.

(b4) Let $S_{u,m} = (s_{i,j})_{i,j=0}^{u-1}$ and $S^*_{u,m} = (s^*_{i,j})_{i,j=0}^{u-1}$. Perform step (b4a) if m_l is prime and for $k = 1, \ldots, e$ perform step (b4b) if m_l is not prime. Finally put $S^{\#}_{u,m} = (s^{\#}_{i,j})_{i,j=0}^{u-1}$.

(b4a) Put $s^{\#}_{i,j} = \sum_{k=0}^{u-1} s^*_{i,k} \cdot s_{(k-1) \bmod u, j}$ for $0 \leq i, j < u$.

For any element $x \in \mathbf{Z}[\eta_u]/n\mathbf{Z}[\eta_u]$, which is represented as a u-dimensional column vector over $\mathbf{Z}/n\mathbf{Z}$, the element $\sigma_n(x)$ will equal $S^{\#}_{u,m} \cdot x$. In order to apply σ_n to an element in $\mathbf{Z}[\eta_u]/n\mathbf{Z}[\eta_u]$ in an easy way, the element will be transformed to its representation with respect to $(\varsigma_{n,0,u}, \varsigma_{n,1,u}, \ldots, \varsigma_{n,u-1,u})$; in this representation the mapping σ_n is simply a shift of the coordinates. By transforming the result back to the representation with respect to η^j, $0 \leq j < u$, one gets the final result. Instead of applying $S_{u,m}$, performing a shift and applying $S^*_{u,m}$, one computes the complete transformation in advance, by multiplying $S_{u,m}$ by a 'shifted' version of $S^*_{u,m}$ to get $S^{\#}_{u,m}$. This is done as follows. By multiplying the element $x = \sum_{j=0}^{u-1} x_j \eta^j$ by S, one gets its representation $(y_0, y_1, \ldots, y_{u-1})$ with respect to the basis $(\varsigma_{n,0,u}, \varsigma_{n,1,u}, \ldots, \varsigma_{n,u-1,u}) = (\sigma_n^0(\eta), \sigma_n^1(\eta), \ldots, \sigma_n^{u-1}(\eta))$. So $y_i = \sum_{j=0}^{u-1} s_{i,j} \cdot x_j$. Next σ_n is applied, giving a representation with respect to the basis $(\sigma_n^1(\eta), \sigma_n^2(\eta), \ldots, \sigma_n^{u-1}(\eta), \sigma_n^0(\eta)) = (\varsigma_{n,1,u}, \varsigma_{n,2,u}, \ldots, \varsigma_{n,u-1,u}, \varsigma_{n,0,u})$. So $\sigma_n(x) = y_{u-1} \cdot \varsigma_{n,0,u} + \sum_{j=0}^{u-2} y_j \cdot \varsigma_{n,j+1,u}$. Expressing this in terms of $(\varsigma_{n,0,u}, \varsigma_{n,1,u}, \ldots, \varsigma_{n,u-1,u})$, one gets $\sigma_n(x) = y_{u-1} \cdot \varsigma_{n,0,u} + \sum_{j=2}^{u-1} y_{j-1} \cdot \varsigma_{n,j,u}$. To represent this element on the basis η^j with $0 \leq j < u$, one has to multiply it by S^*, giving $(\sigma_n(x))_i = \sum_{j=0}^{u-1} s^*_{i,j} \cdot y_{(j-1) \bmod u}$. Substituting $y_i = \sum_{j=0}^{u-1} s_{i,j} \cdot x_j$, gives the result above.

(b4b) Put $s^{\#}_{i,j} = \sum_{k=1}^{e} \sum_{z=l^{k-1}+1}^{l^k-1} s^*_{i,z} \cdot s_{(z-1),j} - \sum_{k=1}^{e} \sum_{z=0}^{l-1} s^*_{i,l^{k-1}} \cdot s_{(l^k - z l^{k-1} - 1),j}$ for $0 \leq i, j < u$.

In this step essentially the same operations are performed as in step (b4a). The only difficult part is the representation of basis elements that are introduced in the basis. Basically, the rules described in step (b3b) are used to solve these problems.

(5.3) Calculation of cyclotomic extensions.

Let t and v be as determined in step (4.1) and introduce for all primes $p \mid \operatorname{lcm}(t, v)$ a boolean variable f_p and put f_p initially equal to *false*. Furthermore, let $\hat{k}_p$ be such that $p^{\hat{k}_p} \parallel \operatorname{lcm}(t, v)$ for all primes $p \mid \operatorname{lcm}(t, v)$. Perform steps (a) through (e) as long as there exists at least one $p \mid \operatorname{lcm}(t, v)$ with f_p equal to *false*.

In this step one generates for each prime-divisor p of $\operatorname{lcm}(t, v)$ a $p^{k(p)}$-th root of unity in an extension of degree u, where $k(p) = \max(1, o_p(t))$ and $u = \operatorname{ord}(n \bmod p^{\hat{k}_p})$. This is done by trying the $(n^u - 1)/p^{k(p)}$-th power of random elements. Every random choice has a probability $\frac{p-1}{p}$ of success if n is prime. Instead of generating a root of unity for each prime $p \mid \operatorname{lcm}(t, v)$ separately, one can try to generate roots of unity for a number of primes simultaneously. Let $\hat{u}$ be such that $\hat{u} = \max\{u_{p,\hat{k}_p} : p \mid \operatorname{lcm}(t, v)\}$ and let $\hat{p}$ be such that $u_{\hat{p},\hat{k}_{\hat{p}}} = \operatorname{ord}(n \bmod \hat{p}^{\hat{k}_{\hat{p}}}) = \hat{u}$. While generating a $\hat{p}^{k(\hat{p})}$-th root of unity, it is conceivable

that one generates $p^{k(p)}$-th roots of unity with $u_{p,\hat{k}_p} \mid \hat{u}$ at the same time. First take the product r of all prime powers $p^{k(p)} \mid \mathrm{lcm}(t,v)$ for which $u_{p,\hat{k}_p} \mid \hat{u}$. Next take a random element to the power $(n^{\hat{u}}-1)/r$, and for each p raise the result to the power $r/p^{k(p)}$. This will be a $p^{k(p)}$-th root of unity if its $p^{k(p)-1}$-th power is not equal to 1. For each such p one has a probability $\frac{p-1}{p}$ of success to find a $p^{k(p)}$-th root of unity independent of finding any other root of unity. In fact, as explained in II.(4.15), finding a p-th root of unity in an extension of degree $\mathrm{ord}(n \bmod p^{\hat{k}_p})$ in this way is sufficient for the proof that a $p^{\hat{k}_p}$-th root of unity exists in this extension; the reason for actually constructing the $p^{k(p)}$-th root of unity for primes p dividing t is that these are needed in step (5.4). For primes p dividing v (and not t), one only constructs a p-th root unity in the extension of degree $u_{p,\hat{k}_p}$ to prove the existence of a $p^{\hat{k}_p}$-th root of unity. The variable f_p indicates whether a $p^{k(p)}$-th root of unity has been found.

(a) Put $\hat{u} = \max\{u_{p,\hat{k}_p} : p \mid \mathrm{lcm}(t,v), f_p = \mathit{false}\}$. Find $\hat{p} \mid \mathrm{lcm}(t,v)$ with $f_{\hat{p}} = \mathit{false}$ and $u_{\hat{p},\hat{k}_{\hat{p}}} = \hat{u}$. Put $r = \hat{p}$ if $\hat{p} \nmid t$ and put $r = \hat{p}^{o_{\hat{p}}(t)}$ if $\hat{p} \mid t$.

For primes p not dividing t, only a p-th root of unity is constructed, which is by construction sufficient for the proof of the existence of a $p^{\hat{k}_p}$-th root of unity.

(b) For all $p \neq \hat{p}$ with $p \mid \mathrm{lcm}(t,v)$, $f_p = \mathit{false}$, and $u_{p,\hat{k}_p} \mid \hat{u}$, replace r by $r \cdot p$ if $p \nmid t$, and replace r by $r \cdot p^{o_p(t)}$ if $p \mid t$.

The variable r will be equal to the product of all primes p with $\mathrm{ord}(n \bmod p^{\hat{k}_p}) \mid \hat{u}$. Finding $p^{k(p)}$-th roots of unity, with $k(p) = \max(1, o_p(t))$, will be attempted simultaneously for all these primes.

(c) Put $a = \lfloor (n-1)/r \rfloor$ and let $c = 0$. For $i = \hat{u}-1, \hat{u}-2, \ldots, 0$ in succession perform step (c1).

In this step $n^{\hat{u}} - 1$ is divided by r. This was done by writing $n^{\hat{u}} - 1$ in base n, i.e., $n^{\hat{u}} - 1 = \sum_{i=0}^{\hat{u}-1} b_i \cdot n^i$ and next sequentially dividing all coefficient b_i by r and keeping track of a carry c. The b_i now satisfy $\sum_{i=0}^{\hat{u}-1} b_i \cdot n^i = (n^{\hat{u}}-1)/r$. In order to raise a random element to the power $(n^{\hat{u}}-1)/r$, the base-n representation of this exponent is used, since one can replace the single large exponentiation by a few smaller exponentiations with exponents b_i and a few applications of σ_n (which can be done by applying the matrices $S^{\#}$, computed in step (5.2)).

(c1) First put $b_i = \lfloor (c \cdot n + n - 1)/r \rfloor$, and next replace c by $(c \cdot n + n - 1) \bmod r$. Finally put $b_{1,i} = \lfloor b_i/r \rfloor$ and $b_{2,i} = b_i \bmod r$.

In this step each element b_i of the base-n representation of $n^{\hat{u}} - 1$ is written as $b_{1,i} \cdot a + b_{2,1}$, where $a = \lfloor (n-1)/r \rfloor$. In this way one can replace each exponentiation with exponent b_i by two exponentiations with relatively small exponents $b_{1,i}$ and $b_{2,i}$ and by one common exponentiation with exponent a. For more information on this, see V.(4.3).

(d) For all prime powers $\bar{u} = l^{\bar{e}} \parallel \hat{u}$ let $\eta_{\bar{u}}$ be a zero of $f_{\bar{u},m_l}$, with m_l as in (5.2)(a) and $f_{\bar{u},m_l}$ retrieved from the tables as in (5.2)(b1). Perform steps (d1) through (d4).

In this step a random element α will be taken to the power $(n^{\hat{u}}-1)/r$. Furthermore it will be checked that $\alpha^n = \sigma_n(\alpha)$ which should be the case if n is prime (see II.(2.3)).

(d1) Let α be a non-zero random multivariate polynomial in all the $\eta_{\bar{u}}$'s over $\mathbf{Z}/n\mathbf{Z}$.

Although any non-zero choice is allowed in this step, one can take $\alpha = \sum_{\bar{u}} \eta_{\bar{u}}$ as a possible first choice.

(d2) Put $\beta = \alpha$ and perform step (d2a) for all prime powers $\bar{u} = l^{\bar{e}} \parallel \hat{u}$.

If n is prime, the mapping σ_n is equal to n-th powering (cf. II.(2.3)). In this step one calculates $\sigma_n(\alpha)$, where α is the random element chosen in step (d1).

(d2a) Write β as a $\bar{u}$-dimensional column vector, i.e., write β as a polynomial in $\eta_{\bar{u}}$. Replace β by $S^{\#}_{\bar{u},m_l} \cdot \beta$ (cf. (5.2)(b4)).

For any element $x \in \mathbf{Z}[\eta]/n\mathbf{Z}[\eta]$, which is written as a $\bar{u}$-dimensional column vector over $\mathbf{Z}/n\mathbf{Z}$, the element $\sigma_n(x)$ equals $S^{\#}_{\bar{u},m_l} \cdot x$. In this step one calculates $\sigma_n(\alpha)$, where α is the random element chosen in step (d1).

(d3) Check that $\alpha^n = \beta$. If equality does not hold, then n is composite and the primality test is terminated.

If σ_n does not give the same result as n-th powering, the number n cannot be prime.

(d4) Put $\beta = \alpha^a$, $i = \hat{u} - 1$ and $\gamma = \alpha^{b_i}$. For $i = \hat{u} - 2, \hat{u} - 3, \ldots, 0$ in succession replace γ by $\sigma_n(\gamma) \cdot \beta^{b_{1,i}} \alpha^{b_{2,i}}$, where $\sigma_n(\gamma)$ is computed using the matrices $S^{\#}$.

In this step one calculates $\gamma = \alpha^{(n^{\hat{u}}-1)/r} = \prod_{i=0}^{\hat{u}-1} \alpha^{n^i \cdot b_i} = \prod_{i=0}^{\hat{u}-1} \alpha^{n^i \cdot (b_{1,i} \cdot a + b_{2,i})}$ by means of a Horner-scheme. Instead of raising an element to the power n, one uses the image under σ_n which is now known to give the same result for powers of α, at least.

It is possible to combine the exponentiations in this step with the exponentiation in step (d3), to speed up this step of the algorithm. Basically the exponentiations are done in the way as described in [29, Remark (3.6)].

(e) For all primes $p \mid r$ perform steps (e1) and (e2).

In this step, the individual $p^{k(p)}$-th roots are extracted from the result of the huge exponentiation in step (d), where $k(p) = \max(1, o_p(t))$. In fact the result γ of step (d) is taken to the power r/p. If the result of the exponentiation is unequal to 1, then this attempt to find a $p^{k(p)}$-th root of unity has been successful. Otherwise another attempt to find a root of unity should be made.

(e1) If $p \nmid t$ perform step (e1a) and if $p \mid t$ perform steps (e1b) and (e1c).

(e1a) Put $\gamma_{p,1} = \gamma^{r/p}$.

(e1b) First put $k(p) = \max(1, o_p(t))$, put $\gamma^* = \gamma^{r/p^{k(p)}}$. Next put $\gamma_{p,k} = \gamma^*$. For all prime powers $\bar{u} = l^{\bar{e}} \parallel \hat{u}$ perform step (e1b1) if $\bar{u}$ does not divide $u_{p,k(p)}$.

The element γ^* is constructed as an element in an extension of degree $\hat{u}$ over $\mathbf{Z}/n\mathbf{Z}$. It can however also be represented as an element in an extension of degree $u_{p,k(p)}$ over $\mathbf{Z}/n\mathbf{Z}$. For each prime $l \mid \hat{u}$ with $o_l(u_{p,k(p)}) < o_l(\hat{u})$, let $u = l^e \parallel u_{p,k(p)}$. The element γ^* is written as a $\bar{u}$-dimensional column vector, i.e., as a polynomial in $\eta_{\bar{u}}$. By applying $S_{\bar{u},m_l}$ to this vector, one writes γ^* as a combination of $(\varsigma_{n,0,\bar{u}}), \varsigma_{n,1,\bar{u}}), \ldots, \varsigma_{n,u-1,\bar{u}}))$. Since γ^* is known to live in a subextension, this representation has a repetitive character if m_l (cf. (2.3)) is prime. If m_l is not prime, then only the first u coordinates of this representation are non-zero. By taking only the first u coefficients and applying S^*_{u,m_l} to these, one gets the representation of γ^* as a polynomial in η_u, both in the case that m_l is prime and in the case that m_l is not prime.

(e1b1) Let e be such that $l^e \parallel u_{p,k(p)}$, and let $u = l^e$. Represent γ^* as a $\bar{u}$-dimensional column vector, i.e., write γ^* as a polynomial in $\eta_{\bar{u}}$. Apply S^*_{u,m_l} to the vector consisting of the first u coordinates of the vector $S_{\bar{u},m_l} \cdot \gamma^*$, and replace γ^* by the result (cf. (5.2)(b3a) and (5.2)(b3b)). Put $\gamma_{p,k(p)} = \gamma^*$.

(e1c) For $i = k(p) - 1, k(p) - 2, \ldots, 1$ in succession first compute $\gamma_{p,i} = \gamma^p_{p,i+1}$ and next perform step (e1c1) for those prime powers $\bar{u} = l^{\bar{e}} \parallel u_{p,i+1}$ such that $\bar{u}$ does not divide $u_{p,i}$.

The elements $\gamma_{p,j}$, for $j = 1, 2, \ldots, k(p)$, will be the p^j-th roots of unity if n is prime, which will be used in step (5.4).

(e1c1) Let e be such that $l^e \parallel u_{p,i}$, and let $u = l^e$. Represent γ^* as a $\bar{u}$-dimensional column vector, i.e., write γ^* as a polynomial in $\eta_{\bar{u}}$. Apply S^*_{u,m_l} to the vector consisting of the first u coordinates of the vector $S_{\bar{u},m_l} \cdot \gamma^*$, and replace γ^* by the result (cf. (5.2)(b3a) and (5.2)(b3b)). Put $\gamma_{p,i} = \gamma^*$.

This step will only be performed for those prime powers $\bar{u}$ which are not divisors of $u_{p,i}$; in particular if $u_{p,i+1} = u_{p,i}$ this step is skipped. This is in fact the same transformation as performed in step (e1b). Explanation can be found there.

(e2) Put $\delta = \gamma_{p,1} - 1$. If $\delta \neq 0$, select any non-zero coefficients a of α and d of δ, and replace G by $(G \cdot a \cdot d) \bmod n$. If $G = 0$ a factor of n can easily be derived and the primality test is terminated. Otherwise, if $G \neq 0$, put $f_p = true$.

If δ is not equal to zero, the $\gamma_{p,k}$ are really p^k-th roots of unity.

(5.4) Jacobi sum tests.

Let $\mathbf{S}_\mathcal{T}$ be as constructed in step (4.1). Perform steps (a) through (d) for all $\mathcal{S} \in \mathbf{S}_\mathcal{T}$.

Each set $\mathcal{S} \in \mathbf{S}_\mathcal{T}$ contains a set of triples (q, p, h) for which a combined Jacobi sum test will be performed. Each triple (q, p, h) represents a Jacobi sum test for a character χ of order p^k, where $k = o_p(q-1)$, and conductor q in an extension of degree $u_{p,1} \cdot p^h$. The sets $\mathcal{S} \in \mathbf{S}_\mathcal{T}$ have been determined in step (4.1).

In this step one proves that

$$\prod_{(q,p,h)\in\mathcal{S}} \tau(\chi_{p,q})^n / \tau(\chi^n_{p,q})$$

is a power of an r-th root of unity, where r is defined by

$$r = \prod_{(q,p,h)\in\mathcal{S}} p^{o_p(q-1)}$$

and $\chi_{p,q}$ is a character of order p^k and of conductor q. This is done by expressing $\tau(\chi_{p,q})^{n-\sigma_n}$ in terms of Jacobi sums, for all $(q, p, h) \in \mathcal{S}$ and showing that the product of all $\tau(\chi_{p,q})^{n-\sigma_n}$ is equal to a power of an r-th root of unity. The Jacobi sums, and their exponents have been calculated in advance in step (1.4).

(a) Let $u^* = \text{lcm}\{u_{p,1} \cdot p^h : (q,p,h) \in \mathcal{S}\} = \prod_{l \text{ prime}} l^{\tilde{e}_l}$. For all prime powers $u = l^e$, with $1 \leq e \leq \tilde{e}_l$, let η_u denote a zero of f_{u,m_l}; here m_l is as in step (5.2)(a), and f_{u,m_l} is as in (5.2)(b1).

By construction, u^* will be one of the $u_{p,1} \cdot p^h$ for some $(q,p,h) \in \mathcal{S}$.

(b) Let $r = \prod_{(q,p,h)\in\mathcal{S}} p^{o_p(q-1)}$. Let $c = \lfloor n/r \rfloor$ and $n_r = n - r \cdot c$. Set α, β, and γ equal to 1; these should be regarded as multivariate polynomials in all the $\eta_{\tilde{u}}$'s over $\mathbf{Z}/n\mathbf{Z}$, for $\tilde{u} = l^{\tilde{e}_l} \parallel u^*$.

For each element $(q,p,h) \in \mathcal{S}$ a Jacobi test for a character χ of order p^k and conductor q in an extension of degree $u_{p,1} \cdot p^h$ should be performed. These tests involve a large exponentiation in an extension of degree $u_{p,1} \cdot p^h$. Since for all elements $(p,q,h) \in \mathcal{S}$ about the same exponentiation has to be carried out, the common part of the exponentiations is done simultaneously, to save time. The exponent of this common part is equal to c.

(c) Perform steps (c1) through (c5) for all triples (q,p,k) such that $(q,p,h) \in \mathcal{S}$ and $k = o_p(q-1)$.

In this step the elements needed for the final huge exponentiation are calculated. These elements are

$$\alpha = \prod_{p|r} \tau(\chi_{p,q})^{n_r - \sigma_{n_p}},$$

and

$$\beta = \prod_{p|r} \tau(\chi_{p,q})^{r},$$

respectively, where $n_p \equiv n_r \bmod p^k$ and $0 \leq n_p < p^k$.

(c1) Let $n_p \equiv n_r \bmod p^k$ and $0 \leq n_p < p^k$. For every $J \in \mathcal{J}_{p^k}$ retrieve e_{π,p^k,n_p} and e_{π,p^k,p^k}, which were tabulated in step (2.4)(f).

For those $J \in \mathcal{J}_{p^k}$ for which $e_{\pi,p^k,n_p} \neq 0$ or $e_{\pi,p^k,p^k} \neq 0$ retrieve $J \in \mathbf{Z}[\zeta_{p^k}]$ from the direct access file created in (2.4) and transform these J to $\mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$ by taking their coefficients modulo n.

Compute

$$J^* = \prod_{\substack{J\in\mathcal{J}_{p^k} \\ e_{\pi,p^k,n_p}\neq 0}} J^{e_{\pi,p^k,n_p}} \in \mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$$

and

$$J^{\#} = \chi(-1) \cdot q \cdot \prod_{\substack{J\in\mathcal{J}_{p^k} \\ e_{\pi,p^k,p^k}\neq 0}} J^{e_{\pi,p^k,p^k}} \in \mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}],$$

where $\chi(-1) = -1$ if $q \equiv 3 \bmod 4$ and p even, and $\chi(-1) = 1$ if $q \equiv 1 \bmod 4$ or p odd.

In this step the elements $J^* = \tau(\chi)^{n_p - \sigma_{n_p}}$ and $J^\# = \tau(\chi)^{p^k}$ are calculated as elements of $\mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$.

(c2) Let $J^* = (a_i^*)_{0 \le i < \phi(p^k)}$ and $J^\# = (a_i^\#)_{0 \le i < \phi(p^k)}$. Put $\gamma^* = \gamma_{p,k}$, where $\gamma_{p,k}$ is as in (5.3)(e1).

The $\gamma_{p,k}$ are p^k-th roots of unity, calculated in step (5.3)(e1).

Put $i = \phi(p^k) - 1$, and put $J_\alpha = a_i^*$ and $J_\beta = a_i^\#$. For $i = \phi(p^k) - 2, \phi(p^k) - 3, \ldots, 0$ in succession, replace J_α by $J_\alpha \cdot \gamma^* + a_i^*$ and J_β by $J_\beta \cdot \gamma^* + a_i^\#$.

The elements $J^\#$ and J^* have to be transformed to elements in an extension of degree $u_{p,k}$ of $\mathbf{Z}/n\mathbf{Z}$. This is done by performing a Horner-scheme, substituting γ^* for ζ_{p^k}. The resulting expressions, J_α and J_β are elements in an extension of degree $u_{p,k}$ of $\mathbf{Z}/n\mathbf{Z}$, where J_α equals $\tau(\chi)^{n_p - \sigma_{n_p}}$ and J_β equals $\tau(\chi)^{p^k}$.

(c3) Put $\alpha^* = J_\alpha \cdot (J_\beta)^{\lfloor n_r/p^k \rfloor}$, and $\beta^* = (J_\beta)^{r/p^k}$.

The element α^* equals $\tau(\chi)^{n_p - \sigma_{n_p} + n_r - n_p} = \tau(\chi)^{n_r - \sigma_{n_p}}$ and β^* equals $\tau(\chi)^r$.

(c4) For all $\tilde{u} = l^{\tilde{e}_l} \parallel u^*$ perform steps (c4a) and (c4b) if $\tilde{u}$ does not divide $u_{p,k}$ and if l divides $u_{p,k}$.

In this step the elements α^* and β^*, which are elements in an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree $u_{p,k}$ will be transformed to elements in a possibly larger extension of $\mathbf{Z}/n\mathbf{Z}$ of degree u^*. The huge exponentiation, mentioned in (4.4)(a), will be performed in the extension of degree u^*. The transformation will be done in the following way. For all prime powers $\tilde{u} = l^{\tilde{e}_l} \parallel u^*$ with $\tilde{u} \nmid u_{p,k}$ and $l \mid u_{p,k}$ let $u = l^e \parallel u_{p,k}$. First the element η_u is lifted to the extension of degree $\tilde{u}$. If m_l is prime then

$$\eta_u = \sum_{\substack{0 \le i < \tilde{u} \\ i \equiv 1 \bmod u}} \sigma_n^i(\eta_{\tilde{u}}) = \sum_{\substack{0 \le i < \tilde{u} \\ i \equiv 1 \bmod u}} \varsigma_{n,i,\tilde{u}} \cdot$$

In this case one simply has to apply $S^*_{\tilde{u},m_l}$ to a $\tilde{u}$-dimensional vector with only elements equal to 1 at the coordinates with index equivalent to 1 mod u. All other coefficients will be equal to 0. If m_l is not prime then

$$\eta_u = \sigma_n^0(\eta_u) = \varsigma_{n,u/l,u} = \varsigma_{n,u/l,\tilde{u}} \cdot$$

In this case one simply has to apply $S^*_{\tilde{u},m_l}$ to a $\tilde{u}$-dimensional vector with only elements equal to 1 at the coordinate $i = u/l$. All other coefficients will be equal to 0. Next α^* and β^* are written as polynomials of $\eta_{\tilde{u}}$ by applying a Horner-scheme.

(c4a) Let $e \ge 1$ be such that $l^e \parallel u_{p,k}$, and let $u = l^e$. If m_l is prime let $y = (y_i)_{i=0}^{\tilde{u}-1}$ be such that $y_i = 1$ if $i \equiv 1 \bmod u$ and $y_i = 0$ otherwise. If m_l is not prime let $y = (y_i)_{i=0}^{\tilde{u}-1}$ be such that $y_i = 1$ if $i = u/l$ and $y_i = 0$ otherwise. Replace y by $S^*_{\tilde{u},m_l} \cdot y$ (cf. (4.2)(b3)).

(c4b) For $\delta^* = \alpha^*, \beta^*$ do the following. First, write δ^* as a u-dimensional vector $(\delta_i^*)_{i=0}^{u-1}$, i.e., write δ^* as a polynomial in η_u. Next, put $\delta = \delta_{u-1}^*$, and for $i = u-2, u-3, \ldots, 0$ in succession, replace δ by $\delta \cdot y + \delta_i^*$. Finally, replace δ^* by δ.

(c5) The elements α^* and β^* are now elements in an extension of degree u^* of $\mathbf{Z}/n\mathbf{Z}$. Replace α and β by $\alpha \cdot \alpha^*$ and $\beta \cdot \beta^*$, respectively.

The α^* and β^* now satisfy

$$\alpha^* = \prod_{p|r} \tau(\chi_{p,q})^{n_r - \sigma_{n_p}},$$

and

$$\beta^* = \prod_{p|r} \tau(\chi_{p,q})^r,$$

where $\chi_{p,q}$ is the character of order p^{k_p} and conductor q such that $(q,p,h) \in \mathcal{S}$, with $h = o_p(u_{p,k_p})$.

(d) Compute $\delta = \alpha \cdot \beta^c$ (cf. (b)). Perform steps (d1) and (d2) for all $p^k \parallel r$.

The element δ satisfies

$$\delta = \prod_{p|r} \tau(\chi_{p,q})^{r\cdot c + n_r - \sigma_{n_p}} = \prod_{p|r} \tau(\chi_{p,q})^{n - \sigma_{n_p}} = \prod_{p|r} \tau(\chi_{p,q})^{n-\sigma_n}.$$

In this step one has to check that δ is a power of an r-th root of unity.

(d1) Compute $\delta^* = \delta^{r/p^k}$ and next perform step (d1a) for those prime powers $\bar{u} = l^{\bar{e}} \parallel u^*$ such that $\bar{u}$ does not divide $u_{p,k}$.

(d1a) Let e be such that $l^e \parallel u_{p,k}$, and let $u = l^e$. Represent δ^* as a $\bar{u}$-dimensional column vector, i.e., write δ^* as a polynomial in $\eta_{\bar{u}}$. Apply S^*_{u,m_l} to the vector consisting of the first u coordinates of the vector $S_{\bar{u},m_l} \cdot \delta^*$, and replace δ^* by the result (cf. (5.2)(b3a) and (5.2)(b3b)). Put $\bar{\delta} = \delta^*$.

This step will only be performed for those prime powers $\bar{u}$ which are not divisors of $u_{p,k}$; in particular if $u^* = u_{p,k}$ this step is skipped. This is in fact the same transformation as performed in step (5.3)(e1b). Explanation can be found there.

(d2) Put $\bar{\gamma} = \gamma_{p,k}$, where $\gamma_{p,k}$ is as in (5.3)(e1). Check that $\bar{\delta} = \bar{\gamma}^i$ for some $i \in \{0,1,\ldots,p^k - 1\}$; if such an integer i does not exist, then n is composite and the primality test is terminated.

By taking δ to the power r/p^k, one kills all other $\tau(\chi)^{n-\sigma_n}$ in δ, for $\chi \neq \chi_{p,q}$, since these will then be units in $\mathbf{Z}/n\mathbf{Z}$. The exponent j in $\tau(\chi)^{n-\sigma_n} = \gamma_{p,k}^j$ will then be equal to $j = i \cdot (r/p^k)^{-1} \bmod p^k$.

6. FINAL TRIAL DIVISIONS.

Let s, t, u, v, and w be as determined in step (4.1). Let $s_1 = s/\prod_{p|t} p^{o_p(s)}$ and $t_2 = \prod_{p|t} p^{o_p(n^u-1)}$ (cf. (3.4)). Put $\bar{s} = \text{lcm}(s_1 \cdot t_2, v)$, and $\bar{t} = \text{lcm}(t, u, w)$. Check that $\gcd(G, n) = 1$. If this does not hold, then n is composite and the primality test is terminated. If $\mu = \frac{1}{2}$ perform step (a), and if $\mu = \frac{1}{3}$ perform step (b).

If the number n passed all tests described in Sections 3–5, one can prove that all divisors of n that are congruent to $n^i \bmod \bar{s}$ for $i \in \{1, 2, \ldots, \bar{t}\}$, as shown in II.7. The chance that n passed all tests in steps 3–5 without being prime is practically zero. Therefore checking the remaining possibilities for the divisors of n will usually not yield any non-trivial factor of n, but is needed to complete the proof of the primality of n.

All gcd-operations, which were necessary in the course of the algorithm, are done simultaneously, by performing only one gcd. The result of this gcd cannot be 0, since every time one changes G, it is checked that $G \bmod n \neq 0$. So, if $\gcd(G, n) \neq 1$ (which is not very likely), then one is able to find a non-trivial divisor of n.

In the optimization part of the algorithm, it is determined which type of final trial division will be performed in this section. If $\mu = \frac{1}{2}$, the algorithm will examine all numbers $r \equiv n^i \bmod \bar{s}$, for $i = 1, \ldots, \bar{t}$. If $\mu = \frac{1}{3}$, the algorithm presented in [88] will be used to find all divisors in the residue classes $r \equiv n^i \bmod \bar{s}$, for $i = 1, \ldots, \bar{t}$, (cf. II.9).

(a) Let $\bar{s}$ and $\bar{t}$ be as above. Put $\tilde{n} = n \bmod \bar{s}$ with $1 \leq \tilde{n} < \bar{s}$ and let $r = \tilde{n}$. Repeat step (a1) until $\tilde{n} = 1$ but at most $\bar{t}$ times.

In this way a divisor which is at most $\sqrt{n}$, if it exists, will be found. Since $\text{ord}(n \bmod s_1)$ divides t, $\text{ord}(n \bmod t_2) = u$, $\text{ord}(n \bmod v) = w$, and $\text{lcm}(s_1, t_2, v) = \text{lcm}(s_1 \cdot t_2, v) = \bar{s}$, it follows that $\text{ord}(n \bmod \bar{s})$ divides $\text{lcm}(t, u, w) = \bar{t}$, and that step (a1) will be performed at most $\bar{t}$ times.

(a1) If $r = 1$, then n is prime and the primality test is terminated. Otherwise, if $r \leq \sqrt{n}$, check if $r \mid n$; if so, then n is composite and the primality test is terminated. Replace r by $(\tilde{n} \cdot r) \bmod \bar{s}$ in such a way that the new value of r satisfies $0 \leq r < \bar{s}$.

If n is composite, at least one divisor does not exceed $\sqrt{n}$. So in order to find a divisor in this step, one only has to check those values r, which do not exceed $\sqrt{n}$. This observation speeds up this step considerably. The value of r is probably of the same magnitude as $\bar{s}$. Therefore it probably does not make sense to incorporate the value of r in G; otherwise one has to reduce G modulo n each time it is multiplied by a value of r.

(b) Let $\bar{s}$ and $\bar{t}$ be as above. Put $\tilde{n} = n \bmod \bar{s}$ and put $\bar{n} = n^{-1} \bmod \bar{s}$ with $1 \leq \tilde{n}, \bar{n} < \bar{s}$ and let $r = \tilde{n}$, $r^* = \bar{n}$ and $r' = 1$.

Find the first K odd primes that do not divide $\bar{s}$ and group them into products m_j, such that $m_j < \sqrt{M}/3$, where M is the maximal representable single precision integer. Let k denote the number of products m_j, and $h = \prod_{j=1}^{k} m_j$. Next, for each product m_j, put $f_{j,l} = 0$ if l is a square modulo m_j, and put $f_{j,l} = 1$ if l is not a square modulo m_j, for $l = 0, \ldots, m_j - 1$, and put $n_j = n \bmod m_j$ and

$\bar{s}_j = \bar{s} \bmod m_j$, for $j = 1, \ldots, k$.

For $l = 1, \ldots, \lfloor \bar{t}/2 \rfloor$ perform steps (b1) through (b3) until $\bar{n} = 1$.

The most expensive part of the final trial division using the method presented by [88] is solving a system of two equations in two variables. Solving such a system of equations can be reduced to solving a single quadratic equation in one variable. Therefore finding a solution can essentially be done by applying Newton's method. Before applying Newton's method on a number, it is cheaper to first determine if the number is a perfect square modulo a few small primes. To determine this, a table of all squares modulo these primes, or in fact modulo a product of these primes will be determined.

A divisor which is congruent to $r \equiv n^l \bmod \bar{s}$, if it exists, will be found. A divisor which is congruent to r modulo $\bar{s}$ also implies that there exists a divisor congruent to $r' \equiv r^* \cdot n \bmod \bar{s}$. Since $\mathrm{ord}(n \bmod s_1)$ divides t, $\mathrm{ord}(n \bmod t_2) = u$, $\mathrm{ord}(n \bmod v) = w$, and $\mathrm{lcm}(s_1, t_2, v) = \mathrm{lcm}(s_1 \cdot t_2, v) = \bar{s}$, it follows that $\mathrm{ord}(n \bmod \bar{s})$ divides $\mathrm{lcm}(t, u, w) = \bar{t}$. Secondly, since $r \equiv n^l \bmod \bar{s}$ and $r' \equiv n^{\bar{t}+1-l} \bmod \bar{s}$, that divisor would also be found in the $(\bar{t} + 1 - l)$-th step. Therefore we have that the (non-)existence of divisors congruent to $n^l \bmod \bar{s}$ for $l = 1, \ldots, \bar{t}/2$ implies the (non-)existence of divisors congruent to $n^l \bmod \bar{s}$ for $l = 1, \ldots, \bar{t}$.

(b1) Put $r_j = r \bmod m_j$ and $r'_j = r' \bmod m_j$, for $j = 1, \ldots, k$, by first reducing r and r' modulo h and next reducing its result modulo all the m_j, for $j = 1, \ldots, k$.

Since one does not expect to find any divisors, the expensive step of finding the actual divisors is preceded by a step to check whether it is plausible that n has any divisors. Therefore all operations first will be performed modulo small products of primes.

(b2) Put $a_0 = \bar{s}$, $b_0 = 0$ and $c_0 = 0$, $a_1 = r' \cdot r^* \bmod \bar{s}$ with $0 < a_1 \leq \bar{s}$, $b_1 = 1$ and $c_1 = ((n - r \cdot r')/\bar{s}) \cdot r^* \bmod \bar{s}$, with $0 \leq c_1 < \bar{s}$. Next put $a_{i,j} = a_i \bmod m_j$, $b_{i,j} = b_i \bmod m_j$ and $c_{i,j} = c_i \bmod m_j$ for $j = 1, \ldots, k$ and $i = 0, 1$.

Perform step (b2d) for $i = 0$ and $i = 1$, and steps (b2a) through (b2d) until $a_i = 0$.

In this step we will perform the Euclidean-like algorithm to find the divisors that are congruent $r = n^l \bmod \bar{s}$ as described in [88]. This comes down to solving for each triple (a_i, b_i, c_i) the system of equations

$$a_i \cdot x + b_i \cdot y = c_i$$
$$(x \cdot \bar{s} + r) \cdot (y \cdot \bar{s} + r') = n.$$

(b2a) Put $q = \lfloor a_{i-2}/a_{i-1} \rfloor$ and $a_i = a_{i-2} - q \cdot a_{i-1}$. If $a_i = 0$ and i is odd, replace a_i by $a_i + a_{i-1}$ and q by $q - 1$. Next put $b_i = b_{i-2} - q \cdot b_{i-1}$ and $c_i = c_{i-2} - q \cdot c_{i-1}$. Put $\bar{q} = \lfloor c_i/\bar{s} \rfloor$ and replace c_i by $c_i - \bar{q}\bar{s}$. If $c_i < 0$ then replace c_i by $c_i + \bar{s}$ and $\bar{q}$ by $\bar{q} - 1$.

(b2b) If $q < \sqrt{M}/3$ put $q_j = q \bmod m_j$ for $j = 1, \ldots, k$. Otherwise put $q_j = q$ for $j = 1, \ldots, k$. If $\bar{q} < \sqrt{M}/3$ put $\bar{q}_j = \bar{q} \bmod m_j$ for $j = 1, \ldots, k$. Otherwise put $\bar{q}_j = \bar{q}$ for $j = 1, \ldots, k$.

If q and $\bar{q}$ are less than $\sqrt{M}/3$, all operations modulo the m_j can be done without reducing q and $\bar{q}$ modulo the m_j.

(b2c) For $j = 1, \ldots, k$, put $a_{i,j} = (a_{i-2,j} - q_j \cdot a_{i-1,j}) \bmod m_j$, put $b_{i,j} = (b_{i-2,j} - q_j \cdot b_{i-1,j}) \bmod m_j$, and put $c_{i,j} = (c_{i-2,j} - q_j \cdot c_{i-1,j} - \bar{q}_j \cdot \bar{s}_j) \bmod m_j$.

This step performs the same operations as in step (b2a) modulo the m_j.

(b2d) If i is odd, perform steps (b2d1) and (b2d2) for $z = 0$ and $z = 1$.

If i is even and $c_i = 0$ perform steps (b2d1) and (b2d2) for $z = 0$ and in the case that i is even and $c_i > 0$ perform steps (b2d1) and (b2d2) for $z = 0$ and $z = -1$.

In this step all possible values for c will be checked modulo the m_j, for $j = 1, \ldots, k$. In fact, for i odd only values for c with $2a_i \cdot b_i \leq c_i \leq n^2/s + a_i \cdot b_i$ are possible. This implies that there is at most one value in this range. It seems to be faster to check first if a c possibly gives rise to a solution and next to check whether c is in the correct range than vice versa.

(b2d1) For $j = 1, \ldots, k$ calculate $d_{1,j} = ((c_{i,j} + z \cdot \bar{s}_j) \cdot \bar{s}_j + a_{i,j} r_j + b_{i,j} r'_j) \bmod m_j$ and $d_{2,j} = a_{i,j} b_{i,j} \bmod m_j$, put $e_j = (d_{1,j}^2 - 4 \cdot d_{2,j} \cdot n) \bmod m_j$, and check if $f_{j,e_j} = 0$. If $f_{j,e_j} \neq 0$ for some $j \leq k$ then terminate step (b2d) for this value of z.

In this step it is checked whether the system of two equations gives rise to a solution. This is done by checking if $(c \cdot \bar{s} + a_i \cdot r + b_i \cdot r')^2 - 4a_i \cdot b_i \cdot n$ is a square modulo all m_j, for $j = 1, \ldots, k$. Here c is equal to $c_i + z \cdot \bar{s}$. If the expression is not a square then no solution can be found in this step.

(b2d2) Calculate $d_2 = a_i \cdot b_i$. If i is even or if both i is odd and $2 \cdot d_2 \leq c + z \cdot \bar{s} \leq n^2/\bar{s} + d_2$, try to find a positive integer e such that $e^2 = d_1^2 - 4 \cdot d_2 \cdot n$, with $d_1 = (d_i + z \cdot \bar{s}) \cdot \bar{s} + a_i r + b_i r'$. If such an integer exists, check whether $(d_1 + e)/(2a_i)$ or $(d_1 - e)/(2a_i)$ are non-trivial divisors of n. If one of these possibilities is a non-trivial divisor of n, then n is composite and the primality test is terminated.

In this step an attempt to find an actual candidate for a divisor of n is made. This is done by solving the system of equations. By identifying u with $a_i \cdot (x \cdot \bar{s} + r)$ and v with $b_i \cdot (y \cdot \bar{s} + r')$, solving the system of equations comes down to finding values for u or v which are the positive roots of the quadratic polynomial $X^2 - (c \cdot \bar{s} + a_i \cdot r + b_i \cdot r')X + a_i \cdot b_i \cdot n$.

(b3) Replace r by $r \cdot \tilde{n} \bmod \bar{s}$, r' by r^*, and r^* by $r^* \cdot \tilde{n} \bmod \bar{s}$, with $1 \leq r, r^* < \bar{s}$.

V. ANALYSIS.

1. PRELIMINARIES.

(1.1) Introduction. In this chapter we will analyze the complexity bounds of the algorithm described in the previous chapters. The heuristics and methods needed to obtain these complexity bounds, will also be presented here, at least if they have not yet been presented in the description of the algorithm itself.

Although the optimization part of the algorithm is a vital part of the algorithm, we will not analyze the complexity bounds and heuristics concerning this part of the algorithm here, since these have been discussed in detail in Chapter III.

In Section 2 we will analyze the size of the parameters. These parameters include the parameters s_0, t_0, and u_0 as well as the parameters s, t, u, v, and w, as chosen in the optimization part of the algorithm.

In Section 3 we will discuss the complexity bounds for the first part of the algorithm, the generation of all tables. These tables will be used for each primality test of integers n less than or equal to some bound N, where the value of N will be determined in advance. The complexity bounds for the generation of the tables will be expressed in terms of parameters s_0, t_0, and u_0. By expressing these parameters in terms of N, which will be done in Section 2, completes the expression of the complexity bounds for the generation of these tables as functions of N.

In all other parts of the algorithm, one is able to express the complexity bounds of the methods in terms of the parameters s, t, u, v, and w, chosen in the optimization part of the algorithm. This will be done in Section 4. The parameters used in this section depend on n, the number that is subjected to the primality test. To complete that part of the analysis, the complexity bounds as functions of n will be given in Section 2.

In Section 5 we will give an indication about the structure of the proof given by the algorithm, and in the last section we will discuss the analysis of inverting an integer matrix. The results of Section 6 will be used in Section 3.

In order to analyze the complexity bounds of the algorithm in the next sections we will use some notations, which will be defined below. Two of them were already introduced earlier (cf. I.(2.1), I.(2.7)), but for the sake of completeness, they will also be defined here.

(1.2) Definitions.

Definition 1. For functions f and g the symbol O in $f(x) = O(g(x))$ is used if there exists a positive constant c such that the inequality

$$|f(x)| \leq c \cdot g(x)$$

holds for all x.

Definition 2. For functions f and g the symbol Ω in $f(x) = \Omega(g(x))$ is used if there exists a positive constant c such that the inequality

$$|f(x)| \geq c \cdot g(x)$$

holds for all x.

Definition 3. For functions f and g the symbol Θ in $f(x) = \Theta(g(x))$ is used if there exist positive constants c_1, c_2 such that the inequality

$$c_1 \cdot g(x) \leq |f(x)| \leq c_2 \cdot g(x)$$

holds for all x.

Definition 4. For functions f and g the symbol $\sim$ in $f(x) \sim g(x)$ is used if

$$\lim_{x \to \infty} f(x)/g(x) = 1$$

holds.

Definition 5. For functions f and g the symbol o in $f(x) = o(g(x))$ is used if

$$\lim_{x \to \infty} f(x)/g(x) = 0$$

holds.

(1.3) Notation.

In a number of cases, we will have to use complexity bounds for elementary operations on ordinary integers, or elements in extensions of $\mathbf{Z}/n\mathbf{Z}$. In the case of multiplication (and division) of integers the number of elementary bit operations depends on whether or not we employ the fast multiplication techniques as for instance introduced by Schönhage and Strassen (cf. [140]). As explained in for instance [64, pp. 278-301], the time needed to multiply two elements of size B is equal to $O(B^2)$ if we use naïve multiplication techniques (as we usually do) and $O(B^{1+\epsilon})$, for any $\epsilon > 0$, if we employ the fast multiplication techniques. To avoid the necessity of specifying both bounds each time we give a complexity bound, we will present the following notation.

Notation. The complexity bound for multiplying two elements of size B is equal to $O(B^\rho)$. In the case that we use the naïve way to multiply the elements, ρ is equal to 2; in the case of employing fast multiplication techniques we have that for each $\epsilon > 0$ and for $B > B_0(\epsilon)$ that $1 \leq \rho < 1 + \epsilon$.

2. SIZE OF THE PARAMETERS.

(2.1) Introduction.

In order to analyze the complexity bounds of the algorithm, we frequently have to use the parameters s, t, u, v, w, and μ that have been calculated by the optimization stage of the algorithm. The complexity bounds of the algorithm and its predecessors will be expressed in terms of these parameters. In this section we will express these parameters in terms of n, the number subjected to the primality test. The complexity bounds for the generation of the tables will be expressed in terms of parameters s_0, t_0, and u_0 independent of n. These parameters will be expressed in terms of N, the upper bound for numbers n that can be handled by the algorithm using these tables. To be specific, we will specify the bounds

(i) $$(\log n)^{\check{c}_0 \cdot \log\log(2\cdot\mu\cdot\log n)} < \operatorname{lcm}(t,w) < (\log n)^{\hat{c}_0 \cdot \log\log(2\cdot\mu\cdot\log n)},$$

(ii) $$\operatorname{lcm}(s,v) = \Omega(n^{\mu}) \text{ and } \operatorname{lcm}(s,v) = O(t\cdot n^{\mu}),$$

(iii) $$(\log N)^{\check{c}_0 \cdot \log\log(2\cdot\mu\cdot\log N)} < t_0 < (\log N)^{\hat{c}_0 \cdot \log\log(2\cdot\mu\cdot\log N)},$$

(iv) $$s_0 = \Omega(N^{\mu}) \text{ and } s_0 = O(t_0\cdot N^{\mu}),$$

(v) $$(\log n)^{\check{c}_0 \cdot \log\log(2\cdot\mu\cdot c\cdot\log n)} < t < (\log n)^{\hat{c}_0 \cdot \log\log(2\cdot\mu\cdot c\cdot\log n)},$$

(vi) $$(\log n)^{\check{c}_0 \cdot \log\log(2\cdot\mu\cdot(1-c)\cdot\log n)} < w < (\log n)^{\hat{c}_0 \cdot \log\log(2\cdot\mu\cdot(1-c)\cdot\log n)},$$

(vii) $$\bar{s} = \Omega(n^{c\cdot\mu}) \text{ and } \bar{s} = O(t\cdot n^{c\cdot\mu}),$$

(viii) $$\bar{v} = \Theta(n^{(1-c)\cdot\mu}),$$

(ix) $$(\log s)/(\log(t+1)) \leq \#\{q : q \mid s, \text{ and } q \text{ prime}\} \leq (\log s)/(\log 2),$$

for all n exceeding a positive effectively computable bound and for some $c = c(n)$ with $0 \leq c \leq 1$ and some positive absolute calculable constants $\check{c}_0$ and $\hat{c}_0$. In the above formulae $\bar{s}$ and $\bar{v}$ denote the expected values of s and v respectively. Taking $P_t = \max\{p^k : p^k \mid t, \text{ and } p \text{ prime}\}$, one can also specify the bounds

(x) $$(\log t)/(\log P_t) \leq \#\{p : p \mid t, \text{ and } p \text{ prime}\} \leq (\log t)/(\log 2),$$

(xi) $\max\{l^e : l^e \mid u, u = \text{ord}(n \bmod t), \text{ and } l \text{ prime}\} \leq P_t,$

(xii) $\pi(\sqrt{P_t}) \leq \#\{l : l \mid u, u = \text{ord}(n \bmod t), \text{ and } l \text{ prime}\} \leq \pi(P_t),$

where $\pi(P_t)$ denotes the number of primes less than or equal to P_t. Furthermore we will present observations and conjectures that suggest the bound

(xiii) $P_t = \Theta(\log t).$

We will also specify some bounds about the expected size of some of the parameters

(xiv) $\bar{c} = \bar{c}(n)$ is such that
$\mu(1 - \bar{c}) = \Theta((\log\log n)^2)/\log n)$, and

(xv) $(\lambda(p^k)^{1-\epsilon} \leq \overline{\text{ord}}(n \bmod p^k) \leq \lambda(p^k),$

where $\overline{\text{ord}}(n \bmod p^k)$ denotes the expected value of $n \bmod p^k$, for $p^k \mid t$. Finally, if $T(n)$ is the time needed to complete the test for the number n, we can specify the bound

(xvi) $\text{lcm}(t, w) \leq T(n)$ if n is prime,
$T(n) \leq (\text{lcm}(t, w))^c$ for all n,

where c is some positive absolute effectively computable constant.

Remarks.

The bounds in (i)—(vi) are not valid for small values of n because some of the functions used in these bounds are not defined for small n. Therefore n should exceed a positive effectively computable bound in (i)—(vi).

Throughout this chapter we will use $\text{lcm}(s, v)$ and $\text{lcm}(t, w)$ instead of $\text{lcm}(s_1 t, v)$ and $\text{lcm}(t_1, u, w)$ from Theorem II.(7.1), where s_1 and t_1 are defined in II.(7.1). Similar formulae can be obtained for $\text{lcm}(s_1 t, v)$ and $\text{lcm}(t_1, u, w)$.

(2.2) Proving the bounds.

Before proving the complexity bounds for the algorithm, we will first mention some complexity bounds which have been proved for its predecessors.

The primality test as described by [30] and [29] roughly uses parameters t and s such that

(a) $s \mid 2 \prod_{\substack{q-1 \mid t \\ q \text{ prime}}} q^{o_q(t)+1}$ and

(b) $s > \sqrt{n},$

to prove that for all divisors r of n there exists an i with $0 \leq i \leq t$ such that $r \equiv n^i \bmod s$. (cf. [30, Theorem 6.3]). Checking the values $r_i \equiv n^i \bmod s$ with $0 \leq r_i < s$ and $1 \leq i \leq t-1$ finishes the test. Here $o_q(t)$ denotes the number of factors q in the number t.

To estimate s and t in this method, a theorem due to Pomerance and Odlyzko is used to show the asymptotic growth of the parameter t (cf. II.(6.21)).

Theorem 1 (cf. [2, Theorem 3]). *Let $g(n)$ be the least positive integer such that the product of primes q with $q-1 \mid g(n)$ exceeds $\sqrt{n}$. There are positive absolute calculable constants $\check{c}$ and $\hat{c}$ such that beyond some computable point*

$$(\log n)^{\check{c}\log\log\log n} < g(n) < (\log n)^{\hat{c}\log\log\log n}.$$

In [2] and [30] it is shown that $\check{c}$ can be taken equal to $(1-\epsilon)/\log 2$, and in [2] it is conjectured that $\hat{c}$ can be taken equal to $(1+\epsilon)/\log 2$, for any $\epsilon > 0$, and for for any n exceeding a bound depending on ϵ. This theorem implies that for $n > e^e$

$$t = (\log n)^{\Theta(\log\log\log n)}.$$

The restriction that s should be larger than $\sqrt{n}$, implies that $s = \Omega(\sqrt{n})$. On the other hand, any $s > \sqrt{n}$ is sufficient to complete the test. As will be explained in (4.5) it may be beneficial to take s somewhat larger, i.e., to bound s from above by $f \cdot \sqrt{n}$, where f is an overshoot factor. This overshoot factor will never be larger than t (cf. (4.5)). Therefore we have $s = O(t \cdot \sqrt{n})$. These bounds already hold for the primality test in [2], which is the predecessor of the test in [30]. Both tests have a running time of $T(n)$ binary operations with

$$\begin{aligned} t \leq T(n) \qquad & \text{if } n \text{ is prime,} \\ T(n) \leq t^c \qquad & \text{for all } n, \end{aligned}$$

where c is some positive absolute effectively computable constant. If for all divisors r of n there exists an integer i with $1 \leq i \leq t$ such that $r \equiv n^i \bmod s$, then at least one $r = r_i$ will be smaller than $\sqrt{n}$. The constraint that s should be larger than $\sqrt{n}$ is to ensure that if a divisor $r_i \equiv n^i \bmod s$ with $1 < r_i < \sqrt{n}$ exists, that it will be found by checking only those values of r_i that are smaller than s. In II.9 a polynomial algorithm is described to find all divisors of n that are congruent to a given value r modulo a number s, where $s > \sqrt[3]{n}$ (cf. [88]). Taking $r \equiv n^i \bmod s$ with $0 \leq r < s$ for $0 \leq i \leq t-1$ enables us to find any divisor that is congruent to $n^i \bmod s$ for some i. We will show that using this algorithm

one can improve the complexity bounds of the algorithm. Theorem 1 can be generalized to the next theorem.

Theorem 2. *Let ν be a real constant with $0 < \nu \leq 1/2$ and and let $h(n) = h_\nu(n)$ be the least positive integer such that*

$$2 \prod_{\substack{q-1 \mid h(n) \\ q \text{ prime}}} q^{o_q(h(n))+1}$$

exceeds n^ν. There are positive absolute effectively calculable constants $\check{c}$ and $\hat{c}$ such that beyond some computable point

$$(\log n)^{\check{c} \log\log(2\cdot\nu\cdot\log n)} < h(n) < (\log n)^{\hat{c} \log\log(2\cdot\nu\cdot\log n)}.$$

In Theorem 2 the constants $\check{c}$ and $\hat{c}$ have the same value as the constants in Theorem 1. The lowerbounds on n should be increased in such a way that $\log\log(2 \cdot \nu \cdot \log n) \geq 0$. Using Theorem 2 we get that

$$(\log n)^{\check{c} \log\log(2\cdot\nu\cdot\log n)} < h(n) < (\log n)^{\hat{c} \log\log(2\cdot\nu\cdot\log n)}$$

with $\check{c}$ and $\hat{c}$ absolute effectively computable constants and that $s > n^\nu$. Using the algorithm of II.9 one can take $\nu = \frac{1}{3}$, thus improving the previous bounds. A polynomial algorithm in $\log n$ that would find all divisors congruent to r mod s with $s > n^\nu$ and $\nu < \frac{1}{3}$ would immediately improve the asymptotic bounds.

Our algorithm (cf. Theorem II.(7.1)) uses, apart from the parameters s and t (which play the same role as in [30]), the parameters u, v, and w, to prove that for all divisors r of n there exists an i such that $1 \leq i \leq \operatorname{lcm}(t, w)$ with $r \equiv n^i \bmod \operatorname{lcm}(s, v)$. By taking $\operatorname{lcm}(s, v) > n^\nu$ it suffices to use an algorithm that determines all divisors $r \equiv n^i \bmod \operatorname{lcm}(s, v)$ of n. This proves the first part of (2.1)(ii). The second part of (2.1)(ii) follows from the fact that $\operatorname{lcm}(s, v)$ can be bounded from above by $f \cdot n^\nu$, with f an overshoot factor which is $O(t)$ (cf. (4.5)).

In [2, Remark 6.3] a heuristic argument is presented to find bounds for w.

Conjecture 3. (cf. [2]) *Let n be an arbitrary integer having no small prime factors. Suppose that $w = w(n)$ is the least positive integer such that the product v of the primes v_i expected to be found in $n^w - 1$ exceeds $\sqrt{n}$. Then there exists a positive absolute effectively calculable constant $\hat{c}$ such that beyond some computable point*

$$w < (\log n)^{\hat{c}\log\log\log n}.$$

Moreover, if all primes v_i are at most w^2, then there is positive absolute calculable constant $\check{c}$ such that beyond some computable point

$$(\log n)^{\check{c}\log\log\log n} < w.$$

Here $\check{c}$ and $\hat{c}$, as well as the lower bound on n can be taken equal to those in Theorem 1.

This result can be generalized like the generalization of Theorem 1 to Theorem 2.

Corollary 4. *Let n be an arbitrary integer having no small prime factors. Let ν be a real constant with $0 < \nu \leq 1/2$ and and let $w = w(n)$ be the least positive integer such that the product v of the primes v_i expected to be found in $n^w - 1$ exceeds n^ν. Then there exists a positive absolute effectively calculable constant $\hat{c}$ such that beyond some computable point*

$$w < (\log n)^{\hat{c}\log\log(2\cdot\nu\cdot\log n)}.$$

Moreover, if all primes v_i are at most w^2, then there is positive absolute calculable constant $\check{c}$ such that beyond some computable point

$$(\log n)^{\check{c}\log\log(2\cdot\nu\cdot\log n)} < w.$$

Combining Corollary 4 with Theorem 2 gives (2.1)(i).

We will show in Sections 3 and 4 that our test also has a running time of $T(n)$ binary operations with

$$\begin{aligned} \operatorname{lcm}(t,w) \leq T(n) \qquad & \text{if } n \text{ is prime,} \\ T(n) \leq (\operatorname{lcm}(t,w))^c \qquad & \text{for all } n, \end{aligned}$$

where c is some positive absolute effectively computable constant, according to (2.1)(xvi).

Since the main theorem of [30] can be regarded as a special case of Theorem II.(7.1), the asymptotic growth of s and t can be bounded from above by the asymptotic growth of these parameters found in [30]. The improvement to decrease the asymptotic bounds by decreasing the value of ν can also be applied to our algorithm.

The values s_0 and t_0 are needed to generate the tables needed by the algorithm to complete the primality test for all $n \leq N$. Therefore bounds for t_0 and s_0 can be found by replacing

n by N in the formulae (2.1)(i) and (2.1)(ii), and assuming that v and w do not play any role, i.e., v and w are equal to 1. This gives (2.1)(iii) and (2.1)(iv).

In order to decrease the asymptotic growth of t, one has to decrease the asymptotic growth of s and therefore find a non-trivial asymptotic lower bound for $v/\gcd(s,v)$. Since v is the product of prime divisors of the small cyclotomic polynomials evaluated in n, the value of v highly depends on the particular number n. Therefore it seems to be hard to get a general lower bound for the value of $v/\gcd(s,v)$.

Finding an asymptotic bound for the *expected* value $\bar{v}$ of v depends on the number of factors v_i of $n^i - 1$ that will be expected to be found, for $i \mid w$.

Suppose that $\bar{v}$ is equal to $\Theta(n^{(1-\bar{c})\mu})$ for a value $\bar{c}$ with $0 \leq \bar{c} \leq 1$, (cf. (2.1)(viii)). Using (2.1)(ii), this implies that $\bar{s} = \Omega(n^{c\mu})$, being the first part of (2.1)(vii). Using $\nu = c\mu$ in Theorem 2, with μ equal to $\frac{1}{2}$ or $\frac{1}{3}$, gives (2.1)(v). The second part of (2.1)(vii) follows from (2.1)(ii) and the fact that $\bar{s}$ can be bounded from above by $f \cdot n^{\nu}$, with f an overshoot factor which is $O(t)$ (cf. (4.5)). Using $\nu = (1-\bar{c})\mu$ in Theorem 3, with μ equal to $\frac{1}{2}$ or $\frac{1}{3}$, gives (2.1)(vi).

The time spent on finding factors can trivially be bounded from above by the worst case bound to complete the test without any extra factors, being $(\log n)^{c \log\log\log n}$ arithmetic operations, for some positive constant c. Spending more time on factoring would increase the time needed to complete the primality test.
Suppose that the number of binary operations to be used by the test is bounded by T, and that we try to find factors of $n-1$ up to a certain bound B. The costs to find these factors using trial division is approximately equal to $(B \log n)/\log B$, which should be equal to T.

This implies that $B \approx T/(\log n)$. Since every prime p has a probability of $1/p$ to divide $n-1$, we have that the expected sum of the logarithms of the factors will be equal to

$$\sum_{p \leq B} \log p/p \sim \log B \sim \log(T/(\log n))$$

as a first order approximate. Since $T = (\log n)^{\Theta(\log\log\log n)}$, we get with a rough approximation that

$$\log(\bar{v}) = (1-\bar{c})\mu \log n = \tilde{c} \log\log n \log\log\log n$$

for some constant $\tilde{c}$. Therefore $(1-\bar{c})\mu \approx (\tilde{c}\log\log n \log\log\log n)/\log n$; this implies that finding more factors using trial division does not change the asymptotic behaviour of the primality test. This does not change if one tries to find factors of $n^i - 1$ for $2 \leq i \leq w$ as well.

If we employ a more sophisticated method of factoring instead of trial division method, like for instance the elliptic curve method, we will get another analysis. Again we assume that $T = (\log n)^{\Theta(\log\log\log n)}$ is an upper bound for the number of binary operations. Suppose that we would try to find all factors of $n-1$ up to a certain bound B using the elliptic curve method. The costs are approximately equal to

$$(\log n)^2 \cdot e^{(\sqrt{2}+o(1))\cdot\sqrt{\log B \log\log B}} \quad \text{(cf. [83])}.$$

These costs should not dominate T, and therefore we get that

$$B \approx e^{(\log\log n)^2\cdot(\tilde{c}^2/2)}.$$

Taking all factors up to B into account, we get that the expected product will be approximately

$$\frac{e^{(\log\log n)^2\cdot(\tilde{c}^2/2)}}{(\log\log n)^2\cdot(\tilde{c}^2/2)}$$

This will again be equal to $\bar{v}$. So we get that

$$\log(\bar{v}) = (1-\bar{c})\mu\log n \approx (\log\log n)^2\cdot(\tilde{c}^2/2)$$

for some constant $\tilde{c}$. This implies that

$$(1-\bar{c})\mu \approx ((\tilde{c}^2/2)\cdot(\log\log n)^2)/\log n.$$

This is a better result than the previous one for trial division. This gives

$$2\mu(1-\bar{c}) = \Theta((\log\log n)^2)/\log n,$$

which is (2.1)(xiv) for $\bar{c} = c(n)$. Finding more factors does not change the asymptotic behaviour of the primality test. This does not change if one tries to find factors of $n^i - 1$ for $2 \leq i \leq w$ as well.

In order to incorporate w in the least common multiple with t, for all $d \mid w$ and for all prime factors $v|n^d - 1$ that are found, one needs to generate the v-th roots of unity in an extension of degree d. In (4.3) we will show that the number of binary operations to generate all v-th roots of unity in an extension of degree d is

$$O(d^\rho(\log n)^{1+\rho} + d^{1+\rho}(\log n)^\rho \cdot \sum_{\substack{v|n^d-1 \\ v \text{ prime}}} \log v).$$

When summing the above expression over all divisors d of w, one gets

$$\sum_{d|w} d^\rho(\log n)^{1+\rho} + \sum_{d|w}(d^{1+\rho}(\log n)^\rho) \cdot \sum_{\substack{v|n^d-1 \\ v \text{ prime}}} \log v =$$
$$= \sigma_\rho(w)(\log n)^{1+\rho} + \sigma_{1+\rho}(w)(\log n)^\rho(2 \cdot (1-c) \cdot \mu \log n) =$$
$$= w^\rho(\log n)^{1+\rho} + w^{1+\rho}(\log n)^\rho(2 \cdot (1-c) \cdot \mu \log n),$$

where $\sigma_k(w)$ is defined as the sum of the k-th powers of all divisors of w (cf. [50, Theorem 274]), for any k.

Since the prime factors of s are bounded from above by $t+1$, one can give a lower bound of $\log(s)/\log(t+1) = \Omega((\log s)^{1-\epsilon})$ for the number of factors in s, for any $\epsilon > 0$. The number of prime factors in s can be bounded from above by $\log s/\log 2$. This proves (2.1)(ix).
The number of prime factors in v can be bounded from above by $\log v/\log 2$.
All remaining bounds can be expressed using bounds for the maximal prime power P_t occurring in t. Therefore it seems to be helpful to have a bound for the maximal prime power in terms of t. This seems to be hard, since the proofs do not give any clue about the precise nature of t. In the algorithm described in [2] the value of t should be square-free. It is conjectured in [2] that in order to get

$$\prod_{\substack{q \text{ prime} \\ q-1|t}} q$$

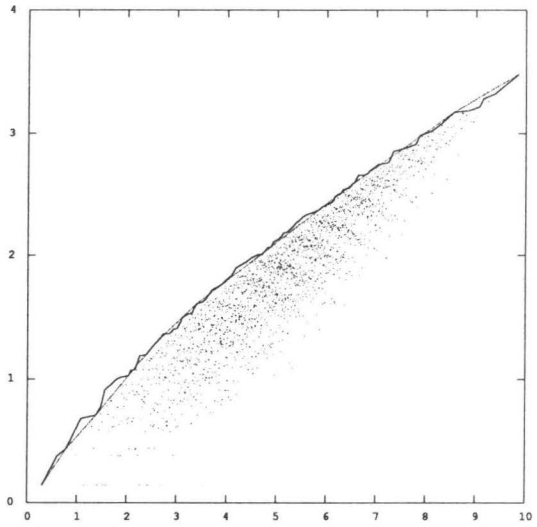
sufficiently large, one can take t equal to the product of the first k primes for some k. If this conjecture would be true we have that $\log t = \sum_{p|t} \log p \sim \max\{p : p|t\}$. So we would have $P_t = \Theta(\log t)$. Here we used that $\sum_{p \le x} \log p \sim x$ (cf. [50, Theorem 6 and 420]).

Unlike the algorithm of [2], the algorithm of [29] and [30] as well as our algorithm do not necessarily need t to be square-free. This implies that the value of t that is used by our algorithm might be different but it gives a better result. We will show what the consequences are for the bounds on P_t. In Appendix.(1.1) all 1920 even divisors t of $t_0 = 6983776800$ are tabulated as well as the values

$$\log_{10}(e(t)) = \log_{10}(2 \prod_{\substack{q-1|t \\ q \text{ prime}}} q^{o_q(t)+1}).$$

In II.6, a subset of this table is shown, containing all 87 values of t, which are optimal in the sense that there does not exist a $t' \mid 6983776800$ with $t' < t$ such that $e(t') \ge e(t)$.

Below for each even divisor t of 6983776800 the value $\log_{10}\log_{10} e(t)$ is depicted by a dot in the diagram, and the 87 optimal values of t are connected by a solid line. The dotted line included in the diagram will be explained later.



$\log_{10}\log_{10} e(t)$ *as a function of* $\log_{10} t$, *with* $0 \le \log_{10} t \le 10$ *and* $0 \le \log_{10}\log_{10} e(t) \le 4$.

Definition 1. *Let the sequence* (t_j), $j = 1, \ldots$ *be defined as*

$$t_j = \prod_{i \le j} p_i^{k_i},$$

where p_i *is the* i*-th prime and* k_i *is defined by*

$$p_i^{k_i} \le 2 \cdot p_j < p_i^{k_i+1},$$

for $i = 1, \ldots, j-1$.

For $j = 1, \ldots, 8$, the integers t_j are tabulated below.

j	t_j
1	$2 = 2$
2	$12 = 2^2 \cdot 3$
3	$360 = 2^3 \cdot 3^2 \cdot 5$
4	$2520 = 2^3 \cdot 3^2 \cdot 5 \cdot 7$
5	$55440 = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11$
6	$3603600 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13$
7	$367567200 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$
8	$6983776800 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$

Note that these 8 integers are optimal in the sense explained above. In the diagram above, the values of t_j are connected by straight dotted lines. For this particular sequence, we have that

$$\max\{p^k : p^k \mid t_j\} \leq 2 \cdot p_j \sim 2 \sum_{i \leq j} \log p_i \leq 2 \sum_{i \leq j} \log {p_i}^{k_i} = 2 \log t_j$$

as well as

$$\log t_j = \sum_{i=1}^{j} \log({p_i}^{k_i}) \leq \sum_{i=1}^{j} \log(2p_j) \leq 2j \log p_j \sim 2j \log(j \log j) \sim$$
$$2j \log j \sim 2p_j < 2 \cdot 2^{k_1+1} \leq 4 \max\{p^k : p^k \parallel t_j\}.$$

Here we used that $\sum_{p \leq x} \log p \sim x$ (cf. [50, Theorem 6 and 420]) and $p_j \sim j \log j$ (cf. [50, Theorem 8]). These data, as well as the implication of the conjecture of [2] suggest

$$\max\{p^k : p^k \parallel t\} = \Theta(\log t),$$

(cf. (2.1)(xiii)). This observation has minor implications for the complexity bounds of our algorithm. If it would not be true, only a few constants in exponents would increase.

Bounds that remain to be specified are the bounds for the asymptotic growth of u and its prime power divisors, as well as the number of prime powers in t and in u. These can be expressed in terms of the maximal prime power P_t occurring in t.

First of all the number of prime powers occurring in t can be bounded from above by $(\log t)/(\log 2)$. Since we do know an upper bound for the maximal prime power in t, we can also specify a lower bound for the number of prime powers, being $\Omega(\log t / \log P_t)$, which is $\Omega((\log t)^{1-\epsilon})$. This proves (2.1)(x).

The maximal prime power occurring in $u = \prod_{l \text{ prime}} l^{e_l}$ will be a divisor of the exponent of $(\mathbf{Z}/p^k\mathbf{Z})^*$ for some prime power $p^k || t$. So $\max\{l^{e_l} : l^{e_l} || u\}$ can be bounded from above by P_t, and consequently we have $\max\{l^{e_l} : l^{e_l} || u\} \leq P_t = O(\log t)$. This proves (2.1)(xi). This implies that the number of prime powers occurring in u can be bounded from above by $\pi(P_t) = O(\log t/(\log \log t))$. Here $\pi(P_t)$ denotes the number of primes less than or equal to P_t. Assuming that t is divisible by all primes less than or equal to the maximal prime in t, which is the case for all values of t in the sequence (t_j) defined above, the number of prime powers occurring in u can be bounded from below by $\#\{p : p^k | t, k > 1, p^k \leq P_t\}$, which is $\pi(\sqrt{P_t}) = O(\sqrt{\log t}/(\log \log t))$. This proves (2.1)(xii).

Although each prime power of u can be bounded by $\log t$, it seems that u itself can only be bounded from above by t.

Similarly we have that the number of prime powers occurring in t_0 as well as in u_0 is bounded from above by $(\log t_0)/(\log 2)$ and from below by $(\log t_0)/(\log P_{t_0})$. Also $u_0 = O(t_0)$. These bounds follow directly by substituting $t = t_0$ in (2.1)(x–xiii).

Next we will try to find bounds for the average order of $n \bmod p^k$ over all integers n coprime with t_0. Suppose that we fix p^k and try to bound the average order modulo p^k. The average order of $n \bmod p^k$ can now be defined as

$$\bar{o}(p^k) = \frac{\sum_{d|\lambda(p^k)} C(p^k, d) \cdot d}{\phi(p^k)},$$

where $C(p^k, d)$ is defined as the number of values $i \in (\mathbf{Z}/p^k\mathbf{Z})^*$ with $\mathrm{ord}(i \bmod p^k) = d$. If $p > 2$ and $d \mid \lambda(p^k)$ then $C(p^k, d) = \phi(d)$, where *phi* is the Euler-phi function. If $p = 2$, $k > 2$ then $C(2^k, d) = d$ for $d \mid \lambda(2^k)$ and $d > 2$. Furthermore, if $p = 2$ and $k > 2$ we have $C(2^k, 2) = 3$ and $C(2^k, 1) = 1$. Finally if $p = 2$ and $k \leq 2$ then $C(2^k, d) = 1$ for all d.

Calculating $\bar{o}(p^k)$ for $p^k = 2^k$, with $k > 2$ we get $\bar{o}(2^k) = \frac{2^{k-1}}{3} + \frac{5}{3}2^{1-k}$ which converges to $\frac{2}{3}\lambda(2^k)$, when k tends to infinity.

For p^k with p odd and $k > 1$ we get

$$\bar{o}(p^k) = \frac{\left(\sum_{i=0}^{k-1} p^i \phi(p^i)\right) \cdot \left(\sum_{d|p-1} \phi(d) \cdot d\right)}{\phi(p^k)}.$$

Since $\lim_{k\to\infty} \left(\sum_{i=0}^{k-1} \phi(p^i) \cdot p^i\right) / p^{k-1} = p^{k-1}$, it remains to bound

$$\bar{o}(p) = \frac{\left(\sum_{d|p-1} \phi(d) \cdot d\right)}{\phi(p)}.$$

If $p - 1 = \prod_l l^{k_l}$, with l prime, this can be reduced to

$$\phi(p) \cdot \bar{o}(p) = \prod_{l|p-1} \frac{l^{2k_l+1} + 1}{l + 1}.$$

Since

$$(p-1) \cdot \phi(p-1) \leq \prod_{l|p-1} \frac{l^{2k_l+1} + 1}{l + 1} \leq (p-1)^2,$$

we have that $\bar{o}(p)$ can be bounded from above by $p - 1$, and from below by $\phi(p - 1)$; therefore the lower bound for $\bar{o}(p^k)$ is $\Omega((\lambda(p^k)^{1-\epsilon})$ for any $\epsilon > 0$, when p tends to infinity (cf. [50, Theorem 328]. This proves (2.1)(xv).

3. ANALYSIS OF THE PREPARATION OF THE TABLES.

(3.1) Introduction. In this section we will analyze the complexity bounds for the generation of the tables used by our algorithm. The bounds will be expressed in terms of the parameters t_0, s_0, and u_0. In the previous section these parameters have been expressed in terms of N, being the upper bound for the numbers n which can be handled by our algorithm using these tables. In this section we will discuss the complexity bounds of the generation of the following five tables:

(i) The table of primes up to a fixed bound B.

(ii) The table containing all even divisors of t_0, which may be used as admissible values of t, all orders $p^k \mid t_0$ and all conductors q with $q-1 \mid t_0$.

(iii) The table containing matrices and related data needed to generate the extensions that may be used by the algorithm.

(iv) The table containing all exponents needed to express Gauss sums in terms of Jacobi sums.

(v) The table of Jacobi sums needed for the set of characters of order $p^k \mid t_0$ and conductor q with $q-1 \mid t_0$.

The analysis and heuristics concerning the preparation of each of these tables will be discussed; this will be done separately for each table.

(3.2) The prime table.

The most common way to create a table containing all primes up to a bound B, is by using Eratosthenes' sieve. It is possible to perform the algorithm in a sequential way, i.e., one does not have to store the complete table in memory, but one can sieve intervals of say length $\sqrt{B}$, by using the prime table of the primes less than $\sqrt{B}$.

This heuristic does not influence the asymptotic behavior of the method. For each prime $p < \sqrt{B}$ we have to delete its multiples from the table, which can be done in

$$\sum_{\substack{p<\sqrt{B} \\ p \text{ prime}}} \frac{B}{p}$$

additions. Since

$$\sum_{\substack{p<\sqrt{B} \\ p \text{ prime}}} \frac{\log p}{p} = \log(\sqrt{B}),$$

(cf. [50, Theorem 425]) we have that the number of binary operations is $\Theta(B \log B)$. In [126] a method is presented to find all primes $p \leq B$ in $\Theta(B \log B)$ binary operations using only $o(\sqrt{B})$ bits of memory. In order not to dominate the time needed to generate all the tables this should be done in at most $(\log N)^{c \log\log\log N}$ binary operations. This implies that B should be at most $t_0/(\log t_0)$. In practice B will be taken constant.

Instead of storing primes in the table, one can store differences between primes, since these differences are considerably smaller. Using the information about maximal gaps between consecutive primes, one is able to store several differences in one computer word. In [132, p. 85] one can find a table of maximal gaps between consecutive primes. For instance, for primes less than 10^6 the maximal gap is 114. This implies that each difference between two adjacent primes less than 10^6 can be expressed in at most 7 bits.

(3.3) The table containing all even divisors of t_0, the sets of primes $\mathcal{Q}_0$ and $\mathcal{P}_0$. Suppose a value for t_0, depending on N, and the factorization $\prod_{p \in \mathcal{P}_0} p^{k_p} = t_0$ is given. One has to determine

(a) the values $\phi(p^k)$ for all prime power divisors $p^k | t_0$, where ϕ is the Euler phi-function,

(b) the exponent $\lambda(t_0)$ of $(\mathbf{Z}/t_0\mathbf{Z})^*$, and its factorization $\prod_{l \in \mathcal{L}_0} l^{e_l} = \lambda(t_0)$,

(c) the set of primes $q \in \mathcal{Q}_0$ with $q - 1 \mid t_0$,

(d) the values $o_p(q-1)$ representing the number of times that a prime p occurs in the prime factorization of $q-1$ for $q \in \mathcal{Q}_0$ and $p \in \mathcal{P}_0$,

(e) for $i = 0, \ldots, \#\{t : t \mid t_0,\ 4 \mid t\} - 1$ all pairs $(t, \log e'_{i(t)})$, with t such that $i(t) = i$. Each pair consists of a divisor t of t_0 with $4 \mid t$, and the logarithm of

$$e'_{i(t)} = \prod_{\substack{q-1 \mid t \\ q \text{ prime}}} q.$$

The index function $i(t)$ is used in order to easily update values related to t. The function is defined by

$$i(t) = \sum_{p \in \mathcal{P}_0} o'_p(t) \prod_{\substack{p' > p \\ p' \in \mathcal{P}_0}} (k_{p'} + 1)$$

for each $t \mid t_0$, with $o'_p(t)$ defined by $o'_2(t) = o_2(t) - 2$ and $o'_p(t) = o_p(t)$ for odd p. The function $o_p(t)$ represents the number of times a prime p occurs in the prime factorization of t. The reason why the function $i(t)$ is used will be explained below.

Remark. In principle it is sufficient to start with t_0 and next calculate the factorization $\prod_{p \in \mathcal{P}_0} p^{k_p}$ of t_0, but in order not to factor at this stage of the algorithm, we assume that

the factorization of t_0 is known. In practice one starts by selecting a suitable set of primes in $\mathcal{P}_0$, and a value for k_p for each $p \in \mathcal{P}_0$. In this way, we do not need to factor t_0 at all. The set of primes $\mathcal{Q}_0$ can be generated by subsequently checking all the values $1 + \prod p^{k'_p}$ with $p \in \mathcal{P}_0$ and $k'_p \leq k_p$. Since we also need to find all divisors t of t_0, we can combine the search for the prime $q \in \mathcal{Q}_0$ with the search for all these divisors, since $q-1|t_0$ for all $q \in \mathcal{Q}_0$. Furthermore the calculation of $e'_{i(t)}$ can be done at the same time. Here we make the following observation.

For any prime q with $q-1 \mid t_0$ we have that $q-1$ divides those t with $o'_p(t) \geq o'_p(q-1)$ for every $p \in \mathcal{P}_0$. Since $i(t)$ is expressed in terms of $o'_p(t)$, one can very easily generate the values of $i(t')$ for all $t|t'$.

Starting with a value of $i(t)$, one can very easily generate the values of $i(t')$ for all $t|t'$, since $i(t')$ is an index generated from the factorization of t' in primes. If $t+1$ is prime, all values of $e'_{i(t')}$ for all $t|t'$ should be updated by adding the value of $\log(t+1)$ to it. Even without the knowledge of the value of all t', the values of $e'_{i(t')}$ for all $t|t'$ can be updated during the generation of all values of t.

These observations suggest the following strategy. First put $\log(e'_i) = 0$ for all $i \leq \prod_{p \in \mathcal{P}_0}(k_p+1)$. Next for all $t = \prod_{p \in \mathcal{P}_0} p^k$ check whether $t+1$ is prime or not. If this is the case, then update all values $\log(e'_{i(t')})$ for all t' with $t \mid t'$, by adding $\log(t+1)$ to each $\log(e'_{i(t')})$.

Heuristics. The calculations in steps (a), (b), and (d) are straightforward and take at most time polynomial in $\log t_0$. The most expensive part of this stage is the determination whether or not q is prime, for all possible even divisors $q-1$ of t_0 in step (c).

Since the factorization of t_0 is completely known, it is straightforward to find the factorization of $q-1$ for each individual q. In for instance [125] it is shown that if the complete factorization of $q-1$ is known, one can prove the primality of q in $O((\log q)^4)$ binary operations. In [2, Theorem 3] it is shown that

$$N^{c_1} < \prod_{\substack{q-1|t_0 \\ q \text{ prime}}} (q-1) \leq \prod_{t|t_0} t < N^{c_2}$$

where c_1 and c_2 are positive absolute effectively computable constants. This immediately implies that $\sum_{t|t_0} \log t = \Theta(\log N)$. Using this bound one can show that finding all primes q with $q-1 \mid t_0$ can be done in at most $O((\log N)^4)$ basic operations.

The calculations in step (e) are polynomial in $\log t_0$ and are, apart from the fact that they are performed simultaneously with the calculations of step (c), dominated by the time

needed to perform (c).

(3.4) The extension table.

The extension table contains for each prime power $u = l^e | \lambda(t_0)$ a set of conductors $m = m_u$ and for each pair (u, m) the table contains a flag $r = r_{u,m}$ indicating whether m_u is a prime or not, an element $g = g_{u,m}$ having maximal order in $(\mathbf{Z}/m\mathbf{Z})^*$, a minimal polynomial $f = f_{u,m}$ of degree u generating the cyclic field of degree u, two matrices $S = S_{u,m}$ and $S^* = S^*_{u,m}$ which will provide an easy way to switch from one representation in the cyclic field to another, and a denominator $D = D_{u,m}$ such that $(S^* \cdot S)/D$ is equal to the $u \times u$ identity matrix.

If the proper conditions (cf. II.(4.5)–II.(4.8)) are met, the algorithm is able to construct from these data in the table a cyclic extension of $\mathbf{Z}/n\mathbf{Z}$ of degree u and conductor m, for each $n \leq N$.

For fixed values of u and m, the time needed to calculate all other elements is dominated by the time needed to calculate the two matrices S and S^*. The matrix S is calculated by first determining the powers of the generator η of the cyclic field, expressed in terms of the powers of ζ_m, i.e., as polynomials of degree $\phi(m)$, and next expressing the powers of η in terms of the conjugates of η, which can be easily determined. Calculating η^i from η^{i-1}, for $i = 2, \ldots, u$ can be done in $O((i-1)\phi(m)\log(\phi(m)))$ binary operations, generating elements of size $O((i-1)\log(\phi(m)))$. So the total number of binary operations is

$$O(\phi(m)u^2 \log(\phi(m))),$$

generating elements of size at most $O(u \log(\phi(m)))$.

As shown in (2.2), u is $O(\log t_0)$. By taking $m = O(\log t_0)$ we get that for a single choice of u and m the number of binary operations can be bounded by

$$O((\log t_0)^3(\log\log t_0)) = O((\log\log N)^{3+\epsilon}),$$

generating integers of size $O((\log\log N)^{3+\epsilon})$, for any $\epsilon > 0$.

The entries of S will have the same magnitude as the coefficients that have been generated in the process analyzed above, being of size $O((\log\log N)^{1+\epsilon})$.

The calculation of S^* is the same as inverting the matrix S. In Section 6 it is shown that inverting a $u \times u$ integer matrix containing entries of size at most M can be done $O(u^3(u(M + \log u))^{1+\rho+\epsilon})$ binary operations, for any $\epsilon > 0$. Using this bound

gives $O((\log\log N)^{4+\rho+\epsilon})$ binary operations to generate all elements for a fixed u and a fixed m, for any $\epsilon > 0$.

In order to calculate the number of operations to generate the extension table, bounds are needed for the number of pairs (u, m). In Section 2 it is shown that the number of prime power divisors of u_0 is $O(\log t_0)$.

In II.4 it is shown that for fixed $u = l^e$ the set of conductors contains the prime conductors m with $m \equiv 1 \bmod u$ as well as $m = l^{e+1}$ if l is odd and $m = l^{e+2}$ if $l = 2$. The number of values of m can then be bounded by the number of primes less than $c \log t_0$ for some constant c, which is $O(\log t_0 / \log\log t_0)$. This implies that the generation of all elements in the extension table can be done in $O((\log t_0)^2(\log\log N)^{4+\rho+\epsilon}) = O((\log\log N)^{6+\rho+\epsilon})$ binary operations, for any $\epsilon > 0$.

Remark. As stated above, in order to generate an extension of degree $u = l^e$ of $\mathbf{Z}/n\mathbf{Z}$ for a particular value of $n < N$, the conductor m should meet the conditions mentioned in II.(4.5)–(4.8). If no such m can be found in the table, the algorithm should generate entries for another m that does meet the required conditions. Since in this case the value of n is known, all operations can then be done modulo n. In this way the size of the elements involved can be bounded.

We will examine the probability that for a particular $n \leq N$ no proper conductor m can be found. Although for each degree u there is one entry in the table with m being a prime power instead of a prime, we will only examine the probability that for a particular $n \leq N$ no proper prime conductor m can be found in the table. The condition that an entry with prime conductor m can be used is

$$n^{(m-1)/l} \not\equiv 1 \bmod m.$$

The probability that such an extension of degree $u = l^e$ cannot be used is equal to $1/l$. This implies that for each $u = l^e$ the expected number of conductors in the table necessary to find a conductor that meets the restrictions is $l/(l-1) \leq 2$.

The number $\pi_{d,a}(x)$ of primes congruent to a modulo d less than or equal to x for arbitrary x and d, and for $a < d$ with $\gcd(a, d) = 1$, is given by a theorem due to de la Vallée Poussin (cf. [129, p. 214] and [56, Ch. 16, §1]) and based on Dirichlet's theorem:

$$\pi_{d,a}(x) \stackrel{\text{def}}{=} \{p \text{ prime } p \leq x, p \equiv a \bmod d\} \sim \frac{x}{\phi(d) \cdot \log x}.$$

This implies that if the table contains only extensions of conductor $m \leq C \cdot l^{e_l}$, then at most $C \cdot l^{e_l - e + 1}/((l-1) \cdot e_l \cdot \log l)$ extensions of degree $u = l^e$ will be in the table. Combining

this with the fact that one cannot use an extension of degree $u = l^e$ with probability $1/l$ gives a probability of $\exp(-\frac{C \cdot l^{e_l - e + 1}}{(l-1) \cdot e_l})$ that the table does not contain a proper extension of degree l^e. This is a negligible probability.

(3.5) The "Jacobi-sum-exponent" table.

In order to be able to complete the primality test for a particular value of $n \leq N$, one needs to compute the quotient of Gauss sums

$$\frac{\tau(\chi)^i}{\tau(\chi^i)}$$

for each character χ of prime power order $\mathrm{ord}(\chi) = p^k$ dividing the parameter $t|t_0$, where i is equal to either p^k or $n \bmod p^k$.

This implies that in order to be able to complete the primality test for each $n \leq N$ one should be able to compute $\tau(\chi)^i/\tau(\chi^i)$ for each character χ of prime power order p^k dividing t_0 and for $i = p^k$ as well as for each $i < p^k$ relatively prime to p.

For each character χ of order p^k and prime conductor q a Gauss sum $\tau(\chi)$ is an element of the ring $\mathbf{Z}[\zeta_{p^k}, \zeta_q]$. In order to speed up the operations on these Gauss sums one can replace the quotients of Gauss sums mentioned above by a product of Jacobi sums $J(\chi^a, \chi^b)$, which live in the considerably smaller rings $\mathbf{Z}[\zeta_{p^k}]$. To express the quotients of Gauss sums in terms of Jacobi sums one needs a table of exponents $e_{\pi,p^k,i}$ such that

$$\frac{\tau(\chi)^i}{\tau(\chi^i)} = \prod_{\pi \in \mathcal{J}} J(\chi^a, \chi^b)^{e_{\pi,p^k,i}}.$$

In this product the set $\mathcal{J}$ consists of the values $\pi = a + b$ representing all Jacobi sums $J(\chi^a, \chi^b)$ needed by the algorithm to complete the primality test for all $n \leq N$. Furthermore $e_{\pi,p^k,i} = \sum_{j \in (\mathbf{Z}/p^k\mathbf{Z})^*} z_j \sigma_j$ with $z_j \in \mathbf{Z}_{\geq 0}$ for every $\pi \in \mathcal{J}$, every $p^k \mid t_0$ and $1 \leq i \leq p^k$ with $p \nmid i$ and $J(\chi^a, \chi^b)^{z_j \sigma_j} = J(\chi^{aj}, \chi^{bj})^{z_j}$.

By using

$$\frac{\tau(\chi)^{\mathrm{ord}\,\chi}}{\tau(\chi^{\mathrm{ord}\,\chi})} = \chi(-1) \cdot q \cdot \frac{\tau(\chi)^{\mathrm{ord}(\chi)-1}}{\tau(\chi^{\mathrm{ord}(\chi)-1})}$$

one can show that one does not need to calculate exponents for $\tau(\chi)^{\mathrm{ord}\,\chi}/\tau(\chi^{\mathrm{ord}\,\chi})$, (cf. Corollary II.(8.8)).

Since the exponents $e_{\pi,p^k,i}$ depend on the Jacobi sums being used, one does not only need to determine a set of exponents, but also one has to determine *which* Jacobi sums $J(\chi^a, \chi^b)$ in terms of the values a and $b = \pi - a$ will be used.

In Lemma II.(8.7) it is shown that

$$\frac{\tau(\chi)^i}{\tau(\chi^i)} = \prod_{j=1}^{i-1} J(\chi, \chi^j).$$

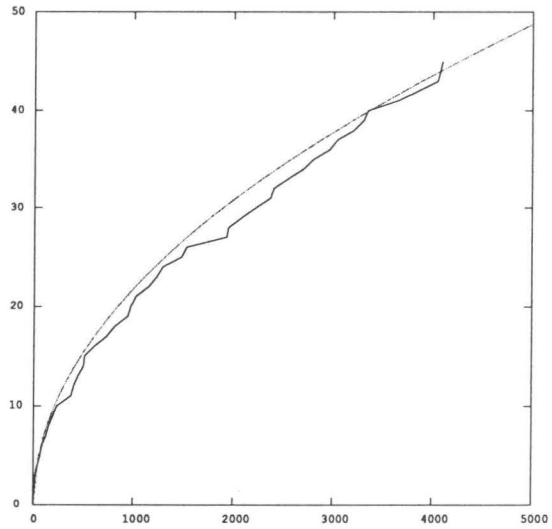
This implies that $\mathcal{J} = \{\pi \in \mathbf{Z}_{>0} : \pi < \max_{p^k | t_0}(\mathrm{ord}(\mathbf{Z}/p^k\mathbf{Z})^*)\}$ and $e_{\pi,p^k,i} = 1$ for all $\pi \in \mathcal{J}$, all $p^k \mid t_0$ and all $i \in (\mathbf{Z}/p^k\mathbf{Z})^*$ with $i < \mathrm{ord}(\mathbf{Z}/p^k\mathbf{Z})^*$ would be a solution. This solution however, would need many Jacobi sums, and since one needs to generate a table of all these Jacobi sums, it is beneficial to attempt to minimize the number of Jacobi sums needed, both with respect to the storage used by them, as well as with respect to the time needed to generate them. The solution above only gives us an upper bound for the number of Jacobi sums needed by the algorithm. The number of Jacobi sums for a fixed conductor q and fixed order p^k will be at most $p^k - 2$. In Example II.(8.12) it is shown that it is possible to use only Jacobi sums of characters of prime order. This implies an upper bound $\pi(p^k - 2)$ for the number of Jacobi sums needed, where $\pi(p^k - 2)$ is the number of primes less than $p^k - 2$. In II.(8.12) it is shown that this bound is not strict. We will try to find a better bound. This will be done by the algorithm described in IV.(2.4).

Using this algorithm sequentially on all prime powers p^k less than or equal to 4096 we can tabulate the number of Jacobi sums that have to be introduced to express all Gauss sums in terms of Jacobi sums. We will list these quantities only for those p^k for which new Jacobi sums had to be introduced, i.e., only the smallest p^k for which a new Jacobi sum had to be introduced is tabulated. In this table we also list *which* Jacobi sum $J(\chi^a, \chi^b)$, represented by $\pi = a + b$, had to be introduced for $\mathrm{ord}(\chi) = p^k$.

$\#\mathcal{J}$	π	p^k	$\#\mathcal{J}$	π	p^k	$\#\mathcal{J}$	π	p^k
1	2	3	16	53	607	31	107	2371
2	3	7	17	61	729	32	181	2401
3	5	16	18	47	811	33	131	2551
4	7	37	19	71	941	34	127	2699
5	11	64	20	109	971	35	157	2801
6	13	81	21	83	1024	36	151	2963
7	17	125	22	67	1151	37	149	3041
8	19	151	23	89	1231	38	139	3209
9	23	191	24	101	1291	39	167	3307
10	29	233	25	73	1481	40	137	3347
11	43	373	26	79	1531	41	179	3643
12	31	401	27	103	1931	42	227	3851
13	37	443	28	97	1949	43	193	4049
14	41	499	29	233	2083	44	163	4073
15	59	509	30	113	2221	45	199	4096

Depicting the number of Jacobi sums needed to express all Gauss sums of characters χ with $\mathrm{ord}(\chi) = p^k \leq P$ into these Jacobi sums as a function of P results in the picture given below. The solid line in this picture represents the number of Jacobi sums needed, while the dotted line is the function $(\sqrt{P})/1.45$, which is the result of curve-fitting.

Based on this empirical evidence, we will assume from this point on that the function of the expected number of Jacobi sums needed for a character χ of order p^k behaves approximately as $c\sqrt{p^k}$, i.e., $\#\mathcal{J} = \Theta(\sqrt{p^k})$. This function is determined by including all prime powers less than 4096. This is far beyond the scope of the algorithm: using these prime powers should suffice to prove the primality of numbers n with $\log n \geq 10^{100}$.



$\#\mathcal{J}$ as a function of P, with $0 \leq P \leq 5000$ and $0 \leq \#\mathcal{J} \leq 50$.

Summing the expected number of Jacobi sums needed for a character χ of order p^k over all conductors q with $q - 1 \mid t_0$ and all orders $p^k | t_0$, one would get that the expected number of Jacobi sums needed to be calculated would be equal to

$$\sum_{\substack{q-1|t_0 \\ q \text{ prime}}} \sum_{\substack{p^k \| q-1 \\ p \text{ prime}}} c\sqrt{p^k} \leq \sum_{\substack{q-1|t_0 \\ q \text{ prime}}} \log_2 q \sqrt{\log t} = \log_2 s \sqrt{\log t}$$

and

$$\sum_{\substack{q-1|t_0 \\ q \text{ prime}}} \sum_{\substack{p^k \| q-1 \\ p \text{ prime}}} c\sqrt{p^k} \geq \sum_{\substack{q-1|t_0 \\ q \text{ prime}}} \log_2 q = \log_2 s.$$

Therefore the total expected number of Jacobi sums is $O(\log s\sqrt{\log t})$ and $\Omega(\log s)$. Since the exponents do not depend on the conductor q, the number of exponents needed to express quotients of Gauss sums in terms of Jacobi sums is independent of the number of conductors. Using these bounds we can analyze the time needed to generate the table of exponents. Assuming that the generation of the exponents needed to express the Gauss sum $\tau(\chi)^i/\tau(\chi^i)$ into Jacobi sums will be done for i in increasing order, it suffices to express it into Jacobi sums and Gauss sums $\tau(\chi)^j/\tau(\chi^j)$ with $j < i$.

Roughly speaking, one tries to express the Gauss sum $\tau(\chi)^i/\tau(\chi^i)$ with character χ of order p^k into Jacobi sums by two different strategies. Both strategies try to express $\tau(\chi)^i/\tau(\chi^i)$ in terms of $\tau(\chi)^{i'}/\tau(\chi^{i'})$ with $i' < i$ and Jacobi sums.

The first strategy, uses the expression

$$\frac{\tau(\chi)^i}{\tau(\chi^i)} = \left(\frac{\tau(\chi)^{i/\pi}}{\tau(\chi^{i/\pi})}\right)^{\pi} \cdot \left(\frac{\tau(\chi)^{\pi}}{\tau(\chi^{\pi})}\right)^{\sigma_{i/\pi}}.$$

This strategy can only be applied when i is divisible by some $\pi \in \mathcal{J}$. Since both i/π and π are smaller than i, we have that $\frac{\tau(\chi)^{i/\pi}}{\tau(\chi^{i/\pi})}$ and $\frac{\tau(\chi)^{\pi}}{\tau(\chi^{\pi})}$ are already expressed in terms of Jacobi sums. Using the formula above enables one to express $\frac{\tau(\chi)^i}{\tau(\chi^i)}$ in terms of Jacobi sums as well.

In the second method, let $i_1 \equiv b((jp^k - i)/a) \bmod p^k$ and $i_2 \equiv a((jp^k - i)/b) \bmod p^k$ for $j = 0, \ldots, \max(a,b) - 1$ and $a + b = \pi \in \mathcal{J}$. If for some $j < a$ we have that a divides $jp^k - i$ and $i_1 < i$ then

$$\frac{\tau(\chi)^i}{\tau(\chi^i)} = \frac{\tau(\chi)^{i_1}}{\tau(\chi^{i_1})} \cdot \frac{\tau(\chi)^{i-i_1}}{\tau(\chi^{i-i_1})} \cdot J(\chi^a, \chi^b)^{\sigma_{(jp^k-i)/a}}.$$

Here both i_1 and $i - i_1$ are smaller than i, which enables us to express $\frac{\tau(\chi)^i}{\tau(\chi^i)}$ in terms of Jacobi sums.

If, on the other hand, we have that for some $j < b$ we have that b divides $jp^k - i$ and $i_1 < i$ then

$$\frac{\tau(\chi)^i}{\tau(\chi^i)} = \frac{\tau(\chi)^{i_2}}{\tau(\chi^{i_2})} \cdot \frac{\tau(\chi)^{i-i_2}}{\tau(\chi^{i-i_2})} \cdot J(\chi^a, \chi^b)^{\sigma_{(jp^k-i)/b}}.$$

Here i_2 as well as $i - i_2$ are smaller than i, enabling us to express $\frac{\tau(\chi)^i}{\tau(\chi^i)}$ in terms of Jacobi sums.

If both strategies fail, one has to introduce a new Jacobi sum $J(\chi^a, \chi^b)$ with $a + b = i$ prime. It follows from II.(8.12) that this only occurs when i is prime. Finally, one chooses the expression which uses the least number of Jacobi sums.

In finding the best expression for a fixed order p^k and fixed i only a constant number of operations are performed. At the end, however, the complete exponent has to be built up, keeping track of the coefficients for each Jacobi sum. For each coefficient one has to perform only a constant number of operations. Since we need $O(\sqrt{p^k})$ Jacobi sums for fixed p^k, this implies that the total number of operations needed to generate this table is

$$\sum_{\substack{p^k \mid t_0 \\ p \text{ prime}}} \sum_{\substack{i<p^k \\ p \nmid i}} c\sqrt{p^k} = \sum_{\substack{p^k \mid t_0 \\ p \text{ prime}}} c\phi(p^k)\sqrt{p^k} < c_2(\log t_0)^2 \log\log t_0$$

for some constant c_2. This implies that this table can be generated in

$$O((\log\log N)^{2+\epsilon})$$

binary operations, for any $\epsilon > 0$.

(3.6) The Jacobi sum table.

The last table that has to be generated is the table containing all Jacobi sums

$$J(\chi^a, \chi^b) = \sum_{x=0}^{q-1} \chi^a(x)\chi^b(1-x)$$

for all characters χ having conductor q with $q-1 \mid t_0$ and having order p^k with $p^k \parallel q-1$ and for all Jacobi sums represented by $\pi \in \mathcal{J}_{p^k}$. As explained in the previous subsection for each $\pi \in \mathcal{J}_{p^k}$ a pair (a,b) is chosen such that $a+b=\pi$.

The generation of these Jacobi sums is done in the following way. For each prime power $p^k \parallel q-1$, we will store the contribution of $\zeta_{p^k}^i$ in $c_{\pi,q,p,i}$ for all $\pi \in \mathcal{J}_{p^k}$. These values are initially set to zero. Next a table of pairs $(x, f(x))$ with $1-g^x \equiv g^{f(x)} \bmod q$ is made. Since each pair $(x, f(x))$ gives a contribution $\zeta_{p^k}^{ax+bf(x)}$ to the Jacobi sum $J(\chi^a, \chi^b)$, we will increase $c_{\pi,q,p,ax+bf(x)}$ by 1 for each $1 \le x \le q-2$. At this stage, $J(\chi^a, \chi^b)$ is expressed in terms of $\zeta_{p^k}^i$ with $0 \le i < p^k$. In order to represent $J(\chi^a, \chi^b)$ in terms of $\zeta_{p^k}^i$ with $0 \le i < \phi(p^k)$, one has to use the relation

$$\zeta_{p^k}^0 + \zeta_{p^k}^{p^{k-1}} + \ldots + \zeta_{p^k}^{(p-1)p^{k-1}} = 0.$$

Therefore, for each prime power $p^k \parallel q-1$, every i such that $\phi(p^k) \le i < p^k$ and every $1 \le j < p$, decrease $c_{\pi,q,p,i-jp^{k-1}}$ by $c_{\pi,q,p,i}$ for all $\pi \in \mathcal{J}_{p^k}$.

Suppose that the maximum number of integers less than q that can be stored in memory is equal to M_0. Let $M = \min(M_0, q-1)$ and $c(q)$ be the number of distinct prime divisors of $q-1$.

In IV.(2.5) two methods are described in detail to compute the Jacobi sums. In both cases, all the Jacobi sums with fixed conductor q are computed simultaneously.

In the first method a table is made of pairs $(x, f(x))$ with $1 - g^x \equiv g^{f(x)} \bmod q$ for a block of length M out of the $q-2$ different powers of g modulo q. This method requires about q^2/M multiplications modulo q, since we have to calculate all the $q-2$ powers of g at most q/M times.

In the second method one first computes the $c(q)$ powers $g^{(q-1)/p^k}$; this can be done in roughly $((c(q)/2)+1)\log_2 q$ multiplications modulo q, if one does the squarings of g followed by assembling $g^{(q-1)/p^k}$ (requiring approximately $\frac{\log q}{2}$ multiplications on the average) for each of the p^k. Next for each positive $x < q-1$ one computes the $c(q)$ powers $\bar{g}_x^{(q-1)/p^k}$, where $\bar{g}_x = 1 - g^x \bmod q$. Finally one has to find $i < p^k$ such that $g^{i(q-1)/p^k} \equiv \bar{g}_x^{(q-1)/p^k} \bmod q$, which can for instance be done by hashing. This method requires $(q-1)((c(q)/2)+1)\log_2 q$ multiplications modulo q for all powers $\bar{g}_x^{(q-1)/p^k}$. At most q more multiplications are needed to find the p^k different powers $(g^{(q-1)/p^k})^i$ for all p^k. In order to minimize the number of operations needed to perform this step, we will perform the first method as long as $q^2/M < (q-1)((c(q)/2)+1)\log_2 q + q$. This is the case when $M \geq \min(q-2, 2q/((c(q)+2)\log_2 q))$. Otherwise the second method is performed.

Asymptotically, i.e., for $q \to \infty$, one will make use of the second method, since for sufficiently large q the condition $M \geq \min(q-2, 2q/((c(q)+2)\log_2 q))$ will not be met. In the case that one uses the first method, the complexity bound of the second method serves as an upper bound for the complexity bound of the first method. Therefore this step takes at most $(q-1)((c(q)/2)+1)\log_2 q + q$ multiplications modulo q, which implies $O(qc(q)(\log q)^{1+\rho})$ binary operations.

Since $c(q)$ is at most $\log q$, we have that for the calculation of all Jacobi sums for one single value of q one needs $O(q(\log q)^{2+\rho})$ binary operations. This bound has to be summed over all primes q for which $q-1 \mid t$. We can bound this expression from above by using the following observations. First of all we have that $q \leq t_0$ and the number of primes q with $q-1 \mid t_0$ is at most $O(\log s_0)$. Therefore at most

$$\sum_{\substack{q-1 \mid t_0 \\ q \text{ prime}}} c \cdot q(\log q)^{2+\rho} \leq c \cdot\ t_0 \cdot (\log s_0)^{2+\rho}$$

binary operations are needed to generate the complete table of Jacobi sums. Using the asymptotic bounds for t_0 and s_0 gives $(\log N)^{\Theta(\log\log\log N)}$ binary operations.

Notice that the generation of this table is not polynomial in the size of N.

4. ANALYSIS OF THE JACOBI SUM PART OF THE ALGORITHM.

(4.1) Introduction.

Suppose the optimization part of the test has given appropriate values for s, t, u, v, and w. To calculate the cost of the Jacobi sum part of the test, we have to analyze

(i) the cost to generate additional extensions and transition matrices, (cf. IV.(5.1), IV.(5.2)),

(ii) the cost of the generation of the p^k-th roots of unity, for all $p^k \parallel \mathrm{lcm}(t, v)$, (cf. IV.(5.3)), and

(iii) the cost of performing all Jacobi sum tests for all primes q with $q - 1 \mid t$ and all primes p with $p \mid q - 1$, (cf. IV.(5.4)), and possibly a few additional Jacobi sum tests (cf. II.(7.1)).

We will analyze the cost for these parts separately. It can be easily seen that the calculations of the last two parts dominate the total cost of the Jacobi sum part.

(4.2) Generation of additional extensions and transition matrices.

Since we already discussed the complexity of generating extensions in (3.4) we can refer to this part for the asymptotic bounds of this step, by replacing s_0 by s, t_0 by t, and N by n. The generation of transition matrices is equivalent to one matrix multiplication modulo n, for each prime power extension that has been found. As has been shown in (3.4), there are $O(\log t)$ prime power extensions and each prime power degree can be bounded by $\log t$. Therefore this step takes $O((\log t)^2(\log\log n)^{4+\rho+\epsilon})$ binary operations, giving $O((\log\log n)^{6+\rho+\epsilon})$ binary operations, for any $\epsilon > 0$.

(4.3) Generating roots of unity.

Suppose one has to generate a p^k-th root of unity ζ_{p^k} in an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree $u_p = \mathrm{ord}(n \bmod p^k)$. Although there are several strategies to do so (cf. [87]), we choose to do this by taking a random element α in the extension of degree u_p and calculating its $(n^{u_p} - 1)/p^k$-th power γ. If, amongst other conditions (which are automatically true if n is prime, but which are checked always) (cf. Proposition II.(4.12)), $\gamma^{p^{k-1}} \neq 1$, then γ is a p^k-th root of unity. Following this strategy, the cost of generating a p^k-th root of unity in an extension of degree u_p is proportional to the cost of taking an element of size $u_p \cdot \log n$ to the power n^{u_p}. This exponentiation takes about $O((u_p \cdot \log n)^{1+\rho})$ binary operations.

This strategy has, however, a probability of $1/p$ to fail, in which case we have to choose another element α. This implies that the expected number of binary operations needed

to generate a p^k-th root of unity successfully in an extension of degree u_p is $O(\frac{p}{p-1}(u_p \cdot \log n)^{1+\rho})$.

In the case that at least two different roots of unity have to be generated, there exists a strategy to generate several roots of unity simultaneously. Suppose that one has to generate a ${p_1}^{k_1}$-th root of unity in an extension of degree u_1, and a ${p_2}^{k_2}$-th root of unity in an extension of degree u_2. Then it is possible to generate both the ${p_1}^{k_1}$-th and the ${p_2}^{k_2}$-th root of unity by taking a random element α in an extension of degree $\mathrm{lcm}(u_1, u_2)$ and raising it to the power $\left(n^{\mathrm{lcm}(u_1,u_2)} - 1\right)/({p_1}^{k_1} \cdot {p_2}^{k_2})$. By raising the resulting element to the power ${p_1}^{k_1}$ and to the power ${p_2}^{k_2}$ respectively, one gets possible candidates for the ${p_1}^{k_1}$-th and the ${p_2}^{k_2}$-th roots of unity γ_1 and γ_2. To prove that γ_1 and γ_2 are really roots of unity, one needs to show that $\gamma_1^{{p_1}^{k_1-1}} \neq 1$ and $\gamma_2^{{p_2}^{k_2-1}} \neq 1$. The probabilities that these elements are roots of unity are equal to $\frac{p_1-1}{p_1}$ and $\frac{p_2-1}{p_2}$ respectively. Notice that whether or not γ_2 is a root of unity is independent of the fact whether or not γ_1 is a root of unity.

As has been shown in the Chapter III, it is only beneficial to combine the generation of several roots of unity, if the least common multiple of the degrees involved is attained by one of the degrees. In any case the complexity bound for the time to generate all roots of unity is equal to the product of the number of roots that have to be generated and the maximum complexity bound for the time needed to generate one root of unity. Whether or not the generation of a root of unity fails is independent of the success or failure to generate any other root in the same combination. Therefore, if the generation of a root of unity fails, only the generation of this particular root of unity has to be performed again.

The maximum number of roots to be generated can be bounded by $\log tv$ which is proportional to $\log n$. The complexity bound for the time needed to generate a p^k-th root of unity in an extension of degree u_p is $O(\frac{p}{p-1}(u_p \cdot \log n)^{1+\rho})$.

Since $\frac{p}{p-1}$ can be bounded by 2, and u_p is $O(\log t)$, we have that the average case complexity bound, expressed in the number of bit operations, for the generation of all the roots of unity is equal to $O(\log n \cdot (\log t \cdot \log n)^{1+\rho}) = O((\log n)^{2+\rho}((\log\log n)(\log\log\log n))^{1+\rho})$.

An alternative approach. There exists a method, which is in most cases faster than the one described above.

Suppose that a P-th root of unity in an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree u_P has to be generated. Here P may be a prime power, or a product of prime powers. In order to do so, one has to take the $(n^{u_P} - 1)/P$-th power of a random element in the extension of degree u_P.

Let D be equal to $\lfloor (n-1/P) \rfloor$ and $r < P$ be such that $n-1 = P \cdot D + r$. We will show that $(n^{u_P}-1)/P$ can be written as

$$(n^{u_P}-1)/P = \sum_{i=0}^{u_P-1} a_i \cdot n^i = \sum_{i=0}^{u_P-1} (b_i \cdot D + c_i) \cdot n^i,$$

where b_i and c_i will be smaller than P. In doing so, one is able to do the $(n^{u_P}-1)/P$-th powering as

$$\prod_{i=0}^{u_P-1} \alpha^{a_i \cdot n^i} = \prod_{i=0}^{u_P-1} ((\alpha^D)^{b_i} \cdot \alpha^{c_i})^{n^i} = \prod_{i=0}^{u_P-1} \sigma^i((\alpha^D)^{b_i} \cdot \alpha^{c_i}).$$

This is considerably faster than the previous method, if P is relatively small in comparison with n.

Writing $n^{u_P}-1$ in base n gives $\sum_{i=0}^{u_P-1}(n-1)\cdot n^i$. Dividing this expression by P can be done as follows:

(1) Put $r_{u_P} = 0$.

(2) For $i = u_P - 1, \ldots, 1, 0$ perform step (2a).

(2a) Put $a_i = \lfloor r_{i+1} \cdot n + (n-1)/P \rfloor$ and put $r_i = r_{i+1} \cdot n + (n-1) - a_i \cdot P$.

In this way we get $(n^{u_P}-1)/P = \sum_{i=0}^{u_P-1} a_i \cdot n^i$, with $0 \leq a_i < n$ for $0 \leq i < u_P$.

Since r_i is a carry that arises from the division of $r_{i+1} \cdot n + n - 1$ and P we have that r_i is less than P. In fact it is even less than $P-1$; otherwise n and P would not be coprime.

If we divide $r_i \cdot n + n - 1 = (r_i+1) \cdot D \cdot P + r_i \cdot (r+1) + r$ by P to get a_i, we get $a_i = (r_i+1) \cdot D + \lfloor (r_i \cdot (r+1) + r)/P \rfloor$.

Since $r_i < P-1$ and $r < P$ we have that $\lfloor (r_i \cdot (r+1) + r)/P \rfloor$ is less than P. So a_i can be written as $(r_i+1) \cdot D + c_i$ with $c_i < P$. Furthermore, since $r_i < P-1$, we have that a_i can be written as $b_i \cdot D + c_i$ with $b_i < P$ and $c_i < P$.

These observations suggest the following strategy to calculate the $(n^{u_P}-1)/P$-th power of a random element α:

(a) First calculate $x = \alpha^D$

(b) Next calculate $x_i = x^{b_i}$ and $y_i = \alpha^{c_i}$ for $i = 0, \ldots, u_P - 1$.

(c) Finally calculate $\prod_{i=0}^{u_P-1} \sigma^i(x_i \cdot y_i)$, using a Horner scheme. Here the automorphism σ is as in IV.(5.3).

In this way the number of binary operations needed to perform the exponentiation is equal to $O(u_P^{\rho+1}(\log n)^\rho \log P + u_P^\rho (\log n)^{\rho+1})$ compared to $O((u_P \log n)^{\rho+1})$ for the previous method. If $\log P$ is significantly smaller than $\log n$, the method given above is much faster than the previous one. To find the total number of binary operations

needed in this step one has to sum the above expression over all divisors of w and all prime powers $p^k \parallel t$. As has been shown in (2.1) summing over all divisors of w gives $O(w^{1+\rho}(\log n)^\rho + w^\rho(\log n)^{1+\rho})$ binary operations. Taking the sum over all maximal prime power divisors of t gives $O((\log t)^{2+\rho}(\log n)^\rho + (\log t)^{1+\rho}(\log n)^{1+\rho})$ binary operations.

(4.4) Performing the Jacobi sum tests.

Suppose one has to perform a combined Jacobi sum test represented by $\mathcal{S}$ where each triple (q, p, h) in $\mathcal{S}$ represents a Jacobi sum test for a character χ of order p^k, where $k = o_p(q-1)$, and conductor q. The element h in each triple indicates that such a test could be performed in an extension of degree $u_{p,1} \cdot p^h$ (cf. IV.(5.4)). Suppose that the combined test will be performed in an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree $\bar{u}$. Essentially this test comes down to raising an element in an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree $\bar{u}$ to a power e, where $e = \Theta(n)$.

In Chapter III, it has been shown that several tests can be combined into one test, if certain restrictions have been met. It might be possible that the tests cannot be combined at all. Therefore the number of Jacobi sum tests $\#_{s,t}$ is at most equal to the number of pairs (p, q) with $q \mid s$ and $p \mid q - 1$. So

$$\#_{s,t} = \sum_{q \mid s} \#\{p : p \mid q-1\} \leq \sum_{q \mid s} \log_2 q \leq \log_2 s$$

The number of tests is at least equal to the number of primes q that divide s, since for each odd q one has to perform a Jacobi sum test with order equal to a power of 2. These tests cannot be combined, since the orders of the characters involved are not coprime. Therefore we have $\#_{s,t}$ is at least $(\log s)/(\log t)$ being equal to $\Omega((\log s)^{1-\epsilon})$. Even if we fix $p^k \parallel t$, the number of tests that have to be performed for characters of order p^k is $\Omega(\#_{s,t}/\phi(p^k)) = \Omega((\log s)^{1-\epsilon})$. Here we used the theorem, due to de la Vallée Poussin (cf. [129, p. 214] and [56, Ch. 16, §1]), based on Dirichlet's theorem, which is also given in Section (3.4). Using this result, one can show that the number of tests of order p^k where the exponent of $(\mathbf{Z}/p^k\mathbf{Z})^*$ is maximal is $\Omega((\log s)^{1-\epsilon})$. In (2.1) it is shown that $u = O(\log t)$ and that the average order of n modulo p^k is $\Omega(((p-1)p^{k-1})^{1-\epsilon})$.

The time needed to perform a Jacobi sum test in an extension of degree u is proportional to $\Theta(\log n \cdot (u \cdot \log n)^\rho)$. Using these expressions, the complexity bounds, expressed in the average number of bit operations for the performance of all Jacobi sum tests is $O(\log n \cdot \log s \cdot (\log t \cdot \log n)^\rho) = O((\log n)^{2+\rho+\epsilon})$ and $\Omega(\log n \cdot (\log s)^{1-\epsilon} \cdot (\log t \cdot \log n)^{\rho-\epsilon}) = \Omega((\log n)^{2+\rho-\epsilon})$ for any $\epsilon > 0$.

(4.5) Analysis concerning the final trial division.

The final trial division consists of checking all remaining divisors in the residue classes $n^i \bmod \mathrm{lcm}(s,v)$, for $i = 1,\ldots,\mathrm{lcm}(t,w)-1$. We will discuss two possibilities: $\mathrm{lcm}(s,v)$ exceeds $\sqrt{n}$ or $\mathrm{lcm}(s,v)$ exceeds $\sqrt[3]{n}$.

In the case that $\mathrm{lcm}(s,v)$ exceeds $\sqrt{n}$, at most $\mathrm{lcm}(t,w)-1$ powers of n modulo $\mathrm{lcm}(s,v)$ have to be calculated for which has to be checked whether or not the result is a divisor of n. The cost of the final trial division is $\Theta(t\cdot(\log n)^{\rho})$ binary operations. Since $t = (\log n)^{\Theta(\log\log\log n)}$, the cost of the final trial division will be $(\log n)^{\Theta(\log\log\log n)}$ binary operations, (cf. Theorem II.(6.21)).

In the case that s exceeds $\sqrt[3]{n}$, at most $t-1$ powers of n modulo s have to be calculated. For each power $r_i \equiv n^i \bmod s$ we have to employ the algorithm described in [88] to find all possible divisors of n in the residue class $r_i \bmod s$ (cf. Section II.9).

The cost of performing the algorithm described in [88] and II.9 is proportional to $O((\log n)^{1+\rho})$. It follows that the cost of the final trial division will be $O(t\cdot\log n^{1+\rho})$ binary operations. Again by using that $t = (\log n)^{\Theta(\log\log\log n)}$, for some effectively computable constant c (cf. Theorem II.(6.21)), we find that the cost of the final trial division will be $(\log n)^{\Theta(\log\log\log n)}$ binary operations.

Heuristics. In the final trial division stage we have to check all possible integers of the form $n^i \bmod \mathrm{lcm}(s,v)$, where $i = 1,\ldots,\mathrm{lcm}(t,w)$.

The probability that a composite number will not be detected by one of the previous stages is negligible. Therefore we do not expect to find any divisors, during this stage. Using this expectation, we can try to minimize the number of trial divisions that have to be performed in this step.

In the case that we use that $\mathrm{lcm}(s,v) > \sqrt{n}$ this leads to the following heuristic. If n is composite, then at least one divisor is less than or equal to $\sqrt{n}$. This proves, that it suffices to check only those integers $n^i \bmod \mathrm{lcm}(s,v)$ which are smaller than $\sqrt{n}$. If $\mathrm{lcm}(s,v) \approx \sqrt{n}$ then almost all integers of the form $n^i \bmod \mathrm{lcm}(s,v)$ are smaller than $\sqrt{n}$. By taking $\mathrm{lcm}(s,v)$ somewhat larger, however, the number of candidates less than $\sqrt{n}$ will be reduced considerably.

The probability that $n^i \bmod \mathrm{lcm}(s,v)$ is less than $\sqrt{n}$ is assumed to be to $\frac{\sqrt{n}}{\mathrm{lcm}(s,v)}$. Apart from the fact that for each $i < t$ one has to perform one reduction modulo $\mathrm{lcm}(s,v)$ the number of trial divisions of n can be reduced. The expected number of divisions in this stage will be equal to $\mathrm{lcm}(t,w)\cdot\left(1+\frac{\sqrt{n}}{\mathrm{lcm}(s,v)}\right)$. The phrase "expected" is because

$n^{t'} \bmod \mathrm{lcm}(s,v)$ might be equal to 1 for some divisor t' of $\mathrm{lcm}(t,w)$. In that particular case, all the trial divisions have been performed using only $t' \cdot \left(1 + \frac{\sqrt{n}}{\mathrm{lcm}(s,v)}\right)$ divisions. Since the reduction and the division have comparable complexities, this indicates that the number of operations can be reduced by a factor of almost 2.

It is not clear that the same kind of heuristic can be performed in the case that we use that $\mathrm{lcm}(s,v) > \sqrt[3]{n}$.

In the case that we use $\mathrm{lcm}(s,v) > \sqrt[3]{n}$, there exist at least three possibilities to reduce the number of operations needed for the final trial division.

In the case $\mathrm{lcm}(s,v) > \sqrt[3]{n}$ the algorithm of [88] (cf. II.9) is used to prove that there do not exist any divisors in the residue classes $r_i \equiv n^i \bmod \mathrm{lcm}(s,v)$ for $i = 1,\ldots,\mathrm{lcm}(t,w)$. Finding a divisor r in the residue class r_i immediately implies that the residue class $r_{i'}$, with $i' = \mathrm{lcm}(t,w) + 1 - i$ contains a divisor r'. If our algorithm does not find any divisors in the residue classes r_i for $i = 0,\ldots,\mathrm{lcm}(t,w)/2$ the remaining residue classes cannot contain any divisors. This observation immediately produces a speed up factor of 2 in the final trial division.

For the second improvement we will make the following observation: the most expensive part of the final trial division is the calculation of exact square-root of integers of length $O(\log s)$ using Newton's method. Williams and Dubner (cf. [38]) noticed that it is beneficial to check if the number is a quadratic residue modulo a number of small primes before employing Newton's method to determine if the number is a perfect square. If this is not the case, one can circumvent the calculation of the square-root. The number of primes depends on the ratio of the cost of the square-root and the reduction of the number modulo the product of all the small primes involved.

Suppose that one tries to find the divisors of n in the residue class r modulo s, and suppose that $0 \leq r' < s$ is defined by $r' \cdot r \equiv n \bmod s$. In the algorithm of [88] for each triple (a_i, b_i, c_i) the system of equations

$$a_i \cdot x + b_i \cdot y = c_i$$
$$(x \cdot \mathrm{lcm}(s,v) + r) \cdot (y \cdot \mathrm{lcm}(s,v) + r') = n$$

is solved. The elements a_i, b_i, and c_i are created during a Euclidean-like algorithm, and are all less than or equal to $\mathrm{lcm}(s,v)$. If the solution of the system of equations is a pair of non-negative integers (x,y), then n is composite. The method to solve the system of equations involves the extraction of a square-root using Newton's method.

Without solving the system of equations the algorithm has the same complexity as the Euclidean algorithm, which is $O((\log s)^2)$.

Trying to solve the system of equations modulo a small prime p not dividing $2 \cdot \mathrm{lcm}(s, v)$ can be done if $a_i \bmod p$, $b_i \bmod p$, and $c_i \bmod p$ are known. By only reducing a_0, b_0, and c_0 modulo p, and updating a_i, b_i, and c_i by using the reduction of the $(i-1)$-th convergent of the Euclidean-like algorithm (which is on average small) modulo p this can be done in $O(\log s \cdot (\log p)^2)$ binary operations. Checking whether the system of equations can be solved modulo p is equivalent to a few multiplications modulo p and a simple look-up, taking another $O(\log s \cdot (\log p)^2)$ operations.

We will determine how many primes should be taken in order to get the expected number of calls of Newton's method less than $(\log s)/B$, where $\log s$ is the expected number of iterations of the Euclidean-like algorithm. The probability that solving the system modulo p is possible is $1/2$: only half of the elements in $(\mathbf{Z}/p\mathbf{Z})^*$ are squares modulo p. Therefore $\log B$ primes should be taken.

The expected number of operations for the algorithm is now

$$O((\log s)^2 + (\log s)^3/B + \log s \cdot \sum_{j=1}^{\log B} (\log p_j)^2) =$$
$$= O((\log s)^2 + (\log s)^3/B + \log s \cdot (\log B \log\log B)^2).$$

Taking $B = \log s$ gives a quadratic algorithm in $\log s$. The constraint that the primes p do not divide $2 \cdot \mathrm{lcm}(s, v)$ is no serious restriction.

Therefore the algorithm for checking all residue-classes $r_i \equiv n^i \bmod \mathrm{lcm}(s, v)$ takes

$$O(\mathrm{lcm}(t, w)(\log s)^2)$$

binary operations.

Instead of performing all operations modulo p_i, for $i = 1, \ldots, \log B$, it is also possible to use suitable products of primes p_i. This last improvement gives another constant speed up factor.

5. GENERATED PROOF.

(5.1) Introduction.

In this section we will discuss the length of the primality proof. If the length of the primality proof cannot be bounded by a polynomial as a function of the number of bits of the prime number involved, then it is also not possible to verify the proof in polynomial time.

As has been shown by Pratt, $O((\log n)^4)$ binary operations suffice to show that a number is prime (cf. [125], I.(11.2)). This result has been improved by Pomerance, who showed that $O((\log n)^3)$ binary operations are sufficient (cf. [121], I.11).

This does not prove however that there exists an algorithm that can prove the primality of a prime number in polynomial time. This only proves that, given the correct type of certificate, one is able to verify the validity of the certificate in polynomial time. We will now analyze the structure and the length of the proof produced by our algorithm. Almost all parts of the proof can be proved to be polynomially bounded. Only the length of the last part of the proof can be proved to be not polynomially bounded.

(5.2) The structure of the primality proof.

The proof consists of 5 parts:

(a) The parameters. This part contains the parameters s, t, u, v, w provided by the optimization part of the algorithm. These parameters are chosen in such a way that $t = \exp(\mathbf{Z}/s\mathbf{Z})^*$, $u = \mathrm{ord}(n \bmod t)$, $w = \mathrm{ord}(n \bmod v)$ and finally $\mathrm{lcm}(s,v) > n^\mu$, where μ may be $\frac{1}{2}$ or $\frac{1}{3}$.

(b) The extensions. This part contains for each prime power divisor $l^e \parallel u$, the conductor m as well as the minimal polynomial $f_{l^e,m}$ of degree l^e of the extension used by the algorithm.

(c) The roots of unity. This part contains tuples of the form $(v_i, k_i, w_i, l_i, \zeta_{{v_i}^{l_i}})$, where $\mathrm{lcm}(t,v) = \prod_i {v_i}^{k_i}$. In this tuple v_i is a prime factor of $\mathrm{lcm}(t,v)$ in (5.2)(a) and k_i is the number of prime factors v_i in $\mathrm{lcm}(t,v)$, and finally $w_i = \mathrm{ord}(n \bmod {v_i}^{k_i})$. The number l_i indicates, that a ${v_i}^{l_i}$-th root of unity has been found in an extension of $\mathbf{Z}/n\mathbf{Z}$ of degree w_i. If $v_i | t$, then l_i will be equal to the number of factors v_i in t; otherwise l_i will be equal to 1. This is because of the fact that generating a v_i-th root of unity (or a ${v_i}^{k_i}$-th root of

unity) in an extension of degree w_i by the method described in IV.(5.3) suffices to show that a ${v_i}^{l_i}$-th root of unity exists in an extension of degree w_i (cf. II.(4.12)–(4.16)). Finally $\zeta_{{v_i}^{l_i}}$ will be equal to the root of unity, that has been found. It will be represented by w_i integers of length $O(\log n)$.

(d) The Jacobi sum tests. This part contains quadruples (q_i, p_i, k_i, e_i), where ${p_i}^{k_i} || q_i - 1$. Each quadruple contains the information regarding one single Jacobi sum test. This Jacobi sum test shows that $\tau(\chi)^{n-\sigma_n}$ is equal to the e_i-th power of $\zeta_{{p_i}^{k_i}}$, where χ is a character of order ${p_i}^{k_i}$ and conductor q_i, and $\tau(\chi)$ is the Gauss sum of this character. The $\zeta_{{p_i}^{k_i}}$ will be a power of one of the roots of unity in (5.2)(c).

(e) Remaining residue classes. This part contains an element of each residue class, that remains to be tested for divisors of n, when the Lucas-Lehmer type tests and all the Jacobi sum tests have been performed.

As shown in Theorem II.(7.1), this list contains the $\operatorname{lcm}(t, w)$ integers $r_i \equiv n^i \bmod \operatorname{lcm}(s, v)$, with $0 \leq r_i < \operatorname{lcm}(t, w)$.

(5.3) The length of the proof.

(a) The parameters. Since $\operatorname{lcm}(s, v) > n^{\mu}$, but certainly $\operatorname{lcm}(s, v) < n$, we have that the length of this part is $O(\log n)$.

(b) The extensions. Each prime power divisor l^e of u as well as each conductor m is of order $O(\log t)$.

(c) The roots of unity. Since $\prod_i v_i | v$ and $v < n$, and the fact that every v_i only occurs in one tuple, we have that this list contains at most $O(\log n)$ tuples. Every tuple has length $O(w_i \log n)$. Since w can serve as an upper bound for w_i, we have that the length of this part is $O(w(\log n)^2)$.

(d) The Jacobi sum tests. The number of quadruples (q_i, p_i, k_i, e_i) is at most equal to the number of pairs (p_i, q_i) with $q_i \mid s$ and $p_i \mid q_i - 1$. So this is

$$\sum_{q_i | s} \#\{p : p \mid q_i\} \leq \sum_{q_i | s} \log_2 q_i \leq \log_2 s.$$

Each entry is dominated by the size of q_i which is $O(\log t)$. Combining these results gives that the length of this part is $O((\log s)(\log t)) = O((\log n)^{1+\epsilon})$.

Remark. If one would include the Jacobi sums used by the test in the proof, each entry would be dominated by the size of the Jacobi sums involved. Each Jacobi sum test uses only $O(\sqrt{p^k})$ Jacobi sums with $\phi(p_i^{k_i})$ entries which are in absolute value less than or equal to $\sqrt{q_i} = O(\log t)$. Since $p_i^{k_i}$ is conjectured to be $O(\log t)$ we get that combining these results gives that the length of this part is $O((\log s)(\log^3 t)) = O((\log n)^{1+\epsilon})$.

(e) Remaining residue classes. The number of residue classes that have to be checked for divisors of n is equal to $\text{lcm}(t, w)$. Each integer r_i is less than $\text{lcm}(s, v)$ and therefore has binary length $O(\log n)$; since $\text{lcm}(t, w) = (\log n)^{\Theta(\log\log\log n)}$, for some effectively computable constant c, we have that the length of this part is $(\log n)^{\Theta(\log\log\log n)}$. Note that this is not polynomially bounded.

(5.4) Time to verify the primality proof.

In the worst case, the time needed to verify the proof is proportional to the time needed to check all remaining candidate-divisors, as listed in (5.2)(e). This can be done in time $O(t(\log n)^\rho)$.

In the best possible case we have that $t|w$. In that particular case we have that the time needed to verify the proof is proportional to the time needed to verify (5.2)(c). Since this is polynomial in $O(\log n)$, the best possible case gives a proof, which can be verified in polynomial time.

In the average case, the time needed to verify the proof is dominated by the time needed to verify (5.2)(e), which is $n^{\Theta(\log\log\log n)}$.

6. INVERTING AN INTEGER MATRIX.

In order to invert an integer matrix, we will distinguish here two different kinds of methods exist. The first kind uses congruence techniques, i.e., modular arithmetic, to perform elimination steps and derives the exact inverse from these results. The second kind of method uses a variant of an elimination technique, known from the theory of inverting *real* matrices, and performs this method in long integer arithmetic.

(6.1) A method using modular arithmetic.

Although some modifications of this method are known, cf. [64], we will describe a simple variant of this method. To invert a $u \times u$ non-singular matrix $A = (a_{ij})_{i,j=1}^{u}$ using the "modular" method, one has to perform steps (1) and (2).

(1) Put $d^\circ = 0$ and $A^\circ = I$, where I is the $u \times u$ identity matrix.

(2) Perform step (2a) and step (2b) for $k = 1, \ldots$, as long as $A \cdot A^\circ \neq d^\circ \cdot I$. As soon as $A \cdot A^\circ = d^\circ \cdot I$ we have $A^\circ = d^\circ \cdot A^{-1}$, and the process is terminated.

(2a) Let p_k be the k-th prime. Perform a Gaussian elimination to calculate d_k^* with $0 \leq d_k^* < p_k$ and the matrix A_k^* containing only non-negative entries which are less than p_k, such that $d^* \equiv \det(A) \bmod p^k$ and $A \cdot A_k^* \equiv d_k^* \cdot I \bmod p_k$, where I is the $u \times u$ identity matrix.

(2b) If $d^* \neq 0$ and $d^\circ = 0$ perform step (2b1); if $d^* \neq 0$ and $d^\circ \neq 0$ perform step (2b2). If $d^* = 0$ then step (2b) is completely skipped.

(2b1) Put $A^\circ = A_k^*$, and $d^\circ = d_k^*$.

(2b2) Put $A' = A^\circ$, $d' = d^\circ$, $P' = P^\circ$, and $P^\circ = P' \cdot p_k$. Use the Chinese remainder algorithm, (cf. [64]), to calculate A° such that $(a_{ij})^\circ \equiv (a_{ij})' \bmod P'$, and $(a_{ij})^\circ \equiv (a_{ij})_k^* \bmod p_k$ for $1 \leq i, j \leq u$ and to calculate d° such that $d^\circ \equiv d' \bmod P'$, and $d^\circ \equiv d_k^* \bmod p_k$. Calculate A° and d° in such a way that all entries are in absolute value less than $P^\circ/2$.

The basic idea of this method is to sequentially solve the problem modulo a list of primes, that are sufficiently small. In this way, one can perform all arithmetic operations in step (2b) in relatively small precision. Long integer arithmetic is only needed to check the solution in step (2), and in step (2b2), using the Chinese remainder algorithm.

By using the Chinese remainder algorithm, one can find the unique solution, where all entries of A° and d° have absolute values less than $P^\circ/2$.

Since one can calculate an upper bound for the size of the entries of the inverse of

A as well as for the size of $\det(A)$ by using Hadamard's inequality (cf. [64]), this process terminates after a finite number of steps.

(6.2) Complexity of the "modular" method.

Suppose that all primes p_i involved in this method are less than C. Since every elimination in step (2a) involves at most $\frac{1}{3}u^3$ multiplications and u divisions modulo a prime of size less than $\log C$, we have that each elimination step takes $O(\frac{1}{3}u^3(\log C)^\rho + u(\log C)^\rho)$ binary operations (cf. (1.7) for the definition of ρ). The exponent 3 in u^3 can be improved by using techniques presented for instance in [20].

Furthermore, in the k-th step A° contains only entries of size $O(\log P_k) = O(k \log C)$. To perform the Chinese remainder algorithm to calculate the u^2 entries of A° and the value of d°, takes $u^2 + 1$ times $O((k \log C)^\rho)$ binary operations. Checking the solution in each step takes $O(u^3(k \log C)^\rho)$ binary operations. Therefore each step takes about $O(u^3(k \log C)^\rho)$ binary operations. This implies that at most $O(u^3 k_0^{1+\rho}(\log C)^\rho)$ binary operations are needed to perform the complete algorithm, where k_0 is an upper bound for the number of steps.

To get an upper bound for k_0, we assume that the set of primes involved in this method are the first k_0 primes, with their product larger than M. By taking $k_0 = \Theta(\log M)$, we have that the product exceeds M, and that $C = O(\log M \cdot \log\log M)$.

Since it suffices to have M larger than the upper bound for the absolute value of the entries of A^{-1}, we can choose $M = \prod_{i=1}^{u} (\sum_{j=1}^{u} a_{ij}^2)^{\frac{1}{2}}$, by using Hadamard's inequality (cf. [64]). So $\log(M) = O(u(\log B + \log u))$, where $\log B$ is a bound on the size of the entries in A. Combining this with the upper bound for the total number of binary operations we get that the algorithm takes $O(u^3(u(\log B + \log u))^{1+\rho+\epsilon})$ binary operations, for any $\epsilon > 0$.

(6.3) A method using long integer arithmetic.

We will describe a method to invert the integral matrix $A = (a_{ij})_{i,j=1}^{u}$, which is similar to Gaussian elimination for real matrices.

For real matrices, the Gaussian elimination method can be regarded as decomposing the original matrix A in a product of a lower triangular matrix L, a diagonal matrix D and an upper triangular matrix U, i.e.,

$$A = L \cdot D \cdot U.$$

In order to obtain numerical stability, row and/or column interchanges can (should) be applied. In this case we get

$$P \cdot A \cdot Q = L \cdot D \cdot U,$$

where P and Q are permutation matrices reflecting the row and column interchanges respectively. In practice, one only applies row interchanges (i.e., $Q = I$, the identity matrix). This variant is called Gaussian elimination with partial pivoting on rows (i.e., $P \cdot A = L \cdot D \cdot U$).

In the case of inverting *integer* matrices, or more generally exact inverting, the problem of numerical stability is replaced by the problem to bound the growth of the elements involved.

For some variants of the Gaussian elimination one is able to construct matrices, for which the size of elements generated during the Gaussian elimination grows exponentially. One can even construct matrices such that both the matrix itself as well as its inverse matrix have small entries, but such that the elements created in the intermediate computations have a size exponential in the size of the input, see [64], [135].

The method for integral matrices will express the matrix A as the product $D^{-1} \cdot L \cdot U \cdot P$, where D is a diagonal matrix, L is a lower triangular matrix, U is an upper triangular matrix and P is a permutation matrix to limit the growth of the size of the elements generated during the decomposition. All matrices have integral coefficients.

We will describe the decomposition process in u steps. For step $k = 1, \ldots, u$, integral matrices $D^{(k)}$, $L^{(k)}$, $U^{(k)}$ and $P^{(k)}$ will be generated. During each step of the reduction process, the equation $A = (D^{(k)})^{-1} \cdot L^{(k)} \cdot U^{(k)} \cdot P^{(k)}$ remains preserved, for $k = 1, \ldots, u$.

To invert a $u \times u$ matrix $A = (a_{ij})_{i,j=1}^{u}$ using a method similar to the Gaussian elimination of real matrices, one has to perform steps (1) through (3).

(1) Initially put $L^{(1)} = A$ and $D^{(1)} = U^{(1)} = P^{(1)} = I$, where I is the $u \times u$ identity matrix.

(2) For $k = 1, \ldots, u-1$ perform steps (2a) to (2f).

(2a) Find the smallest non-zero element among $|L_{kk}^{(k)}|, \ldots, |L_{uk}^{(k)}|$. Suppose that $|L_{ik}^{(k)}|$ is the smallest element, with $i > j$. Replace $L^{(k)}$ by the matrix that is generated by exchanging the i-th and the k-th column of $L^{(k)}$. In the same way replace $P^{(k)}$ by the matrix that is generated by exchanging the i-th and the k-th row of $P^{(k)}$.

(2b) Calculate $g^{(k)} = \gcd\{L_{ik}^{(k)} : k \leq i \leq u\}$.

(2c) Put $D_{ii}^{(k+1)} = D_{ii}^{(k)}$ for $i \leq k$ and $D_{ii}^{(k+1)} = D_{ii}^{(k)} \cdot L_{kk}^{(k)} / g^{(k)}$ for $i > k$.

(2d) Put $U_{ij}^{(k+1)} = U_{ij}^{(k)}$ for $i \neq k$. Put $U_{kj}^{(k+1)} = L_{kj}^{(k)}$ for $j \geq k$ and put $U_{kj}^{(k+1)} = 0$ for $j < k$.

(2e) First put $L_{ij}^{(k+1)} = L_{ij}^{(k)}$ for $1 \leq i < k$ and $1 \leq j \leq u$ as well as for $i = k$ and $j < k$. Next put $L_{kk}^{(k+1)} = 1$ and put $L_{kj}^{(k+1)} = 0$ for $j > k$. Put $L_{ij}^{(k+1)} = L_{ij}^{(k)} \cdot L_{kk}^{(k)} / g^{(k)}$ for $k \leq i \leq u$ and $1 \leq j < k$. Put $L_{ik}^{(k+1)} = L_{ik}^{(k)}$ for $k < i \leq u$, and finally put $L_{ij}^{(k+1)} = (L_{ij}^{(k)} \cdot L_{kk}^{(k)} - L_{ik}^{(k)} \cdot L_{kj}^{(k)})/g^{(k)}$ for $k < i, j \leq u$.

(2f) Put $P^{(k+1)} = P^{(k)}$.

(3) Calculating the inverse of A can now be done by sequentially solving the systems $L \cdot X = D$, $U \cdot Y = X$, $P \cdot Z = Y$, where X, Y, and Z are $u \times u$ matrices. These are straightforward operations, since $L^{(u)}$ is a lower triangular matrix, $U^{(u)}$ is an upper triangular matrix, $D^{(u)}$ is a diagonal matrix and $P^{(u)}$ is a permutation matrix.

Remark. As explained before, Gaussian elimination for real matrices can be performed using row and/or column interchanges to obtain numerical stability. This method is often referred to as Gaussian elimination with complete pivoting. The variant for inverting integral matrices described above can be seen as a variant of Gaussian elimination with partial pivoting. In a somewhat more complicated way, one can also describe a variant similar to Gaussian elimination with complete pivoting.

(6.4) Complexity of the "long integer" method.

As can be verified quite easily, the i-th reduction step takes $O((u-i) \cdot u)$ multiplications and divisions and $O(i)$ applications of a Euclidean algorithm.

Suppose that the entries in the i-th reduction step can be bounded by B_i, we get that the number of operations in this step will be equal to $O((u-i)u(\log B_i)^\rho + i(\log B_i)^\rho)$.

The most simple upper bound for $\log B_i$ is equal to $2^i \log B$, where $\log B$ is an upper bound for the size of the entries of A. This gives an overall upper bound for the "long integer" method of

$$O(u^3(2^u \log B)^\rho + u^2(2^u \log B)^\rho)$$

binary operations.

(6.5) Conclusion.

For large u, the first method is definitely preferable over the second one, since the growth of the elements involved in the second method is hard to control. However, for small u, it seems that the growth of the elements does not yet play an important role in the complexity bounds.

It is possible to improve the second method in such a way that the size of the elements involved in the Gaussian elimination can be bounded by a polynomial in the size of the input. Instead of keeping track of a denominator for the complete inverse matrix, as has been done above, one should express each element as a rational, having its private numerator and denominator. In doing so, one can show that all the elements involved have a size that is bounded by a polynomial in the size of the input.

In the precomputation part of the primality test, one needs to generate the inverse of the matrices used to map the conjugates of a generator of a cyclic extension to the powers of the generator (cf. IV.(2.3)). The dimension of the matrices involved runs through all prime power divisors of $\lambda(t_0)$, the exponent of $(\mathbf{Z}/t_0\mathbf{Z})^*$. Even for fairly large values of t_0, these prime power values are small.

Since only matrices of small size will be inverted in the primality test, the second method is preferred over the first method. If the sizes of the matrices involved tend to grow, i.e., when the size of N grows beyond 6000 decimal digits (cf. IV.2), it seems worthwhile to use the first method.

Since the inversion most of the time takes place in the preliminary part of the algorithm, the calculation of the inverse matrices is done once and for all.

VI. PERFORMANCE.

1. INTRODUCTION.

In this chapter we will discuss the performance of the new primality test, as well as related results.

For two widely used computers, the SUN-4™ and the DEC-3100™, some basic arithmetic operations used by the primality test have been written in assembly language in order to speed up the primality test (cf. [34], [150]). The timings for these basic routines are used to give an estimate of the time needed to perform the complete test. They are given in the second section.

In the third section we will present the performance of the test on numbers up to 200 digits. This will include, apart from a table of timings, performed on 20 randomly chosen probable primes of various sizes, a discussion about the optimal choices for the parameters fed to our test.

Next we will present an example of a proof for $(2^{3539}+1)/3$ to show the capabilities of our test for larger numbers. This particular prime has 1065 decimal digits, and it was proved to be prime by F. Morain, (cf. [110]). It is the first prime of more than a thousand digits, proved to be prime by a general purpose primality test.

In the final section we will compare the test with its two major competitors, namely the old Jacobi sum test due to H. Cohen, A.K. Lenstra and H.W. Lenstra, Jr. (cf. [29] and [30]) and the complex multiplication test due to A.O.L. Atkin and F. Morain (cf. [108] and [109]). These were the two fastest general purpose primality tests known so far.

2. APPROXIMATE FUNCTIONS FOR BASIC OPERATIONS.

In IV.4 functions are given to determine the time needed for the various stages of the algorithm. These functions depend on the value for n, and the parameters s, t, u, v, w, μ, as determined in the optimization step of the algorithm, and some machine dependent constants and functions.

To determine the time needed by the various stages as accurately as possible, one needs the time needed to perform the most frequently used operations on two multi-length integers, i.e., multiplication, division, and multiplication modulo n. The last operation is performed by a method of [107] and is applied quite frequently in the algorithm.

Another function that is determined is an approximation function for the time needed to perform the final trial division, with $\mu = \frac{1}{3}$. In order to determine the time needed to perform the final trial division, one needs to determine the number of prime products which are used during the final trial division stage. These prime products are needed to check if a system of equations is solvable, without actually solving the system. See II.9, IV.6, and V.(4.5) for more details.

Given these functions, the value for n, as well as the values for s, t, u, v, and w, and the value for μ, the time needed for the primality test to complete the primality proof can be accurately predicted.

(2.1) Arithmetical operations.

First we will determine the functions that specify the time needed for multiplication, division, and multiplication modulo n. Clearly, the timing functions will be a function of the *length* of the integers involved. Usually, length signifies the binary logarithm but in this case we use the number of computer words needed to represent the integers. The functions will be determined for a SUN-4 and a DEC-3100. Both computers have a computer-word length of 32 bits. For various reasons, only 30 bits in each word will be used to store the multi-length integers. Consequently the length $l(n)$ of a multi-length integer n will be equal to $\lfloor \log_2(n)/30 \rfloor + 1$. Calculating the time needed for multiplication, division and multiplication modulo n for two multi-length integers was done by taking 100 times a random choice of two multi-length integers of length a and length b and performing the appropriate operation 100 times on these integers.

The time needed to perform multiplication modulo n of two multi-length integers only depends on the length $l(n)$ of n (cf. Remark (2.2)). The time $t_m(a,b)$, $t_d(a,b)$ and $t_n(l(n))$

for multiplication, division and multiplication modulo n of two multi-length integers respectively is measured in seconds per 10000 operations.

For the SUN-4 we obtain the following results:

	$t_m(a,b)$												
(a,b)	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.14												
2	0.22	0.35											
3	0.28	0.45	0.61										
4	0.32	0.52	0.72	0.77									
5	0.37	0.64	0.85	1.10	1.32								
6	0.46	0.69	0.98	1.29	1.53	1.88							
7	0.46	0.79	1.14	1.46	1.76	2.09	2.44						
8	0.53	0.93	1.24	1.58	1.98	2.34	2.72	3.04					
9	0.57	0.98	1.40	1.82	2.27	2.58	3.01	3.43	3.77				
10	0.67	1.05	1.49	1.88	2.44	2.87	3.31	3.78	4.19	4.64			
11	0.69	1.16	1.63	2.10	2.64	3.04	3.57	4.06	4.51	5.11	5.63		
12	0.71	1.24	1.78	2.31	2.84	3.35	3.88	4.39	5.01	5.42	6.00	6.53	
13	0.78	1.32	1.91	2.46	3.05	3.62	4.15	4.67	5.30	5.88	6.40	6.99	7.62
14	0.81	1.44	2.07	2.65	3.24	3.73	4.43	5.12	5.69	6.27	6.92	7.48	8.12
15	0.73	1.49	2.15	2.79	3.48	4.09	4.75	5.43	6.06	6.73	7.36	8.00	8.64
16	0.92	1.62	2.30	2.99	3.69	4.34	5.10	5.73	6.40	7.24	7.99	8.97	9.62
17	0.95	1.73	2.45	3.10	3.83	4.61	5.42	6.04	6.72	7.53	8.26	9.04	9.80
18	1.00	1.78	2.59	3.34	4.10	4.89	5.64	6.43	7.23	7.99	8.79	9.63	10.40
19	1.07	1.94	2.76	3.51	4.36	5.14	6.00	6.75	7.58	8.49	9.19	9.97	10.90
20	1.14	2.00	2.80	3.68	4.51	5.42	6.28	7.12	7.86	8.88	9.80	10.55	11.39
30	1.59	2.88	4.19	5.37	6.74	7.98	9.31	10.57	11.78	13.06	14.18	15.76	17.03
40	2.12	3.82	5.54	7.25	9.04	10.58	12.30	14.07	15.98	17.38	19.06	20.38	22.19
50	2.63	4.69	6.70	8.83	10.87	13.06	15.00	17.23	19.26	21.31	23.43	25.47	27.42
60	3.09	5.64	8.25	10.67	13.17	15.72	18.11	20.65	23.21	25.72	28.21	30.75	33.25
70	3.60	6.51	9.67	12.39	15.33	18.13	21.09	23.99	27.07	30.17	32.86	35.68	38.65
80	4.06	7.37	10.65	13.92	17.30	20.60	23.95	27.15	30.52	33.77	37.03	40.57	43.73
90	4.63	8.35	12.08	15.92	19.67	23.35	27.12	30.78	34.62	38.16	42.06	45.77	49.48
100	5.04	9.13	13.29	17.35	21.52	25.66	29.72	34.06	38.36	42.49	46.77	50.84	54.92
120	6.03	10.98	15.84	20.84	25.71	30.74	35.61	40.56	45.47	50.41	55.30	60.28	65.57
140	7.03	12.85	18.65	24.46	30.31	36.06	41.88	47.65	53.49	59.25	65.06	70.88	76.65
160	8.07	14.72	21.36	28.03	34.68	41.34	47.99	54.63	61.19	67.97	74.65	81.26	87.83
180	9.09	16.53	24.05	31.50	38.94	46.46	54.00	61.37	68.88	76.36	83.91	91.28	98.71
200	10.01	18.36	26.71	34.90	43.22	51.65	59.96	68.07	75.75	84.03	92.11	100.93	109.72

	$t_m(a,b)$									
(a,b)	14	15	16	17	18	19	20	30	40	50
14	8.70									
15	9.41	9.93								
16	10.11	10.87	11.26							
17	10.44	11.29	11.86	12.73						
18	11.13	11.76	12.57	13.39	14.18					
19	11.71	12.41	13.33	14.13	15.05	15.79				
20	12.23	13.03	13.92	14.95	15.51	16.48	17.24			
30	18.23	19.53	20.58	22.07	23.40	24.71	25.66	38.28		
40	24.19	25.85	27.32	29.12	30.85	32.27	33.74	50.70	67.68	
50	29.79	31.65	34.13	35.93	37.86	39.89	42.04	63.26	84.42	105.42
60	35.72	38.22	40.71	43.24	45.81	48.34	50.83	75.84	101.05	126.50
70	41.89	44.05	47.00	50.34	53.02	55.74	58.74	87.70	116.50	145.32
80	46.96	50.22	53.46	56.94	60.23	63.61	67.47	100.93	134.01	167.40
90	53.21	56.97	60.69	64.38	68.21	71.92	76.19	112.85	149.38	186.32
100	59.13	63.32	67.54	71.63	75.69	79.85	83.99	125.58	167.04	208.54
120	70.80	75.76	80.75	85.70	90.67	95.69	100.65	150.42	200.22	249.94
140	82.45	88.28	94.09	99.86	105.72	111.53	117.28	175.28	233.31	291.29
160	94.57	100.72	106.81	113.42	120.70	127.71	134.37	200.84	267.27	333.92
180	105.31	112.65	120.22	128.74	136.20	143.68	151.13	225.95	300.57	375.30
200	118.00	126.28	134.73	142.88	151.38	159.38	168.31	251.06	332.24	417.20

	$t_m(a,b)$									
(a,b)	60	70	80	90	100	120	140	160	180	200
60	151.73									
70	174.27	203.29								
80	200.61	233.95	267.20							
90	223.37	260.78	297.61	334.69						
100	250.05	291.58	333.12	374.62	414.61					
120	299.81	349.48	399.28	448.96	496.49	595.19				
140	349.38	407.72	463.88	525.16	583.33	696.69	815.85			
160	400.29	463.67	533.19	599.70	663.22	798.85	930.86	1063.16		
180	446.92	524.89	599.52	671.98	748.15	898.62	1045.00	1194.57	1346.13	
200	499.91	583.33	664.47	752.86	828.09	999.46	1162.90	1332.37	1495.11	1659.36

	$t_d(a,b)$												
(a,b)	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.29												
2	0.50	0.43											
3	0.65	0.66	0.50										
4	0.69	0.86	0.72	0.55									
5	0.80	1.03	1.02	0.78	0.57								
6	0.96	1.28	1.27	1.13	0.92	0.63							
7	1.08	1.47	1.51	1.47	1.22	1.02	0.64						
8	1.19	1.67	1.79	1.72	1.61	1.35	1.09	0.60					
9	1.35	1.96	2.04	2.04	2.00	1.79	1.47	1.15	0.70				
10	1.47	2.12	2.32	2.33	2.32	2.24	1.92	1.58	1.25	0.79			
11	1.60	2.28	2.53	2.62	2.75	2.59	2.38	2.19	1.72	1.29	0.79		
12	1.75	2.55	2.83	2.94	2.97	2.94	2.93	2.54	2.32	1.95	1.31	0.77	
13	1.87	2.80	3.03	3.21	3.38	3.34	3.31	3.07	2.80	2.46	1.95	1.46	0.75
14	1.96	2.94	3.35	3.59	3.73	3.65	3.76	3.54	3.35	3.09	2.59	2.22	1.53
15	2.09	3.11	3.60	3.89	4.09	4.19	4.20	4.06	3.91	3.58	3.34	2.90	2.22
16	2.28	3.41	3.84	4.20	4.41	4.61	4.71	4.62	4.51	4.22	3.91	3.52	3.04
17	2.43	3.64	4.09	4.49	4.82	4.95	5.12	5.10	5.02	4.82	4.55	4.12	3.75
18	2.51	3.85	4.42	4.90	5.18	5.39	5.52	5.66	5.52	5.42	5.15	4.97	4.51
19	2.58	4.01	4.61	5.15	5.48	5.73	6.00	6.08	6.12	5.99	5.87	5.46	5.20
20	2.76	4.24	4.84	5.46	5.80	6.20	6.42	6.51	6.71	6.59	6.50	6.27	5.98
30	4.00	6.39	7.53	8.54	9.43	10.26	10.96	11.53	12.12	12.50	12.91	13.16	13.26
40	5.31	8.50	10.10	11.55	13.03	14.25	15.37	16.48	17.53	18.33	19.21	19.94	20.62
50	6.52	10.70	12.74	14.65	16.47	18.20	19.93	21.49	22.91	24.30	25.57	26.72	27.88
60	7.77	12.61	15.11	17.52	19.86	22.01	24.21	26.32	28.46	30.22	32.04	33.74	35.21
70	9.07	14.83	17.84	20.81	23.71	26.20	28.83	31.39	33.80	36.27	38.17	40.54	42.56
80	10.30	16.80	20.25	23.65	26.78	30.04	33.45	36.43	39.22	41.89	44.82	47.40	49.95
90	11.64	19.13	22.95	26.84	30.70	34.23	37.75	41.29	44.77	47.89	50.65	53.84	56.91
100	12.80	21.17	25.72	29.86	34.21	38.29	42.36	46.24	50.02	53.95	57.53	61.22	64.20
120	15.39	25.31	30.83	36.08	41.33	46.23	51.36	56.23	61.01	65.07	69.53	74.08	78.68
140	17.95	29.67	36.01	42.23	48.33	54.36	60.23	66.07	71.72	77.55	82.89	88.64	93.94
160	20.46	33.88	41.12	48.32	55.29	62.49	69.43	76.09	82.69	89.36	95.81	102.26	108.45
180	22.67	37.78	46.14	54.45	62.46	70.47	78.32	85.90	93.60	101.31	108.62	116.20	123.27
200	25.51	42.16	51.63	60.66	69.64	78.27	86.66	95.20	104.12	112.73	121.56	130.12	138.08

	$t_d(a,b)$									
(a,b)	14	15	16	17	18	19	20	30	40	50
14	0.78									
15	1.59	0.84								
16	2.38	1.75	0.92							
17	3.04	2.54	1.75	0.90						
18	3.90	3.37	2.67	1.87	0.96					
19	4.80	4.17	3.49	2.76	1.86	0.95				
20	5.54	4.98	4.40	3.58	2.88	1.98	1.01			
30	13.40	13.21	13.13	13.06	12.47	12.09	11.65	1.30		
40	21.06	21.53	21.92	22.07	22.24	22.48	22.32	16.69	1.61	
50	28.92	29.62	30.67	31.14	31.78	32.10	32.85	31.75	21.43	1.98
60	36.73	38.15	39.33	40.37	41.60	42.79	43.50	47.43	41.89	26.76
70	44.52	46.47	48.18	49.80	51.35	52.77	54.14	62.76	61.37	51.22
80	52.16	54.51	56.87	59.14	60.98	63.05	64.79	78.14	82.22	76.41
90	59.59	62.29	65.68	68.23	70.96	73.18	75.61	93.67	102.22	101.37
100	67.17	70.59	74.04	77.14	80.40	83.40	86.12	108.88	122.36	126.27
120	83.56	87.79	92.25	95.92	99.77	103.65	107.66	139.73	162.75	175.79
140	99.17	104.23	109.15	113.35	118.41	124.02	128.80	170.76	202.82	225.64
160	114.22	119.80	125.92	132.72	138.83	144.33	150.21	201.02	243.59	275.52
180	131.06	137.42	144.53	149.83	156.89	165.10	172.92	231.58	282.84	326.38
200	146.53	154.50	162.22	170.27	177.35	184.12	192.58	263.70	323.41	375.57
(a,b)	60	70	80	90	100	120	140	160	180	200
60	2.22									
70	31.41	2.59								
80	61.13	37.15	2.88							
90	91.25	71.04	41.93	3.37						
100	120.62	105.90	81.00	47.24	3.20					
120	179.72	172.69	158.32	135.23	100.83	3.86				
140	238.84	243.03	235.58	221.03	197.88	120.02	4.54			
160	295.48	311.13	315.02	309.74	294.45	236.17	138.68	4.99		
180	356.39	382.84	394.82	394.70	391.28	352.67	274.92	159.97	5.85	
200	414.58	446.91	474.10	485.67	489.84	464.23	410.29	313.33	180.75	5.95

	$t_n(l(n))$										
$l(n)$	1	2	3	4	5	6	7	8	9	10	11
	0.54	0.95	1.68	2.26	3.19	4.23	5.51	6.89	8.44	10.16	12.01
$l(n)$	12	13	14	15	16	17	18	19	20	30	40
	14.17	16.44	18.72	21.21	23.85	26.78	29.86	33.04	36.37	78.90	138.39
$l(n)$	50	60	70	80	90	100	120	140	160	180	200
	216.05	310.64	422.84	554.99	702.85	863.01	1244.95	1689.91	2202.07	2781.43	3568.37

For the DEC-3100 we get:

	$t_m(a,b)$												
(a,b)	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.12												
2	0.14	0.17											
3	0.15	0.23	0.31										
4	0.19	0.28	0.38	0.45									
5	0.21	0.30	0.41	0.52	0.63								
6	0.22	0.35	0.47	0.58	0.75	0.82							
7	0.25	0.38	0.50	0.65	0.79	0.93	1.09						
8	0.27	0.41	0.58	0.72	0.88	1.02	1.17	1.32					
9	0.30	0.45	0.61	0.79	0.93	1.12	1.28	1.45	1.62				
10	0.37	0.49	0.66	0.86	1.07	1.21	1.41	1.58	1.73	1.95			
11	0.33	0.53	0.73	0.92	1.13	1.31	1.49	1.71	1.89	2.09	2.29		
12	0.35	0.57	0.78	0.98	1.22	1.39	1.63	1.79	2.04	2.23	2.46	2.66	
13	0.38	0.60	0.81	1.05	1.27	1.52	1.71	1.95	2.16	2.41	2.60	2.85	3.07
14	0.41	0.64	0.87	1.11	1.35	1.59	1.84	2.07	2.32	2.55	2.77	3.02	3.26
15	0.43	0.72	0.93	1.19	1.44	1.70	1.96	2.21	2.43	2.71	3.02	3.25	3.47
16	0.45	0.71	0.99	1.25	1.52	1.79	2.06	2.33	2.59	2.86	3.12	3.38	3.67
17	0.46	0.76	1.16	1.32	1.59	1.88	2.17	2.44	2.73	3.03	3.27	3.59	3.88
18	0.49	0.79	1.09	1.38	1.68	2.00	2.27	2.52	2.88	3.14	3.44	3.78	4.06
19	0.50	0.84	1.14	1.47	1.77	2.06	2.38	2.68	3.01	3.32	3.65	4.00	4.26
20	0.54	0.87	1.18	1.52	1.85	2.14	2.49	2.84	3.15	3.45	3.81	4.09	4.44
30	0.76	1.23	1.70	2.18	2.64	3.10	3.60	4.00	4.55	5.00	5.50	5.98	6.46
40	0.99	1.59	2.23	2.82	3.46	4.09	4.73	5.32	5.96	6.55	7.17	7.80	8.46
50	1.21	1.98	2.75	3.51	4.27	5.06	5.83	6.61	7.35	8.12	8.87	9.64	10.39
60	1.44	2.34	3.26	4.17	5.10	6.03	6.92	7.81	8.75	9.66	10.58	11.47	12.41
70	1.63	2.73	3.80	4.85	5.89	6.96	8.05	9.07	10.11	11.24	12.30	13.31	14.42
80	1.87	3.10	4.29	5.48	6.70	7.92	9.11	10.32	11.49	12.75	13.93	15.19	16.36
90	2.13	3.47	4.80	6.16	7.50	8.88	10.24	11.62	12.98	14.33	15.62	17.02	18.34
100	2.33	3.82	5.33	6.83	8.30	9.81	11.30	12.78	14.31	15.80	17.31	18.82	20.35
120	2.84	4.55	6.35	8.14	9.95	11.73	13.54	15.32	17.11	18.91	20.69	22.46	24.25
140	3.23	5.31	7.41	9.49	11.60	13.66	15.71	17.81	19.87	22.03	24.10	26.15	28.27
160	3.67	6.00	8.42	10.81	13.19	15.58	17.98	20.35	22.68	25.09	27.45	29.85	32.22
180	4.12	6.80	9.47	12.14	14.78	17.47	20.15	22.82	25.49	28.17	30.81	33.48	36.20
200	4.56	7.52	10.51	13.46	16.41	19.38	22.35	25.33	28.31	31.21	34.13	37.13	40.10

	$t_m(a,b)$									
(a,b)	14	15	16	17	18	19	20	30	40	50
14	3.52									
15	3.73	4.01								
16	3.93	4.20	4.46							
17	4.14	4.41	4.72	5.01						
18	4.34	4.66	4.93	5.25	5.58					
19	4.58	4.86	5.22	5.48	5.80	6.16				
20	4.77	5.09	5.45	5.79	6.08	6.41	6.76			
30	6.92	7.40	7.88	8.35	8.81	9.27	9.75	14.55		
40	9.04	9.72	10.29	10.89	11.51	12.14	12.79	18.98	25.24	
50	11.18	11.95	12.74	13.47	14.26	15.03	15.81	23.39	31.09	38.84
60	13.30	14.21	15.13	16.03	16.90	17.91	18.78	27.94	37.08	46.27
70	15.44	16.50	17.67	18.70	19.72	20.76	21.73	32.45	42.91	53.49
80	17.58	18.74	20.06	21.21	22.52	23.66	24.85	36.96	48.84	60.90
90	19.73	21.05	22.45	23.82	25.16	26.45	27.91	41.38	54.98	68.28
100	21.87	23.43	24.88	26.32	27.86	29.39	30.81	45.84	60.84	75.86
120	26.07	27.87	29.68	31.48	33.26	35.04	36.85	54.75	72.66	90.56
140	30.40	32.49	34.53	36.57	38.62	40.83	42.81	63.65	84.55	105.27
160	34.66	37.02	39.40	41.76	44.09	46.50	48.91	72.70	96.48	120.33
180	38.81	41.52	44.18	46.83	49.50	52.15	54.81	81.52	108.24	134.90
200	43.05	46.04	49.00	52.02	55.11	58.12	61.05	90.65	120.53	150.17

	$t_m(a,b)$									
(a,b)	60	70	80	90	100	120	140	160	180	200
60	55.47									
70	64.07	74.83								
80	73.03	85.00	97.29							
90	81.76	95.48	108.88	122.38						
100	90.86	105.66	120.55	135.56	150.63					
120	108.43	126.38	144.35	162.68	180.98	216.45				
140	126.15	146.96	167.78	188.86	209.68	251.48	293.18			
160	144.12	167.84	191.74	215.45	239.27	286.72	334.03	381.47		
180	161.54	188.35	215.15	241.67	268.51	322.60	376.26	429.51	481.84	
200	179.56	208.99	238.92	268.71	298.20	357.80	416.87	476.23	535.15	594.56

	$t_d(a,b)$												
(a,b)	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.16												
2	0.28	0.29											
3	0.37	0.47	0.32										
4	0.45	0.62	0.51	0.34									
5	0.51	0.75	0.68	0.54	0.35								
6	0.58	0.96	0.86	0.73	0.60	0.40							
7	0.69	1.10	1.05	0.95	0.79	0.63	0.40						
8	0.77	1.25	1.23	1.12	1.07	0.84	0.68	0.40					
9	0.84	1.39	1.41	1.30	1.20	1.11	0.91	0.69	0.39				
10	0.92	1.56	1.59	1.52	1.43	1.36	1.16	0.96	0.72	0.47			
11	1.00	1.73	1.74	1.73	1.69	1.56	1.44	1.22	1.05	0.77	0.47		
12	1.17	1.92	1.93	1.93	1.92	1.80	1.68	1.55	1.36	1.10	0.81	0.47	
13	1.16	2.09	2.17	2.12	2.10	2.04	1.93	1.85	1.64	1.44	1.15	0.86	0.51
14	1.23	2.27	2.32	2.38	2.33	2.25	2.21	2.07	1.93	1.75	1.47	1.26	0.92
15	1.31	2.39	2.50	2.55	2.57	2.56	2.50	2.38	2.23	2.01	1.88	1.55	1.23
16	1.41	2.55	2.72	2.80	2.83	2.79	2.75	2.64	2.51	2.38	2.17	2.00	1.64
17	1.49	2.75	2.85	2.99	2.98	3.05	3.01	2.93	2.86	2.71	2.54	2.31	1.98
18	1.57	2.88	3.04	3.15	3.21	3.29	3.29	3.27	3.10	3.00	2.82	2.65	2.40
19	1.64	3.06	3.22	3.34	3.43	3.51	3.55	3.50	3.44	3.27	3.18	3.00	2.75
20	1.73	3.23	3.38	3.54	3.64	3.77	3.78	3.82	3.75	3.60	3.55	3.43	3.16
30	2.54	4.86	5.25	5.59	5.88	6.20	6.42	6.62	6.83	6.84	6.94	7.00	6.98
40	3.34	6.45	6.98	7.61	8.20	8.62	9.03	9.48	9.79	10.11	10.42	10.59	10.77
50	4.14	8.05	8.90	9.61	10.34	10.99	11.64	12.28	12.75	13.26	13.80	14.19	14.55
60	4.96	9.71	10.69	11.64	12.60	13.46	14.24	15.05	15.92	16.48	17.22	17.83	18.36
70	5.78	11.30	12.48	13.70	14.87	15.90	16.87	17.84	18.80	19.75	20.50	21.46	22.17
80	6.57	13.01	14.34	15.68	17.02	18.30	19.60	20.83	21.75	22.84	24.11	25.03	26.11
90	7.39	14.55	16.16	17.78	19.26	20.72	21.98	23.55	24.88	26.09	27.35	28.73	29.93
100	8.19	16.13	18.00	19.73	21.48	23.21	24.81	26.37	27.83	29.41	30.77	32.35	33.62
120	9.80	19.42	21.67	23.83	25.98	28.00	30.05	32.10	33.93	35.83	37.52	39.40	41.40
140	11.45	22.69	25.29	27.87	30.36	32.90	35.16	37.59	39.79	42.27	44.33	46.70	48.98
160	13.04	25.93	28.96	32.00	34.72	37.92	40.54	43.32	45.97	48.61	51.36	53.82	56.51
180	14.64	29.13	32.42	35.96	39.23	42.68	45.73	48.84	52.07	55.24	58.23	61.14	64.17
200	16.26	32.37	36.12	39.77	43.73	47.40	50.86	54.53	57.89	61.52	65.06	68.42	71.55

	$t_d(a,b)$									
(a,b)	14	15	16	17	18	19	20	30	40	50
14	0.57									
15	0.97	0.54								
16	1.35	1.00	0.57							
17	1.72	1.38	1.02	0.58						
18	2.15	1.80	1.43	1.04	0.61					
19	2.53	2.20	1.88	1.51	1.10	0.63				
20	2.92	2.67	2.28	1.92	1.55	1.12	0.61			
30	6.91	6.82	6.73	6.58	6.34	6.05	5.85	0.86		
40	10.87	11.09	11.14	11.08	11.13	11.19	11.05	8.02	1.11	
50	14.89	15.12	15.50	15.79	15.97	15.97	16.44	15.21	10.17	1.35
60	18.93	19.41	19.92	20.41	20.75	21.13	21.42	22.31	19.38	12.37
70	23.00	23.73	24.37	25.07	25.48	26.02	26.51	29.52	28.50	23.63
80	26.82	27.75	28.67	29.65	30.12	31.06	31.73	36.69	37.79	34.59
90	30.92	31.95	33.20	34.32	35.20	36.11	37.09	43.87	46.92	45.82
100	34.82	36.27	37.81	38.79	39.81	41.20	42.20	50.98	56.12	57.09
120	42.94	44.64	46.41	48.11	49.41	51.03	52.80	65.50	74.55	79.33
140	50.97	52.99	55.23	57.24	59.21	61.16	63.11	80.08	92.90	101.56
160	59.12	61.43	63.88	66.46	68.66	71.01	73.68	93.92	111.52	124.13
180	67.04	69.90	72.78	75.57	77.94	81.14	83.90	108.58	129.75	146.31
200	75.33	78.25	81.78	85.09	87.78	91.33	94.19	123.57	148.58	168.96

	$t_d(a,b)$									
(a,b)	60	70	80	90	100	120	140	160	180	200
60	1.48									
70	14.53	1.64								
80	27.64	16.88	1.88							
90	41.25	31.88	18.96	2.19						
100	54.30	47.22	36.05	21.38	2.18					
120	80.30	77.25	70.51	59.65	44.51	2.64				
140	106.37	107.67	104.39	97.19	86.85	52.82	3.10			
160	132.66	137.69	138.49	135.79	128.63	103.36	61.54	3.44		
180	159.38	168.34	172.42	173.93	170.88	152.92	119.59	70.25	3.88	
200	185.53	198.66	208.00	212.81	213.59	203.80	178.11	136.54	78.39	4.22

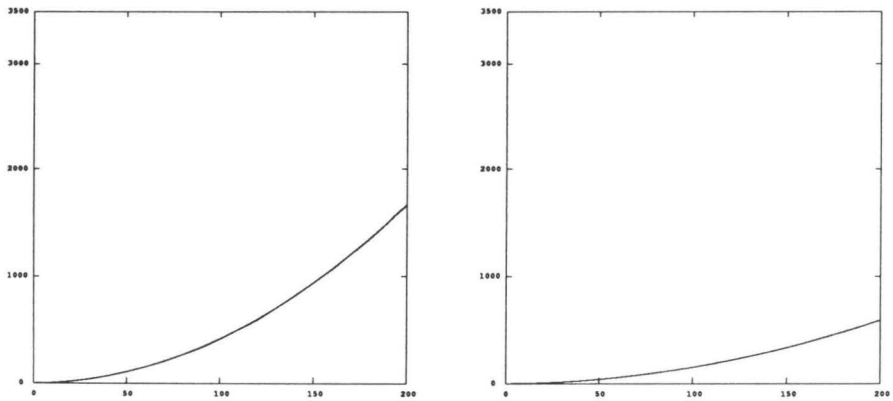
	$t_n(l(n))$										
$l(n)$	1	2	3	4	5	6	7	8	9	10	11
	0.36	0.54	0.78	1.06	1.47	1.88	2.36	2.92	3.53	4.24	4.93
$l(n)$	12	13	14	15	16	17	18	19	20	30	40
	5.69	6.53	7.47	8.42	9.43	10.50	11.68	12.87	14.12	30.08	51.81
$l(n)$	50	60	70	80	90	100	120	140	160	180	200
	79.57	113.18	152.98	198.55	249.96	307.47	440.28	596.74	777.02	981.12	1208.62

To find a function which gives an approximation for the time needed to perform these operations one has to solve a *least squares problem*, which can numerically be done by for instance using a routine from [54]. Doing so for these routines, one gets the following functions

SUN-4	$t_m(a,b)$	$0.0415 \cdot a \cdot b + 0.0032 \cdot a + 0.0005 \cdot b + 0.4040$
SUN-4	$t_d(a,b)$	$0.0474 \cdot a \cdot b - 0.0474 \cdot b^2 + 0.1137 \cdot a - 0.0841 \cdot b + 0.3719$
SUN-4	$t_n(l(n))$	$0.0892 \cdot (l(n))^2 + 0.3830$
DEC-3100	$t_m(a,b)$	$0.01467 \cdot a \cdot b - 0.00001 \cdot b^2 + 0.00772 \cdot a + 0.03324 \cdot b + 0.06065$
DEC-3100	$t_d(a,b)$	$0.0200 \cdot a \cdot b - 0.0200 \cdot b^2 + 0.1173 \cdot a - 0.0981 \cdot b + 0.2314$
DEC-3100	$t_n(l(n))$	$0.0297 \cdot (l(n))^2 + 0.1020 \cdot l(n) + 0.2060$

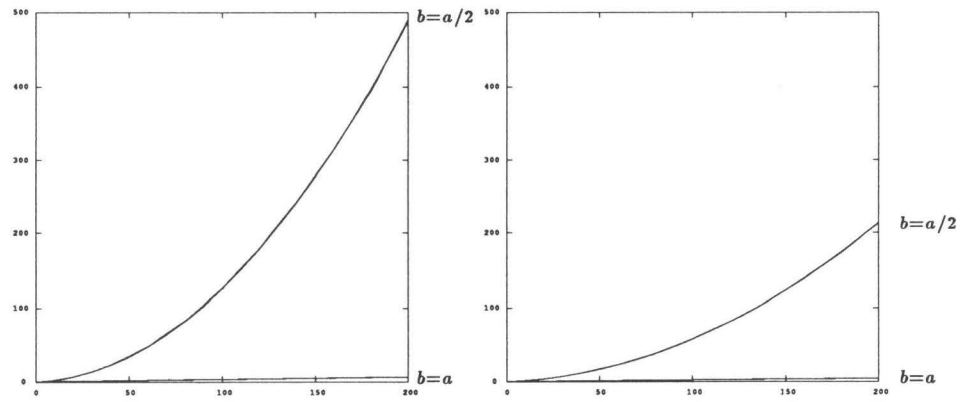
(2.2) Remark. The time to perform multiplication modulo n is independent of a and b, since this operation is performed on integers which have approximately the same length as the length $l(n)$ of n. For more details about this modular multiplication method, see [107].

(2.3) Remark. If we graphically display these least squares functions, along with the original data, we get that the curves of the data and the curves of the functions determined by solving the least squares problem are approximately the same. The pictures are displayed below. The values a and b are equal to the lengths of the integers involved.

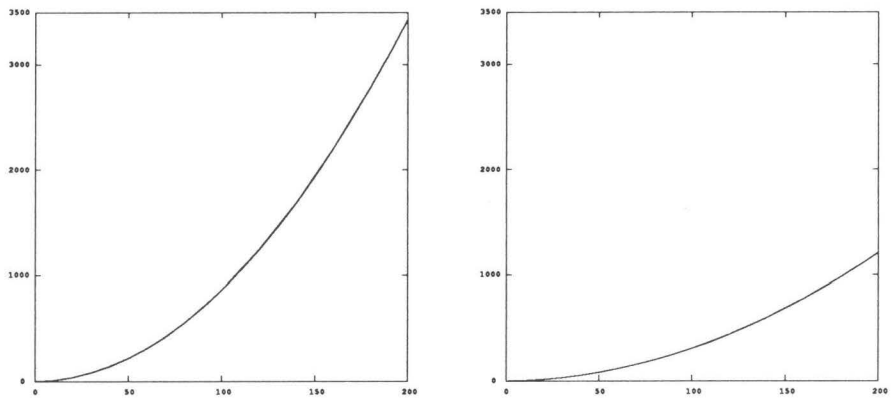


multiplication on the SUN *multiplication on the DEC*

$t_m(a,b)$ with $b = a$ as a function of a, where $0 \leq a \leq 200$ and $0 \leq t_m(a,b) \leq 3500$.



division on the SUN *division on the DEC*
$t_d(a,b)$ *with* $b \in \{a/2, a\}$ *as a function of* a, *where* $0 \leq a \leq 200$ *and* $0 \leq t_d(a,b) \leq 500$.



multiplication modulo n *on the SUN* *multiplication modulo* n *on the DEC*
$t_n(l(n))$, *where* $0 \leq l(n) \leq 200$ *and* $0 \leq t_n(l(n)) \leq 3500$.

(2.4) Final trial division.

In the final trial division with $\mu = \frac{1}{3}$ the most expensive part is solving a system of equations. In order to reduce the time needed for this stage, one uses a number of prime products to perform certain operations modulo these prime products instead of much larger numbers. See II.9, IV.6, and V.(4.5) for more details. The time needed for the final trial division will be a function of the length of n and the number k of prime products used in this stage. By varying the number k of prime products for 40 randomly selected values n of length $d = \lfloor \log_2(n)/30 \rfloor + 1$ one can determine the time needed to try and find divisors of n in the residue classes $r_i \equiv n^i \bmod s$ for $i = 1, \ldots, 100$ for each value of n. Here s is

chosen approximately $\sqrt[3]{n}$. The average of these 4000 residue classes will be scaled to 1000 residue classes in the table. For the SUN-4 we obtain the following results

	k									
d	1	2	3	4	5	6	7	8	9	10
1	4.93	4.06	**3.98**	4.27	4.42	4.54	4.69	4.96	5.26	5.46
2	10.96	8.00	6.65	**6.46**	6.96	7.21	7.72	8.19	8.69	9.33
3	17.44	12.33	9.33	**9.02**	9.79	10.09	10.56	11.39	11.98	12.83
4	31.50	19.32	13.79	**12.37**	12.96	13.66	14.49	15.08	15.98	16.89
5	44.33	25.69	16.85	**15.49**	15.67	16.29	17.51	18.33	19.59	20.46
6	59.85	31.74	21.30	**18.59**	18.65	19.29	20.62	21.78	23.00	24.09
7	85.42	46.23	26.95	22.87	**22.16**	22.54	24.24	25.66	27.13	28.32
8	115.82	70.08	32.96	26.18	**24.87**	25.47	26.88	28.79	30.39	32.05
9	155.85	68.59	40.74	30.94	**28.63**	28.71	30.37	32.16	34.12	35.84
10	178.07	84.11	48.83	34.35	**32.38**	32.81	34.20	36.07	38.20	40.59
11		97.33	57.25	39.40	**35.33**	35.57	37.46	39.19	41.53	44.01
12		119.26	66.12	42.99	**38.99**	39.33	40.37	42.55	44.99	47.65
13		143.53	75.86	49.45	**43.01**	43.60	44.93	46.29	48.78	51.21
14			88.99	55.84	47.95	**46.08**	47.99	49.98	52.72	55.56
15			99.23	64.59	51.92	**49.64**	50.53	52.76	55.65	58.44
16			112.58	72.14	57.75	**54.21**	55.48	58.35	60.69	63.75
17			130.71	78.25	61.83	**58.81**	60.52	61.63	64.45	67.58
18			143.04	87.84	66.37	**60.25**	62.18	65.38	67.48	70.63
19			153.89	88.07	71.64	**66.77**	66.83	69.40	72.86	76.44
20			201.29	102.58	76.32	**69.85**	70.80	73.76	76.68	80.39
25				141.02	100.38	**91.89**	93.45	96.60	98.80	102.85
30				200.58	142.84	119.17	**114.01**	116.69	119.71	125.52
35				337.10	193.35	147.32	**139.06**	141.73	146.58	152.01
40				375.82	232.09	178.32	**164.37**	166.01	172.02	182.41
45				447.92	270.01	210.44	**192.74**	193.38	196.73	205.40
50					380.03	279.06	263.99	**229.61**	233.56	234.90
60					520.03	331.16	296.18	**285.58**	286.56	298.96
70					660.68	431.40	372.22	**358.83**	363.35	373.09
80						608.88	455.71	**439.04**	440.43	444.62
90						727.30	563.90	**515.81**	520.44	526.72
100						979.65	674.31	616.76	**607.72**	619.61
110						1180.74	787.20	701.99	**699.89**	711.98
120						1415.85	935.66	812.66	**796.68**	809.34
130						1781.11	1078.48	993.95	**916.77**	927.13
140						1996.71	1317.54	1136.80	**1049.68**	1054.38
150						2201.46	1419.66	1209.56	**1143.21**	1145.66

By fixing d and varying k one can observe that there exists a local minimum for each d.

This minimum is made bold in the table. For the DEC-3100 we get

	k									
d	1	2	3	4	5	6	7	8	9	10
1	2.65	2.20	**2.11**	2.21	2.37	2.54	2.69	2.86	3.05	3.31
2	6.04	4.51	3.74	**3.68**	3.86	4.12	4.40	4.67	4.98	5.32
3	10.27	6.65	5.47	**5.30**	5.43	5.75	6.06	6.46	6.81	7.22
4	16.61	10.75	8.09	**7.61**	7.78	8.09	8.52	8.98	9.57	10.06
5	23.61	14.66	10.34	**9.52**	9.59	9.93	10.45	10.96	11.55	12.15
6	30.25	17.92	12.82	11.31	**11.29**	11.75	12.30	12.90	13.61	14.31
7	45.27	24.65	16.18	13.86	**13.62**	13.87	14.65	15.42	16.21	16.97
8	58.61	30.42	19.72	16.22	**15.48**	15.84	16.62	17.45	18.28	19.18
9	66.16	36.50	22.85	18.14	**17.47**	17.80	18.59	19.62	20.48	21.48
10	83.51	43.52	27.06	20.96	**19.98**	20.30	21.05	22.07	23.07	24.39
11		56.29	30.89	23.71	**22.18**	22.38	23.10	24.29	25.32	26.67
12		62.53	35.09	26.26	**24.07**	24.38	25.16	26.50	27.61	29.00
13		75.43	41.60	29.75	**26.83**	26.84	27.81	29.10	30.38	31.83
14			49.34	32.91	29.61	**28.99**	29.92	31.38	32.59	34.19
15			52.13	37.08	31.86	**31.20**	32.11	33.60	35.02	36.70
16			60.75	40.58	34.65	**33.93**	35.03	36.54	37.93	39.79
17			68.10	43.66	37.53	**36.29**	37.31	38.87	40.31	42.29
18			78.27	47.53	39.93	**38.51**	39.61	41.26	42.91	44.82
19			83.46	52.70	43.15	**41.94**	42.53	44.24	45.86	47.94
20			91.96	58.07	45.82	**44.23**	44.89	46.52	48.29	50.42
25				85.63	62.21	**58.81**	58.94	60.80	62.92	65.50
30				115.40	83.11	73.87	**73.20**	75.09	77.69	80.60
35				164.11	106.01	91.07	**89.41**	91.42	94.14	97.60
40				198.04	127.66	109.88	**107.46**	109.19	112.14	115.92
45				261.89	160.03	131.19	**124.35**	126.02	129.40	133.51
50					197.56	153.42	**143.65**	145.75	149.09	153.85
60					299.76	216.18	188.44	**187.13**	190.72	195.69
70					384.30	267.79	238.35	**234.66**	238.05	244.21
80						332.02	293.99	**284.12**	287.31	293.71
90						401.67	350.96	**339.33**	341.26	348.07
100						523.67	420.85	**400.60**	401.76	409.30
110						651.25	496.69	465.39	**463.75**	470.69
120						735.69	585.71	532.93	**530.86**	537.31
130						899.75	675.35	607.85	**604.01**	610.86
140						1025.96	757.96	689.95	**677.46**	685.03
150						1187.95	897.31	775.78	**760.70**	763.50

Using this information, one can easily choose the optimal value of k for each value of d. Fixing k to these optimal values, one can determine the least squares function $t_f(d)$ which

gives an approximation for the the time needed for 1000 final trial divisions on integers n of length d. Using the least squares method gives $t_f(d) = 0.032 \cdot d^2 + 2.848 \cdot d + 0.362$ for the SUN-4 and $t_f(d) = 0.022 \cdot d^2 + 1.831 \cdot d - 0.643$ for the DEC-3100.

3. PERFORMANCE OF THE TEST.

Using the results from the previous section, makes it possible to give an accurate approximation of the time needed to complete the primality test, without performing the test itself.

This observation will be used to determine parameters which have to be fed to the algorithm. In particular, a trial division bound $B^{\#}$ and an upper bound $W^{\#}$ for ω in $n^{\omega}-1$ will be determined. In this way we can determine these bounds as a function of the size of n. The algorithm first tries to find all divisors less than a bound $B^{\#}$ in $n^{\omega}-1$, for $\omega = 1, \ldots, W^{\#}$, before trying to find the best values for the parameters in the algorithm.

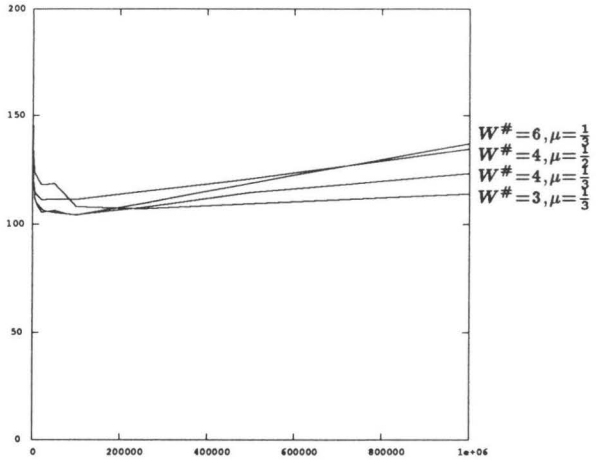
We will now describe the method to find the optimal values of $B^{\#}$, $W^{\#}$ and μ. Suppose we fix choices for $B^{\#}$, $W^{\#}$, and μ. We can estimate the time needed by the algorithm by taking the sum of the time needed to find all prime divisors less than $B^{\#}$ of $n^{\omega}-1$ for $\omega = 1, \ldots, W^{\#}$ and the expected time for the test with the optimal values of the parameters s, t, u, v and w found by the optimization stage of the algorithm. This will be done for 20 randomly chosen probable primes n with $\log_{10}(n) = 100, 120, 140, 160, 180, 200$. For fixed choices of $\log_{10}(n)$, $W^{\#}$, and μ we can express the time needed by the algorithm to complete the test as a function of $B^{\#}$. Taking the choices for which the minimal time needed for the test is expected to be found, gives good initial values for $B^{\#}$, $W^{\#}$, and μ. This minimization-method will be performed for a SUN-4.

The reason that we perform the test for values of $\log_{10}(n)$, instead of $\lfloor \log_2(n)/30 \rfloor$ as we did in the previous section to determine machine dependent functions, is mainly historical. All previously introduced primality tests express the CPU-time as a function of $\log_{10}(n)$.

The list of "randomly chosen probable primes" is generated by taking 20 times a random odd integer n_0 of the proper size, checking if this integer is divisible by primes less than 10^6, and performing four Miller-Rabin probabilistic compositeness tests on the number. If such an integer is not proved to be composite by these tests, then it is added to the list of "randomly chosen probable primes". If the number is proved to be composite by one of the compositeness tests, the tests are repeated on the integers $n_1 = n_0 + 2, n_2 = n_0 + 4, \ldots$, until an integer n_i for some $i \geq 0$ is found that is not proved to be composite by any of four compositeness tests; n_i will then be added to the list.

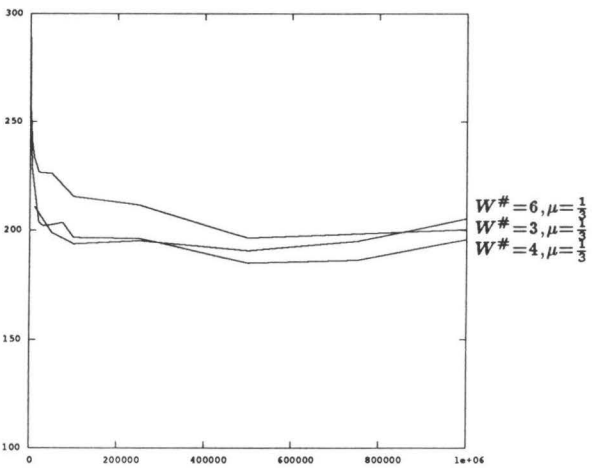
The results on these "probable primes" are graphically presented by separate pictures for each size of n. In these pictures each line gives the expected time for the complete test

as a function of the trial division bound $B^{\#}$. For each line in the picture we have a fixed choice of $W^{\#}$ and μ.
The results are for $\log_{10}(n) = 100$:



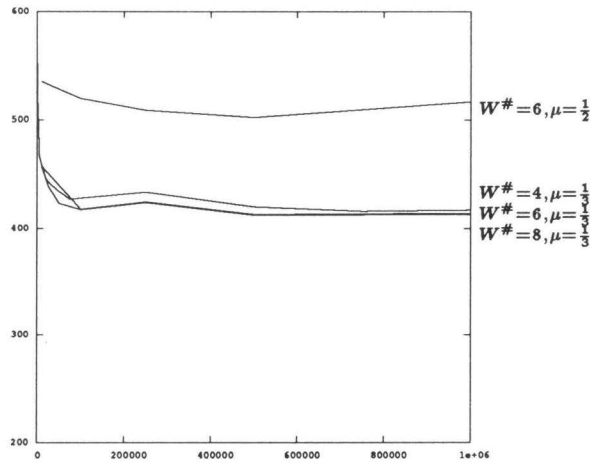
The time T needed to complete the test (in seconds CPU) as a function of $B^{\#}$ for several pairs $(\mu, W^{\#})$ where $0 \leq B^{\#} \leq 10^6$ and $0 \leq T \leq 200$. The minimum is attained for $(B^{\#}, W^{\#}, \mu) = (100000, 4, \frac{1}{3})$.

for $\log_{10}(n) = 120$:



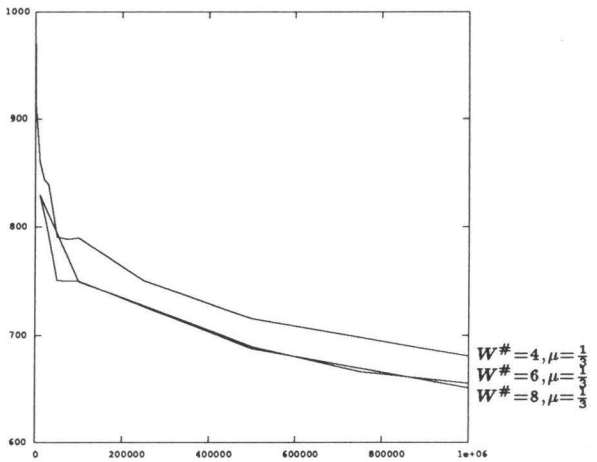
The time T needed to complete the test (in seconds CPU) as a function of $B^{\#}$ for several pairs $(\mu, W^{\#})$ where $0 \leq B^{\#} \leq 10^6$ and $100 \leq T \leq 300$. The minimum is attained for $(B^{\#}, W^{\#}, \mu) = (500000, 4, \frac{1}{3})$.

for $\log_{10}(n) = 140$:



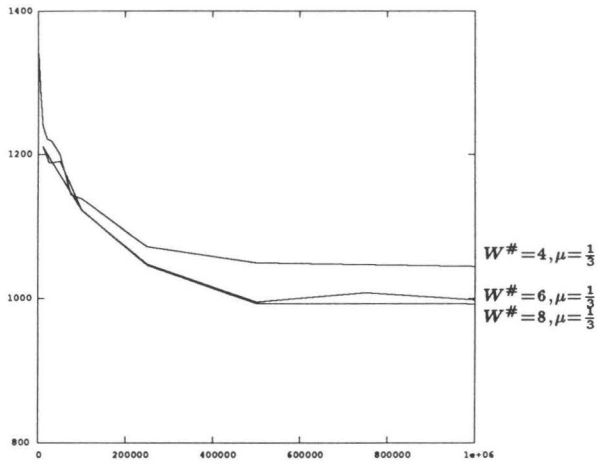
The time T needed to complete the test (in seconds CPU) as a function of $B^{\#}$ for several pairs $(\mu, W^{\#})$ where $0 \leq B^{\#} \leq 10^6$ and $200 \leq T \leq 600$. The minimum is attained for $(B^{\#}, W^{\#}, \mu) = (500000, 8, \frac{1}{3})$.

for $\log_{10}(n) = 160$:



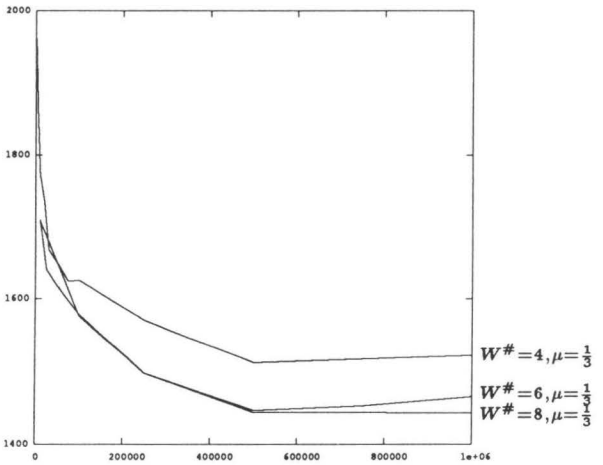
The time T needed to complete the test (in seconds CPU) as a function of $B^{\#}$ for several pairs $(\mu, W^{\#})$ where $0 \leq B^{\#} \leq 10^6$ and $600 \leq T \leq 1000$. The minimum is attained for $(B^{\#}, W^{\#}, \mu) = (1000000, 8, \frac{1}{3})$.

for $\log_{10}(n) = 180$:



The time T needed to complete the test (in seconds CPU) as a function of $B^{\#}$ for several pairs $(\mu, W^{\#})$ where $0 \leq B^{\#} \leq 10^6$ and $800 \leq T \leq 1400$. The minimum is attained for $(B^{\#}, W^{\#}, \mu) = (1000000, 8, \frac{1}{3})$.

for $\log_{10}(n) = 200$:



The time T needed to complete the test (in seconds CPU) as a function of $B^{\#}$ for several pairs $(\mu, W^{\#})$ where $0 \leq B^{\#} \leq 10^6$ and $1400 \leq T \leq 2000$. The minimum is attained for $(B^{\#}, W^{\#}, \mu) = (1000000, 8, \frac{1}{3})$.

Remark. Notice that for the last few examples the minimum is found at $B^{\#} = 1000000$. This does not need to be a global minimum; it might only be a minimum on the boundary. This is due to the fact that the prime table which is generated in the preliminary stage and which is used in the trial division step does not contain more primes.

At this point we are able to present a table of the actual performance of our algorithm on the 20 randomly chosen probable primes of size $\log_{10}(n) = 100, 120, 140, 160, 180, 200$. For each stage we will list the average time, the standard deviation, the maximum time as well as the minimum time measured in seconds of elapsed CPU-time.

number of digits	Trial division	4 Miller-Rabin tests	Optimization step	Cyclotomic extensions	Jacobi sum tests	Final trial division	Total running time
100	1.40	2.16	7.11	22.57	42.74	31.92	108.65
	0.02	0.08	2.50	14.79	30.43	16.69	45.92
	1.46	2.25	12.92	45.80	113.64	68.19	223.40
	1.32	2.00	3.45	5.11	7.40	15.23	64.72
120	7.53	3.61	10.23	37.74	70.13	61.98	190.83
	0.02	0.04	3.75	25.05	47.87	41.22	74.22
	7.69	3.68	18.48	94.90	185.60	150.59	253.07
	7.41	3.53	4.36	7.72	3.50	0.30	47.15
140	23.67	5.38	10.76	78.57	169.45	129.32	418.03
	0.07	0.05	2.45	54.14	73.24	52.29	132.51
	24.23	5.48	14.82	224.06	371.88	294.75	700.30
	23.43	5.27	5.29	15.91	62.41	54.69	174.15
160	47.69	7.73	11.35	117.12	290.87	195.78	671.45
	0.00	0.06	4.52	79.27	179.31	125.82	293.23
	47.93	7.89	24.49	297.21	837.30	433.32	1385.09
	47.44	7.61	4.75	38.51	63.85	1.87	388.81
180	49.64	10.79	12.27	173.43	428.05	342.72	1017.89
	0.00	0.08	4.28	152.59	201.79	197.34	367.25
	49.79	10.88	22.91	669.98	807.32	796.77	1620.49
	49.21	10.57	6.94	40.61	90.42	60.37	460.72
200	52.69	15.03	18.06	242.73	642.29	487.06	1458.87
	0.38	0.16	7.97	170.06	320.59	212.34	491.69
	53.03	15.32	30.61	450.48	1327.81	775.79	2169.14
	52.22	14.61	5.42	34.25	187.71	208.33	884.21

Remark. The results mentioned in the table above are within a few percent of the estimate made by the optimization stage of the algorithm.

For the DEC-3100 we get CPU-times which are about half the size of the CPU-times on the SUN-4. This is what we expected, since this machine is about twice as fast as a SUN-4.

In order to indicate how the method behaves for larger values of $\log_{10}(n)$, we will present a table of the estimated time on a SUN-4 calculated by the optimization routine of the algorithm. This will be done for 20 randomly chosen probable primes of size $\log_{10}(n) = 300, 400, 500$. The optimization routine gives for each probable prime a choice for all the parameters which have to be used by the algorithm, and an accurate estimate (within a few percent) of the time needed for the algorithm to prove the primality of this probable prime using these parameters. For $\log_{10}(n) = 300$ the optimization routine found its best solution at $(B^{\#}, W^{\#}, \mu) = (1000000, 8, \frac{1}{3})$, for $\log_{10}(n) = 400$ its best solution was $(B^{\#}, W^{\#}, \mu) = (1000000, 10, \frac{1}{3})$, and finally for $\log_{10}(n) = 500$, the solution found by the optimization routine was $(B^{\#}, W^{\#}, \mu) = (1000000, 10, \frac{1}{3})$.

For each stage (apart from the optimization step and the probabilistic compositeness tests) we will list the average time, the standard deviation, the maximum time as well as the minimum time measured in seconds of expected elapsed CPU-time.

number of digits	Trial division	4 Miller-Rabin tests	Cyclotomic extensions	Jacobi sum tests	Final trial division	Total running time
300	59.60	48.31	689.50	4220.56	2776.08	7794.05
	0.25	0.40	399.18	1800.18	906.87	2239.20
	59.99	48.91	1511.90	7648.55	4404.94	12006.24
	59.19	47.47	321.45	1593.77	2776.08	3856.59
400	94.57	110.29	2237.43	17779.19	13107.72	33329.21
	0.30	0.88	1866.61	6724.21	4131.92	8245.22
	94.92	112.27	9070.41	30441.17	18629.44	47600.09
	93.73	108.72	563.21	4728.48	13107.72	21674.76
500	105.70	211.98	5081.86	52171.18	28813.95	86384.67
	0.32	1.99	3509.75	19358.62	8499.59	24060.33
	105.99	215.33	12608.43	91082.17	37756.21	140442.95
	104.80	207.94	881.35	13534.23	28813.95	51161.01

4. A LARGE EXAMPLE.

(4.2) Introduction.

In this section we will present a primality proof for $n = (2^{3539}+1)/3$ to show the capabilities of our test for larger numbers. This particular prime has 1065 decimal digits, and it was proved to be prime by F. Morain, (cf. [110]). In fact, it is the first prime of more than one thousand digits proved to be prime by a general purpose primality test.

Before presenting our proof, which will be done in the format presented in V.3, we will first informally discuss the properties of the number n, as well as the time needed for the various stages of the primality test. We will *not* present the final part of the primality proof, i.e., the enumeration of all residue classes $r_i \equiv n^i \bmod \mathrm{lcm}(s,v)$ for $i = 1,\ldots,\mathrm{lcm}(t,w)$, since the number of residue classes takes too much space to specify here. Since μ will be $\frac{1}{3}$ in this proof, the enumeration of the residue classes $r_i \equiv n^i \bmod \mathrm{lcm}(s,v)$ for $i = 1,\ldots,\mathrm{lcm}(t,w)/2$ would be sufficient, but this is also too much. For each of these residue classes, one needs to show that it does not contain an actual divisor d of n. For each residue class this can be done in polynomial time.

The test has been performed on a DEC-3100, a machine which is approximately twice as fast as a SUN-4 and about 6 times as fast as a SUN-3/60.

First we will present the product of divisors P_i of $n^i - 1$, which were found for $i = 1,\ldots,20$:

$P_1 = 2 \cdot 59 \cdot 233 \cdot 1103 \cdot 2089 \cdot 3539 \cdot 3033169 \cdot 39232883 \cdot 2278390627 \cdot 114219291889$

$P_2 = P_1 \cdot 2^2 \cdot 3^3 \cdot 19 \cdot 787 \cdot 1049 \cdot 2593 \cdot 82531 \cdot 87211 \cdot 198073 \cdot 4744297 \cdot 57384289,$

$P_3 = P_1 \cdot 7,$

$P_4 = P_2 \cdot 2 \cdot 5 \cdot 90529 \cdot 171049,$

$P_5 = P_1 \cdot 31 \cdot 521,$

$P_6 = P_2 \cdot (P_3/P_1) \cdot 3 \cdot 321169,$

$P_7 = P_1 \cdot 29 \cdot 16073,$

$P_8 = P_4 \cdot 2,$

$P_9 = P_3 \cdot 6823,$

$P_{10} = P_{10} = P_5 \cdot (P_2/P_1) \cdot 11,$

$P_{16} = P_8 \cdot 2,$

$P_{18} = P_9 \cdot (P_6/P_3) \cdot 3,$ and

$P_{20} = P_{10} \cdot (P_4/P_2) \cdot 5.$

In fact, P_i for $10 < i \leq 20$ were generated by multiplication of only trivial factors that were already found in P_j, for $j = 1, \ldots, 10$. Finding all these divisors took less than a day.

The rest of the test consists of:

— Finding all roots of unity:

	total	92774	seconds

— The Jacobi sum tests:

20 tests in an extension of degree 10:	$\approx 20 \cdot 34800$	$= 696000$	seconds
41 tests in an extension of degree 4:	$\approx 41 \cdot 5840$	$= 239440$	seconds
37 tests in an extension of degree 3:	$\approx 37 \cdot 3480$	$= 128760$	seconds
	total	1064200	seconds

— Final trial division:

1580040 residue classes in $\approx$ 74600 sec/100000 residue classes

	total	1178710	seconds
	grand total	2335684	seconds
		($\approx$ 27 days)	

(4.2) Remark. To prove the primality of $n = (2^{3539} + 1)/3$, Morain needed 12 SUN workstations, among which four 3/50, seven 3/60 and one 3/160 with a special chip designed for 512-bit multiplication. These machines together consumed 319 days of CPU time. Since one DEC-3100 is about 6 times faster than one of these machines, we may state that our test has proven the primality of $n = (2^{3539} + 1)/3$ substantially faster than the test of Morain.

(4.3) Remark. The test on $n = (2^{3539} + 1)/3$ has been performed with an early version of the program. If all improvements made ever since would have been incorporated in the program, the test would have taken only about two weeks.

(4.4) Remark. In the proof which follows on the next pages, the information regarding each individual Jacobi sum test needs some explanation. Each first line contains 8 entries, referring to the 8 different primes p_i in t_0, for $i = 1, \ldots, 8$. If the i-th entry in the first line is non-zero, it is equal to the value of the conductor q of the character whose order is a power of p_i. If the i-th entry is zero, this indicates that no test is performed for a character whose order is a power of p_i.

Primality Certificate of (2^3539+1)/3

n is

737960982013072251717827114247527699664069926110661926761160893989171826141119\
521929101931369383490982766847861989655122674917393968130587829514313057964897\
260666603966014551135454876960320216137860473967819079126559583788823053318306\
935359861441837366855882806862688379176110552962626629416447849146630431273272\
995724167758732603285383186777757585738831334876769323045488159378902977467713\
837514310434260339455526629562477584638188948029810999272419754692499619209623\
001613278324156579182194830724641306437197442253758831506979191793352123575900\
078074622075861772974340085081499467073364163279258937714462799026957182335844\
641686166117068880549847126723693094451226465616453469916241963304631414866552\
321857691895960230071111758208813268761188738075593564976176900029491437349453 3\
422904682091065786979568675767012114940615503154724612851260976803588566734355\
505597767200432478724025353578867345462382931420365933103311751585395715447345\
608189334178143604853988249079586934366138135363609181014186399426964456547292\
271975820619495426338199733732972086304686486497963

Parameters

s

815255541870520878744200739051019183269977342218220964296307483283540022488522\
394194945590326690669350591791443319220986260492554136570962975902227027251927\
309950779325925670263769913498822631606866850472584826406357191420571966424543\
7778254368368929

t 3160080

u 60

v

141190424735824435339708555547840960043155165112394474431212872914358486891626\
538484150615101143149431704 0

w 60

lcm(t, w) 3160080

lcm(s, v)

115106276224933544725190441176588798011191923244413529429374449435074502102615\
484917446691777532232278084400540231943415238514982495573423866726689479275145\
068902710991341607351612770279071016403466956850146515139623167087525675649756\
127628029257449541354607868210976502304579786811572722886888880677726654691784\
961081524822939776514525179799665976112501 60

Extensions

u = 2	m = 5	f = x^2 + x - 1
u = 4	m = 5	f = x^4 + x^3 + x^2 + x + 1
u = 3	m = 7	f = x^3 + x^2 - 2x - 1
u = 5	m = 11	f = x^5 + x^4 - 4x^3 -3x^2 + 3x + 1

Roots of Unity

(5, 1, 4, 1, zeta_5) = (5, 1, 4, 1, (

0,

0,

1,

0))

(2, 4, 4, 4, zeta_16) = (2, 4, 4, 4, (

259263455133166599441423707965572776210925082003438386757283323469744839779616\
129413309211489251326801881322966314143432139243647876366833791105653190720001\
433868044082633527373702558839531115118389459962058890123347425078021640370102\
817077096675051970641863173655087482598594673870625315993334955689987233661732\
136658022314903024275997361933645261074767512658808392200291287818074991934548\
542067795381316425111515741365669852419327254333467035595864640875033736075109\
299518767615344575876257908786587666629315949382088297159184932678689454467078\
552000075656406079112481924223573249252814591394523180683003137180783364517799\
914078661584512854928349885363596196756648342366178339320855419614963320156333\
908789789083837969792461696139011765771777451050304642921760371164490016053649\
156026486820038504227660293454984726500957353182961307519764883091496991477843\
334897857781302506282544346107663879013012638420113364999147505328184490223788\
362190371509356791274586434092246166074491692820462385332186838002501212140713\
020061395604299703801214500242570533247984185656115,

518526910266333198882847415931145552421850164006876773514566646939489679559232\

258826618422978502653603762645932628286864278487295752733667582211306381440002\
867736088165267054747405117679062230236778919924117780246694850156042380740205\
634154193350103941283726347310174965197189347741250631986669911379974467323464\
273316044629806048551994723867290522149535025317616784400582575636149983869097\
084135590762632850223031482731339704838654508669340711917292817500674692150218\
599037535230689151752515817573175333258631898764176594313869865357378908934157\
104000151312812158224963848447146498505629182789042761366006274361566729035599\
828157323169025709856699770727192393513296684732356678641710839229926640312667\
817579578167675939584923392278023531543554902100609285843520742328980032107298\
312052973640077008455320586909969453001914706365922615039529766182993982955686\
669795715562605012565088692215327758026025276840226729998295010656368980447576\
724380743018713582549172868184492323214898338566409247706643736760050024242814 26\
040122791208599407602429000485141066495968371312230,

717493653263796867863099000461332305016121211113195088357249228585662796005814\
000533308899232457953791610079326862011415236552941956662548671954671040660564\
782454606148553096880329004878839966526128826605302565938095103434571325185249\
771714356680221092080204348813916841559525502541154912427509725569849665915342\
212409689749441572239949479080653202972918877424895453211762076975939767406332\
490938061328440825740561828750296861473224599678581393574540877396076654546297\
843583608411095809827572704424027222408191807697753643893604231498349377358793\
133590322900397156370709528281298099654740423824753517280518172595812042031515\
558623794890547478852140834690107470196508454574527575039468763169183557904821\
224198100406347371610943171616640143874630581686165631050269821345351032919198\
599821178823244376402649663471485696817274747643704154900256167133456782904292\
674841407873718150312032365470965458174979525820154927464892291946042741727397\
895777144590400948359522417978746755489757439012306592167631541645540854075916\
689364585434849663680376916031707108684497286490853,

538994239015608582737575529717340947069798879004343611918478312342998709694537\
780222411455115428190794919414467755930571716851747764201706739770948398744335\
345948085982785090025309897605424798485105672866342934351593305102950008873262\
797799698111720216059940480535894650281377439816272234897560803495675523268139 5\
056630522639097079597428431564394904915447482769490654234308658039113193930478\
430711839868452363937996283543520428003618857020570317615171694797097656813543\
757067205143749921107137943873789417287637533201817813917446825652381655151264\
048484450488827677482859440524734834555822517529221513785693438166845040523766 8\
911219694395547111554406062760778017768014605774282573518484039365374497274398\
915239169657288798045097802749516075280811701170379304555019921278543372682633\
135136476907898419032239599205495871125255461876943072990535457585312576678574 9\
500552074889319340977081680323229645313428682440437735636714470295721954167524\
436792932606456239043638699285332511025364081992227359510928533786426026752801\
622734026393245170260251818186406044116157571319340))

(19, 1, 2, 1, zeta_19) = (19, 1, 2, 1, (

245911395712065214433484089888357782703394237145336852724124936072660706636258\
735797834710076473930402152683014727080575632441545643401615299798727273991406\
219542041358148095796508835765337579937349232078308526334994966958167834756 35\
462636947851947343986058753231445082945223874047990281637657807643999564163614\
225528950843355059458391295016131378777145232782455236900456702968126562934356\
198639906014384567888493198713950543475202822262969455818880454866989218313015\
023479513041214313692185167263857555101229850762335169109810456509555561749947 9\
232372160803538484012996802394938723926744406144867238991308314794689259105974\
890985847862181128356589224790993016636151641742845949371661806359723706201920\
657660282166420626599568130032046896130599058900188424205729534006523789441890\
458327674998539750343169394658860243477731045067633432709654057660158493386317\
093018360667747348591905839349062726830051100443387823921485133890005434217 35\
999041134113852463693699387802923452829248198405112485735429631194317327126889\
484858381673939651981448009054618519074854172639610,

286318820240682224163784485400527521295870294543831539044059092183742582319064\
502077165547402508927738236841781139045103351268522574235539161108840976096\
365316129798504997409216204703649210617308432999011188376717268089908550058281\
355739262105003445275686774343323526156703178881761441767696902852804797001 44\
730970679077956223383788357025960165861208422432730884219948929927725036511663\
391599026187333996052269388661089384632412061438535497778886889645036953917643\
547278974018723460490840261292385122430575518663130798673768800175853473122231\
890251492241413687010074254206864315583266592167171366926443487335926834963029\
097746875685882791681394060909370725361511864843996751383750127564657740691096\

```
490928744832702959579969850541334438614080534711027708972983816851404009190706\
344592840515387205358221953781499601748664467595684228374422316542571266102775\
408352156444410490332736970986384452056844167481947209114921844021128403719156\
889000986942214831030364031567912195475684378079264532334009525973570115215645\
986218926411656525002474077729767955672378078457896))

(59, 1, 1, 1, zeta_59) = (59, 1, 1, 1, (
361297136568996482986040934080255140499807653597740317627779857824348903252735\
173645592763889205141218101945443032192051735398798223114594717482462659880985\
356865179380543105610480840830549335024596951429338433853060006326522635814193\
191967826855652621518162834717240704416438734995978877586735864932218432896546 3\
200727983466496333696000646230394808263433511778793738609510022085598717056052 04\
395597971268717925707322747899523895184065085142531738778955301018351979821726\
000531720774748555484350771130643135391897247098351374322410146907785134773438\
514151186600091425705151850159141894676630986758296171393774915064244168918513\
987319330601299981617367580378162438461336737443946441074985523211236639125682\
135658859294006503765543508935949192671076508070987252024457652546511561704320\
408500316714340000562569539724422176151420163447584082345721729174683170699727\
626504175289078649058927132783349521770117180594006857614477029087646158299405\
514679554091175273485071988822027071027272715282803005781700907896813566131689\
079771446855127387411375124679443668816080006952688))

(233, 1, 1, 1, zeta_233) = (233, 1, 1, 1, (
927707049308971998000393038857354560267723496532732190329258661445311287923746\
375452933974571233102121191498786306554236191844920549694127998843830168193770\
778592113652313083337786665223273208646010836988212089211228321746665091372768\
913603153613524034227941928419678757258897844091232784962517203470108779839 48\
318803313069578407060690808484057452334551269601887609468885747296109727353397\
604667607443587128466243806060623875822387367091842400702171606994195788726258\
950143004497167075158071342510910985807884007197465836053640708596122188082383\
102946119233650115978591086527032494244665938224793832726151419477132834625770\
372947315479311950396770767411179179658280623057551490093352419466549397233544\
918192228048360934127051209486959241256325675755884780331259601921459999918757\
674988182473136360258880843313971517970909028695851350671672737182911442480 90\
756033738862496661215911330348144751099452144354583205457629696172841051474734\
854028013419662598449212561966880625193702522716946954575080967530881281707324\
587572604314559487411516576282573855853039986646 7))

(787, 1, 2, 1, zeta_787) = (787, 1, 2, 1, (
373031640218081603351269512738547997246641942844625970695000150195004097948 11\
841556344527609756194424975020161860249222787652246897269132511290535399275084\
228561953548601349203410045416524099051391479581100516670144115403350378887461\
440216977799588369856569776760397939457801708698385991648146120886289488814183\
510226101257296012783897444031586925092540722939210689908368993725588143299 80\
292401293475815381053838862359488687311386648011098943697539803267074936431800\
874957708396235169184251527832827709421795795873132830149985231238992837688614\
608790721911955067871912234819076370344367607230007431619279198599601900835136\
031303059834040570124308235918385038558132251872121547417439858255326543276620\
887730760708033031731218543812598424706874271040227111140586634091977062685154\
743588883957186738020125720664253579840251895112472791731465890876288203968884\
745082386524175752833358832256105685143421219260243446676233301550226856957671\
315835656543906086633737425595174014471246572487564202988963793057849027833 86\
253189554283893278786083081213988591939965991212472,
155955795075547857263825241577799376118999721833966043355433752778986546807 35\
699925763443178221591427409457243609657540571680352743569491038806249646055140\
710277840420291145646950358320479651235866157632677850488412162019171660918847\
464882857070150290165879660321515228445283753340299452209132638544687354121813\
821096753927585888411219618151963802985579264466022629451420388184247935012 65\
608502750586817112084147091231125671003492453199745526129575776379256777216296\
064457051917597182363106090888283684285869603164035714449501094891077403283664 8\
929444775169178633509689775425203664954072390236240159129088461366222494010109\
705838086655127090357876022982639838189940390920811588403077339004606442623726\
300136970564838172481975897869996466821955477357012200774718258661192120325117\
126144704338457291645263377361275883193159324897299044331054196862540107342105 63\
100149683119056884378878471322320277293684368899663467892259832234569349414 8\
009152675938288156123036622997600262359534337714251841178392619948769083921608 3\
```

```
548231258088180234467207625041403455977462840208 42))

(1049, 1, 2, 1, zeta_1049) = (1049, 1, 2, 1, (
656811407197910832585210816537826397189910510329351037276225981176478690271756\
813228868635076204272257554586696737762993788113471376970641104901587392284692\
990941545900609326104108643257942943817378314527347130111664726139578069086869\
831250925694461327242397619634932082485370898719771799820687174572391677610506\
124134350035185835633667341432357342123084719522800368367699878826972030876019\
243648287888671207898530318290285273783857869820464740403608637866176472234778\
856937109164538446468042107741215360810364530911291623601429440727465238801755\
175314126409438639766431812450608883459641698356852734533341283824125042526410\
550914761148526824923556877245919335309731280598143605555653499338093487937857 1\
553291201429002276652367629605786819591199883438473661439186882417860001503035\
152214629285043805316799979827636387940345660744311889457684786192329436114665\
507732605811854454339331315962021399574644988303452064046757156144557990409692 8\
267604357400666119750375890684774145614492581301769208876636668344734665476195\
087768635191125421784645949190855477436555307094634,
162140013646059887070111222065548483404729982217741276357433601048732739756105 8\
175711678185067411225679125274197893340573854962964188670527312651198250404027\
157458574160471812343306615592923784580298826241850975094564772667496539045085\
910891086504405233888664676029191123919937480664049056862860531165132728337931\
406324443404091402467536341267349017400007389783941562872816093998551752511409 0\
284024865073658408576205908698592354950625018445751544948592498145456245562415\
891207714129531553559859576805214911138033365168624578035977193113250785163759\
113920511244544222865221038813908408592376881211776633133836346012858822129738\
822852227454331888687592982755515373798020752802367525437541974121927863857304\
730551275932477550319776629272968644688256648558478095205040996152874295059236\
724032367488358524484092023535505594545218579269360236225550255304780930083072\
032339497581559787255874587705059525188729610646750303730458782610065000031112\
616641700941668408195639181588020316120778980455011357505078289037608816799080\
311371259364761649776661539501321485188519718139044))

(1103, 1, 1, 1, zeta_1103) = (1103, 1, 1, 1, (
278302544730744203841652472686555105951803353361601331587335592365322837388 35\
483601096730052690126283947144274768187414537083860570087689320512517730697751\
171173283383589119167014147710535370655019742096587720866626358575315076057153\
830192001425676486995025999909767395078387811204105146342500855714415189043 21\
444051867623328718398840971385381098689537214582371943169995275837573652130 5\
130066694047647942782363777771456248076016175337142906538600006005303133054107 35\
289642392651726123950357630277919001778953642517418635567275579063934677922954\
641793748954808388731818486248057896473534120292835052682072066188505277010711\
270868525754240944502713057912081516659421192466411484885017698055851047053 25\
499596744006809761826722822478983579098887919082037305262941668198492400801401\
738322966399779946263154999337485111002498217492647930435275931033984993443 83\
699467937177762950322239834799939059045851509318778102089587433862773327197685\
852564862988721793752742548906607764308310014047038414686115649733105634816 58\
919946368143046118523681659331364311510352934759169 2))

(2089, 1, 1, 1, zeta_2089) = (2089, 1, 1, 1, (
612928436016315525558308204945778504618304074826329327152029091394648367815767 9\
761618476875872430128512304997437389546244488164211743366916437612328232999561\
259639971036842735200551970731500200339905044970170099901761586510666587815231\
449735036330324879952208270351634657523086526724723386460634539711828563275236\
492022220036042292744718840681512077790421306420455778871160051084906030483572 5\
698837704264151168086850904714139594177950422304495620515500029386377741030874\
698942566790770358563616082807087303897037291101074616786154707685560245119085 2\
105138200138844458563142002191267948716944752323736522616427176736568910461095\
032436723809653574143117655159547244662209001267998900666507732100381636546968\
184364959814258880926509077750863584043654373831148865834894502719663711066774\
333023915751975821774581514218065755792970122330710320909363102677208429126872\
534074481307905554559473393745452375562824671834193117624426328026425847701105\
426355553419646882593314255078637532954771057283794843974992100469427927973681 1\
873155808095584233603238997552429849241851865431441))

(3539, 1, 1, 1, zeta_3539) = (3539, 1, 1, 1, (
737960982013072251717827114247527699664069926110661926761160893989171826141119\
```

```
52192910193136938349098276684786198965512267491739396813058782951431305796489 7\
26066660396601455113545487696032021613786047396781907912655958378888230533183 06\
93535986144183736685588280686268883791761105529626266294164478491466304312737 2\
99572416775873260328538316877775758573883133487676932304548815937890297746771 3\
83751431034326033945552662956247758463818898402981099927241975469249961920962 3\
00161327832415657918219483072464130643719744225375883150697919179335212357590 0\
07807462207586177297434008508149994670733641632792589377144627990269571823358 4\
64168616611706888054984712672369309445122646561645346991624196330463141486655 2\
32185769189596023007111758208813268761188738075593564976176900029491437349453 3\
42290468209106578697956867576701211494061550315472461285126097680358856673435 5\
50559776720043247872402535357886734546238293142036593310331175158539571544734 5\
26160562241304174740668647606170147143658350094163210911787645185013679037158 9\
06909582381873240889093755983160191085820786 4728235))

(25939, 1, 2, 1, zeta_25939) = (25939, 1, 2, 1, (
57918323107163038362608843061824285325233166426833676454820718852231024829082\
46068090578895772201154956013936197646714835397832712434613472962167136222482 3\
13320713939594466432119734594018916522275424593982683852952593074253353522263 9\
88000721750424416014660353833782842696706600448420657468070868209551886668901 69\
07168048265646011235838822820679655764468688534781225315535829787817184388010 6\
29264074575338739334515875914962199364439086964426273736340189292440397462528 3\
64266487398806468345808201755644330317393315622898079635061854012888977878000 4\
48148305447097260060460656316534421748234516222474486507974815693218428026405 99\
17056077554332457057923889164090499943632485878834277717711495280156161651933 22\
90392124427756313077178356396020902542837442570714674333463492667235484727563 1\
10515123903754742644324020848981602279708438777989223860167720555913425000261 0\
25900897850995722673098688924488282420611625439234608629408053601822133732477 9\
92877635100161414700412986675953923906459518890828871008548432388465130314536 5\
52074578695951243977600520274948672267861385317207 1,
65611660823256135218158174742227311020372140728030804420152817165524460884202 0\
00769255253972683484975597640056912759585494725155760785640113786592534191061 8\
63008530475302264671286306944188266692301864570105589110364277024000811355272 8\
64603076942599850101818120857829729472758939733170013829361383813724361593928 8\
63098375471035353766131928200024795131897505252344904608867298888687046751876 0\
11282635194587791209417755704990705790256644692389170425447376374942993323959 6\
20591744714978081879597941844946961067257799983158175988873384832689724816 31\
66499020014510262882415581350567793031033711407677916807097899049029848048237 4\
58693644311620968637452564977372510908013770728170311146665023533876045213644 5\
50911225159789414185194582641014929626454142088856333031395220480556365578441 92\
88403966879360240308571871619479810873605019558409029590038267128977609897447 1\
33834546629055741893706893974870117345278311851009474896805217866507665722941 1\
79051403555729259382413306361195513391158778281838307695305722848267859936354 0\
00050470990237918839273880851233542760043617645021 3))

(82531, 1, 2, 1, zeta_82531) = (82531, 1, 2, 1, (
29296071095141957357004415262735364033729368521278456261941772889737803441926 43\
03226501410218921541738934074696621740336905521564658383896596759010535616324 0\
44804860353722783536361421809037098945285447336633283389922077280573382476838 1\
86601278666390766310589874225443108396782264636173892416673467745006678790315 4\
85780670791853766328774402861145798035618354619170072526377147318605533836457 7\
41835290591259238324491667654403635166774681714776967952332787004009136229639 0\
38185272426282866201087414964251495856604292751176724213501214358600586157277 44\
42688540612019576067651370287064578393174148983903826833986159953604911703540 5\
55970078631515676215399358264997664764875399135022039247761014930641421939632 1\
36978070252939043531528440585444025534991071816456709359935411633325235270369 6\
89862837571989230080874634082355695434386458130317555052876800951664802627773 3\
34795369888266389473286105159342940972230406076213342782449493257017402236407 3\
57840328114222243947007063023137774894031693136026950626960280196741798120075 5\
51498203755348266274825468153421968840028150081900 9,
10480497500818651174771144835639638725675426722873240029254087294754312337 31\
36022662325025607500260998553331668998723638044003957399261430354870172091913 5\
80548324479615446102499887247957428889894037925229577165640412065302888065353 6\
42431284102267508857394097652102450579648454891515193789585650235983985352596 27\
83397325521197030519606929861362773340167267037064754822041099487837251409503 6\
38336480771785854888868327285567922711134392653839319630497731084654304206651 6\
57897365276413733081582818471964376211365688237555263893654015683002570784437 0\
```

```
18957878508996953325016922124908593808432361537522958121398194502913820363946 7\
67126845851769961126409086004211346834269029059378888230533183061 6\
65346060312550708820883863475267102094551855695153131034239645906554728404575 2\
40964286322508038734899507171080754951815753943612189991880088797292190117509 8\
53852614184173043072524558763065660064871122236935539238703466648807053526789 5\
67335765029478851031211375219939273674684710260706546904325443432108839696234 3\
62863969794948181751858985981297113511873951284290 2))

(87211, 1, 2, 1, zeta_87211) = (87211, 1, 2, 1, (
30472702691158703094921455110921796604533767719929452582979841803439760091773 7\
95440070085609234405842949715805441365878874220443579067172322967019492560111 5\
35178873786290277217904133837732116047980519240695647497565936870079701453123 4\
53125152996995270879076139532659734285023263261496335786640796790629904772968 5\
08811074373685718170768544980242745072918511418550143798323935802440915137331 41\
87212576266378521560129825292276610865643785283283550354351836559030466490047 1\
14189812590503018070149967320466502595613912681987372393581323158433615318551 1\
36742550089665567052554972117033610574542545945616213224359697800666804743635 5\
21615324397028836813813467703955290123790665049341139849833569962287647927755 3\
86459886850984350195526754408478654185395154591550365416605809814583828975709 3\
25331846961507513718550819039347432149578298381591197864249562960722351624740 7\
09881451205673136166459558450667893522103278136759057599041310145611833992353 7\
68439426262021044709867795951196776926466564449116205348605394352896881077552 26\
16764306918267108810254185730325200420094410290307 9,
41054471818815988180479572356785026062462451986306618199315010831711031247010 3\
04271960867103896631607170648526694174540625824635017767803446016201899893904 2\
96015655923627701355539516984669810218175863062828550005181273485120401056417 4\
75988960937391213988479002780548057928920244436446303524056583629801727216316 5\
68194256123656197666426245238569646761236956042117715189252308785005006638442 7\
00267727886854770067229677317978318974044761364365514720119973046545313398050 80\
00695959357638104231077931587615874739309308810122666322828678392446085733929 2\
57702796402660868472191604548443825959426662269664617890961846159117108739339 6\
51498921588420620351932353791738061400804472703321355780259198009162090298511 7\
44071610225667813870217766941248939597276131965164408391791057863018997370592 5\
04320695230602857274374169796134224830368492029799132737822409145309063498847 6\
60403278550922694767073037935161989595379318093531595657150611875721804838099 2\
89673137076514528337732441644955956225108449909883958280615874237114906173817 84\
43804030778936202107398281350714189641102992852709 4))

(90529, 1, 4, 1, zeta_90529) = (90529, 1, 4, 1, (
57518322145883489278406060724611591535180274115885803970621800913754729859656 92\
79609199317609977704238027010819037360261527646931715439937829417876265839225 6\
67252649687905565854328218447802216773893073813690717223520108314887551364 1\
37283579062434514232436190110985841238616457109284972736746869736777703946074 1\
33050741834000415905528576787897469710934675953034615135780705483827283832814 9\
51441915215870251218291902384084520493337291940542865086447018640397989642187 7\
92997761585980146232306766243985772370854699437030597200632892528590060448079 1\
00118234561504113989943084748090485819456403619661244300469546545286638907129 0\
88751528491277895207223601381802218840856071354130340802510708881879085772237 80\
62542386639593337225724559058295270098945786801651788122003910525357456034074 55\
43081179159365140520746196050945164228742446768298736246302322643694507229842 0\
26659805491991050997997508706846170574919189393597429175037533376600665167929 24\
45563315850460044073333620747191464900754834732376935668909671012294266180140 1\
17521746156417681409170519849468559320641892050304 9,
55035962018700516095077031628446824409742505866603045341263226349993146097287 4\
49286289009203427191069670916325569611045610583028368117585525874755306340426 1\
24713879034237875399144548397421698888850334701937440642616649018663226338496 37\
24869888246427653743603784747681954332046019510638451265836418447972142742654 8\
72286968432401335490955610570587831320257719170979326669059485863805282334458 94\
84304257501368839115095777846827142770717684747074299712193887887676664922258 8\
60635250751592270578277456876688512896788356089058338856217170227267101300750 14\
64637973111605740522473614074199282712109271856870965122402899937387897467893 2\
87585910880792735106899554330011757731890871768664455894598415578848137019429 10\
64910259334343281216821965719851205559362502596742224461717494523932433587676 2\
82650368806631405299431011208995948885258152990153208334566931415538863746210 \
07119854828528461174425865838296623015879452224461717494523932433587862 6\
53093603829508524493052757690403953054527076127687090800011704105263198786011 57\
```

```
383230362163644854072877906881711179751037562830622,
1482089600646530830773743926840228471638695331406496915608491610577346272436911\
6943967944586696173079171696137322281789943693654395958940484187221422260247161\
3253542765006399309338244411050388553693532631913158287533770037719951758182337\
1756939169843285176987978865354280040861662638655971352605126450384608228476852\
5191920283807342767123040844047180454561870067300619161279851273389340565754022\
6795599969015985595327942423920532869088161698140290778640038824232264696222250\
3300166261184325315405149171578638288802378952035899108962297181368027334483973\
2966832972691947406442606187848584179835138205320402296871121958136724923900061\
5990807413891624122771960090969965407053995782564105383755989979137208149491811\
3692940471779456648553676788297283352540989706628146401752527004554075817743331\
1825531398702761292088949731068918222582607641137257071431516102547666836077052\
9457010826343239858485274515361659467845300143613658555220979204022988172020351\
9688772191717219337417727346710897314428034316054802535983261625754964950679107\
9809297893386714789291116774561238300581797453190444,
3881886746500658991102791650407125158755319654892190532784374746029694606636841\
4120468560469779556858706517716409273510550425774768291830899642732586255635161\
9024570363364521315183860637518416097581213854136243025501841662709167757674541\
6950733792671345612924406780991961898982875249974692144214888782507685486966701\
7304643487595724938880453287132066751897244027518694847778890242758706580738601\
2497381320675654082395452506972569045820650283724749844407118430650967716264571\
5900723522501808370769905903361595703583828822090186382367354322350292347566091\
6302726928613714772162792863420403056908335961600664556351159669195925746301891\
8449528678674950798750776042838103341191173755012714456300158443258390029326361\
8967959853689646595454999567927631004579405901819362928925145629811104248326221\
2204357359633450601120448544313926307868140169104123712902696939776385063677051\
0981182541062323407684594810155889978818644772409109618883580554248090841555671\
1537405614491466227618007613428829666494834391798597270811411788242876225777547\
8624169492912685482933197156564346421666620096181931))

(171049, 1, 4, 1, zeta_171049) = (171049, 1, 4, 1, (
2027577323065449173388371438362015966489016857086220208163283511989298982558621\
9242560671589381285674137435769853909308241186412276757734818332880321397189371\
0324022195237687851501792934349480987533630423155833199061253077695817988394680\
6707822253953425908066929229462779473214030795735204865789462053885845814673985\
9778127543922616778848827942625454020587251261145246865309785593410880345843241\
7874930469259385798376718390894637017772207290936878143282431315297240051112741\
0912340497036504717259378996879711176668274910253036664706996773876826224495121\
9120315868008054422731924680744436861892255723552481321662470962258523691483081\
0052232548317492281406271423469787247146666129569851326985997506786929634354281\
4927146485243184571080491556755973928012843334801639317510172131568740494428381\
9312754555528201040757010620937140012295306704150787506969521205199176129012804\
0670764100405652359860514348507161234551411589476638050176242622234140308172381\
6323859788498261644934582937415600368040983769326900578604507736359425388180301\
6406346253133597501587371181492627993280766410950721,
4921729378106434580399980138592630869033350473697876566972564577937449332285\
7774878921644570409455682444882519198515438468015011906407474021995647429889801\
7011435122593047871780597730559919968028388102188906250912571951632945032000771\
6819894851857089133120233764294145977496498946050731240504150860350855179447821\
6182804620199842547544482232690011807297905506360980370699669817578038884324451\
1089459508153010094723361859099430442408417486814313210787334686857259476302271\
6225291498832315604710914110417465732016892982477021808116482154776334010644831\
0247014979158014247537951683484455963803732053305577188516962073311184670497196\
3531069958139617588959842044898779700770756718827055456609661020142296855034441\
4051911610492956781091590505048190469439646371848976984232110837226224303063281\
1494518885136484418296686925437770467464420023686344293545857180700773894168661\
3696574387884678732872312052764474576942904289239555686704744414181195274748131\
0850103537076405268436827959629481299957597284437066739792115486018648215131671\
1738336316095910268538029751718124180161250810611943,
8254041779884885950338313862555853904183236091764008694678989879909909901151961\
5992150868682350910382085208857774560781672477026178117700892797485710063275771\
6901193943123284610410621518461359148289710772844600757967909966444202570447821\
2230429838559162717160230651602200381684361627676216849216250032993943877511591\
6752608871102754723349910044242227900256836022586897772187266855070883906301251\
7510230304751127049283745446341617206562288121726312483149583952986502054932961\
8452499016925158898616872959879931487678479732981217313981635512839316283852281\
```

```
6834841274419949877616759090361849523043904866049265630980712667982760116550980\
7054621743475113334269323710081847933407036206470995888490847320360360349659321\
2424153891190658615787991912396036924778065443854721410812351207061032993638031\
7696338739040079034987507861367684284563331741294669618031826910033821\
7156483523246538447200247050187151420455147701849382350726815052303717067040261\
0813234961587675667862037466056798228692312657960299937270709601927932260591011\
2377026018973101312890668628872713507084406688756 3,
5171840227191431053439522861205509745005940823705413609337648652728177849101651\
3330142241403567418896700587219198127203500809050104980893124428333232156059366\
1366073259896888298995095267842465728442980768097752295220750110578877134405661\
7456758425613083407238171884059570274299639926245594559421756402134207096005551\
6787647497599981285424530213854941349747127847303517712304836106508226809871691\
1027673220815007438253870704204507458798242380473168300583012663808891894737041\
0398246141883282049277504775883279161789112902049129936783460743339806283769098\
8828271043897214862489529128066280332233103071490498482042222626664054852680276\
9450837747057024741304046538923321803354359173836185387014162847853804880386871\
9708118159307597121827617617642248085839950831053139087923895982480053688067631\
2625321112459269453351497418851611343913659389194019146687435960453680114670931\
8246514805366814378688896176622002895860811032499230816678855594248241464910110\
3520832172263450617214847289714074550775672198523540295446290271658303251654901\
6231046045406642052815944769864274041780356382309001))

(198073, 1, 2, 1, zeta_198073) = (198073, 1, 2, 1, (
7363786597772805836826111411937657157221752511889854268395486816681092661592171\
1973562257415511297394389926238394101161866358586656060325595332730154318377621\
2058838455726336339037553196073195236172823385477310568294730827576957135433721\
1288554153852905102700921197449328632866349047099986362699343388204334786347471\
5395468746001995773856538954812517314238727630801910129369339061625235591918250\
8052560756195871546365184754413704205360863070026503985818430215773155032886091\
3870859126853883224455169099854671342605291903251197104413019478600817614234231\
6610273414681103541901802731987839285390114882307224697707575936833408478438391\
3671039582792283815044770603044062746028043250442958721314333436316188085139871\
1058681883000946668372945715203486946849725977723694929766126177872164645160301\
9620566289920744164010358924327252267498647626230438853389870736001018400935731\
2554268642359336922210986036008460702015698887728384429439630385447610915131031\
9661515794412432343188977921666513796372473745546220275613394455790827185742413732\
3732402710809832251383483262370767824123582348804021,
4014091748823556306240111815685185726474633780425095066562578297549085637036061\
0074750773646909497926522646494416785689001997265295129541465936887579366925571\
0561140551035745362160958172951013830576026320634995363617360048960241290202761\
6412570977373508775096575701588992641413348405016291689680192535120710399374811\
3275411895496153986766860631703127247960861455849339062818702021274798068745401\
9805484056609207344074131559896274824273570479436356036171038556383794060501331\
4901757439553145548299683881866910485859453701778803045567596748905094915497\
3142740397987607450261812363079255784927948902527245226561322274239575855544191\
6898365148187766223997326543425617675943892934317931763618533635429538362994811\
9604358215172664957723931154333439747073869282759869670222997233881333978951813\
2\
7879597961295095520821756154565048226932046493736600810024403012347616084171251\
9136285561653999965551768972790334590716205734959933518049257570298483522025970\
9989344495695339802169955585483820536534587125396272191052428813342858632819281\
8925200683076557131696578346740924457653997156124811))

(3033169, 1, 1, 1, zeta_3033169) = (3033169, 1, 1, 1, (
3334914601401952547634277679515927236477684419179104033374536060964385426340461\
7106138287861053867482876002828789111343891346150266473707111721808841891246441\
4227553955580117158082057030870282234014973135200726452213695094956161594427426\
6811521539971425350868785285817019997523874917114833284870297816343444013530\
0232506419797297959808098357766163296897928252868874624966338512872934966902401\
3329711796354726050108671008144165911270749444272914830282035711960910418760881\
0232216559464831324489934666037264422873295273009784528036389918096582177532686\
9487434471475459462446822735295840263623504963344278167771191823674536626655299\
0941101132726585758680696357782613669174814814231075161194388422822708170453899\
9135617238876404917995096516009426683542391971692908889567070188590775527057865\
4248823333044903697228949399283253853890572482796321246393714039348851460086271\
1230325593490165027078825936202035005312197553611058840531277471287719812216271\
8654540219284516631399122143768292525873334741498447243406170349703782262250881\
```

```
8635454499777303959500929457844013571308603577686 17))

(39232883, 1, 1, 1, zeta_39232883) = (39232883, 1, 1, 1, (
5303884860550305923635126962000675693755879843204302521094564688247981073429 15\
1555672624813833061862333183547414725804376966323717897490100371028002367013 14\
3522198279858252716123723030659293853172357115736752636613762111180479043629 73\
7525895969223428213900943264419879220757007927616054502344213212521983345504 32\
9613150039717997217281458258357056419707796249782499774577843514159580039142 23\
1278354950494089169257623882851190843710698251614644849805900667251265719613 11\
0491796129970653914730604411590237874422983195347734479828530498550543831686 21\
9124442791563906417909668606465107121212235505250205474589605720123116081437 35\
7016200205172096057835058048590270181091935586514826179916296454074233619077 52\
6617956329292317705017339825317524981827906006582325871897265114754790227699 89\
3813890985410321855972041754423693412425419049159104425821512921146187025489 77\
1073706766128358062003526023597404320668875570082927766719977018981804779854 43\
7692793312805090186313567672504624711937098449524950823187188457165005309340 39\
3571805497209841238309728334761956132672544952915 69))

(7, 1, 3, 1, zeta_7) = (7, 1, 3, 1, (
3452421493454237469530858552075400150340889822951892560815649904991451851237 21\
5129571848199906478272329611634751629373738863466467877534497787292301250038 43\
2148341100894309963577199408783046837657602372615047083367139269920672310555 54\
6840829646559784932334857774406681389270681967961189906120263364543420340500 88\
9253633089868161757270161671655449049977150365842981064388829352786251055381 25\
9917255717925026136234346040138566544547346898073689980614403354505035364873 13\
3572204682049779675028308779776599251979468186722844320355270351892449513138 63\
8155129994629915038809635375999341752046792235559788895730784869447897995428 06\
0131949453279718948820218797839060177789346100357316867368181511858355703002 13\
2882313205458771876936985943206063191228658856795720800508695561422730079533 34\
1258150724325795461544059468416895201293828061315069359515595871775339557893 56\
8604415239574558073072472363339862002340207906346607310883794019219403414162 34\
7427496936366976036294276171219075193534629217539671418048452873047932356162 43\
4542059637687552672812396849196431913490224802602 23,
3927188326676485047647412590399876846299809438154726706795959034900266410173 98\
0089719171113787356637498056843868267177487885707471803771380507850829326105 4\
0458324938765835547782554681772733784802581013527719952898456567967558222692 52\
2512771667858588736223970294220202402490423561665076388044215126922883972231 84\
0703608587719164275583670196122126807411162982924712166066052241002778715923 87\
8457887386417577258320920255486209301834542582224420012109794192419960827223 09\
6443928101191786116793639527469813812392506235814743994714521566041071722602 36\
2625616226128702690933765474815657715026571927681369980406614104559059186907 78\
6284912207890969856678252469397870767229185558072178317942381211879584566339 36\
0336263713500830423774189877675263684890214950763635697108994415264136554119 99\
2970896096584862408251627289253225948112326970232176768997013896260546109449 98\
6451562432429766714167781172448811452283621407857052020149323496634553740311 10\
8654396405414460012245606318666794150126752136096420392093411212217122093104 8\
8177698568507401590569600488133288949556640062370 40,
3768932715602402521608561244291717947668362330871153248025226582639948905 2839\
1769670063475827063849108575107496054576238211627137161692419600997986636419 83\
7688330326141993686379034483791978648727061917735303581388017468618595621935 73\
0622123427425654134927599454282362064750509697097114227402897872796396094988 19\
0220283421768830102812500687966557933158777230801798840311278263936164657 67\
2277676830253393555095872885037032838273881068750751000161133057978165233977 310\
8820020294811117302871862611572075625588160219450777436594771161324864319459 78\
7802120815629106806892388775210219060699978696986876306040358967862005456141 54\
4233926423020552887392241245544933903745061070657250843652192514114241981096 3\
7851613544153477574828455232852196853669696252774330731575561481491852464352 44\
2399980972498506759349104682307782365839494000593140965836541221432143925597 84\
7169180034811363833802678236079161635635816907353570450394147004162836964928 18\
8245429915731965353595162936487557449792937782991083740074509170516378559159447\
029915225823411861798386594182100327086783497578334))

(3, 3, 2, 3, zeta_3) = (3, 3, 2, 3, (
5207351494307361008738661708021874785507080695009543975852860011175618495054 68\
5505380855120061251573720732755326865858725678227994750962798426438570805206 39\
1129081762207015600970274410252515857730755215989343427857341192083519818552 2\
```

```
1575709186233928040695098194343091011933688702884375472448225213562178670812 55\
9550444299752520156871025888893638283579109127466023093163669905212750535317 99\
3145236150579093586060594982124807663349873369779195180446487060661605441972 29\
5278734237947395888024811490195602123795232290727736952689587293071825954598 27\
9954828403033575845904644886490986073176641242918204841854794398314916389125 7\
6539974105907319972417153885047824584449226048267334368500945869683891125926 \
2592777974062105504796001456177682560144769402530067539425787569731485259037 2\
4675142067554494201893383602482986681629055220113590201931791204977151271369 16\
2880216422662410213871191680506892586235578525734181930017105055778846011469 69\
9304185786739498185552866589728514031611214792947749197786950388999126929146 23\
1396695395707566115698317782152019635748882113552 46,
4771008152056192957392024444642518859194373800070383064473279343623906148733 44\
1712719264902530603270667970763008189902922881266607013418577351503946796994 85\
7906630882432824794211623948147575252969891510119329545697130104093817096993 37\
2512441056516604564045969560436072207555099922143539712573197936498601654275 22\
1388524652355822677043163666680706930239368962947761874046276158960314695875 \
7122057012239353026460687478253405483725882068381556859397143301074515903768 84\
6513535461657467437370865868300951802336527859695929635455205413365032470407 87\
5772353332476576453917805009532778445597267561960480755339488896625142793553 70\
7296837314974624308498669160246277249522998703415514264857900337741581374401 25\
3252212451419723364746501667959081691398730213296185030807533717129786236032 96\
0295532143801785203022174244650309850190196138535345492116763071867383889007 8\
6500412376633814320984812654815294807866973492140850436396956001070563834077 10\
6984848977723256843763124453677347809499612492408483229014364106094739314051 83\
2197713946691664856830906002036048178783555291744)) 

(4744297, 1, 2, 1, zeta_4744297) = (4744297, 1, 2, 1, (
5726279320760629732657564987175342737432037816806598316656924052859115101002 45\
8066693631129833115042272073413760406002106620226962816330819779632078567375 79\
9226456270789498233932749666326858037614094984704029040307998953742131865667 44\
9954450518307615334490733704423403511130215710164888036155207026235939258573 66\
8633660257999073012796500737993988921519787685206220594764586658556650705288 \
5784129517562180558464753880758009008816970194210439273386534908126161855081 35\
4527039250014071571761112400536160026228804914918440148628778462778089213212 27\
5383058129297806358037085303336026732293704743627970931223924120113736640227 14\
4745480395580923318739535352173435470295506639823443582123511489136939033871 51\
4896834150011672556965190434363538085760634113793433009124801368613181554438 29\
3745090519404655908962017090737483546362111991704606823298904674927736644830 7\
1743149154094611995160207960847868879619869953510330133210065691317419942357 51\
0582352039798966017835152676892087761101835283116621610043385231890980356378 98\
2479801111917795177347446878158720846403048183798 70,
2530115459503722152504310169925728311485897545178376231554480673044385758763 \
5033862545094179980767623219323122845901311584534224447884704574790938713775 73\
0181259513882308789186457053653508078515160511775316791342540903140851242359 51\
6839678575275403327620683209445924368741973046102912976552570620533694563269 30\
9597666053054232102956376366218352429870131419671141031450449070546884581961 6\
1304777526983150134044232575836918653941407227900380563020402580742580505125 41\
2105171402854068912603981666821308732290914162459761621555729534082400983967 4\
4121646279955314838737663923997371511852649158564461818673413319143291189154 50\
2438584454345955214574114110072850670635447992210233878437344488779973728516 09\
8031310224453310821256462359606501032567098259185294538522233931631487830017 48\
5206117129439272187279028460261371704762959467744022175239304814822278524800 72\
2916835243154058509475838782581977336692596863403925202256494105442079983534 59\
3213125044236687756894288409458492242064311917570956063020664379481361650184 97\
9970880080075004738893251025415621962525494809548297))

(57384289, 1, 2, 1, zeta_57384289) = (57384289, 1, 2, 1, (
2112704707748819810245206481459994603590949110209553521010400012863920278282 4\
9321055440180328222183137948406689599357872601993650151562977547043960762807 31\
8082672037218285399314754962745101428038350620800434949465218918572960992855 75\
9289013690178929740102531403970352598570639159023398747661385373341746089774 50\
1204078378675191424835902190447613412650078890658666006760829500174469049953 99\
4106331358178877503462020826077716052600197165634900231758837549558571754636 39\
5004947923237730432338501325062476980398682256385767568375514105110479783322 90\
2058771575935449384101497960270562313536139160379231716828126717946582180821 73\
8745897551509760921038303838206640156426713544220512707953222623981930348230 96\
```

```
289106738648027465115643116289598101892616540617259965788637502513907584837387\
225325523252557076735947947109718165109579619422179309472311561680600738568935\
630503741711664946129669518743765681202192406147520625522461185193783700100160\
113014522750483341709515052667684299073434042736477264247130180631825957379633\
391555715083014793908928945373653470181883776448174900387742061372817337747594\
374373379624357229272738317567353633474897895189647208224081567916716929986700\
841623565279870767227357548849968039344418934375976553263639814238538708852340\
880144058186420518320033835473982615324212544039677,
104336804378885946260911769261950442813130531006877125778303804559865788486418\
593279052383200959651299365383107103980325199225394883843837717745906273841749\
568132454576089569959283467278710773388808323311086754500520694999255694061061\
574496239830256618193629365296898867894827083435477233488872571032702441306116\
689174974116070190893512449630421162269468552401641338158192669941226351978776\
684355223473462727958304397430522034518163637158423406373822538372383435066967\
289106738648027465115643116289598101892616540617259965788637502513907584837387\
225325523252557076735947947109718165109579619422179309472311561680600738568935\
630503741711664946129669518743765681202192406147520625522461185193783700100160\
113014522750483341709515052667684299073434042736477264247130180631825957379633\
391555715083014793908928945373653470181883776448174900387742061372817337747594\
374373379624357229272738317567353633474897895189647208224081567916716929986700\
841623565279870767227357548849968039344418934375976553263639814238538708852340\
880144058186420518320033835473982615324212544039677,
0,
316812088817093152728457672492788628425469697551892400491428544714653018827350\
464325024774084211919841700732377442837398737845999542143375055884203392061573\
846267074694962490588085704840804721374526075328366162313019444394783679628622\
680431810805790374331126720782894755640641734763574697963787639056963994983578\
153274596821331206195935368573668211734681391237563992443647744718838312744468\
576579543480398805748611116065977775060012655435693796449298608160058092071327\
856253269838064557033275857217521602272290450818249432859170844639722269369256\
426374549411652348119196068985890907988783984528732921495673591110474898323241\
505591212202701967210088803989963706624517029734466422196890389055423857383196\
104421584572738444180801264710224194269226669009729192757319409831544208057450\
015674483504025496535319865196679322379365863353274856231759457715385614493380\
565612193788037624725643518005756855993742518115359362439615091834339392730322\
383282884449136418813315350114809447510859600493816313875273292594212873847475\
695915881216537454009082949129494735490236971229143))
```

Sets of Jacobi sum tests

TEST NR: 1								
q	29	43	0	0	8779	0	0	191
p^k	4	3			11			19
e	0	1			0			9
TEST NR: 2								
q	229	8779	0	0	43891	0	0	8779
p^k	4	3			11			19
e	3	0			1			10
TEST NR: 3								
q	1597	31	0	0	131671	0	0	571
p^k	4	3			11			19
e	0	0			7			2
TEST NR: 4								
q	61	571	0	0	11287	0	0	43891
p^k	4	3			11			19
e	3	1			8			5
TEST NR: 5								
q	421	211	0	0	56431	0	0	11971
p^k	4	3			11			19
e	0	1			7			17
TEST NR: 6								
q	37	43891	0	0	22573	0	0	131671
p^k	4	3			11			19
e	3	2			4			15
TEST NR: 7								
q	4789	127	0	0	790021	0	0	11287
p^k	4	9			11			19
e	2	1			8			14
TEST NR: 8								
q	181	631	0	0	105337	0	0	56431
p^k	4	9			11			19
e	2	2			0			13
TEST NR: 9								
q	109	11971	0	0	526681	0	0	35911
p^k	4	9			11			19
e	2	2			4			17
TEST NR: 10								
q	2053	131671	0	0	16633	0	0	229
p^k	4	9			11			19
e	1	2			3			11
TEST NR: 11								
q	22573	11287	0	0	225721	0	0	1597
p^k	4	27			11			19
e	0	22			7			16
TEST NR: 12								
q	757	379	0	0	1580041	0	0	4789
p^k	4	27			11			19
e	0	0			0			5
TEST NR: 13								
q	541	271	0	0	117041	0	0	2053
p^k	4	27			11			19
e	3	21			0			0
TEST NR: 14								
q	71821	56431	0	0	351121	0	0	22573
p^k	4	27			11			19
e	0	19			1			5
TEST NR: 15								
q	790021	35911	0	0	30097	0	0	71821
p^k	4	27			11			19
e	2	12			7			3
TEST NR: 16								
q	41	229	0	0	55441	0	0	790021
p^k	8	3			11			19
e	6	1			2			15
TEST NR: 17								
q	761	1597	0	0	1053361	0	0	761
p^k	8	3			11			19
e	5	0			10			1
TEST NR: 18								
q	281	61	0	0	90289	0	0	457
p^k	8	3			11			19
e	6	2			2			18
TEST NR: 19								
q	457	421	0	0	23761	0	0	2281
p^k	8	3			11			19

```
e        1        2                  1                  2

TEST NR: 20
q     2281       37        0        0   451441        0        0   105337
p^k      8        9                  11                 19
e        4        7                   8                 13

TEST NR: 21
q       17     4789      191        0        0        0        0     6841
p^k     16        9        5                                        19
e       13        3        3                                         0

TEST NR: 22
q      113      181       31        0        0        0        0    47881
p^k     16        9        5                                        19
e        7        1        2                                        13

TEST NR: 23
q     2129      109      571        0        0        0        0   526681
p^k     16       27        5                                        19
e        9        7        1                                         6

TEST NR: 24
q   117041     2053      211        0        0        0        0    28729
p^k     16       27        5                                        19
e        4        2        1                                        10

TEST NR: 25
q      337    22573    43891        0        0        0        0    20521
p^k     16       27        5                                        19
e       12        2        3                                         7

TEST NR: 26
q      241      757      631        0        0        0        0   225721
p^k     16       27        5                                        19
e        4       22        3                                         4

TEST NR: 27
q     4561      541    11971        0        0        0        0  1580041
p^k     16       27        5                                        19
e       12       19        4                                        10

TEST NR: 28
q   351121    71821   131671        0        0        0        0     2129
p^k     16       27        5                                        19
e        5        8        4                                        18

TEST NR: 29
q    30097   790021      271        0        0        0        0   117041
p^k     16       27        5                                        19
e        3       22        4                                         1

TEST NR: 30
q     1009      457    56431        0        0        0        0     4561
p^k     16        3        5                                        19
e        9        1        3                                         6

TEST NR: 31
q    13681     2281    35911        0        0        0        0   351121
p^k     16        3        5                                        19
e       13        1        4                                        13

TEST NR: 32
q    55441       73       61        0        0        0        0    30097
p^k     16        9        5                                        19
e        1        4        3                                        12

TEST NR: 33
q  1053361   105337      421        0        0        0        0    13681
p^k     16        9        5                                        19
e        5        3        4                                         4

TEST NR: 34
q      433     6841      181        0        0        0        0  1053361
p^k     16        9        5                                        19
e       13        8        2                                        16

TEST NR: 35
q     8209     2521      541        0        0        0        0     8209
p^k     16        9        5                                        19
e        8        7        3                                         1

TEST NR: 36
q    90289    47881    71821        0        0        0        0    90289
p^k     16        9        5                                        19
e        5        6        3                                        14

TEST NR: 37
q    57457   526681   790021        0        0        0        0    57457
p^k     16        9        5                                        19
e        1        1        2                                        16

TEST NR: 38
q     2161    28729       41        0        0        0        0   451441
p^k     16       27        5                                        19
e       13       17        3                                         0

TEST NR: 39
q    23761    16633      761        0        0        0        0   287281
p^k     16       27        5                                        19
e        9        7        3                                         9

TEST NR: 40
q   451441    20521      281        0        0        0        0        0
p^k     16       27        5
e        9        6        2

TEST NR: 41
q    15121   225721     2281        0        0        0        0        0
p^k     16       27        5
e        0        4        1

TEST NR: 42
q   287281     7561     6841        0        0        0        0        0
p^k     16       27        5
e        0       26        0

TEST NR: 43
q       73  1580041     2521        0        0        0        0        0
p^k      8       27        5
e        7        3        4

TEST NR: 44
q   105337      337    47881        0        0        0        0        0
p^k      8        3        5
e        5        1        0

TEST NR: 45
q     6841      241   526681        0        0        0        0        0
p^k      8        3        5
e        6        0        2
```

```
TEST NR: 46
q     2521     4561    20521        0        0        0        0        0
p^k      8        3        5
e        2        2        2

TEST NR: 47
q    47881   351121   225721        0        0        0        0        0
p^k      8        3        5
e        5        2        0

TEST NR: 48
q   526681    30097     7561        0        0        0        0        0
p^k      8        9        5
e        2        1        3

TEST NR: 49
q    28729     1009  1580041        0        0        0        0        0
p^k      8        9        5
e        3        2        4

TEST NR: 50
q    16633    13681   117041        0        0        0        0        0
p^k      8        9        5
e        1        2        0

TEST NR: 51
q    20521    55441      241        0        0        0        0        0
p^k      8        9        5
e        3        3        1

TEST NR: 52
q   225721  1053361     4561        0        0        0        0        0
p^k      8        9        5
e        1        7        3

TEST NR: 53
q     7561      433   351121        0        0        0        0        0
p^k      8       27        5
e        1        0        3

TEST NR: 54
q  1580041     8209    13681        0        0        0        0        0
p^k      8       27        5
e        7       25        4

TEST NR: 55
q      191    90289    55441        0        0        0        0        0
p^k      2       27        5
e        1        7        0

TEST NR: 56
q       43    57457  1053361        0        0        0        0        0
p^k      2       27        5
e        0       20        1

TEST NR: 57
q     8779     2161     2161        0        0        0        0        0
p^k      2       27        5
e        1       18        1

TEST NR: 58
q       31    23761    23761        0        0        0        0        0
p^k      2       27        5
e        0       22        4

TEST NR: 59
q      571   451441   451441        0        0        0        0        0
p^k      2       27        5
e        0       21        2

TEST NR: 60
q      211    15121    15121        0        0        0        0        0
p^k      2       27        5
e        1       15        4

TEST NR: 61
q    43891   287281   287281        0        0        0        0        0
p^k      2       27        5
e        0        1        1

TEST NR: 62
q      127        0        0       43        0        0        0        0
p^k      2                 7
e        1                 5

TEST NR: 63
q      631        0        0     8779        0        0        0        0
p^k      2                 7
e        1                 6

TEST NR: 64
q    11971        0        0      211        0        0        0        0
p^k      2                 7
e        0                 0

TEST NR: 65
q   131671        0        0    43891        0        0        0        0
p^k      2                 7
e        0                 2

TEST NR: 66
q    11287        0        0      127        0        0        0        0
p^k      2                 7
e        1                 6

TEST NR: 67
q      379        0        0      631        0        0        0        0
p^k      2                 7
e        1                 2

TEST NR: 68
q      271        0        0    11971        0        0        0        0
p^k      2                 7
e        0                 3

TEST NR: 69
q    56431        0        0   131671        0        0        0        0
p^k      2                 7
e        0                 4

TEST NR: 70
q    35911        0        0      379        0        0        0        0
p^k      2                 7
e        1                 1

TEST NR: 71
q        0        0        0    35911        0        0        0        0
p^k                        7
e                          5

TEST NR: 72
q        0        0        0       29        0        0        0        0
```

```
p^k                                   7
e                                     5

TEST NR: 73
q          0     0     0          1597     0     0     0     0
p^k                                   7
e                                     5

TEST NR: 74
q          0     0     0           421     0     0     0     0
p^k                                   7
e                                     3

TEST NR: 75
q          0     0     0          4789     0     0     0     0
p^k                                   7
e                                     3

TEST NR: 76
q          0     0     0           757     0     0     0     0
p^k                                   7
e                                     1

TEST NR: 77
q          0     0     0         71821     0     0     0     0
p^k                                   7
e                                     2

TEST NR: 78
q          0     0     0        790021     0     0     0     0
p^k                                   7
e                                     5

TEST NR: 79
q          0     0     0           281     0     0     0     0
p^k                                   7
e                                     5

TEST NR: 80
q          0     0     0        105337     0     0     0     0
p^k                                   7
e                                     6

TEST NR: 81
q          0     0     0          2521     0     0     0     0
p^k                                   7
e                                     6

TEST NR: 82
q          0     0     0         47881     0     0     0     0
p^k                                   7
e                                     3

TEST NR: 83
q          0     0     0        526681     0     0     0     0
p^k                                   7
e                                     2

TEST NR: 84
q          0     0     0         28729     0     0     0     0
p^k                                   7
e                                     6

TEST NR: 85
q          0     0     0         16633     0     0     0     0
p^k                                   7
e                                     0

TEST NR: 86
q          0     0     0          7561     0     0     0     0
p^k                                   7
e                                     6

TEST NR: 87
q          0     0     0       1580041     0     0     0     0
p^k                                   7
e                                     4

TEST NR:  88
q          0     0     0           113     0     0     0     0
p^k                                   7
e                                     4

TEST NR: 89
q          0     0     0          2129     0     0     0     0
p^k                                   7
e                                     2

TEST NR: 90
q          0     0     0        117041     0     0     0     0
p^k                                   7
e                                     1

TEST NR: 91
q          0     0     0           337     0     0     0     0
p^k                                   7
e                                     1

TEST NR: 92
q          0     0     0        351121     0     0     0     0
p^k                                   7
e                                     3

TEST NR: 93
q          0     0     0          1009     0     0     0     0
p^k                                   7
e                                     4

TEST NR: 94
q          0     0     0         55441     0     0     0     0
p^k                                   7
e                                     0

TEST NR: 95
q          0     0     0       1053361     0     0     0     0
p^k                                   7
e                                     3

TEST NR: 96
q          0     0     0         57457     0     0     0     0
p^k                                   7
e                                     6

TEST NR: 97
q          0     0     0         15121     0     0     0     0
p^k                                   7
e                                     0

TEST NR: 98
q          0     0     0        287281     0     0     0     0
p^k                                   7
e                                     4
```

5. COMPARISON.

In this section we will compare the test with its two major competitors, namely the old Jacobi sum test (cf. [29], [30]) and the complex multiplication test (cf. [108], [109]). These were the two fastest general purpose primality tests.

(5.1) The Jacobi sum test of Cohen and Lenstra.

In [29] an implementation of the Jacobi sum test is presented. To prove the primality of a number n, the program runs through seven stages. First the program looks for possible factors of n less than 10^6. If no factors are found, four Miller-Rabin compositeness tests are performed. If n is a pseudoprime for all four compositeness tests, the remaining five stages, together forming the actual Jacobi sum test, are performed on the number n. The computations were initially done on a CDC 170/750 computer.

number of digits	Trial division up to 10^6	4 Miller-Rabin tests	Lucas-Lehmer test	Selection of t and s	Jacobi sum tests	Additional tests	Final trial division	Total running time
100	7.965	0.567	2.211	0.017	37.334	0.000	2.336	50.442
	0.039	0.015	0.936	0.003	15.696	0.000	1.379	15.203
	8.019	0.602	3.930	0.023	62.705	0.000	6.216	75.416
	7.824	0.544	0.724	0.011	12.426	0.000	1.099	26.031
120	7.972	0.759	2.419	0.017	78.151	0.000	8.468	97.797
	0.025	0.023	0.777	0.003	24.042	0.000	7.062	28.274
	8.010	0.803	4.348	0.024	113.357	0.000	27.571	147.259
	7.887	0.723	0.864	0.012	34.503	0.000	2.442	51.077
140	7.963	0.957	3.705	0.016	130.251	0.000	13.525	156.429
	0.027	0.029	1.547	0.003	42.919	0.000	5.257	43.122
	8.022	0.999	6.371	0.023	186.919	0.000	28.782	210.756
	7.904	0.906	0.480	0.012	52.947	0.000	2.546	77.316
160	7.951	1.292	5.086	0.015	205.347	0.000	26.501	246.204
	0.047	0.054	2.722	0.002	45.350	0.000	8.301	44.144
	8.010	1.387	12.615	0.019	252.452	0.000	33.927	298.144
	7.778	1.181	2.147	0.011	64.833	0.000	16.045	111.888
180	7.973	1.558	5.354	0.014	308.475	0.000	36.341	359.728
	0.016	0.059	2.031	0.002	56.701	0.000	0.658	55.833
	7.999	1.680	9.494	0.020	392.170	0.000	37.930	439.039
	7.926	1.472	1.365	0.011	206.021	0.000	35.280	259.021
200	7.950	1.998	6.653	0.015	438.143	0.000	40.978	495.748
	0.035	0.127	2.214	0.002	80.472	0.000	1.606	80.025
	8.000	2.191	10.469	0.020	560.381	0.000	43.292	614.254
	7.859	1.552	2.834	0.012	205.896	0.000	35.761	258.859

Above a table from [29] is presented, where for each stage the average time, the standard deviation, the maximum time as well as the minimum time measured in seconds of elapsed CPU time is listed.

About two years later, an especially dedicated version of the program was designed by A.K. Lenstra on a Cray-1 at AT&T-Bell Labs. A table of the elapsed CPU time on the Cray-1 measured in seconds is listed below, cf. [81].

number of digits	Trial division up to 10^6	4 Miller-Rabin tests	Lucas-Lehmer test	Selection of t and s	Jacobi sum tests	Additional tests	Final trial division	Total running time
50	1.997	0.021	0.101	0.001	0.318	0.005	0.089	2.538
	0.020	0.000	0.054	0.001	0.266	0.022	0.063	0.259
	2.049	0.021	0.226	0.002	0.912	0.098	0.303	3.087
	1.973	0.021	0.027	0.000	0.000	0.000	0.000	2.139
60	2.050	0.026	0.110	0.001	0.827	0.000	0.139	3.160
	0.012	0.000	0.043	0.001	0.389	0.000	0.092	0.390
	2.071	0.026	0.214	0.002	1.601	0.000	0.338	3.825
	2.031	0.025	0.047	0.000	0.000	0.000	0.000	2.300
70	2.130	0.033	0.131	0.002	1.389	0.000	0.223	3.914
	0.013	0.000	0.031	0.001	0.651	0.000	0.100	0.676
	2.161	0.033	0.195	0.002	2.312	0.000	0.371	4.976
	2.121	0.032	0.089	0.001	0.345	0.000	0.090	2.823
80	2.178	0.039	0.158	0.002	1.886	0.000	0.306	4.576
	0.011	0.000	0.042	0.000	0.741	0.000	0.284	0.702
	2.206	0.040	0.245	0.002	3.450	0.000	1.099	6.009
	2.168	0.039	0.104	0.001	0.444	0.000	0.090	2.984
90	2.248	0.046	0.194	0.002	3.130	0.000	0.346	5.972
	0.008	0.000	0.064	0.000	1.104	0.000	0.239	1.059
	2.262	0.047	0.342	0.002	5.414	0.000	1.217	8.290
	2.240	0.046	0.061	0.001	1.072	0.000	0.108	4.016
100	2.337	0.058	0.234	0.002	3.888	0.000	0.522	7.047
	0.015	0.001	0.082	0.000	1.466	0.000	0.366	1.549
	2.362	0.061	0.395	0.002	6.144	0.000	1.354	9.174
	2.307	0.057	0.121	0.002	0.852	0.000	0.242	3.822
110	2.368	0.066	0.276	0.002	6.545	0.000	0.895	10.159
	0.015	0.001	0.110	0.000	2.095	0.000	0.839	2.558
	2.408	0.068	0.514	0.002	10.438	0.000	2.751	15.784
	2.354	0.066	0.085	0.001	3.465	0.000	0.131	6.847
120	2.462	0.081	0.322	0.002	8.999	0.000	1.867	13.740
	0.025	0.001	0.176	0.000	2.440	0.000	1.241	3.028
	2.553	0.082	0.782	0.003	12.377	0.000	3.259	18.228
	2.446	0.080	0.102	0.001	4.262	0.000	0.281	8.636

number of digits	Trial division up to 10^6	4 Miller-Rabin tests	Lucas-Lehmer test	Selection of t and s	Jacobi sum tests	Additional tests	Final trial division	Total running time
130	2.504	0.092	0.355	0.002	11.186	0.000	2.536	16.682
	0.009	0.001	0.130	0.000	3.102	0.000	1.703	4.016
	2.525	0.093	0.637	0.002	17.413	0.000	7.018	23.901
	2.493	0.092	0.127	0.001	3.953	0.000	0.607	8.852
140	2.599	0.112	0.407	0.002	15.966	0.000	4.326	23.417
	0.021	0.002	0.141	0.000	4.225	0.000	1.755	4.265
	2.660	0.120	0.597	0.002	23.988	0.000	7.673	30.610
	2.586	0.110	0.134	0.001	4.614	0.000	0.697	11.561
150	2.665	0.126	0.451	0.002	21.450	0.000	4.214	28.916
	0.032	0.001	0.166	0.000	5.287	0.000	1.328	5.253
	2.776	0.131	0.749	0.002	28.411	0.000	7.909	35.349
	2.634	0.124	0.203	0.001	10.715	0.000	2.000	17.475
160	2.746	0.149	0.492	0.002	27.136	0.000	5.137	35.670
	0.018	0.002	0.206	0.000	5.990	0.000	1.721	5.206
	2.790	0.155	1.024	0.002	37.494	0.000	8.563	44.974
	2.727	0.146	0.238	0.001	19.432	0.000	4.216	28.674
170	2.787	0.167	0.479	0.002	32.589	0.000	6.875	42.907
	0.013	0.001	0.137	0.000	8.555	0.000	2.355	8.600
	2.809	0.168	0.708	0.002	43.430	0.000	9.408	55.805
	2.772	0.166	0.188	0.001	11.769	0.000	4.502	19.963
180	2.816	0.186	0.678	0.002	35.584	0.000	8.589	47.862
	0.008	0.001	0.263	0.000	7.976	0.000	1.934	8.613
	2.841	0.187	1.327	0.002	52.153	0.000	9.996	65.327
	2.810	0.185	0.218	0.001	18.338	0.000	4.800	27.051
190	2.926	0.217	0.777	0.001	50.340	0.000	9.722	63.991
	0.016	0.001	0.241	0.001	7.323	0.000	1.552	7.448
	2.977	0.219	1.204	0.002	64.763	0.000	10.705	78.540
	2.912	0.215	0.465	0.001	35.841	0.000	5.164	50.891
200	2.965	0.240	0.888	0.002	53.542	0.000	10.724	68.369
	0.013	0.001	0.320	0.000	15.458	0.000	1.182	15.827
	3.005	0.243	1.772	0.002	76.681	0.000	11.416	90.975
	2.959	0.239	0.323	0.001	25.895	0.000	5.814	40.960
210	3.056	0.277	1.011	0.002	63.687	0.000	11.605	79.646
	0.003	0.001	0.385	0.000	20.628	0.000	1.355	20.822
	3.064	0.279	2.078	0.002	92.904	0.000	12.399	108.821
	3.052	0.275	0.507	0.001	24.337	0.000	5.999	40.411
220	3.113	0.298	0.980	0.002	79.242	0.000	12.226	95.932
	0.013	0.002	0.278	0.001	17.781	0.000	0.263	17.747
	3.137	0.303	1.429	0.002	104.506	0.000	12.757	121.736
	3.098	0.295	0.447	0.001	47.260	0.000	11.811	64.061

(5.2) The complex multiplication test.

In [108] an algorithm is presented to prove the primality of a number n using elliptic curves over $\mathbf{Z}[i]$ and quadratic forms. For further details concerning the theory of this algorithm, we refer to [108] and to I.10. This algorithm is called the complex multiplication test. Although this test is called "Elliptic Curve Primality Proving (ECPP) algorithm" in [109], we prefer the name "complex multiplication test" in order to make a distinction between this test and the Elliptic Curve Primality Proving algorithm of [46], and because the "complex multiplication test" does not solely uses elliptic curves, in the way the algorithm in [46] does.

To prove the primality of a number n, the program runs through three stages. First the program looks for possible factors of n less than 10^4. If no factors are found four probabilistic compositeness tests are performed on the number n. If n is a probable prime for all four probabilistic compositeness tests, the complex multiplication test is performed on the number n.

The computations of Morain were done on a SUN-3/60. Below we present a table from [108], where for each stage we list the average time, the standard deviation, the maximum time, as well as the minimum time measured in seconds of elapsed CPU time.

number of digits	50	100	120	140	160	180	200
	0.7	0.8	0.8	0.8	0.8	0.9	0.9
Trial	0.0	0.0	0.0	0.0	0.0	0.0	0.0
division	0.8	0.8	0.8	0.9	0.9	0.9	0.9
up to 10^4	0.7	0.8	0.8	0.8	0.8	0.8	0.8
four	1.7	6.1	9.4	13.2	17.1	21.8	27.7
probabilistic	0.1	0.4	0.6	1.0	1.4	1.5	2.0
compositeness	1.9	6.9	10.5	14.7	20.4	24.9	30.9
tests	1.6	5.6	8.4	12.0	15.3	19.6	24.9
	383.6	4108.1	6567.8	10917.0	17762.5	27404.4	36905.6
complex	192.4	2181.7	1965.2	4648.3	6423.1	9327.9	11212.0
multiplication	715.6	10407.1	10504.4	16651.3	33091.7	43803.5	66597.8
test	76.9	1441.3	2637.2	4862.9	8391.4	13113.8	18436.8

In [109] a comparison is made between the complex multiplication test and the Jacobi sum test. In these the average time needed for the two primality tests is compared to the time needed by both algorithms to perform four Miller-Rabin probabilistic compositeness tests.

In the comparison as presented by [109] the time needed for the complex multiplication test to prove the primality of 100-digit numbers is a factor 3.781 faster than the results

from [108] as given above. No data are given for numbers of any other size. Nevertheless we will mention this comparison.

	old Jacobi sum			complex multiplication		
	test	4×MR	ratio	test	4×MR	ratio
mean	50.442	0.567	88.96	1086.3	4.4	246.88
standard dev.	15.203	0.015		355.9	0.4	
maximum	75.416	0.602		1948.4	5.0	
minimum	26.031	0.544		546.2	4.0	

Since the tests are performed on different machines having different sizes for one computer-word, it is hard to compare these tests with each other. The time needed by the basic routines to multiply, divide and add multiple length integers have an enormous impact on the time needed for the complete tests. These are also influenced by the word lengths.

However both in the tests as well as in the probabilistic compositeness test long integer routines are used. This implies that, regardless of the basic routines that are used the ratio of the time needed by the tests and the time needed to perform four probabilistic compositeness tests indicates which method is faster. If we would compare these ratios above, one could easily conclude that for numbers of 100 digits the old Jacobi sum test is superior to the complex multiplication test.

If we would use the table from [108] to compare the ratios of the mean time needed for both tests of Cohen and Lenstra, the complex multiplication test as well as our new Jacobi sum test and the mean time needed for four probabilistic compositeness tests for $\log_{10}(n) = 100, 120, \ldots, 200$ we get

	digits	100	120	140	160	180	200
old Jacobi	test	50.442	97.797	156.429	246.204	359.728	495.748
sum test	4×MR	0.567	0.759	0.957	1.292	1.558	1.998
Cyber-170/750	ratio	88.962	128.849	163.457	190.560	230.885	248.122
old Jacobi	test	7.047	13.740	23.417	35.670	47.862	68.369
sum test	4×MR	0.058	0.081	0.112	0.149	0.186	0.240
Cray-1	ratio	121.500	169.629	209.080	239.396	257.323	284.870
complex	test	4108.100	6567.800	10917.000	17762.500	27404.400	36905.600
multiplication	4×MR	6.100	9.400	13.200	17.100	21.800	27.700
SUN-60/3	ratio	673.459	698.702	827.045	1038.742	1257.082	1332.332
new Jacobi	test	108.650	190.83	418.030	671.450	1017.890	1458.870
sum test	4×MR	2.160	3.610	5.380	7.730	10.790	15.030
SUN-4	ratio	50.404	52.86	77.744	86.874	94.345	97.096

This table seems to indicate that the ratio of the time needed for the primality test of [29] and the time needed for four compositeness tests grows faster than the ratio of the time needed for the complex multiplication test and the time needed for four compositeness tests. The cross-over point, probably around 3500 decimal digits, seems to be beyond the capabilities of both methods, although this is a somewhat premature conclusion.

Comparing the old Jacobi sum test and the test of [108] to our test, the table given above indicates that our test is the fastest test for each size. Furthermore, if we compare our test with the test of [108] we do have the advantage that the computer-word-length for the SUN-3/60 is the same as for a SUN-4 (as well as for the DEC-3100). Therefore it is possible to compare the performance of the complex multiplication test directly to our test. Since the SUN-4 is about three times as fast as a SUN-3/60 (and the DEC-3100 is about six times as fast as a SUN-3/60) this gives us another indication that our test is superior to the complex multiplication test. Even for very large examples (cf. VI.4) our test is superior to the complex multiplication test.

Remark. Very recently new information about the performance of the complex multiplication method was published in [6]. The timings mentioned in [6] are faster than the timings of the complex multiplication method that were mentioned above. There are two reasons however that the new results are hard to compare with the results of the other methods. First of all, this paper does not supply information about the CPU-time for compositeness tests, which were used to compare the different methods. The second objection is that the new results are given as a function of the number of words (usually a block of 30 or 32 bits) needed to represent a prime. This differs from the earlier results of the complex multiplication method as well as from the results of all the Jacobi sum tests. However if we discard all these problems, and try to estimate the time needed for compositeness tests in [6], by using earlier results we find that the complex multiplication test is still inferior to the old Jacobi sum test, and therefore also inferior to the new Jacobi sum test.

Remark. There is one aspect for where our algorithm is inferior in comparison to the complex multiplication method. The complex multiplication method is able to provide a proof that can be verified in polynomial time. As mentioned in V.5, our algorithm is not able to do so.

VII. INSTRUCTIONS FOR USE.

1. INTRODUCTION.

This chapter is intended to provide guidance to a user who, having all the sources of the primality testing program, wants to run the program. Before using the program itself, one has to change the program in such a way that it is suitable to be run on one's own favorite computer. The changes that have to be made to the program are mainly a change of parameters that depend on the computer's hardware.

The program has been written in standard **Fortran-77**, since this language is available on most computers.

The Fortran programming language has, as most programming languages, some limitations for integers which it can represent. These integers, which we will call Fortran integers (not to be confused with integers in mathematics), have a limited range. The Fortran-program is initially set up to execute the primality-test on a computer with 32-bits Fortran-integers and 8-bits bytes; it is assumed that the maximal representable Fortran-integer is equal to $2^{31}-1$. Integers larger than $2^{30}-1$ are represented as arrays with each entry at most $2^{30}-1$. The length of these arrays is at most 200. In particular this implies that the computer program is initially set up to execute the primality-test for numbers of up to around $200 \cdot \log_{10}(2^{30}-1) \approx 1806$ decimal digits.

The amount of memory needed by the program is highly dependent on the magnitude of various parameters, which are dependent on the size of integers one likes to handle in the primality test. Since Fortran needs upper bounds of arrays to be fixed, i.e., not dynamically determined during execution of the program, this implies that the amount of memory needed to execute the program is independent of the size of the number n handled by the primality test, and only dependent on the size of N, the maximal number that can possibly be handled by the primality test.

For integers of up to 1806 decimal digits, the program needs, apart from the memory space needed to load the program itself, approximately 2 megabytes ($= 2 \cdot 10^6$ bytes) of memory. This amount of memory is almost linear in $\log N$.

In Fortran, input and output operations are read from or written to "units". In our program, the input is assumed to be on standard unit 5, and the output will be put on unit 6. This is the Fortran-standard on most computers. If you are not familiar with these terms, please consult a local expert.

To change the parameters in such a way that it is suited for the machine's hardware is straightforward. To change the parameters in such a way that the program needs less

memory is not recommended to users with little experience on computers.

Next, after all parameters have been changed in the proper way, a few files containing various tables have to be generated which will be used by the program (except if one tries to prove the primality of a number less than or equal to 2).

Although the program has not been written to run on a particular computer or for one particular operating system, we will assume in our exposé that we try to use the program on a system running under the UNIX® operating system. It should not be difficult to replace typical UNIX-commands by commands in any other operating system.

The source of the computer program for primality testing is divided into several parts in order not to load all the routines into one large executable file. This is also done because not all of the routines and functions are needed while executing various parts of the program. The source consists of 5 files containing programs:

`extgn.f` . . . . . Program to generate cyclic extensions
`jgen.f` . . . . . Program to generate Jacobi sums
`pqgn.f` . . . . . Program to generate p-primes and q-primes
`prmgen.f` . . . . . Program to generate a table of primes
`test.f` . . . . . The primality test

and 20 files containing various functions and subroutines:

`bufop.f` . . . . . Buffered output routines
`cyclic.f` . . . . . Routines to generate cyclic extensions of $\mathbf{Z}/n\mathbf{Z}$
`cyclotomic.f` . . . . . Routines to generate roots of unity
`ftrt23.f` . . . . . Routines for the final trial division
`gnop1a.f` . . . . . Operations on Jacobi sums
`gnop1b.f` . . . . . Routines to express Gauss sums in terms of Jacobi sums
`ibasic.f` . . . . . Elementary routines on integers
`init.f` . . . . . Initialization routines
`jpqbas.f` . . . . . Basic routines for p-primes, q-primes and Jacobi sums
`jstst.f` . . . . . Routines for the Jacobi sum test
`lbas.f` . . . . . Elementary routines on long integers
`lop.f` . . . . . Operations on long integers
`lopje.f` . . . . . Operations to speed up the operations in `lop.f`
`mchdep.f` . . . . . Machine dependent routines

® UNIX is a registered trademark of AT&T.

`nop.f` . . . . . Operations on long integers modulo n
`optim.f` . . . . . Optimization routines
`pop.f` . . . . . Operations on polynomials modulo n
`millerrabin.f` . . . . . Miller-Rabin's compositeness test
`ran.f` . . . . . Random generators
`sort.f` . . . . . Sorting routines

Finally there are a few files with helpful programs, subroutines and functions:
`btoi.f` . . . . . Conversions of binary files to integer files
`chkmch.f` . . . . . Routines to check machine-dependent constants
`itob.f` . . . . . Conversion of integer files to binary files
`jaccheck.p` . . . . . Program to validate a Jacobi sum

In order not to generate the Jacobi sum table once more, the file:
`deci.19` . . . . . Decimal coded file of Jacobi sums
has been added to the set of source files. The reason why will be explained in VII.2.

(1.1) Remark. For those machines having the UNIX operating-system a *Makefile* is provided which performs all the necessary operations on these Fortran source files.

2. SETTING UP.

In this section we will describe how to modify the sources of the primality test in order to run it on your own computer. First we will describe how to modify parameters that depend on the machine's hardware. Next we will describe how to modify the program in such a way that it needs less memory. Most users will not be interested in that section. Moreover modifying the program to save memory space needs great care, since the values of various parameters should be changed in such a way that they do not conflict. Changing these parameters is not recommended to users having little experience in programming.

Finally, it is explained how to generate the files containing tables needed by the primality program.

(2.1) Changing the parameters depending on the machine's hardware.

In order to change the parameters needed by the program to make the program suited for your computer, one has to modify the contents of the file `mchdep.f`. This file contains two routines:

```
real function second()
```

and

```
subroutine getcon(maxint, iwords)
```

Since there does not exist a standard Fortran routine to calculate the elapsed time that the program resides in the computer, there is no standard way to measure the computing time used by the program.

In order to enable the program to calculate the elapsed time one has to provide a function such that the difference of two values provided by two subsequent calls to the function `second()` gives the elapsed computing time between two calls. On most computers there is a built-in function which performs this task. These functions are mostly called `time` or `etime`. In these cases one can solve the problem by writing a function `second()` that uses this built-in function.

If the built-in function is called `second`, of course one does not have to provide a function `second` that calls this built-in function. If one is not able to write such a function, which is very unlikely, then a dummy function:

```
real function second()
second=0.0
return
end
```

will do. In this case the program will not be able to calculate the time it needs to run, but it will be able to minimize the time needed for the test to complete, although the minimization will be done using time-functions which will not be especially suited for the machine one is using.

In the second routine one has to change the machine dependent constants `maxint` and `iwords`. The constant `maxint` should have the value of the maximal representable integer on your machine. Usually, this is $2^{k-1}-1$, where k is the number of bits in a machine-word. This is not always the case, however. Unfortunately, there is no standard way to calculate this constant.

Another constant, which cannot be routinely calculated is the constant `iwords`, which specifies the number of bytes which is used to store an integer.

For some machines (a VAX, a Sun-3, a Sun-4, a DEC-3100, and some CDC-models), these constants have already been specified. These are the only machine dependent changes one has to make to run the programs on your own machine.

(2.2) Saving memory space.

As has been indicated in VII.1, the memory used by the algorithm is almost linear in $\log N$. Therefore the easiest way to save memory is to change the upper bounds of the arrays used to represent the multiple-length integers. This reduces the bound on $\log N$, which implies that the program is only capable of performing the test on smaller numbers. The upper bound of the arrays used to represent the multiple-length integers is called `maxml`. This constant is initiated in almost all routines handling multiple-length integers. Each initialization of `maxml` (having the value 200) should be changed.

Any other change to save memory is not recommended.

(2.3) Generating the tables.

In order to use the primality test, a few files containing tables have to be generated. The generation of these files will be discussed separately.

Generating the prime table.

In order to generate the same file as described in IV.2, no changes have to be made. If one wishes to generate more (or less) primes, we have to make a few changes in the file `prmgen.f`. In the program a call is made to a subroutine prgen having parameters pbound, gapbnd and dbnd.

The parameter pbound represents the bound on the primes that have to be generated. This should be a single precision integer, i.e., pbound should be at most `maxint`.

The parameter gapbnd represents the bound on the maximal gap between two consecutive primes at most pbound. A table containing these gaps can for instance be found in [132].

Finally the parameter dbnd represents the number of differences that have to be *packed* in one computer word. It should be at most $\lfloor \log(\texttt{maxint})/\log(\texttt{gapbnd}) \rfloor$.

If these parameters are set to the proper values, one can compile the files `prmgen.f` and `mchdep.f` and execute the binary program, generating a *binary-formatted* prime file on standard unit 13. This file is usually called `fort.13` or `tape13`. On the standard output all primes up to pbound are listed.

Generating the table containing all information about the values of t, the p-primes, and the q-primes.

In order to generate the same file as described in IV.2, no changes have to be made. By compiling the files `pqgn.f`, `bufop.f`, `lbas.f`, `lop.f`, `lopje.f`, `mchdep.f`, `init.f`, and `sort.f` and executing the binary program, a *binary-formatted* file is generated on standard unit 17 (usually called `fort.17` or `tape17`). On the standard output the maximal value t_0 for t and its factorization is listed, and for all primes q with $q-1 \mid t_0$ the factorization of $q-1$ and the index of q is listed.

Generating the Jacobi sum table (first method).

In order to generate the same Jacobi sum file as described in IV.2, no changes have to be made. By compiling the files `jgen.f`, `bufop.f`, `ibasic.f`, `init`, `jpqbas.f`, `lbas.f`, `lop.f`, `lopje.f` and `mchdep.f` and executing the binary program, a file is generated on standard unit 19 (usually called `fort.19` or `tape19`). The program is able to generate Jacobi sums for a sequence of q-primes, or to restart the generation just for one single q-prime, to list the Jacobi sums for specified primes, to clear the contents of the Jacobi sum file, and to list Jacobi sums in TEX® -format. The program uses the information of the

® TEX is a registered trademark of the American Mathematical Society.

table stored on unit 17. Apart from specifying what listing should be made the program needs and asks as input the indices of the first and the last q-prime in the sequence.

In case of the generation of the Jacobi sums the program will list for each prime q with $q-1|t$ and each prime $p|q-1$ all Jacobi sums on the standard output.

Remark. The program takes a considerable amount of time to generate the Jacobi sums for the largest q-primes. If $t_0 = 6983776800$, generation of the Jacobi sums for the largest q-prime takes about twenty days on a SUN-4. Therefore a second method will be specified to generate this file.

Generating the Jacobi sum table (second method).

The Jacobi sum file generated on standard unit 19 by the method described above is formatted in a *binary* format. This implies that a file generated on one machine cannot always be used on another machine. However first converting the binary file into a text-file, specifying all integers in *text*-format, next transferring the file to another machine, and finally converting it back to its original form, makes it possible to use the Jacobi sum file generated on another machine. Since we did generate this file, we converted it to text format by the program on the files `btoi.f` and `mchdep.f` to a text-file `deci.19`. The user who wants to use this file simply has to use the program from the compiled source-files `itob.f` and `mchdep.f` to convert `deci.19` into the binary file `fort.19` or `tape19`.

Generating the extension table.

In order to generate the same extension file as described in IV.2, no changes have to be made. By compiling the files `extgn.f`, `bufop.f`, `ibasic.f`, `init`, `lbas.f`, `lop.f`, `lopje.f` and `mchdep.f` and executing the binary program, a *binary-formatted* file is generated on standard unit 23 (usually called `fort.23` or `tape23`).

The program does not need any input and while generating the binary extension file it will list all extensions of prime power degree $u = 2^{k_2}$, $1 \leq k_2 \leq 4$, of prime power degree $u = 3^{k_3}$, $1 \leq k_3 \leq 2$, and of degree $u = 5$. The corresponding conductors m will be less than 160. For each pair (u,m), the minimal polynomial f of the generator, the discriminant of f, the matrices S and S^* and the denominator D (cf. IV.2) will be listed. This is sufficient for $t_0 = 6983776800$ and its divisors.

Generating the primality testing program.

The last setting up that has to be done is the generation of the binary primality testing program. By compiling the main program `test.f` and the files `bufop.f`, `optim.f`, `cyclic.f`, `cyclotomic.f`, `frt23.f`, `gnop1b.f`, `gnop1a.f`, `ibasic.f`, `init.f`, `jpqbas.f`,

`jstst.f`, `lbas.f`, `lop.f`, `lopje.f`, `mchdep.f`, `nop.f`, `pop.f`, `millerrabin.f`, and `ran.f`, we will get a binary program that can perform the primality test. This concludes all the setting up we have to do in order to be able to run the primality test.

3. RUNNING A PRIMALITY TEST.

To show how to run the primality test, we will give an example and show what occurs and what information has to be given.

In order to run the program, all the files of which the generation has been described in the previous section have to be *locally* present. In UNIX-terms this means that these files should be in the current working directory. Suppose that the primality testing program has been called `PRIT`. We will execute `PRIT` in an interactive environment, i.e., input is tranferred to the standard input (logical unit 5), which is the keyboard, and output is tranferred to the standard output (logical unit 6), which is the terminal-screen. Suppose that we would like to prove the primality of 4899633286613100914950380950653312638379, which is a 40-digit number that is found by a procedure which randomly chooses a 40-digit number and checks if the number fails to pass 4 compositeness tests.

If we execute the program `PRIT`, the program prompts

```
enter n (number to be tested prime)
```

By typing in

```
4899633286613100914950380950653312638379
```

the program prompts with

```
enter start=0/restart=1
```

Since we would like to start, we type

```
0
```

Restarting the program will be discussed in VII.5.

Next the program finds the maximal powers of 2 in $n^w - 1$, where $w \leq 20$. So

```
2 ^ 1 | n ^ 1 - 1
2 ^ 3 | n ^ 2 - 1
2 ^ 4 | n ^ 4 - 1
2 ^ 5 | n ^ 8 - 1
2 ^ 6 | n ^ 16 - 1
```

Next, the program needs a trial division bound and prompts with

```
enter trial division bound
```

Any bound ≤ 1000000 can be specified. Any larger bound will be changed into 1000000. Here we take

```
10000
```

Furthermore the program needs an upper bound for ω in $n^\omega - 1$, i.e., which cyclotomic polynomials have to be examined. This upper bound is at most 20.

```
enter search limit for w
```

which will here be given as

```
4
```

in this case. The factors that will be found are

```
3 ^ 1 | n ^ 1 - 1
3 ^ 2 | n ^ 3 - 1
3 ^ 3 | n ^ 9 - 1
5 ^ 1 | n ^ 2 - 1
5 ^ 2 | n ^ 10 - 1
7 ^ 1 | n ^ 2 - 1
7 ^ 2 | n ^ 14 - 1
17 ^ 1 | n ^ 4 - 1
29 ^ 1 | n ^ 1 - 1
43 ^ 1 | n ^ 3 - 1
139 ^ 1 | n ^ 3 - 1
193 ^ 1 | n ^ 2 - 1
trial division of n^i-1 (1 <= i <= 4) up to 10000 in 0.270 seconds
no factor of n found
```

Fortunately, no factors have been found of n, which is what we expected, since we chose a prime number as input.

```
enter number of Miller-Rabin tests
```

Next we may specify the number of Miller-Rabin compositeness tests. Any positive integer $\leq 2^{31} - 1$ will do, but since this number already failed to pass four Miller-Rabin compositeness tests during the procedure of finding this probable prime, we give

```
0
```

Suppose that we know any nice factors of some $n^\omega - 1$, then we can specify them here.

```
enter any known factors div^k of n^w-1
enter w, if no known factors:  w = 0
```

No factors are known, so we type

```
0
```

At this point the program will list the values of n mod p^k, where p^k is a prime power

divisor of $t_0 = 6983776800 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$, as well as the order of n modulo these prime powers. Among these prime powers might be some additional factors of $n^{\text{ord } n \bmod t_0} - 1$ not known before. Next all the factors will be listed again, including their multiplicities and the smallest power w of n for which $n^w - 1$ is divisible by a power of these prime factors. The product of all factors found at this point is

6214562360441154720 0

which is a 20-digit number. This implies that using the generation of roots of unity, the program is able to complete the proof that n is indeed prime. The rings in which the generation of roots of unity has to be performed may have very large order, so that there may be a faster way to prove the primality of n. Therefore the program tries to find a faster way to complete the proof. In order to perform the optimization the program needs to know which final trial division will be chosen (cf. IV.6). This depends on whether $\text{lcm}(s, v)$ is at least $\sqrt[2]{n}$ or $\sqrt[3]{n}$. When the program prompts

```
enter e2or3
```

one may only specify 2 of 3 for one of these methods. In order to make the optimization procedure not too time consuming, we enable the user to choose a proper value instead of the program. We choose

2

Now the program starts to optimize. One may influence this optimization, by forcing the program to perform the optimization stage more than once. This will be discussed in VII.4. Here we will assume that this will not take place. So when the program prompts

```
optimize?(0=no more iterations/1=more iterations)
```

we simply type

0

and the program will display the best solution found up till this point, i.e., without doing any further optimization. This will be a better solution than the solution to use only the generation of roots of unity. First it will display which Jacobi sum tests have to be performed according to the optimal solution.

```
final constants
take 6 of 2
take 2 of 1
```

This means that 6 tests have to be performed in an extension of degree 2, and 2 tests have to be performed in an extension of degree 1. Next the Jacobi sum tests for

combinations of characters will be displayed.

```
set of combinations, expressed in terms of q primes
(       13      43      11      71      0       0       0       0       )
(       61      31      71      43      0       0       0       0       )
(       421     211     31      211     0       0       0       0       )
(       11      13      211     421     0       0       0       0       )
(       71      61      61      0       0       0       0       0       )
(       43      421     421     0       0       0       0       0       )
(       31      0       0       0       0       0       0       0       )
(       211     0       0       0       0       0       0       0       )
```

Each entry represents the conductor of a character in the test. Furthermore in the first column tests with characters of 2-power order are listed, the second column tests of 3-power order, in the third column tests of 5-power order, etc. The parameters that belong to the optimal solution are

```
s is 73336287640759
t is 420
u is 2
v is 4701480
w is 2
```

Next the factorization of t is given.

```
set of p-primes and their powers
        2       4
        3
        5
        7
```

The factorization of $\mathrm{lcm}(t, v)$ in prime powers is given. For these prime powers the program should generate roots of unity or prove the existence of these roots of unity.

```
prime factors and their maximum multiplicity
maximum multiplicity=3, prime factor=2
maximum multiplicity=1, prime factor=3
maximum multiplicity=1, prime factor=5
maximum multiplicity=1, prime factor=7
maximum multiplicity=1, prime factor=29
```

```
maximum multiplicity=1, prime factor=193
product of all factors :
4701480=lcm(t,v)
orders, multiplicities and prime factors
w=2, multiplicity=3, prime factor=2
w=1, multiplicity=1, prime factor=3
w=2, multiplicity=1, prime factor=5
w=2, multiplicity=1, prime factor=7
w=1, multiplicity=1, prime factor=29
w=2, multiplicity=1, prime factor=193
```

This means that the factors $9|n^3-1$, $27|n^9-1$, $25|n^10-1$, $49|n^14-1$, $17|n^4-1$, $17|n^4-1$, $43|n^3-1$ and $139|n^3-1$ will not be used by the test. The generation of the roots of unity corresponding to these factors takes too much time. The only thing that remains to be done is to perform the test. First, the cyclic extension of degree 2 will be generated. Next the roots of unity have to be generated. In this case ζ_8, ζ_5, ζ_7, and ζ_{193} have to be generated in an extension of degree 2. Furthermore ζ_3 and ζ_{29} have to be generated in $\mathbf{Z}/n\mathbf{Z}$.

Only ζ_8, ζ_3, ζ_5, and ζ_7 will be needed to perform the Jacobi sum test, for the other roots of unity the test only needs the proof of their existence. If n would not be prime and if no Rabin-Miller tests would have been performed, then the test will most likely detect that n is composite during the generation of the roots of unity. This would mean that no factor would be found. In this case, since n will be prime, all rots of unity will be generated. After being generated, these roots of unity will be given on the standard output.

All these roots are generated simultaneously. Unfortunately, the generation of ζ_3 and ζ_7 have to be performed once more, since the random elements chosen in this part of the test (which are generated by using our own pseudo-random generator), did not suffice to find these roots of unity. The other roots of unity were found in the first attempt.

After all roots of unity have been generated, the Jacobi sum tests will be performed. We will only list their results, by giving the exponent e, which signifies what power of ζ_{p^k} is equal to $\tau(\chi)^{n-\sigma_n}$.

Test nr 1:

p^k	4	3	5	7

q	13	43	11	71
e	1	1	3	3
Test nr 2:				
p^k	4	3	5	7
q	61	31	71	43
e	1	1	0	0
Test nr 3:				
p^k	4	3	5	7
q	421	211	31	211
e	2	0	3	1
Test nr 4:				
p^k	8	3	5	7
q	11	13	211	421
e	1	2	4	1
Test nr 5:				
p^k	2	3	5	
q	71	61	61	
e	1	1	4	
Test nr 6:				
p^k	2	3	5	
q	43	421	421	
e	0	2	2	
Test nr 7:				
p^k	2			
q	31			
e	1			
Test nr 8:				
p^k	2			
q	211			
e	1			

Finally, the final trial division has to be performed, to exclude the only remaining 420 candidate-divisors of n. If n would not be prime, it is not very likely at all, that n will reach this stage, since the test would probably detect the compositeness of n during the

generation of the roots of unity; therefore, the chance of finding a factor of n in this stage is small.

```
final division started;
lcm(t, w) is 420
lcm(s, v) is 344789089617275623320
>>>final test:  420 trial divisions in 0.100 seconds
>>>final division
>>>did not produce any divisors
>>>n is prime
```

This concludes the primality-proof of 489963328661310091495038095065331263837 9.

Remark. If one chooses not to run the program interactively, but to run the program PRIT by supplying an input file, this input file consists in the most simple form of

```
489963328661310091495038095065331263837 9 . . . . . The number n
0 . . . . . No restart
10000 . . . . . Trial division bound
4 . . . . . Upper bound for w
0 . . . . . No extra factors known
2 . . . . . lcm(s,v) > ∛n
0 . . . . . No further optimization
```

4. HELPING YOUR PRIMALITY TEST.

There are two ways to assist the primality test in generating a primality proof. The first method is by supplying known prime factors of $n^\omega - 1$ for small ω. The second method is by interfering with the optimization stage of the test.

(4.1) Supplying extra prime factors.

It might occur that, before starting the primality test for a number n, properties of n are already known. Suppose for instance that someone already invested a lot of computing time in order to get factors of $n^\omega - 1$ for some ω. In these cases one would like to benefit from these factors by supplying them to the primality test. This is made possible in our test.

After the test did find all the factors of $n^w - 1$ less than a given trial division bound, where w is also less than a requested bound, the program prompts

```
enter any known factors div^k of n^w-1
enter w, if no known factors:  w = 0
```

In the case of the example discussed in the previous section, we happened to know that $524201 \mid n^1 - 1$. So we might type

```
1
```

The program next prompts for the prime divisor. These divisors might be arbitrary long integers but should be prime. This is not checked.

```
enter div
```

So we type

```
524201
```

Finally, the program asks for the exponent of the divisor

```
enter k
1
```

After verifying that 524201 is indeed a divisor of $n^1 - 1$, the program responds with

```
524201 ^ 1 | n ^ 1 - 1
enter w, if no known factors:  w = 0
```

Since we do not know any more factor, we type

```
0
```

In this case the program is able to complete the primality test in roughly 60% of the time it should need if the factor was not given to the program. Initially searching for factors up

to say 600000 would take a considerable part of the time needed to complete the primality test.

(4.2) Interfering with the optimization.

In the ideal situation the primality test should find the optimal solution by itself. Interfering with the optimization would only slow down the total running time. The user of the test, may however possess information about special properties of the number n (for instance additional factors) which would be hard to find by the test. In these cases, interfering might speed up the test.

In these cases one could force the test to invest more time in the optimization test. It tries to find a better solution by performing more iterations and it might try to find more divisors of $n^w - 1$ for some w. So after the prompt

```
optimize?(0=no more iterations,1=more iterations)
```

one types

```
1
```

to force the primality test to spend more time in optimizing and to display after this extended optimization stage the best solution found. The option of forcing the test to spend more time during the optimization stage may be repeated. Since the optimization might be expensive in this case, this seems only to be beneficial for large values of n. For smaller values of n, the initial solution should be good enough to complete the test using this solution. For instance if one would perform the optimization test on the example given in VII.3, the test finds the additional factor mentioned in VII.(4.1). The time to complete the test using this "improved" final solution is less then the time needed by the solution mentioned in VII.3. This is because of the fact that the optimization consumes in this case too much time, as well as the fact that finding the factor takes more time.

5. RESTARTING OR PARALLELIZING THE PRIMALITY TEST.

(5.1) Restarting the primality test. If a primality test stops at a preliminary stage (for instance, when the computer crashes), it would be beneficial to restart the primality test at a suitable point, and to feed all the information obtained from the previous execution to the test. In this way these do not have to be recalculated.

Especially when executing the primality test for large numbers, this is of vital importance, since one cannot expect that the computer system will not be restarted during a long execution-run of the primality test. The information obtained from the output of the previous execution will be put in the input of the new run.

At the start of the test (after specifying the number n), the program prompts

```
enter start=0/restart=1
```

Contrary to the discussion in VII.3, we will now specify

```
1
```

The program now runs along the same stages as it did the first time. Eventually we could specify all important factors of $n^w - 1$ which were found in the previous run, in the same way as in VII.(4.1). As soon as the generation of the roots of unity is started, the program demands for any known roots of unity.

```
enter v of previously generated root of unity
0 if none
negative to prevent generation
```

Suppose that we already generated a root of unity ζ_{p^k} in the previous run. Then we only have to specify the prime p here. So for instance for an 8^{th} root of unity, we have to specify

```
2
```

If for some reason we want to prevent the generation of a p^k-th root of unity (cf. VII.(5.2)), we have to specify `-p`. If no further root of unity is known, we specify `0`. In case of a positive answer (in our case 2), the program prompts for the root of unity

```
trying to check cyclotomic extension for
prod = 8 which is the product of v^k with
k = 3 v = 2
w = 2
enter gamma
```

By specifying the root of unity, generated and given by the program during the previous run, this root does not have to be generated again. The program will check if the input is

correct.

`gamma is ok`

If the input was not correct, one can give the input again, or one can let the program generate the root of unity for itself. When all roots of unity have been generated (or specified), the Jacobi sum tests have to be performed. Each test can be specified by its index. During a restart run, the user is allowed to specify *which* tests have to be performed. The program asks for the starting index and the last index.

`enter istart`

`enter iend`

If all the tests have to be performed then `istart` $= 1$ and `iend` $= -1$. If all tests with index ≥ 5 have to be performed, then `istart` $= 5$ and `iend` $= -1$. If no tests have to be performed, then `istart` $= -1$ and `iend` $= -1$.

These indices were displayed in previous executions of the program.

For the final trial division the user is allowed to specify *which* residue classes modulo $\text{lcm}(s, v)$ have to be checked for divisors of n. This can be done for both final trial division methods. Again the program asks for the starting index and the last index

`enter istart`

`enter iend`

If all the tests have to be performed then `istart` $= 1$ and `iend` $= \text{lcm}(t, w)$ or larger. If no tests have to be performed, then `istart` should be larger than `iend`.

(5.2) Parallelizing the primality test.

Parallelizing the program, i.e., sharing the various tasks of the program among different processors, can be regarded as restarting the program on all these processors with different inputs. The input will specify which particular task has to be done. All other tasks can be skipped in the same way as has been described in the previous section.

In this way, the generation of roots of unity, performing the Jacobi sum tests and the final trial division can all be performed in parallel.

In the sequential method, the generation of roots of unity is performed in such a way that, during the generation of a root of unity, roots living in sub-extensions can be generated almost for free.

If the generation of each such set of roots is performed on a different processor, the generation of the roots of unity can be completed in the time needed to do the generation of the set of roots in the largest extension ring of $\mathbf{Z}/n\mathbf{Z}$.

Each Jacobi sum test is independent of any other Jacobi sum test. So each Jacobi sum test can be performed by another processor. In this way the Jacobi sum test part of the algorithm can be completed in the time needed to perform the Jacobi sum test in the largest extension of $\mathbf{Z}/n\mathbf{Z}$.

The final trial division can also be shared among several processors, by specifying for each processor, for which set of exponents i in $n^i \bmod \mathrm{lcm}(s,v)$ the processor has to check whether $n^i \bmod \mathrm{lcm}(s,v)$ is an actual divisor of n. For instance each processor can get an equally large range in the range $1,\ldots,\mathrm{lcm}(t,w)-1$. In this way k processors can do the task k times as fast as one processor.

Of course there is a restriction to this approach. The Jacobi sum tests can only be performed when the particular roots of unity used in the test have already been generated. This does not mean however that *all* the roots of unity have to be generated before the *first* Jacobi sum test can be started. After having performed the optimization routine, one can exactly determine which roots are necessary for a particular Jacobi sum test. This is the only restriction that we have to obey. The final trial division for instance can be performed in parallel with the generation of the roots of unity and the performance of the Jacobi sum tests. The same applies to the generation of roots of unity and the performance of Jacobi sum tests that do not depend on these roots.

So when the optimization routine has been performed, one can generate specific inputs for the various tasks. This has to be done very carefully, in order not to get wrong results.

TABLES.

1. *Values of t and $\log_{10}(e(t))$.* 288

In this table, which is generated using the methods described in IV.(2.2), for all 1920 even divisors t of $t_0 = 6983776800 = 2^5 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$ the values of

$$\log_{10}\Big(\prod_{\substack{q \text{ prime} \\ q-1|t}} q\Big) = \sum_{\substack{q \text{ prime} \\ q-1|t}} \log_{10} q$$

and

$$\log_{10}(e(t)) = \log_{10}\Big(2 \cdot \prod_{\substack{q \text{ prime} \\ q-1|t}} q^{o_q(t)+1}\Big)$$

are tabulated. Here $o_q(t)$ denotes the number of factors q in t.

2. *Extensions (minimum polynomials, matrices).* 301

In this table, which is generated using the methods described in IV.(2.3), we tabulate the data necessary to generate Galois extensions of prime power degree $l^e \mid 720$. For each l^e and for all primes $m \leq 160$ we tabulate:

(a) the element g of maximal order modulo m;

(b) the minimum polynomial f of the generator η of the cyclic ring;

(c) the value of D, being the denominator of the matrix S^{-1} (cf. IV.(2.3));

(d) the transition matrix S between the bases consisting of powers of the generator η of the ring (see II.(4.11)) and that consisting of the elements ς_{g,i,l^e} for $i = 0, \ldots, l^e - 1$. The value of the transition matrix S^* is not tabulated here, since the entries of these matrices are very large for some of the larger values of l^e and m.

3. *Gauss sums as products of Jacobi sums.* 312

In this table, which is generated using the methods described in IV.(2.4), we tabulate for each prime power $p^k \mid t_0$ all formulae that are needed to express the quotient of Gauss sums $\tau(\chi)^i/\tau(\chi^i)$ in terms of Jacobi sums, for $i = 1, 2, \ldots, p^k$. Here χ denotes a character of order p^k. These formulae do not depend on the conductor q of χ, except that $q^* = q \cdot \chi(-1)$ occurs for $i = p^k$.

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
2	0.778	1.380	144	9.376	11.836	408	10.197	11.878
4	1.477	2.380	150	6.334	7.414	416	4.431	6.237
6	1.623	2.702	152	1.477	2.681	418	4.762	5.364
8	1.477	2.681	154	2.139	2.741	420	17.649	20.574
10	1.819	2.421	156	9.253	11.748	432	14.050	16.987
12	3.436	4.816	160	5.361	7.866	440	7.442	10.386
14	0.778	1.380	168	6.531	9.058	442	3.424	4.026
16	2.707	4.212	170	1.819	2.421	450	7.613	9.169
18	2.901	4.458	176	6.018	7.523	456	8.455	10.137
20	2.518	4.120	180	12.858	15.415	462	9.110	11.034
22	2.139	2.741	182	0.778	1.380	468	12.100	15.072
24	3.436	5.117	190	4.100	4.702	476	5.317	6.220
26	0.778	1.380	198	8.388	9.944	480	14.966	17.948
28	2.939	3.842	200	6.135	8.737	494	0.778	1.380
30	4.155	5.235	204	5.448	6.829	504	13.346	16.349
32	2.707	4.513	208	4.431	5.936	510	6.168	7.247
34	0.778	1.380	210	9.964	11.889	520	10.689	12.592
36	6.283	8.140	216	10.183	12.819	528	9.803	11.785
38	0.778	1.380	220	3.880	6.523	532	2.939	3.842
40	4.131	6.034	224	6.222	8.029	540	20.062	23.095
42	3.256	5.180	228	5.795	7.176	544	4.844	7.880
44	2.838	3.741	234	4.799	6.355	546	7.892	9.816
48	4.666	6.648	238	3.156	3.758	550	3.181	4.824
50	1.819	2.421	240	12.979	15.660	560	13.177	15.381
52	3.201	4.104	252	11.482	14.185	570	9.193	10.272
54	2.901	4.935	260	6.359	7.962	572	4.563	5.466
56	2.939	4.143	264	8.573	10.254	594	8.388	10.422
60	7.754	9.833	266	0.778	1.380	600	16.329	19.408
66	4.810	5.890	270	7.867	9.901	608	2.707	4.513
68	1.477	2.380	272	4.844	7.579	612	13.570	15.427
70	3.670	4.272	280	9.893	11.796	616	9.040	10.244
72	8.146	10.304	286	2.139	2.741	624	12.979	16.076
76	1.477	2.380	288	11.363	14.124	630	16.147	18.548
78	3.520	4.599	300	11.937	14.715	646	3.588	4.191
80	5.361	7.565	304	2.707	4.212	650	3.936	4.538
84	6.531	8.757	306	7.401	8.958	660	16.282	19.402
88	4.788	5.992	308	4.301	5.204	672	17.157	20.286
90	5.434	6.990	312	11.749	14.544	680	6.267	8.171
96	6.653	8.936	330	9.863	11.984	684	8.642	11.778
100	4.522	6.823	336	12.343	15.170	700	10.682	12.983
102	3.636	4.715	340	2.518	4.120	702	4.799	6.832
104	3.201	4.405	342	2.901	5.736	714	7.647	9.572
108	8.320	10.654	350	3.670	4.272	720	19.947	23.105
110	3.181	4.824	352	8.566	10.372	728	4.663	5.867
112	6.222	7.728	360	16.334	19.192	748	2.838	3.741
114	1.623	2.702	364	4.663	5.566	756	18.977	22.157
120	9.366	11.747	374	2.139	2.741	760	9.293	11.196
126	6.639	9.040	378	9.217	12.096	770	5.032	6.675
130	3.936	4.538	380	4.799	6.401	780	15.689	18.882
132	6.623	8.004	390	8.170	9.250	792	18.181	20.339
136	3.613	4.817	396	14.368	16.225	798	3.256	5.180
140	5.832	7.434	400	9.969	12.872	800	9.969	13.173

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
816	11.427	14.640	1428	14.078	16.303	2288	7.742	9.248
836	5.460	6.364	1430	5.298	6.941	2310	21.701	24.667
840	21.711	24.936	1440	21.934	25.393	2340	27.231	30.902
850	1.819	2.421	1456	7.947	9.452	2376	23.594	26.230
858	9.642	10.721	1482	6.691	7.771	2380	11.587	13.189
864	16.037	19.275	1496	6.924	8.129	2394	6.639	10.319
880	11.617	14.863	1512	20.841	24.321	2400	27.611	31.292
884	5.847	6.750	1520	10.524	12.728	2448	21.412	25.102
900	17.042	20.297	1530	13.119	14.675	2464	14.872	16.678
910	8.747	9.349	1540	7.193	9.837	2470	6.217	6.819
912	9.686	11.668	1560	22.514	26.008	2508	11.605	12.986
918	10.365	12.398	1584	19.411	21.871	2520	36.984	40.687
924	12.385	14.610	1596	12.095	14.320	2550	11.754	12.833
936	19.431	22.703	1632	13.414	16.928	2574	13.220	14.776
950	4.100	4.702	1638	11.274	13.676	2584	6.424	7.628
952	10.433	11.637	1650	12.042	14.163	2600	15.808	18.410
988	3.201	4.104	1672	7.410	8.614	2618	4.518	5.120
990	16.437	19.034	1680	29.904	33.430	2640	29.522	33.245
1008	22.161	25.465	1700	4.522	6.823	2652	17.036	19.530
1020	12.776	14.855	1710	10.472	13.307	2660	8.113	9.715
1026	2.901	6.214	1716	15.375	17.869	2700	24.245	27.978
1040	11.920	14.124	1760	14.165	17.712	2720	10.632	14.367
1050	15.165	17.089	1768	7.984	9.188	2730	23.113	25.037
1056	14.338	16.621	1800	26.552	30.108	2736	14.396	18.134
1064	2.939	4.143	1820	12.633	14.235	2772	22.233	24.936
1080	23.538	26.872	1824	11.673	13.956	2800	24.077	26.980
1092	18.126	21.465	1836	18.571	20.905	2808	21.468	25.218
1100	5.884	9.226	1848	17.125	19.651	2850	14.827	15.906
1120	13.177	15.682	1870	6.453	8.096	2856	25.261	27.788
1122	9.874	10.953	1872	23.934	27.507	2860	11.178	13.821
1140	15.151	17.230	1890	21.159	24.037	2912	7.947	9.753
1144	6.512	7.716	1900	10.082	12.383	2926	8.228	8.830
1170	12.518	14.074	1904	13.717	16.452	2964	14.784	17.279
1188	16.405	18.740	1938	6.446	7.526	2970	22.343	25.417
1190	6.049	6.651	1950	13.640	14.719	2992	8.155	10.890
1200	25.624	29.004	1976	3.201	4.405	3024	32.292	36.074
1224	20.182	22.340	1980	29.280	32.878	3040	14.007	16.512
1232	12.324	13.829	2002	5.441	6.043	3060	29.826	32.382
1248	18.063	21.460	2016	30.280	33.886	3080	15.994	18.939
1254	7.433	8.512	2040	19.137	21.517	3094	5.802	6.404
1260	27.658	31.059	2052	13.992	17.605	3120	29.621	33.416
1292	4.287	5.191	2080	15.238	17.743	3150	21.348	23.749
1300	11.478	13.779	2090	8.084	9.727	3168	27.447	30.207
1320	22.965	26.386	2100	27.700	31.323	3192	14.755	17.281
1326	11.302	12.382	2128	9.551	11.056	3230	6.911	7.513
1330	5.951	6.553	2142	16.848	19.249	3276	23.077	26.893
1350	10.046	12.080	2160	33.122	36.757	3300	23.983	27.803
1360	10.632	14.066	2184	20.621	24.262	3344	8.640	10.145
1368	13.166	16.603	2200	9.446	13.090	3360	38.245	42.073
1386	14.791	17.192	2210	6.583	7.185	3366	15.938	17.495
1400	14.743	17.345	2244	11.687	13.067	3400	8.272	10.874
1404	14.138	17.586	2280	25.663	28.044	3420	20.256	24.091

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
3432	23.356	26.151	5016	16.215	17.896	7072	9.214	12.251
3458	0.778	1.380	5040	48.181	52.185	7106	7.572	8.174
3510	18.496	20.529	5100	24.073	26.851	7140	35.134	38.058
3536	9.214	11.950	5130	12.905	16.217	7150	9.152	10.796
3570	17.908	19.833	5148	23.120	26.091	7182	9.217	13.375
3600	35.847	39.705	5168	7.655	10.390	7200	37.834	41.992
3640	19.411	21.314	5200	19.641	22.544	7280	22.695	24.899
3672	28.748	31.383	5236	10.398	11.301	7344	32.614	36.781
3696	26.504	29.331	5280	37.779	41.803	7392	37.735	40.863
3740	7.152	9.795	5304	24.279	27.075	7410	20.249	21.328
3744	29.017	32.892	5320	15.056	16.959	7480	16.725	19.669
3762	11.010	13.845	5400	33.756	37.789	7524	19.350	22.486
3780	40.319	44.198	5434	4.762	5.364	7560	53.524	57.704
3800	14.576	17.179	5460	37.757	41.795	7600	18.410	21.313
3808	13.717	16.753	5472	16.383	20.422	7650	18.705	20.261
3850	8.618	10.261	5544	28.836	31.840	7700	15.629	18.971
3876	14.208	15.588	5600	24.077	27.281	7722	17.107	19.141
3900	26.277	30.169	5610	18.198	20.319	7752	25.505	27.187
3952	4.431	5.936	5616	28.608	32.658	7800	35.881	40.074
3960	37.826	41.725	5700	29.824	32.603	7854	16.551	18.475
3978	15.068	16.625	5712	31.072	35.130	7904	4.431	6.237
3990	15.002	16.926	5720	17.457	20.401	7920	44.384	48.583
4004	9.327	10.230	5814	10.212	13.047	7956	25.157	28.128
4080	25.883	29.795	5850	21.754	23.310	7980	28.250	31.174
4104	18.515	22.430	5852	10.389	11.292	8008	17.970	19.174
4158	20.989	23.867	5928	19.940	22.735	8160	31.782	35.994
4180	8.783	11.426	5940	39.956	44.031	8190	36.278	38.679
4200	38.164	42.088	5950	6.049	6.651	8208	26.297	30.512
4256	9.551	11.357	5984	10.703	13.739	8316	37.267	40.447
4284	27.634	30.337	6006	23.759	25.684	8360	15.226	18.171
4290	16.812	18.933	6048	40.412	44.494	8398	6.235	6.837
4320	35.108	39.045	6120	41.837	44.694	8400	55.487	59.712
4368	26.433	30.374	6160	22.223	25.469	8550	16.106	18.941
4400	16.225	20.169	6188	9.688	10.591	8568	40.681	43.684
4420	12.651	14.253	6240	38.022	42.118	8580	34.541	38.775
4446	11.618	14.453	6270	21.320	23.441	8736	38.285	42.528
4488	18.384	20.066	6300	41.508	45.608	8778	19.142	21.066
4522	9.622	10.224	6384	23.894	26.721	8800	18.773	23.018
4550	8.747	9.349	6426	26.198	29.077	8840	19.118	21.021
4560	32.935	35.616	6460	7.610	9.212	8892	25.228	29.478
4576	10.290	12.096	6552	34.224	38.341	8976	19.615	22.828
4590	22.177	24.211	6600	33.445	37.566	9044	11.784	12.687
4620	35.871	39.837	6630	15.952	17.032	9100	20.597	22.898
4680	38.892	42.863	6650	5.951	6.553	9120	38.405	41.387
4752	27.461	30.398	6688	15.013	16.820	9180	47.617	50.651
4760	20.764	22.667	6732	28.534	30.391	9240	51.759	56.026
4788	20.726	24.707	6800	15.239	19.373	9282	22.020	23.944
4862	4.786	5.388	6840	36.466	40.603	9350	6.453	8.096
4896	23.399	27.390	6864	24.586	27.682	9360	49.271	53.543
4914	13.853	16.731	6916	8.503	9.406	9450	26.359	29.238
4940	8.641	10.243	6930	33.179	36.622	9504	35.497	38.734
4950	22.310	24.908	7020	37.980	42.128	9520	31.160	34.595

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
9576	25.249	29.531	13090	10.682	12.326	17290	15.266	15.868
9690	14.017	15.096	13104	46.311	50.730	17442	17.417	20.729
9724	7.209	8.112	13200	45.686	50.107	17550	31.977	34.010
9828	34.564	38.858	13260	30.125	33.319	17556	27.980	30.205
9880	15.852	17.755	13300	16.242	18.543	17680	27.730	31.164
9900	44.672	48.969	13338	15.743	19.056	17784	35.219	39.770
10010	13.410	15.054	13464	37.095	39.254	17850	30.767	32.692
10032	17.445	19.427	13566	18.246	20.170	17952	24.149	27.663
10080	59.827	64.132	13600	15.239	19.674	18018	29.441	31.842
10098	22.906	24.939	13650	31.604	33.528	18088	21.157	22.361
10200	33.213	36.292	13680	47.874	52.311	18200	27.375	29.977
10260	30.772	35.084	13728	36.355	39.752	18360	63.194	66.528
10296	35.935	39.208	13832	8.503	9.707	18480	70.732	75.300
10336	11.669	14.706	13860	53.773	58.216	18564	35.409	38.748
10374	11.063	12.987	14040	49.641	54.089	18700	13.428	16.770
10400	22.960	26.164	14212	8.271	9.174	18720	57.672	62.246
10450	8.084	9.727	14280	54.534	57.759	18810	27.894	31.770
10472	20.254	21.458	14300	20.151	23.493	18900	54.169	58.747
10608	25.510	29.837	14364	31.533	35.991	19040	31.160	34.896
10640	21.667	23.871	14560	30.176	32.681	19152	37.392	41.975
10710	37.124	39.525	14586	20.475	21.554	19380	30.860	32.939
10800	49.022	53.356	14630	13.402	15.045	19448	11.295	12.499
10868	7.185	8.088	14688	34.601	39.069	19656	45.711	50.306
10920	47.031	51.370	14820	34.298	37.491	19760	17.082	19.286
11050	6.583	7.185	14850	32.388	35.463	19800	63.550	68.147
11088	41.219	44.524	14960	24.034	28.510	19890	30.271	31.827
11220	27.626	30.746	15048	25.823	29.260	19950	23.658	25.582
11232	33.691	38.043	15120	74.872	79.352	20020	25.054	27.698
11286	15.063	18.375	15200	21.893	25.097	20064	25.805	28.088
11400	43.115	46.195	15300	41.123	44.379	20196	37.539	39.873
11424	35.887	40.246	15400	28.618	32.261	20400	45.642	50.253
11440	21.632	24.878	15444	29.045	32.493	20520	51.295	55.908
11550	34.550	37.516	15470	13.772	14.374	20592	44.752	48.325
11628	22.329	25.465	15504	26.736	29.949	20748	35.017	38.356
11700	45.655	50.024	15600	52.863	57.357	20790	45.282	49.202
11704	15.129	16.333	15708	26.701	28.926	20900	14.066	17.409
11856	21.170	24.266	15840	56.142	60.643	20944	23.537	26.273
11880	51.879	56.255	15912	41.438	44.710	21216	30.593	35.221
11900	16.437	18.738	15960	41.211	44.436	21280	25.150	27.655
11934	18.032	20.065	16016	21.253	22.759	21318	19.636	20.715
11970	25.263	28.943	16150	6.911	7.513	21420	64.449	67.850
12012	33.994	37.333	16302	15.435	16.515	21450	26.136	28.256
12240	52.671	57.060	16380	62.331	66.846	21600	55.343	59.979
12320	24.771	28.317	16416	32.499	37.015	21736	13.471	14.675
12350	6.217	6.819	16632	51.467	54.948	21840	63.057	67.698
12376	18.896	20.101	16720	19.402	22.647	22100	17.770	20.071
12540	34.197	37.317	16796	8.658	9.561	22176	59.256	62.862
12600	64.592	68.994	16800	63.828	68.354	22230	28.244	31.079
12768	28.709	31.837	16830	34.670	37.268	22440	47.282	50.704
12852	46.010	49.189	17100	34.929	39.463	22572	33.106	36.719
12870	26.454	29.052	17136	53.730	58.265	22610	14.796	15.398
12920	14.241	16.144	17160	49.972	54.507	22800	56.070	59.450

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
22880	27.498	31.045	30030	44.865	47.830	38896	12.526	15.261
22950	27.763	29.796	30096	31.532	35.270	39270	35.968	38.934
23100	57.089	61.753	30240	90.998	95.780	39312	65.030	69.926
23256	35.490	38.927	30600	59.169	62.725	39520	28.480	30.985
23400	63.349	68.019	30800	40.897	44.841	39600	75.790	80.688
23408	21.741	23.246	30888	45.237	48.986	39780	57.912	61.582
23562	32.423	34.824	30940	29.170	30.772	39900	53.392	57.015
23712	26.254	29.651	31008	32.737	36.251	40040	40.476	43.420
23760	68.783	73.460	31122	22.587	26.267	40392	53.041	55.677
23800	29.991	32.593	31200	61.265	66.060	40698	32.057	35.737
23868	34.536	37.984	31350	26.954	29.075	40800	56.152	61.063
23940	46.017	50.697	31416	42.624	45.150	40950	48.536	50.937
24024	48.668	52.308	31824	45.941	50.745	41040	72.588	77.502
24310	11.216	12.860	31920	56.391	59.918	41184	60.022	63.896
24480	62.958	67.648	32032	23.801	25.607	41496	40.173	43.813
24570	49.225	52.104	32130	52.569	55.447	41580	77.447	82.367
24700	17.038	19.339	32300	12.893	15.194	41800	25.131	28.774
24752	22.180	24.916	32604	23.528	26.023	41888	26.085	29.122
25080	49.779	53.201	32760	83.658	88.474	41990	11.675	12.277
25194	17.284	18.364	33150	29.349	30.428	42636	27.397	28.777
25200	84.919	89.622	33264	66.487	70.268	42840	97.532	101.235
25650	18.539	21.851	33440	29.258	32.805	42900	57.134	62.067
25704	62.621	66.102	33592	10.795	11.999	43472	14.702	16.207
25740	58.388	63.099	33660	60.624	64.222	43680	85.918	90.859
25840	23.017	26.452	34034	10.466	11.068	43758	31.180	32.737
26180	19.940	22.583	34200	57.174	62.009	43890	45.211	48.177
26208	65.887	70.607	34272	66.384	71.220	44200	28.882	31.484
26334	24.823	28.503	34320	60.024	64.860	44460	53.438	58.387
26400	53.943	58.665	34580	22.991	24.593	44880	56.974	61.927
26520	41.699	45.193	34650	54.262	57.705	45144	42.955	46.869
26600	23.185	25.787	34884	34.884	38.497	45220	20.334	21.936
26676	34.703	39.431	35100	60.648	65.494	45600	61.539	65.221
26928	38.326	42.015	35112	35.380	37.906	45900	58.915	62.647
27132	33.828	36.053	35360	31.048	34.783	46200	83.566	88.532
27170	10.201	11.845	35530	18.718	20.361	46410	45.461	47.385
27300	54.212	58.949	35568	44.273	49.125	46512	36.721	41.689
27360	57.781	62.519	35700	56.551	60.174	46800	83.604	88.576
27664	15.115	16.620	35910	34.830	38.987	46816	32.784	34.590
27720	74.926	79.670	36036	48.399	52.215	47124	53.355	56.057
27846	35.665	38.067	36176	27.769	30.504	47520	85.218	90.196
28050	28.232	30.353	36400	36.709	39.612	47600	46.437	50.571
28080	70.439	75.189	36720	84.564	89.430	47736	59.060	62.810
28424	12.358	13.562	36960	89.212	94.081	47880	72.758	77.739
28560	69.840	74.596	37050	29.173	30.252	48048	62.729	66.670
28600	26.430	30.073	37128	53.181	56.821	48450	23.058	24.137
28728	40.515	45.274	37400	23.001	26.644	48620	20.742	23.385
29070	20.968	23.803	37620	47.195	52.072	48906	27.350	30.186
29172	30.673	33.167	37800	81.132	86.011	49140	86.921	91.913
29260	15.563	18.206	38038	16.110	16.712	49400	24.249	26.851
29640	54.495	57.989	38304	45.512	50.396	49504	22.180	25.217
29700	59.520	64.294	38610	44.380	47.455	49742	17.073	17.675
29920	31.058	35.835	38760	50.010	52.390	50050	25.550	27.193

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
50160	59.996	63.719	64600	24.334	26.936	82992	49.312	53.253
50388	28.966	31.460	65208	38.506	41.301	83160	110.075	115.296
50400	96.565	101.569	65450	14.268	15.911	83538	45.015	47.894
50490	51.206	54.280	65520	110.777	115.895	83600	31.910	35.854
51300	45.445	50.456	66300	57.170	61.062	83980	17.743	19.345
51408	78.307	83.319	66528	89.347	93.429	84150	48.399	50.996
51480	83.365	88.378	66690	38.348	41.660	85272	40.644	42.325
51680	30.515	34.250	67184	12.025	14.761	85680	124.164	129.397
51870	44.145	46.069	67320	85.931	89.829	85800	75.344	80.578
52360	42.450	45.394	67830	33.545	35.469	86450	15.266	15.868
52668	41.509	45.490	68068	18.071	18.974	86944	21.075	22.881
53040	56.187	61.213	68400	74.264	79.400	87210	39.208	42.520
53200	40.573	43.476	68640	78.833	83.971	87516	57.103	60.074
53352	49.421	54.449	69160	32.651	34.554	87780	69.042	73.008
53550	54.712	57.113	69300	91.020	96.162	88400	40.097	44.230
53856	51.092	55.083	69768	51.610	55.524	88920	82.305	87.555
54054	39.526	42.405	70200	83.189	88.336	89760	73.619	78.873
54264	55.818	58.345	70224	48.087	50.914	90090	63.324	66.767
54340	16.081	18.724	70686	54.246	57.124	90288	60.170	64.385
54600	74.625	79.664	71060	19.417	22.060	90440	36.650	38.553
55328	15.115	16.921	71136	49.356	54.509	91800	85.489	89.522
55440	101.647	106.691	71400	86.729	90.654	92378	10.219	10.821
55692	53.410	57.227	71820	71.402	76.559	92400	116.635	121.901
56100	55.911	59.730	72072	76.582	80.700	92820	82.749	86.787
56160	78.841	83.892	72352	36.642	39.679	93024	42.722	47.992
56430	42.604	46.957	72800	44.191	47.395	93366	29.290	33.448
56848	13.588	16.324	72930	35.779	37.900	93600	96.977	102.250
57120	82.093	87.150	73150	16.987	18.631	94050	37.223	41.099
57200	33.208	37.153	73440	94.851	100.018	94248	71.141	74.145
57456	63.968	69.028	74100	60.246	64.138	95200	46.437	50.872
58140	47.910	51.745	74256	63.863	69.034	95472	66.200	71.481
58344	43.402	46.197	74800	32.913	38.088	95760	95.079	100.361
58520	27.246	30.190	75240	68.476	73.654	96096	90.118	94.361
58786	17.038	17.640	75582	29.577	32.412	96900	52.647	55.426
59280	70.033	73.829	75600	111.610	116.790	97240	38.019	40.964
59400	81.774	86.849	76076	23.835	24.738	97812	48.550	52.800
59670	47.650	49.683	77220	81.084	86.273	98280	112.126	117.420
59850	33.918	37.599	77350	18.660	19.262	98800	33.077	35.981
60060	77.685	82.764	77520	69.717	73.629	99450	47.434	48.990
60192	43.392	47.432	77792	15.073	18.110	99484	22.953	23.856
61200	75.686	80.774	78540	68.293	72.258	100100	45.158	48.500
61600	43.444	47.690	78624	84.606	89.802	100320	75.562	79.585
61776	56.690	60.740	79002	35.073	39.231	100776	42.759	45.554
61880	45.157	47.060	79200	92.447	97.646	100980	90.897	94.972
62244	55.738	60.833	79560	87.210	91.181	102102	45.947	47.871
62700	57.186	61.006	79800	77.657	81.581	102600	72.002	77.314
62832	52.003	56.061	80080	46.704	49.950	102816	90.961	96.274
63648	55.828	60.933	80784	56.908	61.075	102960	101.003	106.317
63840	73.021	76.848	81396	55.674	59.656	103740	76.679	80.717
63954	25.700	28.535	81510	35.310	37.431	103950	75.554	79.474
64260	95.615	99.494	81900	95.334	100.549	104720	55.791	60.267
64350	43.240	45.837	82080	86.710	91.925	105336	55.794	60.076

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
106080	68.500	73.827	135660	69.342	72.266	172900	34.235	36.536
106400	44.056	47.260	135850	19.189	20.832	174420	78.196	82.508
106590	46.374	48.494	136136	35.922	37.126	175032	78.869	82.141
106704	65.026	70.355	136800	84.171	89.608	175560	93.829	98.096
107100	99.424	103.524	138320	39.263	41.467	176358	40.559	42.483
108108	76.347	80.641	138600	134.416	139.858	176800	43.415	47.850
108528	69.993	74.051	139230	80.402	82.803	177650	18.718	20.361
108680	29.579	32.523	139536	64.536	69.982	177840	115.053	120.604
109200	109.013	114.353	140400	119.011	124.459	179550	43.485	47.643
110880	131.978	137.324	140448	72.962	76.090	180180	124.376	129.932
111150	40.936	43.771	141372	88.123	91.303	180576	76.246	80.762
111384	84.035	88.153	142120	31.871	34.815	180880	54.787	58.221
112200	78.346	82.467	142800	111.165	116.621	183600	112.541	118.106
112860	74.342	79.695	143640	110.792	116.251	184756	12.642	13.545
113050	19.849	20.451	144144	101.233	105.652	184800	135.115	140.683
113696	23.975	27.012	145350	30.008	32.843	185640	116.722	121.062
114114	41.543	43.467	145860	69.792	74.026	186732	79.935	85.507
114400	39.074	43.320	146300	27.278	30.620	188100	77.875	83.450
114912	81.363	86.725	146718	44.582	47.895	188496	87.758	92.293
116280	76.545	80.681	148200	88.392	92.585	190190	30.598	32.242
116688	49.699	54.026	148512	80.887	86.360	190944	76.087	81.669
117040	41.871	45.116	149226	41.511	43.435	191520	119.450	125.033
117572	24.764	25.667	149600	39.937	45.413	191862	40.962	44.274
117810	74.148	77.591	150150	74.734	77.700	193050	74.867	77.942
118560	86.515	90.611	150480	87.307	92.786	193800	79.387	82.466
118800	109.436	114.812	151164	49.563	53.813	194480	49.576	54.052
119340	88.403	92.550	151200	137.251	142.731	195624	68.363	72.914
119700	80.036	85.415	152152	36.816	38.020	196350	60.923	63.889
120120	111.304	116.684	154440	109.438	114.928	196560	163.733	169.327
121550	15.071	16.714	154700	42.023	44.324	197600	44.476	47.680
122094	45.648	49.805	155040	83.113	87.326	198900	98.090	102.460
122400	95.672	101.060	155610	69.528	73.209	198968	37.066	38.270
122850	65.727	68.606	157080	117.659	121.926	200200	70.068	73.712
123552	77.051	81.403	158004	68.261	72.720	201552	43.990	48.316
123760	59.801	63.235	159120	109.059	114.561	201960	128.450	132.826
124488	74.640	80.037	159600	106.693	110.918	203490	61.448	65.128
125400	80.169	84.289	160160	56.733	60.280	204204	67.520	70.860
125664	63.234	67.593	160650	75.362	78.241	205200	104.290	109.903
125970	30.842	31.921	161568	74.883	79.351	205920	123.314	128.929
127908	44.244	47.380	162792	79.528	83.810	207480	104.641	108.981
128520	141.251	145.431	163020	68.880	73.115	207900	123.882	129.501
128700	96.507	101.917	163800	135.157	140.672	209440	68.135	72.912
129200	35.714	39.848	165984	61.165	65.407	209950	11.675	12.277
130416	39.737	42.833	166320	151.322	156.844	210672	75.984	80.567
130900	32.647	35.990	167076	85.379	89.673	213180	75.464	78.584
131040	146.479	151.897	167200	41.766	46.011	213408	74.325	79.955
131274	42.036	44.069	167960	27.091	28.994	214200	150.642	155.043
131670	65.886	70.608	168300	96.599	100.896	216216	117.463	122.057
132600	76.168	80.361	170170	21.707	23.351	217056	89.019	93.377
133380	71.624	77.051	170544	41.874	45.087	217360	39.091	42.337
134368	21.168	24.205	171360	148.645	154.180	218400	137.213	142.854
134640	99.710	105.140	171600	95.272	100.807	218790	64.260	66.857

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
219450	66.856	69.822	284240	43.592	48.068	369512	21.065	22.269
222300	87.221	92.869	285600	128.029	133.786	371280	154.550	160.420
222768	105.227	110.876	287280	157.396	163.155	373464	108.022	113.896
224400	99.071	104.723	288288	139.847	144.567	376200	114.108	119.984
225720	108.665	114.319	290700	75.161	79.695	376992	115.062	119.897
226100	33.516	35.817	291720	108.649	113.185	377910	59.264	62.099
226746	40.907	44.219	292600	53.235	56.879	380380	46.082	48.725
228228	65.497	68.836	293436	75.485	80.212	383724	69.209	72.823
232050	76.385	78.309	293930	31.526	32.128	386100	132.905	138.793
232560	104.476	110.144	296400	118.801	123.295	387600	104.776	109.387
233376	61.468	66.096	298350	69.058	71.091	388960	65.508	70.285
234080	56.397	59.944	298452	60.812	63.037	391248	91.801	96.653
235144	38.229	39.434	300300	129.147	134.926	392700	114.345	119.010
235620	128.371	132.814	300960	116.290	122.069	393120	209.509	215.405
237600	135.104	140.781	302328	77.874	82.425	395010	91.349	96.548
238680	131.322	135.771	304304	43.428	44.933	397800	138.068	142.738
239400	125.437	131.117	306306	70.905	73.306	397936	43.678	46.413
240240	142.792	148.473	308880	141.871	147.662	400400	82.347	86.292
243100	39.372	42.715	309400	67.032	69.634	403104	58.216	62.843
244188	81.604	86.062	311220	121.102	126.896	403920	157.141	163.048
244530	53.290	57.166	314160	149.241	155.040	406980	116.634	121.315
245700	124.168	129.860	316008	94.602	99.362	407550	53.222	55.342
247520	67.282	71.018	318240	130.564	136.368	408408	97.471	101.111
248710	30.069	31.712	319200	128.827	133.353	410400	128.360	134.274
248976	94.607	100.304	319770	62.845	66.722	414960	132.428	137.069
250800	96.068	100.489	321300	141.303	145.880	415800	184.372	190.292
251940	64.823	68.016	325584	100.941	106.754	417690	114.179	117.057
252450	69.106	72.181	326040	102.020	106.555	419900	26.141	28.442
255816	59.355	62.792	327600	180.638	186.454	421344	107.664	112.548
257040	182.598	188.309	332640	201.155	206.978	426360	107.909	111.331
257400	137.226	142.938	333450	60.807	64.119	428400	192.035	197.968
258400	43.211	47.646	334152	124.248	128.842	432432	154.980	159.876
259350	56.090	58.015	335920	40.115	43.550	434720	56.862	60.409
260832	55.331	58.728	336600	132.237	136.835	436050	48.249	51.561
261800	63.721	67.365	339150	60.443	62.367	437580	127.520	132.232
262548	74.374	77.822	340340	48.583	51.226	438900	115.489	120.154
263340	99.822	105.544	341088	54.249	57.763	444600	127.293	133.242
265200	100.532	106.256	342342	65.589	69.269	445536	144.963	150.913
266760	109.530	115.258	343200	114.081	119.918	447678	61.992	65.672
269280	134.406	140.137	345800	43.894	46.496	448800	125.979	131.932
270270	97.270	101.189	348840	114.709	119.322	450450	110.849	114.291
271320	105.788	109.013	350064	92.753	97.557	451440	152.367	158.323
271700	33.466	36.808	351120	130.403	134.971	452200	64.675	67.277
272272	39.206	41.941	352716	71.257	74.596	453492	70.621	75.348
276640	54.824	57.329	353430	105.538	109.458	456456	92.828	96.468
277134	33.408	34.487	355300	28.972	32.314	461890	29.146	30.789
277200	175.232	180.975	355680	135.972	141.824	464100	130.166	134.903
278460	135.373	139.888	359100	110.976	116.833	465120	126.698	132.667
279072	80.198	85.945	360360	179.617	185.475	466830	91.156	95.313
280800	136.718	142.468	361760	67.144	70.879	470288	44.841	47.577
282150	56.104	60.458	364650	57.478	59.599	471240	197.094	201.838
282744	117.072	120.552	367200	142.427	148.293	477360	173.834	179.814

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
478800	167.294	173.275	628320	181.431	187.530	831402	57.329	60.164
480480	184.912	190.895	629850	47.693	48.773	831600	244.789	251.010
486200	61.295	64.939	632016	131.058	136.118	835380	198.465	203.457
488376	113.481	118.240	639540	108.462	113.338	839800	50.869	53.471
489060	115.693	121.684	642600	210.036	214.915	850850	38.735	40.379
491400	172.715	178.708	646646	32.370	32.972	852720	130.561	135.514
497420	39.326	41.970	651168	133.619	139.734	856800	226.215	232.449
497952	114.183	120.181	652080	131.655	136.492	864864	203.509	208.706
501600	117.334	122.056	655200	226.649	232.767	872100	105.447	110.458
503880	93.657	97.151	656370	103.408	106.483	875160	188.813	193.826
504900	136.747	141.521	658350	101.584	106.306	877800	165.856	170.821
510510	83.407	86.372	666900	120.999	127.124	881790	81.860	83.785
511632	70.773	75.741	668304	152.671	158.797	889200	174.912	181.162
514080	217.271	223.283	671840	60.656	64.392	895356	101.708	105.689
514800	164.740	170.753	673200	162.878	169.007	900900	210.270	216.526
518700	113.095	117.832	678300	116.433	120.057	902880	190.242	196.498
523600	83.113	88.288	680680	92.627	95.571	904400	93.588	97.721
525096	107.760	111.509	684684	110.887	115.982	906984	111.902	116.931
526680	149.134	155.156	691600	66.277	69.180	912912	110.217	114.158
529074	71.834	75.514	696150	119.821	122.223	918918	96.615	99.494
530400	123.180	129.206	697680	168.079	174.223	923780	38.671	41.314
532950	65.589	67.709	700128	123.403	128.508	928200	184.301	189.339
533520	156.613	162.641	702240	170.815	175.684	933660	167.812	174.083
540540	184.650	190.684	705432	99.835	103.476	940576	58.843	61.880
542640	142.418	147.175	706860	184.407	189.327	942480	244.746	251.021
543400	51.585	55.228	710600	50.857	54.501	950950	47.871	49.514
544544	41.753	44.790	718200	169.027	175.184	954720	195.340	201.621
554268	55.298	57.792	720720	231.256	237.414	957600	203.150	209.432
554400	210.461	216.506	729300	133.559	138.492	959310	97.367	101.721
556920	198.920	203.737	733590	95.177	99.530	972400	75.455	80.630
564300	114.945	120.997	739024	22.296	25.031	976752	151.348	157.639
565488	142.077	147.089	742560	186.494	192.666	978120	162.215	168.506
568480	67.694	72.471	746130	83.983	86.949	982800	253.823	260.117
570570	88.948	91.914	746928	143.894	150.068	994840	68.975	71.919
574560	195.523	201.583	752400	138.622	144.799	1007760	125.879	130.905
581400	114.641	119.476	755820	110.663	115.611	1009800	189.594	194.669
583440	131.149	137.216	760760	68.722	71.666	1017450	93.075	96.755
585200	78.637	82.581	767448	91.261	95.175	1021020	157.114	162.194
586872	103.401	108.429	772200	181.847	188.036	1023264	91.379	96.649
587860	50.765	52.367	775200	122.783	127.695	1027026	95.030	99.188
589050	112.066	115.509	778050	91.132	94.813	1029600	202.933	209.247
592800	135.283	140.078	782496	116.790	121.943	1037400	168.286	173.324
596700	132.825	137.672	785400	178.678	183.643	1047200	95.458	100.933
596904	87.542	90.068	790020	155.275	161.473	1050192	124.280	129.561
600600	189.173	195.253	795600	175.693	181.895	1053360	208.093	214.416
604656	86.928	93.010	795872	63.595	66.632	1058148	120.541	125.636
608608	60.255	62.062	800800	98.280	102.526	1065900	130.791	134.610
610470	90.630	94.788	807840	201.722	207.931	1067040	181.747	188.076
612612	118.547	122.363	813960	181.190	186.171	1081080	256.702	263.037
617760	179.741	185.833	815100	116.764	121.697	1085280	177.170	182.228
618800	87.726	91.859	816816	127.381	132.553	1086800	68.695	72.640
622440	174.728	180.823	829920	168.174	173.115	1093950	99.459	102.057

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
1108536	84.958	87.753	1493856	172.745	179.221	2042040	249.859	255.239
1113840	256.162	262.510	1504800	184.381	190.859	2054052	171.128	176.700
1128600	170.272	176.626	1511640	166.537	171.787	2074800	230.471	235.811
1130976	179.412	184.725	1521520	88.684	91.930	2088450	169.369	172.247
1133730	96.005	99.317	1531530	141.953	145.396	2100384	165.230	170.812
1141140	148.970	154.049	1534896	119.330	124.776	2106720	270.271	276.895
1162800	148.255	154.621	1544400	234.378	240.868	2116296	172.548	177.944
1166880	163.936	170.305	1556100	185.035	191.528	2131800	175.446	179.567
1170400	93.163	97.409	1570800	229.707	236.204	2162160	342.839	349.475
1173744	138.346	143.676	1580040	236.385	242.884	2173600	92.804	97.049
1175720	73.890	75.793	1587222	89.551	93.708	2187900	224.491	229.901
1178100	206.283	211.424	1591200	217.593	224.096	2217072	97.601	101.928
1193400	196.234	201.381	1598850	85.755	89.632	2227680	320.323	326.972
1193808	105.285	109.342	1627920	233.282	239.795	2238390	142.320	147.042
1201200	250.068	256.448	1630200	162.474	167.709	2257200	230.045	236.699
1209312	105.958	112.341	1633632	166.156	171.629	2267460	163.827	169.254
1220940	169.707	174.864	1662804	103.884	108.134	2282280	217.632	223.013
1222650	84.751	88.627	1663200	309.036	315.558	2309450	38.133	39.777
1225224	166.208	170.326	1670760	290.845	296.138	2325600	180.175	186.843
1237600	95.207	99.642	1679600	71.491	75.624	2334150	122.528	126.685
1243550	38.708	40.351	1701700	83.233	86.575	2347488	172.642	178.272
1244880	227.545	233.941	1705440	164.284	169.538	2351440	102.646	106.080
1259700	113.327	117.219	1711710	134.970	139.692	2356200	301.625	307.067
1264032	182.938	188.300	1744200	157.767	163.079	2386800	259.670	266.349
1279080	156.500	161.677	1750320	222.988	229.532	2387616	144.370	148.728
1285200	266.145	272.555	1755600	221.251	226.518	2402400	309.811	316.493
1293292	43.815	44.718	1763580	155.449	159.487	2441880	255.586	261.044
1304160	162.370	167.508	1767150	157.851	161.771	2445300	191.578	198.268
1312740	184.784	189.973	1778400	200.802	207.353	2450448	210.943	216.592
1316700	173.195	179.616	1790712	141.577	145.859	2487100	60.366	63.708
1333800	174.956	181.383	1801800	308.982	315.538	2489760	286.964	293.661
1336608	192.406	198.833	1808800	112.202	116.636	2494206	85.770	89.083
1343034	92.109	96.266	1813968	138.911	145.470	2519400	165.491	169.684
1346400	217.824	224.254	1825824	157.034	161.277	2552550	140.157	143.122
1351350	163.513	167.433	1837836	182.094	186.387	2558160	197.563	204.272
1356600	179.026	182.950	1847560	63.167	66.112	2570400	332.006	338.717
1361360	110.215	114.691	1856400	240.490	247.059	2586584	70.261	71.465
1369368	167.981	173.378	1867320	245.085	251.657	2625480	274.799	280.289
1383200	81.839	85.043	1884960	307.871	314.448	2633400	259.739	266.460
1385670	76.660	78.781	1889550	86.159	88.994	2645370	150.508	154.189
1392300	214.395	219.610	1901900	74.597	77.940	2667600	253.795	260.522
1395360	199.961	206.407	1918620	164.387	169.741	2686068	152.436	156.895
1410864	118.881	124.053	1939938	89.715	91.639	2702700	295.695	302.428
1413720	278.585	283.806	1944800	91.387	96.863	2713200	229.512	234.968
1421200	65.182	70.357	1953504	198.748	205.340	2722720	135.631	140.408
1436400	240.478	246.937	1956240	223.444	230.037	2738736	210.582	216.280
1441440	294.762	301.222	1965600	319.424	326.019	2771340	157.306	161.540
1458600	186.004	191.239	1975050	136.235	141.434	2784600	305.460	310.976
1467180	170.017	176.484	1989680	95.125	99.601	2821728	155.244	160.717
1469650	41.468	42.070	2015520	155.415	160.742	2827440	351.081	357.833
1478048	37.812	40.848	2019600	240.222	246.828	2842400	89.283	94.759
1492260	150.481	154.446	2034900	187.825	193.204	2852850	132.747	135.712

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
2872800	305.218	311.978	4176900	298.765	304.456	6395400	244.743	250.619
2917200	223.731	230.497	4232592	213.282	220.210	6466460	97.571	100.214
2934360	240.775	247.543	4263600	209.132	214.784	6520800	249.732	255.569
2939300	71.950	74.251	4324320	448.962	455.899	6563700	301.159	307.047
2984520	223.368	227.635	4375800	312.335	318.047	6651216	195.445	201.527
3023280	220.056	226.838	4408950	133.467	135.391	6683040	462.765	469.891
3043040	121.073	124.620	4434144	122.338	126.966	6715170	203.079	208.278
3052350	133.947	138.105	4476780	252.609	258.330	6806800	176.259	181.434
3063060	270.095	275.652	4514400	294.643	301.599	6846840	369.144	376.280
3069792	154.806	160.553	4534920	242.363	248.090	6928350	112.674	114.794
3088800	298.955	305.746	4564560	282.645	288.327	6976800	279.226	286.371
3112200	273.245	280.039	4594590	211.399	215.319	7054320	286.554	292.424
3141600	278.656	285.454	4618900	65.714	69.056	7068600	426.151	432.071
3160080	334.612	341.413	4668300	252.892	259.862	7162848	244.571	250.685
3174444	169.460	175.032	4702880	132.209	135.945	7207200	487.039	494.198
3197700	176.942	182.518	4712400	380.183	387.158	7335900	276.702	283.869
3233230	66.855	68.499	4773600	311.470	318.449	7351344	314.285	320.411
3255840	290.513	297.327	4796550	124.449	128.802	7390240	127.208	131.985
3260400	206.980	212.515	4883760	347.513	354.503	7461300	250.672	255.337
3281850	152.309	155.384	4890600	263.634	270.624	7469280	415.291	422.466
3325608	148.061	152.612	4900896	286.506	292.456	7558200	273.077	279.026
3341520	382.818	389.642	4974200	119.132	122.775	7607600	152.547	156.492
3359200	92.031	96.466	4988412	146.406	151.133	7657650	227.126	230.569
3403400	145.787	149.431	5038800	219.286	225.010	7674480	329.294	336.481
3423420	241.532	248.368	5105100	270.964	276.743	7759752	191.139	194.780
3488400	222.132	228.976	5116320	259.252	266.262	7900200	385.753	392.952
3500640	284.475	291.321	5135130	210.154	215.353	7936110	208.203	212.360
3511200	279.413	284.981	5173168	76.873	79.609	8139600	369.753	376.964
3527160	216.258	220.597	5250960	340.733	347.754	8168160	405.951	413.164
3534300	287.923	293.542	5266800	343.199	350.222	8216208	319.437	325.612
3581424	176.745	182.559	5290740	261.043	266.838	8314020	255.075	261.066
3603600	390.027	396.884	5335200	293.848	300.876	8353800	435.393	441.385
3627936	174.161	181.022	5372136	214.656	219.415	8465184	278.170	285.398
3667950	145.862	150.216	5405400	421.682	428.716	8527200	258.818	264.771
3675672	250.931	255.526	5426400	280.636	286.392	8558550	221.153	225.875
3695120	84.473	88.949	5477472	274.517	280.516	8751600	373.465	380.708
3712800	294.678	301.549	5542680	223.806	228.342	8817900	246.155	250.893
3730650	140.617	143.583	5569200	392.596	399.643	8953560	382.383	388.406
3734640	336.521	343.394	5654880	451.380	458.434	9069840	337.707	344.965
3779100	190.618	196.266	5668650	132.668	135.980	9129120	357.078	363.061
3803800	116.814	120.458	5705700	235.664	241.443	9189180	401.775	407.809
3837240	240.921	246.576	5819814	142.526	146.206	9237800	110.211	113.854
3879876	134.341	137.680	5834400	279.272	286.339	9336600	369.595	376.867
3912480	280.062	286.956	5868720	338.093	345.163	9424800	470.055	477.330
3950100	255.740	262.637	5878600	120.487	123.089	9593100	248.494	254.547
3979360	140.677	145.454	5969040	286.889	292.687	9699690	175.227	178.193
4039200	314.952	321.860	6046560	270.003	277.086	9767520	436.647	443.938
4069800	292.493	298.173	6104700	270.436	276.292	9781200	346.724	354.016
4084080	326.235	333.147	6126120	407.657	413.514	9948400	156.058	161.233
4108104	250.340	256.213	6224400	372.935	380.030	9976824	206.931	211.959
4149600	283.678	289.319	6320160	431.670	438.772	10077600	259.157	265.183
4157010	130.552	134.429	6348888	245.699	251.573	10210200	405.302	411.381

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
10270260	366.736	374.049	18139680	403.875	411.434	36951200	188.187	193.663
10346336	107.703	110.740	18378360	592.825	599.159	37346400	660.666	668.539
10501920	430.010	437.332	18475600	139.115	144.290	38372400	486.295	494.180
10533600	440.185	447.508	18673200	530.783	538.355	38798760	466.689	472.070
10581480	376.428	382.523	19186200	363.089	369.442	39680550	308.251	312.408
10744272	276.987	283.278	19399380	318.235	323.315	39907296	353.256	360.117
10810800	553.441	560.776	19562400	444.731	452.323	40840800	656.428	664.340
11085360	281.537	287.604	19896800	207.868	213.343	41081040	737.765	745.679
11138400	489.061	496.409	19953648	277.224	283.784	41570100	425.984	432.673
11191950	224.785	229.506	20420400	523.268	530.879	42325920	606.516	614.443
11337300	259.374	265.499	20540520	548.947	556.560	44767800	611.050	617.771
11411400	363.967	370.047	20785050	192.430	196.307	45349200	521.664	529.621
11639628	237.816	242.911	21162960	486.702	494.329	45645600	604.545	611.227
11737440	421.555	428.926	21488544	375.667	382.259	45945900	638.812	645.544
11757200	165.014	169.147	21621600	709.073	716.709	46558512	441.206	448.134
11938080	367.574	373.674	22170720	341.127	347.495	48498450	275.433	278.399
12209400	401.389	407.547	22383900	415.419	421.839	48837600	674.686	682.676
12252240	512.505	519.894	22674600	374.304	380.730	49884120	534.221	540.990
12448800	460.768	468.164	22822800	481.784	488.164	51351300	593.102	601.114
12471030	215.994	220.348	22972950	334.178	338.098	51731680	269.329	274.106
12697776	320.845	328.250	23279256	340.314	345.710	52509600	695.000	703.021
12790800	302.668	310.076	23514400	208.206	212.640	52907400	601.877	608.672
12932920	153.091	156.035	24418800	523.797	531.486	53721360	727.678	735.709
13127400	427.535	433.724	24504480	648.990	656.680	55426800	449.488	456.254
13226850	222.778	226.459	24942060	373.696	380.163	58198140	535.781	542.617
13302432	244.956	251.340	25395552	407.014	414.719	58687200	666.321	674.390
13430340	356.433	362.632	25581600	403.892	411.601	59690400	588.077	594.875
13613600	207.579	213.055	25675650	327.988	333.187	61261200	811.901	819.989
13693680	484.338	491.776	25865840	195.197	199.673	62355150	297.097	301.451
13856700	264.218	269.151	26254800	530.647	538.367	63488880	723.270	731.374
14108640	350.721	356.893	26453700	412.641	419.134	64664600	246.194	249.838
14137200	534.629	542.080	26860680	545.299	551.799	66512160	582.852	590.976
14671800	383.893	391.360	27387360	616.393	624.131	67151700	583.848	590.746
14702688	404.971	411.398	27713400	364.833	370.067	68468400	797.865	806.001
14922600	371.155	376.120	28274400	683.163	690.916	69837768	484.290	490.163
15116400	354.070	361.551	29099070	295.793	300.514	70543200	597.958	604.829
15215200	197.177	201.423	29343600	525.032	532.801	73513440	958.682	966.850
15315300	451.884	458.140	29845200	458.848	465.345	76744800	622.578	630.764
15348960	410.530	418.017	30232800	424.412	432.193	77597520	603.645	610.557
15519504	235.759	240.931	30630600	648.102	654.659	79361100	564.540	571.510
15800400	518.869	526.369	31039008	313.300	318.773	82162080	934.882	943.097
15872220	361.493	367.764	31600800	673.361	681.162	83140200	585.339	592.330
16166150	94.070	95.713	31744440	539.782	546.355	87297210	435.637	440.836
16279200	461.636	469.149	32332300	145.686	149.028	89535600	759.552	767.805
16432416	408.664	415.140	33256080	465.379	473.202	90698400	623.739	631.997
16628040	359.434	365.725	33415200	702.139	709.964	91891800	917.717	924.751
16707600	568.399	575.923	33575850	306.423	311.621	93117024	574.098	581.326
17117100	406.088	413.622	34234200	610.403	618.238	96996900	508.972	514.750
17459442	191.834	195.991	34918884	337.898	343.470	99768240	717.351	725.651
17503200	485.920	493.464	35271600	485.795	492.365	102702600	868.534	876.847
17635800	366.850	371.889	35814240	631.332	639.187	105814800	784.807	793.133
17907120	489.572	497.126	36756720	754.888	762.754	107442720	942.457	950.789

t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$	t	$\log_{10}(\prod q)$	$\log_{10}(e(t))$
110853600	543.869	550.936	193993800	771.350	777.430	436486050	667.983	673.182
116396280	790.130	797.267	199536480	885.903	894.504	465585120	1291.766	1300.735
122522400	1030.044	1038.433	205405200	1157.572	1166.186	498841200	1094.146	1103.145
124710300	581.108	588.274	211629600	975.868	984.494	537213600	1453.816	1462.848
126977760	887.140	895.545	232792560	1028.035	1036.703	581981400	1270.468	1278.304
129329200	319.016	324.191	249420600	828.172	835.639	634888800	1398.685	1407.788
134303400	877.466	884.665	258658400	419.018	424.494	698377680	1523.645	1532.790
136936800	1013.581	1022.018	268606800	1119.129	1127.859	775975200	1244.929	1252.841
139675536	635.938	643.343	279351072	811.335	819.041	872972100	1233.655	1241.667
145495350	462.476	467.198	290990700	870.603	878.137	997682400	1376.302	1385.602
155195040	747.367	754.580	317444400	1120.091	1128.893	1163962800	1641.613	1650.980
158722200	833.412	840.684	332560800	938.282	947.105	1396755360	1904.157	1913.604
166280400	736.928	745.450	349188840	1150.434	1158.048	1745944200	1820.718	1829.031
174594420	777.189	784.502	367567200	1492.926	1501.792	2327925600	2073.179	2082.847
179071200	986.620	995.174	387987600	995.654	1003.265	3491888400	2378.628	2388.472
183783600	1163.210	1171.776	410810400	1475.039	1483.954	6983776800	3000.728	3010.873

l^e	m	g	f	D	S
2	3	2	x^2+x+1	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	5	2	x^2+x-1	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	7	3	x^2+x+2	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	11	2	x^2+x+3	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	13	2	x^2+x-3	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	17	3	x^2+x-4	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	19	2	x^2+x+5	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	23	5	x^2+x+6	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	29	3	x^2+x-7	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	31	3	x^2+x+8	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	37	2	x^2+x-9	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	41	6	x^2+x-10	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	43	3	x^2+x+11	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	47	5	x^2+x+12	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	53	2	x^2+x-13	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	59	2	x^2+x+15	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	61	2	x^2+x-15	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	67	2	x^2+x+17	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	71	7	x^2+x+18	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	73	5	x^2+x-18	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	79	3	x^2+x+20	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$
2	83	2	x^2+x+21	1	$\begin{pmatrix}-1 & 1\\ -1 & 0\end{pmatrix}$

l^e	m	g	f	D	S
2	89	3	x^2+x-22	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	97	5	x^2+x-24	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	101	2	x^2+x-25	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	103	5	x^2+x+26	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	107	2	x^2+x+27	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	109	6	x^2+x-27	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	113	3	x^2+x-28	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	127	3	x^2+x+32	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	131	2	x^2+x+33	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	137	3	x^2+x-34	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	139	2	x^2+x+35	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	149	2	x^2+x-37	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	151	6	x^2+x+38	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
2	157	5	x^2+x-39	1	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$

l^e	m	g	f	D	S
4	5	2	$x^4+x^3+x^2+x+1$	1	$\begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$
4	13	2	$x^4+x^3+2x^2-4x+3$	3	$\begin{pmatrix} -1 & 1 & 0 & -5 \\ -1 & 0 & 1 & -3 \\ -1 & 0 & 2 & -6 \\ -1 & 0 & 0 & -3 \end{pmatrix}$
4	17	3	$x^4+x^3-6x^2-x+1$	2	$\begin{pmatrix} -1 & 1 & -4 & 9 \\ -1 & 0 & -2 & 1 \\ -1 & 0 & -3 & 3 \\ -1 & 0 & -4 & 3 \end{pmatrix}$
4	29	2	$x^4+x^3+4x^2+20x+23$	7	$\begin{pmatrix} -1 & 1 & 2 & 9 \\ -1 & 0 & 3 & 13 \\ -1 & 0 & 0 & 12 \\ -1 & 0 & 2 & 15 \end{pmatrix}$

l^e	m	g	f	D	S
4	37	2	$x^4+x^3+5x^2+7x+49$	21	$\begin{pmatrix} -1 & 1 & 2 & 0 \\ -1 & 0 & 1 & 6 \\ -1 & 0 & 2 & -2 \\ -1 & 0 & 4 & 3 \end{pmatrix}$
4	41	6	$x^4+x^3-15x^2+18x-4$	2	$\begin{pmatrix} -1 & 1 & -10 & 39 \\ -1 & 0 & -6 & 18 \\ -1 & 0 & -7 & 21 \\ -1 & 0 & -8 & 22 \end{pmatrix}$
4	53	2	$x^4+x^3+7x^2-43x+47$	13	$\begin{pmatrix} -1 & 1 & 2 & -42 \\ -1 & 0 & 3 & -32 \\ -1 & 0 & 6 & -42 \\ -1 & 0 & 2 & -33 \end{pmatrix}$
4	61	2	$x^4+x^3+8x^2+42x+117$	39	$\begin{pmatrix} -1 & 1 & 4 & 21 \\ -1 & 0 & 3 & 33 \\ -1 & 0 & 2 & 22 \\ -1 & 0 & 6 & 27 \end{pmatrix}$
4	73	5	$x^4+x^3-27x^2-41x+2$	8	$\begin{pmatrix} -1 & 1 & -13 & 9 \\ -1 & 0 & -12 & -18 \\ -1 & 0 & -14 & -14 \\ -1 & 0 & -16 & -18 \end{pmatrix}$
4	89	3	$x^4+x^3-33x^2+39x+8$	8	$\begin{pmatrix} -1 & 1 & -19 & 81 \\ -1 & 0 & -14 & 40 \\ -1 & 0 & -16 & 48 \\ -1 & 0 & -18 & 48 \end{pmatrix}$
4	97	5	$x^4+x^3-36x^2+91x-61$	2	$\begin{pmatrix} -1 & 1 & -22 & 129 \\ -1 & 0 & -18 & 87 \\ -1 & 0 & -17 & 85 \\ -1 & 0 & -16 & 81 \end{pmatrix}$
4	101	2	$x^4+x^3+13x^2+19x+361$	95	$\begin{pmatrix} -1 & 1 & 6 & 0 \\ -1 & 0 & 9 & 10 \\ -1 & 0 & 6 & -6 \\ -1 & 0 & 4 & 15 \end{pmatrix}$
4	109	6	$x^4+x^3+14x^2-34x+393$	105	$\begin{pmatrix} -1 & 1 & 6 & -41 \\ -1 & 0 & 9 & -27 \\ -1 & 0 & 8 & -48 \\ -1 & 0 & 4 & -27 \end{pmatrix}$
4	113	3	$x^4+x^3-42x^2-120x-64$	8	$\begin{pmatrix} -1 & 1 & -19 & -27 \\ -1 & 0 & -20 & -68 \\ -1 & 0 & -22 & -66 \\ -1 & 0 & -24 & -72 \end{pmatrix}$
4	137	3	$x^4+x^3-51x^2-214x-236$	2	$\begin{pmatrix} -1 & 1 & -22 & -81 \\ -1 & 0 & -26 & -134 \\ -1 & 0 & -27 & -135 \\ -1 & 0 & -28 & -138 \end{pmatrix}$
4	149	2	$x^4+x^3+19x^2-121x+635$	155	$\begin{pmatrix} -1 & 1 & 8 & -114 \\ -1 & 0 & 11 & -90 \\ -1 & 0 & 12 & -120 \\ -1 & 0 & 6 & -95 \end{pmatrix}$
4	157	5	$x^4+x^3+20x^2-206x+517$	111	$\begin{pmatrix} -1 & 1 & 8 & -183 \\ -1 & 0 & 7 & -159 \\ -1 & 0 & 14 & -182 \\ -1 & 0 & 10 & -153 \end{pmatrix}$

l^e	m	g	f	D
			S	
8	17	3	$x^8+x^7-7x^6-6x^5+15x^4+10x^3-10x^2-4x+1$	1
			$\begin{pmatrix} -1 & 1 & -2 & 3 & -6 & 10 & -20 & 35 \\ -1 & 0 & -2 & 1 & -6 & 5 & -20 & 21 \\ -1 & 0 & -2 & 0 & -6 & 0 & -20 & 0 \\ -1 & 0 & -2 & 0 & -6 & 0 & -20 & 1 \\ -1 & 0 & -2 & 0 & -5 & 0 & -14 & 0 \\ -1 & 0 & -2 & 0 & -6 & 1 & -20 & 7 \\ -1 & 0 & -1 & 0 & -2 & 0 & -5 & 0 \\ -1 & 0 & -2 & 0 & -6 & 0 & -19 & 0 \end{pmatrix}$	
8	41	6	$x^8+x^7+3x^6+11x^5+44x^4-53x^3+153x^2-160x+59$	853738
			$\begin{pmatrix} -1 & 1 & 0 & 3 & 12 & -95 & 191 & -462 \\ -1 & 0 & 2 & 0 & 16 & -60 & 31 & -56 \\ -1 & 0 & 1 & 3 & 12 & -70 & 65 & -196 \\ -1 & 0 & 2 & 3 & 18 & -85 & 30 & -147 \\ -1 & 0 & 0 & 0 & 29 & -85 & 141 & -357 \\ -1 & 0 & 0 & 6 & 6 & -50 & 81 & -406 \\ -1 & 0 & 0 & 3 & 16 & -54 & 66 & -343 \\ -1 & 0 & 0 & 7 & 16 & -65 & 60 & -489 \end{pmatrix}$	
8	73	5	$x^8+x^7+5x^6-17x^5-46x^4-136x^3+320x^2+512x+4096$	12288
			$\begin{pmatrix} -1 & 1 & 1 & -9 & -11 & 15 & 261 & -609 \\ -1 & 0 & 2 & -6 & -18 & 26 & 246 & -174 \\ -1 & 0 & 0 & -6 & -28 & -30 & 260 & -350 \\ -1 & 0 & 0 & -12 & -8 & -20 & 176 & -252 \\ -1 & 0 & 2 & -6 & -18 & -30 & 302 & -462 \\ -1 & 0 & 2 & -12 & -30 & 20 & 210 & -588 \\ -1 & 0 & 2 & -8 & -14 & 0 & 346 & -168 \\ -1 & 0 & 0 & -6 & -20 & 10 & 308 & -462 \end{pmatrix}$	
8	89	3	$x^8+x^7+6x^6+46x^5-143x^4-575x^3+1160x^2+16x+512$	17152
			$\begin{pmatrix} -1 & 1 & 1 & 9 & -127 & -215 & 2145 & 5705 \\ -1 & 0 & 2 & 16 & -94 & -400 & 1282 & 8912 \\ -1 & 0 & 2 & 24 & -134 & -480 & 1394 & 12264 \\ -1 & 0 & 0 & 12 & -80 & -340 & 1136 & 6412 \\ -1 & 0 & 0 & 24 & -64 & -520 & 672 & 8792 \\ -1 & 0 & 2 & 12 & -102 & -380 & 1506 & 8316 \\ -1 & 0 & 4 & 12 & -116 & -524 & 2180 & 10332 \\ -1 & 0 & 0 & 12 & -88 & -300 & 1216 & 6300 \end{pmatrix}$	
8	97	5	$x^8+x^7-42x^6-59x^5+497x^4+719x^3-1792x^2-2295x+193$	2374833
			$\begin{pmatrix} -1 & 1 & -10 & 9 & -156 & 10 & -2963 & -3339 \\ -1 & 0 & -10 & -9 & -174 & -359 & -3748 & -11865 \\ -1 & 0 & -11 & -12 & -198 & -495 & -4419 & -16156 \\ -1 & 0 & -10 & -9 & -182 & -380 & -4075 & -12894 \\ -1 & 0 & -12 & -9 & -219 & -410 & -4824 & -14301 \\ -1 & 0 & -10 & -3 & -162 & -185 & -3253 & -7287 \\ -1 & 0 & -12 & -11 & -224 & -465 & -5017 & -15708 \\ -1 & 0 & -10 & -6 & -158 & -260 & -3199 & -9015 \end{pmatrix}$	

l^e	m	g	f	D
			S	
8	113	3	$x^8+x^7-49x^6+16x^5+511x^4-367x^3+$ $-1499x^2+798x+1372$	9296
			$\begin{pmatrix} -1 & 1 & -14 & 51 & -479 & 2540 & -20483 & 123509 \\ -1 & 0 & -10 & 13 & -264 & 870 & -10037 & 47250 \\ -1 & 0 & -12 & 18 & -338 & 1140 & -12982 & 61418 \\ -1 & 0 & -12 & 27 & -378 & 1626 & -15513 & 83762 \\ -1 & 0 & -11 & 21 & -326 & 1295 & -13059 & 67676 \\ -1 & 0 & -12 & 15 & -350 & 1090 & -13469 & 60998 \\ -1 & 0 & -14 & 30 & -444 & 1830 & -18054 & 95172 \\ -1 & 0 & -14 & 21 & -440 & 1470 & -17369 & 81270 \end{pmatrix}$	
8	137	3	$x^8+x^7+9x^6+105x^5+954x^4+3767x^3+$ $+9149x^2+12828x+7607$	2561927876
			$\begin{pmatrix} -1 & 1 & 4 & 33 & 302 & 385 & -7259 & -72366 \\ -1 & 0 & 0 & 37 & 436 & 1595 & -2522 & -81725 \\ -1 & 0 & 3 & 39 & 440 & 1350 & -5127 & -90278 \\ -1 & 0 & 0 & 36 & 468 & 1736 & -2553 & -85120 \\ -1 & 0 & 0 & 12 & 331 & 1525 & 459 & -49595 \\ -1 & 0 & 4 & 45 & 408 & 1115 & -6558 & -93177 \\ -1 & 0 & 2 & 39 & 444 & 1420 & -4358 & -88277 \\ -1 & 0 & 4 & 48 & 440 & 1100 & -7165 & -98756 \end{pmatrix}$	

l^e	m	g	f	D
			S	
16	17	3	$x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+$ $x^6+x^5+x^4+x^3+x^2+x+1$	1
			$\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	

l^e	m	g	f	D
			S	
16	97	5	$x^{16}+x^{15}-45x^{14}-98x^{13}+650x^{12}+2183x^{11}-2576x^{10}+$ $-17205x^9-9748x^8+44003x^7+63779x^6-18576x^5+$ $-86644x^4-43324x^3+15475x^2+17690x+3721$	187125655 07118089

$$\begin{pmatrix}
-1 & 1 & -4 & 3 & -30 & -20 & -360 & -945 \\
-1 & 0 & -6 & -12 & -90 & -359 & -1990 & -9569 \\
-1 & 0 & -5 & -6 & -56 & -160 & -945 & -3850 \\
-1 & 0 & -6 & -9 & -74 & -240 & -1350 & -5845 \\
-1 & 0 & -6 & -12 & -89 & -340 & -1884 & -8813 \\
-1 & 0 & -6 & -12 & -90 & -360 & -2019 & -9765 \\
-1 & 0 & -6 & -11 & -78 & -285 & -1510 & -6909 \\
-1 & 0 & -6 & -12 & -90 & -355 & -1974 & -9471 \\
-1 & 0 & -6 & -12 & -90 & -350 & -1949 & -9247 \\
-1 & 0 & -6 & -12 & -90 & -350 & -1944 & -9177 \\
-1 & 0 & -6 & -12 & -86 & -325 & -1770 & -8197 \\
-1 & 0 & -4 & -6 & -42 & -140 & -720 & -3150 \\
-1 & 0 & -6 & -9 & -74 & -240 & -1350 & -5845 \\
-1 & 0 & -6 & -12 & -84 & -320 & -1710 & -7973 \\
-1 & 0 & -6 & -12 & -90 & -355 & -1959 & -9331 \\
-1 & 0 & -6 & -12 & -86 & -325 & -1770 & -8217
\end{pmatrix}$$

$$\begin{pmatrix}
-6370 & -25914 & -142131 & -676434 & -3537215 & -17800848 & -92329341 & -474317467 \\
-50134 & -254202 & -1320489 & -6809880 & -35383194 & -183601626 & -955138105 & -4968588947 \\
-20174 & -94995 & -487226 & -2431847 & -12500828 & -63908117 & -330215717 & -1705718742 \\
-30274 & -146361 & -750545 & -3788213 & -19510182 & -100228466 & -518714027 & -2685339851 \\
-46024 & -230277 & -1192146 & -6110203 & -31671441 & -163862972 & -851205368 & -4421638594 \\
-51618 & -263064 & -1374036 & -7110708 & -37062806 & -192725676 & -1004362645 & -5231049172 \\
-35097 & -173472 & -885146 & -4505412 & -23185800 & -119498626 & -618552311 & -3206383127 \\
-49706 & -252264 & -1312026 & -6774108 & -35228127 & -182946465 & -952274609 & -4956148481 \\
-48502 & -244662 & -1270881 & -6541898 & -33983906 & -176221825 & -916612605 & -4766961847 \\
-47978 & -240840 & -1247451 & -6400108 & -33179630 & -171712111 & -892004217 & -4633801827 \\
-42378 & -210585 & -1083896 & -5532538 & -28584533 & -147547673 & -765100907 & -3969286552 \\
-15610 & -75396 & -378866 & -1905552 & -9724440 & -49812308 & -256709557 & -1326508092 \\
-30254 & -146171 & -749079 & -3778269 & -19449330 & -99868249 & -516679995 & -2674031631 \\
-40858 & -203754 & -1045376 & -5344372 & -27585756 & -142489893 & -738715158 & -3833578864 \\
-48594 & -244866 & -1266126 & -6504773 & -33699006 & -174474144 & -906105549 & -4707582317 \\
-42588 & -212241 & -1095306 & -5603334 & -29005602 & -149941922 & -778443159 & -4042098032
\end{pmatrix}$$

l^e	m	g	f	D
			S	
16	113	3	$x^{16}+x^{15}+4x^{14}+20x^{13}+110x^{12}+525x^{11}+$ $325x^{10}-425x^9+12062x^8-21729x^7+$ $64244x^6-119403x^5+154492x^4+$ $-132177x^3+210865x^2+$ $-281708x+132937$	23550755868325548454 55643233624699782 48

$$
\begin{pmatrix}
-1 & 1 & 0 & 3 & 16 & 75 & -429 & -1036 \\
-1 & 0 & 0 & 1 & 18 & 150 & -300 & -476 \\
-1 & 0 & 2 & 0 & 28 & 115 & -345 & -1939 \\
-1 & 0 & 0 & 6 & 4 & 126 & -160 & -1519 \\
-1 & 0 & 0 & 3 & 24 & 80 & -264 & -1043 \\
-1 & 0 & 2 & 6 & 18 & 125 & -390 & -2016 \\
-1 & 0 & 0 & 9 & 12 & 135 & -340 & -1400 \\
-1 & 0 & 0 & 6 & 24 & 155 & -165 & -1792 \\
-1 & 0 & 0 & 0 & 5 & 145 & -15 & -1707 \\
-1 & 0 & 2 & 0 & 24 & 70 & -234 & -1701 \\
-1 & 0 & 0 & 0 & 24 & 175 & -355 & -833 \\
-1 & 0 & 0 & 3 & 28 & 50 & -78 & -1190 \\
-1 & 0 & 1 & 3 & 12 & 110 & -105 & -1428 \\
-1 & 0 & 0 & 0 & 36 & 135 & -119 & -1078 \\
-1 & 0 & 0 & 3 & 34 & 85 & -30 & -1890 \\
-1 & 0 & 0 & 6 & 36 & 105 & -204 & -1988
\end{pmatrix}
$$

$$
\begin{pmatrix}
4592 & -32565 & 99240 & 82038 & 483912 & -2475759 & 9593857 & -58629855 \\
778 & -27438 & 17235 & 161876 & 556534 & -502489 & 13123448 & -48536157 \\
3780 & -29285 & 41055 & 323972 & 523908 & -1580241 & 6547282 & -44916519 \\
1464 & -20844 & 34115 & 83688 & 1128139 & -2102022 & 7273630 & -35100047 \\
1261 & -25203 & 58335 & 134530 & 557425 & -673582 & 5643275 & -46952499 \\
1288 & -21579 & 50362 & 287188 & 944262 & -2287506 & 4144049 & -55322665 \\
896 & -27282 & 61825 & 126335 & 1154373 & -2116906 & 9138129 & -60263719 \\
-308 & -33102 & 41350 & 184525 & 1371690 & -479414 & 8200192 & -55353177 \\
6132 & -35571 & 21072 & 50391 & 936003 & -2091180 & 14210378 & -15742399 \\
3228 & -21978 & 34335 & 257598 & 376157 & -1100528 & 3111668 & -30282551 \\
1778 & -35865 & 24315 & 233432 & 638550 & -609115 & 15316548 & -57982439 \\
1492 & -29970 & 65295 & 70972 & 626604 & 344149 & 4063696 & -37620245 \\
2136 & -24324 & 20227 & 127985 & 831403 & -1009931 & 7627074 & -27373825 \\
490 & -38058 & 28230 & 181797 & 756624 & 1234454 & 10603490 & -47141155 \\
2604 & -35460 & 52125 & 129635 & 894939 & 99164 & 5515993 & -32785077 \\
28 & -32919 & 70306 & 218856 & 1109020 & -190632 & 3399865 & -59182367
\end{pmatrix}
$$

l^e	m	g	f	D	S
3	7	3	x^3+x^2-2x-1	1	$\begin{pmatrix}-1&1&-2\\-1&0&-2\\-1&0&-1\end{pmatrix}$
3	13	2	x^3+x^2-4x+1	1	$\begin{pmatrix}-1&1&-4\\-1&0&-3\\-1&0&-2\end{pmatrix}$
3	19	2	x^3+x^2-6x-7	1	$\begin{pmatrix}-1&1&-4\\-1&0&-5\\-1&0&-4\end{pmatrix}$
3	31	3	$x^3+x^2-10x-8$	2	$\begin{pmatrix}-1&1&-7\\-1&0&-6\\-1&0&-8\end{pmatrix}$
3	37	2	$x^3+x^2-12x+11$	1	$\begin{pmatrix}-1&1&-10\\-1&0&-7\\-1&0&-8\end{pmatrix}$
3	43	3	$x^3+x^2-14x+8$	2	$\begin{pmatrix}-1&1&-11\\-1&0&-8\\-1&0&-10\end{pmatrix}$
3	61	2	$x^3+x^2-20x-9$	3	$\begin{pmatrix}-1&1&-14\\-1&0&-15\\-1&0&-12\end{pmatrix}$
3	67	2	$x^3+x^2-22x+5$	3	$\begin{pmatrix}-1&1&-16\\-1&0&-13\\-1&0&-16\end{pmatrix}$
3	73	5	$x^3+x^2-24x-27$	3	$\begin{pmatrix}-1&1&-16\\-1&0&-18\\-1&0&-15\end{pmatrix}$
3	79	3	$x^3+x^2-26x+41$	1	$\begin{pmatrix}-1&1&-20\\-1&0&-17\\-1&0&-16\end{pmatrix}$
3	97	5	$x^3+x^2-32x-79$	1	$\begin{pmatrix}-1&1&-20\\-1&0&-23\\-1&0&-22\end{pmatrix}$
3	103	5	$x^3+x^2-34x-61$	3	$\begin{pmatrix}-1&1&-22\\-1&0&-22\\-1&0&-25\end{pmatrix}$
3	109	6	$x^3+x^2-36x-4$	4	$\begin{pmatrix}-1&1&-25\\-1&0&-26\\-1&0&-22\end{pmatrix}$
3	127	3	$x^3+x^2-42x+80$	2	$\begin{pmatrix}-1&1&-31\\-1&0&-26\\-1&0&-28\end{pmatrix}$
3	139	2	$x^3+x^2-46x+103$	1	$\begin{pmatrix}-1&1&-34\\-1&0&-29\\-1&0&-30\end{pmatrix}$
3	151	6	$x^3+x^2-50x-123$	3	$\begin{pmatrix}-1&1&-32\\-1&0&-33\\-1&0&-36\end{pmatrix}$
3	157	5	$x^3+x^2-52x+64$	4	$\begin{pmatrix}-1&1&-37\\-1&0&-36\\-1&0&-32\end{pmatrix}$

l^e	m	g	f	D
			S	
9	19	2	$x^9+x^8-8x^7-7x^6+21x^5+15x^4-20x^3-10x^2+5x+1$	1
			$\begin{pmatrix} -1 & 1 & -2 & 3 & -6 & 10 & -20 & 35 & -70 \\ -1 & 0 & -1 & 0 & -2 & 0 & -5 & 0 & -14 \\ -1 & 0 & -2 & 0 & -5 & 0 & -14 & 0 & -42 \\ -1 & 0 & -2 & 0 & -6 & 0 & -20 & 0 & -69 \\ -1 & 0 & -2 & 1 & -6 & 5 & -20 & 21 & -70 \\ -1 & 0 & -2 & 0 & -6 & 0 & -19 & 0 & -62 \\ -1 & 0 & -2 & 0 & -6 & 0 & -20 & 1 & -70 \\ -1 & 0 & -2 & 0 & -6 & 1 & -20 & 7 & -70 \\ -1 & 0 & -2 & 0 & -6 & 0 & -20 & 0 & -70 \end{pmatrix}$	
9	37	2	$x^9+x^8-16x^7-11x^6+66x^5+32x^4-73x^3-7x^2+7x+1$	1333
			$\begin{pmatrix} -1 & 1 & -4 & 9 & -36 & 100 & -393 & 1197 & -4576 \\ -1 & 0 & -3 & 0 & -20 & 5 & -175 & 126 & -1764 \\ -1 & 0 & -4 & 3 & -35 & 50 & -364 & 707 & -4052 \\ -1 & 0 & -4 & 3 & -36 & 50 & -379 & 707 & -4227 \\ -1 & 0 & -4 & 0 & -36 & 10 & -365 & 252 & -3892 \\ -1 & 0 & -2 & 0 & -12 & 1 & -100 & 43 & -980 \\ -1 & 0 & -4 & 0 & -30 & 5 & -280 & 154 & -2884 \\ -1 & 0 & -4 & 0 & -32 & 10 & -310 & 238 & -3262 \\ -1 & 0 & -4 & 1 & -32 & 25 & -310 & 413 & -3304 \end{pmatrix}$	
9	73	5	$x^9+x^8-32x^7-11x^6+278x^5-34x^4-427x^3+150x^2-8x-1$	49053
			$\begin{pmatrix} -1 & 1 & -8 & 21 & -138 & 460 & -2688 & 10325 & -55776 \\ -1 & 0 & -6 & 6 & -96 & 196 & -1779 & 5236 & -35686 \\ -1 & 0 & -8 & 9 & -134 & 290 & -2555 & 7707 & -52142 \\ -1 & 0 & -6 & 3 & -86 & 100 & -1409 & 2730 & -25276 \\ -1 & 0 & -6 & 3 & -84 & 105 & -1394 & 2912 & -25480 \\ -1 & 0 & -8 & 6 & -128 & 220 & -2314 & 6118 & -45380 \\ -1 & 0 & -8 & 4 & -124 & 165 & -2180 & 4838 & -41599 \\ -1 & 0 & -8 & 6 & -119 & 180 & -1998 & 4634 & -36728 \\ -1 & 0 & -7 & 6 & -112 & 190 & -2015 & 5068 & -39114 \end{pmatrix}$	
9	109	6	$x^9+x^8-48x^7-73x^6+660x^5+1454x^4-2149x^3-8350x^2-7432x-2008$	29632996
			$\begin{pmatrix} -1 & 1 & -10 & 9 & -186 & 160 & -4188 & 3073 & -102331 \\ -1 & 0 & -12 & -12 & -248 & -370 & -6002 & -11172 & -155306 \\ -1 & 0 & -12 & -9 & -238 & -245 & -5593 & -7105 & -141678 \\ -1 & 0 & -11 & -12 & -226 & -350 & -5455 & -10227 & -140714 \\ -1 & 0 & -10 & -12 & -188 & -344 & -4348 & -9778 & -110528 \\ -1 & 0 & -10 & -9 & -172 & -215 & -3675 & -5453 & -88270 \\ -1 & 0 & -10 & -6 & -187 & -115 & -4198 & -2835 & -102898 \\ -1 & 0 & -12 & -8 & -234 & -230 & -5374 & -6846 & -134308 \\ -1 & 0 & -10 & -15 & -190 & -445 & -4455 & -12621 & -114492 \end{pmatrix}$	

l^e	m	g	f	D
			S	
9	127	3	$x^9+x^8-56x^7-118x^6+573x^5+1249x^4-1582x^3+$ $-2700x^2+1576x+32$	33536
			$\begin{pmatrix} -1 & 1 & -11 & 9 & -299 & -295 & -11243 & -32599 & -485035 \\ -1 & 0 & -10 & -20 & -306 & -1380 & -13442 & -78036 & -651698 \\ -1 & 0 & -12 & -24 & -396 & -1680 & -17428 & -96600 & -838332 \\ -1 & 0 & -14 & -30 & -482 & -2070 & -21474 & -118902 & -1035026 \\ -1 & 0 & -14 & -24 & -478 & -1800 & -20894 & -107240 & -992094 \\ -1 & 0 & -14 & -24 & -478 & -1840 & -20918 & -109368 & -996014 \\ -1 & 0 & -12 & -18 & -408 & -1434 & -17720 & -87290 & -835624 \\ -1 & 0 & -12 & -18 & -376 & -1370 & -16104 & -81850 & -761192 \\ -1 & 0 & -14 & -36 & -510 & -2420 & -23358 & -136388 & -1140734 \end{pmatrix}$	

l^e	m	g	f	D	S
5	11	2	$x^5+x^4-4x^3-3x^2+$ $3x+1$	1	$\begin{pmatrix} -1 & 1 & -2 & 3 & -6 \\ -1 & 0 & -1 & 0 & -2 \\ -1 & 0 & -2 & 0 & -5 \\ -1 & 0 & -2 & 1 & -6 \\ -1 & 0 & -2 & 0 & -6 \end{pmatrix}$
5	31	3	$x^5+x^4-12x^3-21x^2+$ $x+5$	5	$\begin{pmatrix} -1 & 1 & -4 & 3 & -30 \\ -1 & 0 & -6 & -8 & -62 \\ -1 & 0 & -6 & -9 & -64 \\ -1 & 0 & -4 & -6 & -41 \\ -1 & 0 & -5 & -6 & -52 \end{pmatrix}$
5	41	6	$x^5+x^4-16x^3+5x^2+$ $21x-9$	9	$\begin{pmatrix} -1 & 1 & -8 & 25 & -140 \\ -1 & 0 & -5 & 6 & -68 \\ -1 & 0 & -6 & 12 & -93 \\ -1 & 0 & -6 & 9 & -90 \\ -1 & 0 & -8 & 12 & -122 \end{pmatrix}$
5	61	2	$x^5+x^4-24x^3-17x^2+$ $41x-13$	29	$\begin{pmatrix} -1 & 1 & -10 & 21 & -210 \\ -1 & 0 & -9 & 4 & -190 \\ -1 & 0 & -12 & 0 & -253 \\ -1 & 0 & -8 & -3 & -156 \\ -1 & 0 & -10 & 0 & -208 \end{pmatrix}$
5	71	7	$x^5+x^4-28x^3+37x^2+$ $25x+1$	23	$\begin{pmatrix} -1 & 1 & -14 & 63 & -472 \\ -1 & 0 & -9 & 28 & -262 \\ -1 & 0 & -10 & 30 & -289 \\ -1 & 0 & -12 & 39 & -354 \\ -1 & 0 & -12 & 36 & -352 \end{pmatrix}$
5	101	2	$x^5+x^4-40x^3+93x^2+$ $-21x-17$	17	$\begin{pmatrix} -1 & 1 & -20 & 117 & -1026 \\ -1 & 0 & -17 & 78 & -776 \\ -1 & 0 & -14 & 66 & -645 \\ -1 & 0 & -16 & 75 & -734 \\ -1 & 0 & -14 & 64 & -636 \end{pmatrix}$

l^e	m	g	f	D	S
5	131	2	$x^5+x^4-52x^3-89x^2+109x+193$	79	$\begin{pmatrix} -1 & 1 & -20 & 15 & -886 \\ -1 & 0 & -21 & -30 & -972 \\ -1 & 0 & -24 & -38 & -1129 \\ -1 & 0 & -18 & -33 & -820 \\ -1 & 0 & -22 & -24 & -1018 \end{pmatrix}$
5	151	6	$x^5+x^4-60x^3-12x^2+784x+128$	32	$\begin{pmatrix} -1 & 1 & -25 & 57 & -921 \\ -1 & 0 & -24 & 16 & -800 \\ -1 & 0 & -24 & 24 & -856 \\ -1 & 0 & -22 & 18 & -746 \\ -1 & 0 & -26 & 30 & -934 \end{pmatrix}$

$\mathrm{ord}(\chi) = 2$		
$\tau(\chi)^{1-\sigma_1}$	$=$	1
$\tau(\chi)^2$	$=$	q^*

$\mathrm{ord}(\chi) = 3$			
$\tau(\chi)^{1-\sigma_1}$	$=$	1	
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$
$\tau(\chi)^3$	$=$	q^*	$j(\chi,\chi)$

$\mathrm{ord}(\chi) = 4$			
$\tau(\chi)^{1-\sigma_1}$	$=$	1	
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)^2$
$\tau(\chi)^4$	$=$	q^*	$j(\chi,\chi)^2$

$\mathrm{ord}(\chi) = 5$			
$\tau(\chi)^{1-\sigma_1}$	$=$	1	
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)^{1+\sigma_2}$
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$
$\tau(\chi)^5$	$=$	q^*	$j(\chi,\chi)^{2+\sigma_2}$

$\mathrm{ord}(\chi) = 7$				
$\tau(\chi)^{1-\sigma_1}$	$=$	1		
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$	
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$	
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$	$j(\chi,\chi^2)$
$\tau(\chi)^{6-\sigma_6}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^2$
$\tau(\chi)^7$	$=$	q^*	$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^2$

$\mathrm{ord}(\chi) = 8$				
$\tau(\chi)^{1-\sigma_1}$	$=$	1		
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)$
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^2$
$\tau(\chi)^8$	$=$	q^*	$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^2$

$\mathrm{ord}(\chi) = 9$			
$\tau(\chi)^{1-\sigma_1}$	$=$	1	
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2+\sigma_2+\sigma_4}$
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{3+2\sigma_2+\sigma_4}$
$\tau(\chi)^{8-\sigma_8}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$
$\tau(\chi)^9$	$=$	q^*	$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$

$\mathrm{ord}(\chi) = 11$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$		
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$	
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$		
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$	$j(\chi,\chi^2)^{\sigma_6}$	
$\tau(\chi)^{6-\sigma_6}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^2$	
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{3+\sigma_2+\sigma_4}$	$j(\chi,\chi^2)$	
$\tau(\chi)^{8-\sigma_8}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$		
$\tau(\chi)^{9-\sigma_9}$	$=$		$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$	
$\tau(\chi)^{10-\sigma_{10}}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_5}$	$j(\chi,\chi^2)^{2\sigma_6}$	
$\tau(\chi)^{11}$	$=$	q^*	$j(\chi,\chi)^{4+2\sigma_2+\sigma_5}$	$j(\chi,\chi^2)^{2\sigma_6}$	

$\mathrm{ord}(\chi) = 13$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$		
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$	
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$		
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2}$	$j(\chi,\chi^2)^{1+\sigma_8}$	
$\tau(\chi)^{6-\sigma_6}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^2$	
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{2+\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^2$	
$\tau(\chi)^{8-\sigma_8}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$		
$\tau(\chi)^{9-\sigma_9}$	$=$		$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$	
$\tau(\chi)^{10-\sigma_{10}}$	$=$		$j(\chi,\chi)^{4+\sigma_5}$	$j(\chi,\chi^2)^{2+2\sigma_8}$	
$\tau(\chi)^{11-\sigma_{11}}$	$=$		$j(\chi,\chi)^{4+\sigma_2+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$	
$\tau(\chi)^{12-\sigma_{12}}$	$=$		$j(\chi,\chi)^{4+2\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^4$	
$\tau(\chi)^{13}$	$=$	q^*	$j(\chi,\chi)^{4+2\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^4$	

$\mathrm{ord}(\chi) = 16$						
$\tau(\chi)^{1-\sigma_1}$	$=$	1				
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$		
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^2$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)$	
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^3$	$j(\chi,\chi^2)^{1+\sigma_9}$	$j(\chi^2,\chi^3)$	
$\tau(\chi)^{9-\sigma_9}$	$=$		$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$		
$\tau(\chi)^{11-\sigma_{11}}$	$=$		$j(\chi,\chi)^{4+\sigma_3+\sigma_5}$	$j(\chi,\chi^2)^3$	$j(\chi^2,\chi^3)$	
$\tau(\chi)^{13-\sigma_{13}}$	$=$		$j(\chi,\chi)^{5+\sigma_3+\sigma_5}$	$j(\chi,\chi^2)^3$	$j(\chi^2,\chi^3)^2$	
$\tau(\chi)^{15-\sigma_{15}}$	$=$		$j(\chi,\chi)^{6+\sigma_5}$	$j(\chi,\chi^2)^{3+\sigma_5}$	$j(\chi^2,\chi^3)^3$	
$\tau(\chi)^{16}$	$=$	q^*	$j(\chi,\chi)^{6+\sigma_5}$	$j(\chi,\chi^2)^{3+\sigma_5}$	$j(\chi^2,\chi^3)^3$	

$\mathrm{ord}(\chi) = 17$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$		
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$	
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$		
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2}$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)$
$\tau(\chi)^{6-\sigma_6}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^{2}$	
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{3+\sigma_2}$	$j(\chi,\chi^2)^{1+\sigma_{10}}$	
$\tau(\chi)^{8-\sigma_8}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$		
$\tau(\chi)^{9-\sigma_9}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_4+\sigma_8}$		
$\tau(\chi)^{10-\sigma_{10}}$	$=$		$j(\chi,\chi)^{4+\sigma_2+\sigma_7}$	$j(\chi,\chi^2)^{2+\sigma_{10}}$	
$\tau(\chi)^{11-\sigma_{11}}$	$=$		$j(\chi,\chi)^{5+2\sigma_2+\sigma_4}$	$j(\chi,\chi^2)^{1+\sigma_3}$	
$\tau(\chi)^{12-\sigma_{12}}$	$=$		$j(\chi,\chi)^{4+2\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^{4}$	
$\tau(\chi)^{13-\sigma_{13}}$	$=$		$j(\chi,\chi)^{6+3\sigma_2+2\sigma_4+\sigma_8}$		
$\tau(\chi)^{14-\sigma_{14}}$	$=$		$j(\chi,\chi)^{6+2\sigma_2+\sigma_7}$	$j(\chi,\chi^2)^{2+2\sigma_{10}}$	
$\tau(\chi)^{15-\sigma_{15}}$	$=$		$j(\chi,\chi)^{7+4\sigma_2+2\sigma_4+\sigma_8}$		
$\tau(\chi)^{16-\sigma_{16}}$	$=$		$j(\chi,\chi)^{8+4\sigma_2+2\sigma_4+\sigma_8}$		
$\tau(\chi)^{17}$	$=$	q^*	$j(\chi,\chi)^{8+4\sigma_2+2\sigma_4+\sigma_8}$		

$\mathrm{ord}(\chi) = 19$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{2-\sigma_2}$	$=$		$j(\chi,\chi)$		
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$	
$\tau(\chi)^{4-\sigma_4}$	$=$		$j(\chi,\chi)^{2+\sigma_2}$		
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^{2}$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)$
$\tau(\chi)^{6-\sigma_6}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^{2}$	
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^{2+\sigma_6}$	
$\tau(\chi)^{8-\sigma_8}$	$=$		$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$		
$\tau(\chi)^{9-\sigma_9}$	$=$		$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$	
$\tau(\chi)^{10-\sigma_{10}}$	$=$		$j(\chi,\chi)^{3+\sigma_3+\sigma_9}$	$j(\chi,\chi^2)^{3+\sigma_3}$	
$\tau(\chi)^{11-\sigma_{11}}$	$=$		$j(\chi,\chi)^{5+2\sigma_2+\sigma_4+\sigma_8}$	$j(\chi,\chi^2)$	
$\tau(\chi)^{12-\sigma_{12}}$	$=$		$j(\chi,\chi)^{4+2\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^{4}$	
$\tau(\chi)^{13-\sigma_{13}}$	$=$		$j(\chi,\chi)^{4+2\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^{4+\sigma_6}$	
$\tau(\chi)^{14-\sigma_{14}}$	$=$		$j(\chi,\chi)^{4+2\sigma_3+\sigma_7}$	$j(\chi,\chi^2)^{4+2\sigma_6}$	
$\tau(\chi)^{15-\sigma_{15}}$	$=$		$j(\chi,\chi)^{7+3\sigma_2+2\sigma_4+\sigma_8}$	$j(\chi,\chi^2)$	
$\tau(\chi)^{16-\sigma_{16}}$	$=$		$j(\chi,\chi)^{8+4\sigma_2+2\sigma_4+\sigma_8}$		
$\tau(\chi)^{17-\sigma_{17}}$	$=$		$j(\chi,\chi)^{8+4\sigma_2+2\sigma_4+\sigma_8}$	$j(\chi,\chi^2)$	
$\tau(\chi)^{18-\sigma_{18}}$	$=$		$j(\chi,\chi)^{6+2\sigma_3+\sigma_9}$	$j(\chi,\chi^2)^{6+2\sigma_3}$	
$\tau(\chi)^{19}$	$=$	q^*	$j(\chi,\chi)^{6+2\sigma_3+\sigma_9}$	$j(\chi,\chi^2)^{6+2\sigma_3}$	

$\mathrm{ord}(\chi) = 25$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{2-\sigma_2}$	$=$	$j(\chi,\chi)$			
$\tau(\chi)^{3-\sigma_3}$	$=$	$j(\chi,\chi)$	$j(\chi,\chi^2)$		
$\tau(\chi)^{4-\sigma_4}$	$=$	$j(\chi,\chi)^{2+\sigma_2}$			
$\tau(\chi)^{6-\sigma_6}$	$=$	$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^2$		
$\tau(\chi)^{7-\sigma_7}$	$=$	$j(\chi,\chi)^3$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)^{1+\sigma_9}$	
$\tau(\chi)^{8-\sigma_8}$	$=$	$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$			
$\tau(\chi)^{9-\sigma_9}$	$=$	$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$		
$\tau(\chi)^{11-\sigma_{11}}$	$=$	$j(\chi,\chi)^{5+2\sigma_2+\sigma_4}$	$j(\chi,\chi^2)^{1+\sigma_{14}}$		
$\tau(\chi)^{12-\sigma_{12}}$	$=$	$j(\chi,\chi)^{4+2\sigma_3+\sigma_6}$	$j(\chi,\chi^2)^4$		
$\tau(\chi)^{13-\sigma_{13}}$	$=$	$j(\chi,\chi)^{4+2\sigma_3+\sigma_6+\sigma_{12}}$	$j(\chi,\chi^2)^4$		
$\tau(\chi)^{14-\sigma_{14}}$	$=$	$j(\chi,\chi)^{6+2\sigma_2+\sigma_4+\sigma_{11}}$	$j(\chi,\chi^2)^{2+\sigma_{14}}$		
$\tau(\chi)^{16-\sigma_{16}}$	$=$	$j(\chi,\chi)^{8+4\sigma_2+2\sigma_4+\sigma_8}$			
$\tau(\chi)^{17-\sigma_{17}}$	$=$	$j(\chi,\chi)^{7+2\sigma_2+\sigma_3+\sigma_4+\sigma_8}$	$j(\chi,\chi^2)^{3+\sigma_3}$		
$\tau(\chi)^{18-\sigma_{18}}$	$=$	$j(\chi,\chi)^{6+2\sigma_3+\sigma_9}$	$j(\chi,\chi^2)^{6+2\sigma_3}$		
$\tau(\chi)^{19-\sigma_{19}}$	$=$	$j(\chi,\chi)^{6+3\sigma_3+2\sigma_6+\sigma_{12}}$	$j(\chi,\chi^2)^6$		
$\tau(\chi)^{21-\sigma_{21}}$	$=$	$j(\chi,\chi)^{9+3\sigma_2+\sigma_3+2\sigma_4+\sigma_8}$	$j(\chi,\chi^2)^{3+\sigma_3}$		
$\tau(\chi)^{22-\sigma_{22}}$	$=$	$j(\chi,\chi)^{10+4\sigma_2+2\sigma_4+\sigma_{11}}$	$j(\chi,\chi^2)^{2+2\sigma_{14}}$		
$\tau(\chi)^{23-\sigma_{23}}$	$=$	$j(\chi,\chi)^{10+4\sigma_2+\sigma_3+2\sigma_4+\sigma_8}$	$j(\chi,\chi^2)^{3+\sigma_3}$		
$\tau(\chi)^{24-\sigma_{24}}$	$=$	$j(\chi,\chi)^{8+4\sigma_3+2\sigma_6+\sigma_{12}}$	$j(\chi,\chi^2)^8$		
$\tau(\chi)^{25}$	$=$	$q^*\ j(\chi,\chi)^{8+4\sigma_3+2\sigma_6+\sigma_{12}}$	$j(\chi,\chi^2)^8$		

$\mathrm{ord}(\chi) = 27$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{2-\sigma_2}$	$=$	$j(\chi,\chi)$			
$\tau(\chi)^{4-\sigma_4}$	$=$	$j(\chi,\chi)^{2+\sigma_2}$			
$\tau(\chi)^{5-\sigma_5}$	$=$	$j(\chi,\chi)^2$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)$	
$\tau(\chi)^{7-\sigma_7}$	$=$	$j(\chi,\chi)^{3+\sigma_2}$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)^{\sigma_{10}}$	
$\tau(\chi)^{8-\sigma_8}$	$=$	$j(\chi,\chi)^{4+2\sigma_2+\sigma_4}$			
$\tau(\chi)^{10-\sigma_{10}}$	$=$	$j(\chi,\chi)^{4+\sigma_5}$	$j(\chi,\chi^2)^2$	$j(\chi^2,\chi^3)^2$	
$\tau(\chi)^{11-\sigma_{11}}$	$=$	$j(\chi,\chi)^{5+2\sigma_2+\sigma_4}$	$j(\chi,\chi^2)^{1+\sigma_8}$		
$\tau(\chi)^{13-\sigma_{13}}$	$=$	$j(\chi,\chi)^{6+3\sigma_2+\sigma_4}$	$j(\chi,\chi^2)^{\sigma_4+\sigma_{14}}$		
$\tau(\chi)^{14-\sigma_{14}}$	$=$	$j(\chi,\chi)^{6+3\sigma_2+\sigma_4+\sigma_{13}}$	$j(\chi,\chi^2)^{\sigma_4+\sigma_{14}}$		
$\tau(\chi)^{16-\sigma_{16}}$	$=$	$j(\chi,\chi)^{8+4\sigma_2+2\sigma_4+\sigma_8}$			
$\tau(\chi)^{17-\sigma_{17}}$	$=$	$j(\chi,\chi)^{7+\sigma_2+\sigma_5+\sigma_{10}}$	$j(\chi,\chi^2)^3$	$j(\chi^2,\chi^3)^{2+\sigma_{10}}$	
$\tau(\chi)^{19-\sigma_{19}}$	$=$	$j(\chi,\chi)^{9+4\sigma_2+2\sigma_4+\sigma_8}$	$j(\chi,\chi^2)^{1+\sigma_8}$		
$\tau(\chi)^{20-\sigma_{20}}$	$=$	$j(\chi,\chi)^{9+4\sigma_2+\sigma_4+\sigma_{13}}$	$j(\chi,\chi^2)^{\sigma_2+\sigma_4+\sigma_7+\sigma_{14}}$		
$\tau(\chi)^{22-\sigma_{22}}$	$=$	$j(\chi,\chi)^{10+4\sigma_2+2\sigma_4+\sigma_{11}}$	$j(\chi,\chi^2)^{2+2\sigma_8}$		
$\tau(\chi)^{23-\sigma_{23}}$	$=$	$j(\chi,\chi)^{11+5\sigma_2+3\sigma_4+\sigma_8}$	$j(\chi,\chi^2)^{1+\sigma_8}$		
$\tau(\chi)^{25-\sigma_{25}}$	$=$	$j(\chi,\chi)^{12+6\sigma_2+3\sigma_4+\sigma_8}$	$j(\chi,\chi^2)^{1+\sigma_8}$		
$\tau(\chi)^{26-\sigma_{26}}$	$=$	$j(\chi,\chi)^{12+6\sigma_2+2\sigma_4+\sigma_{13}}$	$j(\chi,\chi^2)^{2\sigma_4+2\sigma_{14}}$		
$\tau(\chi)^{27}$	$=$	$q^*\ j(\chi,\chi)^{12+6\sigma_2+2\sigma_4+\sigma_{13}}$	$j(\chi,\chi^2)^{2\sigma_4+2\sigma_{14}}$		

$\mathrm{ord}(\chi) = 32$					
$\tau(\chi)^{1-\sigma_1}$	$=$	1			
$\tau(\chi)^{3-\sigma_3}$	$=$		$j(\chi,\chi)$	$j(\chi,\chi^2)$	
$\tau(\chi)^{5-\sigma_5}$	$=$		$j(\chi,\chi)^2$	$j(\chi,\chi^2)$	$j(\chi^2,\chi^3)$
$\tau(\chi)^{7-\sigma_7}$	$=$		$j(\chi,\chi)^{2+\sigma_3}$	$j(\chi,\chi^2)^2$	$j(\chi^2,\chi^3)^{\sigma_{19}}$
$\tau(\chi)^{9-\sigma_9}$	$=$		$j(\chi,\chi)^{3+\sigma_3}$	$j(\chi,\chi^2)^{3+\sigma_3}$	
$\tau(\chi)^{11-\sigma_{11}}$	$=$		$j(\chi,\chi)^{4+\sigma_5}$	$j(\chi,\chi^2)^{2+\sigma_{21}}$	$j(\chi^2,\chi^3)^2$
$\tau(\chi)^{13-\sigma_{13}}$	$=$		$j(\chi,\chi)^{4+2\sigma_3}$	$j(\chi,\chi^2)^{4+\sigma_{19}}$	$j(\chi^2,\chi^3)^{\sigma_{19}}$
$\tau(\chi)^{15-\sigma_{15}}$	$=$		$j(\chi,\chi)^{6+\sigma_5}$	$j(\chi,\chi^2)^{3+\sigma_5}$	$j(\chi^2,\chi^3)^3$
$\tau(\chi)^{17-\sigma_{17}}$	$=$		$j(\chi,\chi)^{7+\sigma_5+\sigma_{15}}$	$j(\chi,\chi^2)^{3+\sigma_5}$	$j(\chi^2,\chi^3)^3$
$\tau(\chi)^{19-\sigma_{19}}$	$=$		$j(\chi,\chi)^{6+3\sigma_3+\sigma_{13}}$	$j(\chi,\chi^2)^{6+\sigma_{19}}$	$j(\chi^2,\chi^3)^{\sigma_{19}}$
$\tau(\chi)^{21-\sigma_{21}}$	$=$		$j(\chi,\chi)^{6+3\sigma_3+\sigma_7}$	$j(\chi,\chi^2)^{6+\sigma_7}$	$j(\chi^2,\chi^3)^{3\sigma_{19}}$
$\tau(\chi)^{23-\sigma_{23}}$	$=$		$j(\chi,\chi)^{7+3\sigma_3+\sigma_7+\sigma_9}$	$j(\chi,\chi^2)^{7+\sigma_3}$	$j(\chi^2,\chi^3)^{2\sigma_{19}}$
$\tau(\chi)^{25-\sigma_{25}}$	$=$		$j(\chi,\chi)^{10+2\sigma_5}$	$j(\chi,\chi^2)^{5+\sigma_5}$	$j(\chi^2,\chi^3)^{5+\sigma_5}$
$\tau(\chi)^{27-\sigma_{27}}$	$=$		$j(\chi,\chi)^{9+3\sigma_3+\sigma_9}$	$j(\chi,\chi^2)^{9+3\sigma_3+\sigma_9}$	
$\tau(\chi)^{29-\sigma_{29}}$	$=$		$j(\chi,\chi)^{9+5\sigma_3+\sigma_{13}}$	$j(\chi,\chi^2)^{9+2\sigma_{19}}$	$j(\chi^2,\chi^3)^{2\sigma_{19}}$
$\tau(\chi)^{31-\sigma_{31}}$	$=$		$j(\chi,\chi)^{13+2\sigma_5+\sigma_{15}}$	$j(\chi,\chi^2)^{6+2\sigma_5}$	$j(\chi^2,\chi^3)^6$
$\tau(\chi)^{32}$	$=$	q^*	$j(\chi,\chi)^{13+2\sigma_5+\sigma_{15}}$	$j(\chi,\chi^2)^{6+2\sigma_5}$	$j(\chi^2,\chi^3)^6$

BIBLIOGRAPHY.

This bibliography can be used in two directions. The numbers in [square] brackets in the main text refer to (alphabetically arranged) entries in the list below. Conversely, the numbers at the end of each entry refer to the Chapter(s) (I–VII) and Section(s) (1–11) in which it has been cited.

[1] L. M. Adleman, *On distinguishing prime numbers from composite numbers*, Proc. 21st Annual I.E.E.E. Symp. Found. Comp. Sci. (1980), 387–406. *I.9*

[2] L. M. Adleman, C. Pomerance and R. S. Rumely, *On distinguishing prime numbers from composite numbers*, Ann. of Math. **117** (1983), 173–206. *I.9, IV.1, V.2, V.3*

[3] L. M. Adleman, M. A. Huang, *Recognizing primes in random polynomial time*, Proc. 19th Annual ACM Symp. on Theory of Computing (1987), 462–469. *I.2*

[4] N. C. Ankeny, *The least quadratic non-residue*, Ann. of Math. **55** (1952), 65–72. *I.8*

[5] R. C. Archibald, *Mersenne's numbers*, Scripta Mathematica **3** (1935), 112–119. *I.3, I.7*

[6] A. O. L. Atkin, F. Morain, *Elliptic curves and primality proving*, preprint 1990. *I.10, VI.5*

[7] E. Bach, *Explicit bounds for primality testing and related problems*, Math. Comp. **55** (1990), 355–380. *I.8*

[8] R. Baillie, S. S. Wagstaff, *Lucas pseudoprimes*, Math. Comp. **35** (1980), 1391–1417. *I.7*

[9] C. B. Barker, *Proof that the Mersenne number M_{167} is composite*, Bull. Amer. Math. Soc. **51** (1945), 389. *I.7*

[10] Z. I. Borevich, I. R. Shafarevich, *Number Theory*, Orlando: Academic Press 1966. *I.5*

[11] W. Bosma, *Primality testing using elliptic curves*, Report 85–12 Mathematisch Instituut, Universiteit van Amsterdam (1985). *I.10*

[12] R. P. Brent, *Succinct proofs of primality for the factors of some Fermat numbers*, Math. Comp. **38** (1982), 253–255. *I.11*

[13] R. P. Brent, *Factorization of the eleventh Fermat number (preliminary report)*, Abstracts Amer. Math. Soc. **10** (1989), 89T-11-73. *I.6*

[14] R. P. Brent, J.M. Pollard, *Factorization of the eighth Fermat number*, Math. Comp. **36** (1981), 627–630. *I.6*

[15] R. P. Brent, *Algorithms for minimization without derivatives*, New Jersey: Prentice Hall 1973. *V*

[16] B. W. Brewer, *Tests for primality*, Duke Math. J. **18** (1951), 757–763. *I.7*

[17] J. Brillhart, M. A. Morrison, *A method of factoring and the factorization of* F_7, Math. Comp. **29** (1975), 183–205. *I.6*

[18] J. Brillhart, D. H. Lehmer, J. L. Selfridge, B. Tuckerman, and S. S. Wagstaff, Jr., *Factorizations of* $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$, *up to high powers*, Providence: A. M. S., Contemporary Mathematics **22** 1983. *I.6, I.7*

[19] J. Brillhart, J. L. Selfridge, *New primality criteria and factorizations of* $2^m \pm 1$, Math. Comp. **29** (1975), 620–647. *I.6, I.7*

[20] J. R. Bunch, J. E. Hopcroft, *Triangular factorization and inversion by fast matrix multiplication*, Math. Comp. **28** (1974), 231–236. *V.6*

[21] R. D. Carmichael, *On composite numbers P which satisfy the Fermat congruence* $a^{P-1} = 1 \bmod P$, Amer. Math. Monthly **19** (1912), 22–27. *I.6*

[22] P. A. Cataldi, *Trattato de Numeri Perfetti*, Bologna 1603. *I.3*

[23] J. Chernick, *On Fermat's simple theorem*, Bull. Amer. Math. Soc. **45** (1939), 269–274. *I.6*

[24] S. Chowla, *An extension of Heilbronn's class-number theorem*, Quart. J. Math. **5** (1934), 304-307. *I.5*

[25] S. Chowla and W. E. Briggs, *On discriminants of binary quadratic forms with a single class in each genus*, Canad. J. Math. **6** (1954), 463–470. *I.5*

[26] D. V. Chudnovsky, G. V. Chudnovsky, *Sequences of numbers generated by addition in formal groups and new primality and factorization tests*, Adv. in Appl. Math. **7** (1986), 187–237. *I.10*

[27] M. Cipolla, *Sui numeri composti P, che verificano la congruenza di Fermat* $a^{P-1} \equiv 1 \pmod{P}$, Ann. Mat. Pura Appl. (3) **9** (1904), 139–160. *I.6*

[28] Codex Lat. Monac., see: Bibliotheca Mathematica, (2) **9** (1895), 41. *I.7*

[29] H. Cohen, A. K. Lenstra, *Implementation of a new primality test*, Math. Comp. **48** (1987), 103–121. *I.2, I.9, III.3, IV.1, IV.5, V.2, VI.1, VI.5*

[30] H. Cohen, H. W. Lenstra, Jr., *Primality testing and Jacobi sums*, Math. Comp. **42** (1984), 297–330. *I.2, I.9, IV.1, V.2, VI.1, VI.5*

[31] F. N. Cole, *On the factoring of large numbers*, Bull. Amer. Math. Soc. **10** (1903), 134–137. *I.7*

[32] D. A. Cox, *Primes of the form* $x^2 + ny^2$, New York: John Wiley 1989. *I.5*

[33] Cullen and Cunningham, *Report*, British Association (1901). *I.5*

[34] DEC, *Machine Language Reference Manual for the DEC-3100*, 1988. *VI.1*

[35] L. E. Dickson, *Introduction to the theory of numbers*, Chicago: University of Chicago press 1929 (63–90). *I.5*

[36] J. D. Dixon, *Factorization and primality tests*, Amer. Math. Monthly **91** (1984), 333–352. *I.2*

[37] H. Dubner, *A new method for producing large Carmichael numbers*, Math. Comp. **53** (1989), 411–414. *I.6*

[38] H. Dubner, H. C. Williams, *The primality of* $R1031$, Math. Comp. **47** (1986), 703–711. *I.11, V.4*

[39] L. Euler, *Opera Omnia (1)*, **III** (1917), 336-337. *I.3, I.7*

[40] E. Fauquembergue, Sphinx Œdipe, **9** (1914), 85, 103–105. *I.7*

[41] A. Ferrier, *The determination of a large prime*, Mathematical Tables and other Aids to Computation **6** (1952), 256. *I.7*

[42] F. G. Frobenius, *Über quadratische Formen die viele Primzahlen darstellen*, in: Gesammelte Abhandlungen **III**, Berlin: Springer 1968. *I.5*

[43] M. R. Garey, D.S. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*, New York: Freeman 1979. *I.2, III.2, III.3, V*

[44] C. F. Gauss, *Disquisitiones Arithmeticae* (English edition), New York: Springer 1986. *I.4*

[45] D. B. Gillies, *Three new Mersenne primes and a statistical theory*, Math. Comp. **18** (1964), 93–97. *I.7*

[46] S. Goldwasser, J. Kilian, *Almost all primes can be quickly certified*, Proc. 18th Annual ACM Symp. on Theory of Computing (1986), 316–329. *I.10, VI.5*

[47] D. M. Gordon, *On the number of elliptic pseudoprimes*, Math. Comp. **52** (1989), 231–245. *I.10*

[48] N. A. Hall, *Binary quadratic discriminants with a single class of forms in each genus*, Math. Z. **44** (1938), 85-9-90. *I.5*

[49] N. A. Hall, *The number of representations function for binary quadratic forms*, Amer. J. Math. **62** (1940), 589–598. *I.5*

[50] G. H. Hardy, E. M. Wright, *An introduction to the theory of numbers (fourth edition)*, Oxford: Clarendon Press 1960. *V.2, V.3*

[51] H. Hasse, *Vorlesungen über Zahlentheorie*, Berlin: Springer, Grundlehren der mathematischen Wissenschaften **59** 1964. *II.3*

[52] A. Hurwitz, *New Mersenne primes*, Math. Comp. **16** (1962), 249–251. *I.7*

[53] A. Hurwitz, J. L. Selfridge, *Fermat numbers and Mersenne numbers*, Math. Comp. **18** (1964), 146–148. *I.6, I.7*

[54] I. M. S. L. Math. Library, *Fortran Subroutines for Mathematical Applications*, 1987. *VI.2*

[55] K. Inkeri, *Tests for primality*, Ann. Acad. Sc. Fenn. Ser. A I Math. **279** (1960), 1–18. *I.7*

[56] K. Ireland, M. Rosen, *A classical introduction to modern number theory*, New York: Springer 1982. *II.2, V.3*

[57] H. Iwaniec, M. Jutila, *Primes in short intervals*, Ark. Mat. **17** (1979), 167–176. *I.10*

[58] G. J. Janusz, *Algebraic number fields*, New York: Academic Press 1973. *II.2*

[59] J. P. Jones, D. Sato, H. Wada, D. Wiens, *Diophantine representation of the set of prime numbers*, Amer. Math. Monthly **83** (1976), 449–464. *I.2*

[60] J. S. Judd, H. C. Williams, *Determination of the primality of N by using factors of $N^2 \pm 1$*, Math. Comp. **30** (1976), 157–172. *I.7*

[61] E. Kaltofen, T. Valente and N. Yui, *An improved Las Vegas primality test*, Rensselaer report **89-12** (1989), 1–8. *I.10*

[62] I. Kaplansky, *Lucas's test for Mersenne numbers*, Amer. Math. Monthly **52** (1945), 188–190. *I.7*

[63] W. Keller, *Factors of Fermat numbers and large primes of the form $k2^n + 1$*, Math. Comp. **41** (1983), 661–673. *I.6*

[64] D. E. Knuth, *The art of computer programming*, vol. 2, *Seminumerical algorithms*, Reading: Addison-Wesley 1973. *V.1, V.6*

[65] G. C. Kurtz, D. Shanks and H. C. Williams, *Fast primality tests for numbers less than* $50 \cdot 10^9$, Math. Comp. **46** (1986), 691–701. *I.8*

[66] F. Landry, C. R. Acad. Sc. Paris, **91** (1880), 138. *I.6*

[67] E. Lawler, *Combinatorial optimization*, New York: Holt, Rinehart and Winston 1976. *III.2, III.3, V*

[68] D. J. Lehmann, *On primality tests*, SIAM J. Comput. **11** (1982), 374–375. *I.8*

[69] D. H. Lehmer, *Tests for primality by the converse of Fermat's theorem*, Bull. Amer. Math. Soc. **33** (1927), 327–340. *I.4*

[70] D. H. Lehmer, *A further note on the converse of Fermat's theorem*, Bull. Amer. Math. Soc. **34** (1928), 54–56. *I.6*

[71] D. H. Lehmer, *On the number* $10^{23} - 1/9$, Bull. Amer. Math. Soc. **35** (1929), 349–350. *I.4*

[72] D. H. Lehmer, *An extended theory of Lucas functions*, Ann. of Math. **31** (1930), 419–448. *I.7*

[73] D. H. Lehmer, *Note on Mersenne numbers*, Bull. Amer. Math. Soc. (1932), 383–384. *I.7*

[74] D. H. Lehmer, *On Lucas's test for the primality of Mersenne's numbers*, J. London Math. Soc. **10** (1935), 162–165. *I.7*

[75] D. H. Lehmer, *On the converse of Fermat's theorem*, Amer. Math. Monthly **43** (1936), 347–354.

[76] D. H. Lehmer, *On the converse of Fermat's theorem II*, Amer. Math. Monthly **56** (1949), 300–309.

[77] D. H. Lehmer, *Recent discoveries of large primes*, Mathematical Tables and other Aids to Computation **6** (1952), 61. *I.7*

[78] D. H. Lehmer, *A new Mersenne prime*, Mathematical Tables and other Aids to Computation **6** (1952), 205. *I.7*

[79] D. H. Lehmer, *Two new Mersenne primes*, Mathematical Tables and other Aids to Computation **7** (1953), 72. *I.7*

[80] D.H. Lehmer, *Strong Carmichael numbers* , J. Austral. Math. Soc. **21** (1976), 508–510. *I.8*

[81] A.K. Lenstra, Private communication, 1986. *VI.5*

[82] A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse and J.M. Pollard, *The number field sieve*, Proc. 22nd Annual Symp. on Theory of Computing (1990), 564–572. *I.2*

[83] A.K. Lenstra, H.W. Lenstra, Jr., *Algorithms in number theory*, in: j. van Leeuwen (ed.), *Handbook of Theoretical Computer Science (Vol. A), Algorithms and Complexity*, Amsterdam: Elsevier (1990), 673–715. *I.10, V.2*

[84] A.K. Lenstra, M.S. Manasse, *Complete factorization of the ninth Fermat number*, sci.math (electronic bulletin board) 1990. *I.6*

[85] H.W. Lenstra, Jr., *Miller's primality test*, Inform. Process. Lett. **8** (1979), 86–88. *I.8*

[86] H.W. Lenstra, Jr., *Primality testing algorithms (after Adleman, Rumely and Williams)*, exp. 576 in: Sém. Bourbaki **33** (1980/81), Berlin: Springer, Lecture Notes in Mathematics **901** (1981), 243–257. *I.9*

[87] H.W. Lenstra, Jr., *Galois theory and primality testing*, in: I. Reiner, K. Roggenkamp (eds.), *Orders and their applications*, Heidelberg: Springer, Lecture Notes in Mathematics **1142** (1985), 169–189. *I.9, II.2, V.4*

[88] H.W. Lenstra, Jr., *Divisors in residue classes*, Math. Comp. **42** (1984), 331–334. *I.11, IV.6, V.2, V.4*

[89] H.W. Lenstra, Jr., *Elliptic curves and number-theoretic algorithms*, Proc. International Congress of Mathematicians (1986), 99–120. *I.10*

[90] H.W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. **126** (1987), 649–673. *I.10*

[91] H.W. Lenstra, Jr., R. Tijdeman (eds.), *Computational methods in number theory*, Amsterdam: Mathematisch Centrum, Mathematical Centre Tracts **154/155**, 1982. *I.2, I.10*

[92] R. Long, *Algebraic number theory*, New York: Marcel Dekker 1977. *II.3*

[93] E. Lucas, *Note sur l'application des séries récurrentes à la recherche de la loi de distribution des nombres premiers*, C.R. Acad. Sc. Paris **82** (1876), 165–167. *I.7*

[94] E. Lucas, *Sur les rapports qui existent entre la théorie des nombres et le Calcul intégral*, C. R. Acad. Sc. Paris **82** (1876), 1303–1305. *I.7*

[95] E. Lucas, *Sur l'extension du théorème de Fermat généralisé, et du "Canon arithmeticus"*, C. R. Acad. Sc. Paris **84** (1877), 439–442. *I.7*

[96] E. Lucas, *Sur la division de la circonférence en parties égales*, C.R. Acad. Sc. Paris **85** (1877), 136–139. *I.6, I.7*

[97] E. Lucas, *Théorie des fonctions numériques simplement périodiques (I)*, Amer. J. Math. **1** (1878), 184–239. *I.7*

[98] E. Lucas, *Théorie des fonctions numériques simplement périodiques (II)*, Amer. J. Math. **1** (1878), 289–321. *I.7*

[99] E. Lucas, *Sur la recherche des grands nombres premiers*, Assoc. Française p. l'Avancement des Sciences **5** (1876), 61–68. *I.6*

[100] E. Lucas, *Théorie des Nombres*, Paris 1891. *I.7*

[101] Y. V. Matijasevic, *Primes are nonnegative values of a polynomial in 10 variables*, J. Soviet Math. **15** (1981), 33–44. *I.2*

[102] G. L. Miller, *Riemann's hypothesis and tests for primality* , J. Comput. and System Sc. **13** (1976), 300–317. *I.2, I.8*

[103] J. C. P. Miller, D. J. Wheeler, *Large prime numbers*, Nature **168** (1951), 838. *I.7*

[104] I. Miyamoto, M. Ram Murty, *Elliptic pseudoprimes*, Math. Comp. **53** (1989), 415–430. *I.10*

[105] L. Monier, *Evaluation and comparison of two efficient probabilistic primality testing algorithms*, Theoret. Comput. Sc. **12** (1980), 97–108. *I.8*

[106] H. L. Montgomery, *Topics in multiplicative number theory*, Berlin: Springer 1971. *I.8*

[107] P. Montgomery, *Modular multiplication without trial division*, Math. Comp. **44** (1985), 519–521. *VI.2*

[108] F. Morain, *Implementation of the Atkin-Goldwasser-Kilian primality testing algorithm*, INRIA-Rocquencourt: Rapport de Recherche **911** (1988), 1–101. *I.10, VI.1, VI.5*

[109] F. Morain, *Atkin's test: news from the front*, Proc. Eurocrypt '89 1989. *VI.1, VI.5*

[110] F. Morain, *Implementation of distributed primality proving and the primality of* $(2^{3539}+1)/3$, Preprint (1989), 1–13. *VI.1, VI.4*

[111] M. A. Morrison, *A note on primality testing using Lucas sequences*, Math. Comp. **29** (1975), 181-182. *I.7*

[112] L. Nickel, C. Noll, *The 25th and 26th Mersenne primes*, Math. Comp. **35** (1980), 1387–1390. *I.7*

[113] P. Pepin, *Sur la formule* $2^{2^n}+1$, C.R. Acad. Sc. Paris **85** (1877), 329–331. *I.6*

[114] I. M. Pervouchine, Bull. Acad. d. Sci. St. Pétersbourg, (3) **31** (1887), cols. 532–533. *I.7*

[115] J. Pintz, W. L. Steiger and E. Szemerédi, *Infinite sets of primes with fast primality tests and quick generation of primes*, Math. Comp. **53** (1989), 399–406. *I.11*

[116] D. A. Plaisted, *Fast verification, testing and generation of large primes*, Theoret. Comput. Sc. **9** (1979), 1–16. *I.11*

[117] G. A. A. Plana, R. Accad. d. Sc. Turin, Memorie (2) **20** (1863), 130. *I.7*

[118] H. C. Pocklington, *The determination of the prime or composite nature of large numbers by Fermat's theorem*, Proc. Cambridge Phil. Soc. **18** (1914–16), 29–30. *I.6*

[119] J. M. Pollard, *Theorems on factorization and primality testing*, Proc. Cambridge Phil. Soc. **76** (1974), 521–528. *I.2*

[120] C. Pomerance, *The distribution of pseudoprimes*, Math. Comp. **37** (1981), 587–593. *I.6*

[121] C. Pomerance, *Very short primality proofs*, Math. Comp. **48** (1987), 315–322. *I.2, I.11, V.5*

[122] C. Pomerance, J. L. Selfridge and S. S. Wagstaff, Jr., *The pseudoprimes to* $25 \cdot 10^9$, Math. Comp. **35** (1980), 1003–1026. *I.6, I.7, I.8*

[123] R. E. Powers, *The tenth perfect number*, Amer. Math. Monthly **18** (1911), 195–197. *I.7*

[124] R. E. Powers, Sphinx Œdipe, **9** (1914), 105–108. *I.7*

[125] V. R. Pratt, *Every prime has a succinct certificate*, SIAM J. Comput. **4** (1975), 214–220. *I.2, I.11, V.3, V.5*

[126] P. Pritchard, *Fast compact prime number sieves (among others)*, J. Algorithms **4** (1983), 332–344. *I.3, V.3*

[127] F. Proth, *Théorèmes sur les nombres premiers*, C.R. Acad. Sc. Paris **87** (1878), 926. *I.6*

[128] M. O. Rabin, *Probabilistic algorithms for testing primality*, J. Number Theory **12** (1980), 128–138. *I.8, II.1*

[129] P. Ribenboim, *The book of prime number records*, New York: Springer 1988. *V.3*

[130] H. Riesel, *A new Mersenne prime*, Mathematical Tables and other Aids to Computation **12** (1958), 60. *I.7, V*

[131] H. Riesel, *Lucasian criteria for the primality of* $N = h2^n - 1$, Math. Comp. **23** (1969), 869–875. *I.7*

[132] H. Riesel, *Prime numbers and computer methods for factorization*, Boston: Birkhäuser, Progr. Math. **57** 1985. *I.6, I.7, II.5, V.3, VII.2*

[133] R. M. Robinson, *Mersenne and Fermat numbers*, Proc. Amer. Math. Soc. **5** (1954), 842–846. *I.7*

[134] R. M. Robinson, *A report on primes of the form* $k2^n + 1$ *and on factors of Fermat numbers* , Proc. Amer. Math. Soc. **9** (1958), 673–681. *I.6*

[135] J. B Rosser, *A method of computing exact inverses of matrices with integer coefficients*, J. Res. Nat. Bur. Standards **49** (1952), 349–358. *V.6*

[136] A. Rotkiewicz, *On the pseudoprimes with respect to the Lucas sequence*, Bull. Ac. Pol. Sc. **21** (1973), 793–797. *I.7*

[137] H. G. Ruck, *A note on elliptic curves over finite fields*, Math. Comp. **49** (1987), 301–304. *I.10*

[138] R. Rumely, *Recent advances in primality testing*, Notices Amer. Math. Soc. **30** (1983), 475–477. *I.9*

[139] P. Samuel, *Théorie algébrique des nombres*, Paris: Hermann 1971 (2). *II.4*

[140] A. Schönhage, V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing **7** (1971), 281–292. *I.2, V.1*

[141] R. Schoof, *Elliptic curves over finite fields and the computation of square roots mod* p, Math. Comp. **43** (1985), 483–494. *I.10*

[142] R. Schoof, *Non-singular plane cubic curves over finite fields*, J. Combin. Theory (Ser. A) **46** (1987), 183–211. *I.10*

[143] A. Shamir, *Factoring numbers in $O(\log n)$ arithmetic steps*, Inform. Process. Lett. **8** (1979), 28–31. *I.2*

[144] C. L. Siegel, *Über die Classenzahl quadratischer Zahlkörper*, Acta Arith. **1** (1936), 83–86. *I.5*

[145] J. H. Silverman, *The arithmetic of elliptic curves*, New York: Springer 1986. *I.10*

[146] D. Slowinski, *Searching for the 27th Mersenne prime*, J. Recr. Math. **11** (1978), 258–261. *I.7*

[147] R. Solovay, V. Strassen, *A fast Monte-Carlo test for primality*, SIAM J. Comput. **6** (1977), 84–85, erratum, ibid., **7** (1978), 118. *I.8*

[148] A. J. Stephens, H. C. Williams, *An open architecture number sieve*, in: J.H. Loxton (ed.), *Number Theory and Cryptography*, Cambridge: Cambridge University Press, London Math. Soc. Lecture Note Ser. **154** (1990), 38–75. *I.3*

[149] V. Strassen, *Einige Resultate über Berechnungskomplexität*, Jahresber. Deutsch. Math.-Verein **78** (1976/77), 1–8. *I.2*

[150] SUN, *SUN-4 Assembly Language Reference Manual*, 1988. *VI.1*

[151] J. D. Swift, *Note on discriminants of binary quadratic forms with a single class in each genus*, Bull. Amer. Math. Soc. **54** (1948), 560–561. *I.5*

[152] J. D. Swift, *Table of Carmichael numbers to* 10^9, Math. Comp. **29** (1975), 338–339. *I.6*

[153] B. Tuckerman, *The 24th Mersenne prime*, Proc. Nat. Acad. Sci. **68** (1971), 2319–2320. *I.7*

[154] H. S. Uhler, *A brief history of the investigations on Mersenne numbers and the latest immense primes*, Scripta Mathematica **18** (1952), 122–131. *I.7*

[155] J. F. Voloch, *A note on elliptic curves over finite fields*, Bull. Soc. Math. France **116** (1988), 455–458. *I.10*

[156] S. S. Wagstaff, Jr., *Large Carmichael numbers*, Math. J. Okayama Univ. **22** (1980), 33–41. *I.6*

[157] S. S. Wagstaff, Jr., *Divisors of Mersenne numbers*, Math. Comp. **40** (1983), 385–397.

[158] W. C. Waterhouse, *Abelian varieties over finite fields*, Ann. scient. Ec. Norm. Sup. (4) **2** (1969), 521–560. *I.10*

[159] A. Weil, *Number theory, an approach through history*, Boston: Birkhäuser 1984. *I.5*

[160] P. J. Weinberger, *Exponents of the class groups of complex quadratic fields*, Acta Arith. **22** (1973), 117–124. *I.5*

[161] A. E. Western, *On Lucas's and Pepin's test for the primeness of Mersenne's numbers*, J. London Math. Soc. **7** (1932), 130–137. *I.7*

[162] H. C. Williams, *On numbers analogous to the Carmichael numbers*, Canad. Math. Bull. **20** (1977), 133–143. *I.7*

[163] H. C. Williams, *Primality testing on a computer*, Ars Combinatoria **5** (1978), 127–185. *I.3, I.6*

[164] H. C. Williams, *A class of primality tests for trinomials which includes the Lucas-Lehmer test*, Pacific J. Math. **98** (1982), 477–494. *I.7*

[165] H. C. Williams, *Some primes with interesting digit patterns*, Math. Comp. **32** (1978), 1306–1310. *I.11*

[166] M. Yorinaga, *Numerical computation of Carmichael numbers I*, Math. J. Okayama Univ. **20** (1978), 151–163. *I.6*

[167] M. Yorinaga, *Numerical computation of Carmichael numbers II*, Math. J. Okayama Univ. **21** (1979), 183–205. *I.6*

[168] M. Yorinaga, *Carmichael numbers with many prime factors*, Math. J. Okayama Univ. **22** (1980), 169–184. *I.6*

[169] J. Young, D.A. Buell, *The twentieth Fermat number is composite*, Math. Comp. **50** (1988), 261–265. *I.6*

[170] S. E. Zarantonello, J. Brown, L. C. Noll, B. K. Parady, G. W. Smith, and J. F. Smith, *Letter to the editor*, Amer. Math. Monthly **97** (1990), 214. *I.7*

LIST OF SYMBOLS.

This list of symbols consists of three parts. The first list contains the symbols denoting the parameters for the primality test. The second one contains most of the *Latin* symbols from the text in alphabetical order. The third list comprises the *Greek* symbols that are most frequently used.

Looking up a symbol in the lists below may help clarify its meaning in two ways. Either the summarized, informal definition in the second column may help the reader, or helpful information should be found on one of the pages referred to in the third column.

Numbers refer to a page where the symbol is defined, where it occurs for the first time, or where it plays an important role. For important symbols that are not used very frequently, all occurrences have been listed.

Parameters.

The symbols listed in this part occur very frequently, some on virtually every page. The summary below is intended to provide a superficial idea of the role they play in the algorithm, and to indicate the pages where more precise information can be found. The introductions to Chapter III (110–113), Chapter IV (142–144) and Chapter V (188–197) also contain an overview of the meaning of most of the parameters below. The example run on pages 274–282 may be enlightening too.

n	the odd positive integer to be tested for primality	*2–285*
p^k	a power of the small prime p, dividing t or v; if p^k divides v the existence of a p^k-th root of unity must be established (a Lucas-Lehmer test), but if it divides t, also a Jacobi sum test for a character of order p^k must be performed	*90, 114, 145–146, 157–180, 198–200, 209–211*
q	a prime dividing s and the conductor of a character in the Jacobi sum test; $q-1$ is built up from primes p dividing t	*90, 114, 157–180, 198–200*
s	the product of primes q; it divides s_0 and should be large: to complete the test we need roughly $sv > \sqrt{n}$	*90, 122–129, 157–180, 198–200*
s_0	the product of all primes q for which $q-1$ is built up from primes p in t_0	*142, 145, 186*
s_1	maximal factor of s that is coprime to t	*90, 93, 181*
t	integer built up from powers of small primes p; it divides t_0 and should not be too large since the number of trial divisions in the final stage is proportional to t	*60, 90–92, 130–132*
t_0	integer that is chosen in advance and determines the size of s_0 and hence the size of the integers n that can be tested by (the Jacobi sum part of) the test	*91, 130, 142, 145, 198–200*
t_1	maximal divisor of $n^u - 1$ that is built up from primes in t only	*88–91, 93, 181*
u	the multiplicative order 0f n modulo t; determines the total degree of the extension in which all Jacobi sum tests could take place (in fact they will usually take place in smaller extensions, with degrees built up from powers of primes l in u)	*60, 77, 93, 110–113, 157–180*
u_0	multiplicative order of n modulo t_0	*130, 156, 186*
v	the product of all prime factors found in $n^w - 1$; forms the Lucas-Lehmer contribution towards reaching $sv > \sqrt{n}$ (compare s above)	*93, 133–139, 154, 156*
w	total degree of the extension in which all Lucas-Lehmer tests could be performed (in fact they will usually take place in sub-extensions, compare u above)	*93, 133–139, 154, 156, 169*

Latin symbols.

a, b	coefficients in Weierstrass form of elliptic curve	*43–45*
a, b	pair of integers with sum π representing a Jacobi sum $J(\chi^a, \chi^b)$	*105–106, 149–152, 203–207*
a_i, b_i, c_i	integer sequences connected with Euclidean-like algorithm	*108, 182–183, 214*
A	ring with 1	*64–69, 98*
A	lower bound for factor search	*136–137, 159–161*
B	upper bound for factor search	*136–137, 154, 159–161, 167–169, 198, 241–246*
$\mathbf{C}$	field of complex numbers	*148, 171*
$c_d(a, b)$	machine dependent cost function for division of integers of length a and b	*160, 166*
$c_f(a)$	machine dependent cost function for final trial division for integers of length a	*166*
$c_m(a)$	machine dependent cost function for multiplication of integers of length a	*166*
$c_n(a)$	machine dependent cost function for modular multiplication of two integers	*160, 166*
c_1	first cost function for conductors q	*127, 158, 165*
c_2	second cost function for conductors q	*127, 165*
cond	conductor of a character	*65*
$d(x)$	degree function	*115–125*
D	discriminant of an elliptic curve	*42–45*
D	denominator of transition matrix S	*147–148, 170–174, 287, 302–312*
det	determinant of a matrix	*61*
e^x	exponential function	*8, 148, 171*
$e(t)$	maximal modulus for given exponent	*89–92, 123, 130, 146, 194, 287–300*
e_i	recurrent sequence in Lucas-Lehmer test for Mersenne numbers	*27, 82*
$e_{\pi,p^k,i}$	exponents in the product of Jacobi sums expressing Gauss sums	*149–152, 178, 203*
$E_{a,b}$	elliptic curve with coefficients a and b	*42–48*
exp	the exponent of a group	*64*
f	minimal polynomial	*73, 147, 170–171, 288, 302–312*
$\mathcal{F}_\omega$	set of prime factors of $n^\omega - 1$	*154–156, 168*
$\mathbf{F}_q$	finite field of q elements	*43–44, 59–60*
g	(almost) primitive root	*147, 152, 170–171, 287, 302–312*
G	group	*64*
G	product of integers for which gcd with n must be determined	*154, 172, 181*
$G(u)$	cost of multiplying ring elements on u coordinates	*115–127*
Gal	Galois group	*57*
gcd	greatest common divisor	*84*
Hom	group of homomorphisms	*64, 85–86*
$\bar{I}$	set of orders of characters to be combined	*124–129*
id	identity homomorphism	*81*
index	index of a subring in a ring	*73*
$\bar{J}$	set of conductors of characters to be combined	*124–129*
$\mathcal{J}$	set of primes representing Jacobi sums	*149–152, 203*
$J(\chi_1, \chi_2)$	Jacobi sum	*98–105, 149–153, 203*
ker	kernel	*67, 70, 93*
$k(p), k_p$	number of factors p (in a product)	*2, 145, 174–177*
K	number field	*43, 57–97*

l	prime divisor of the extension degree u	*73, 77, 93, 158–162*
lcm	least common multiple	
L	L-series	*7, 17, 37*
L	number field; in particular the cyclic field in a cyclotomic constellation	*60, 111*
$\mathcal{L}_0$	set of primes from which u_0 is built up	*145*
$\mathcal{L}^+$	set of prime powers for which suitable extension is found	*158, 162–164*
$\mathcal{L}^-$	set of prime powers for which suitable extension is not found	*158, 162–164*
$L(n)$	sub-exponential function	*8–9*
log	natural logarithm	
$\log_2$	logarithm to the base 2	
$\log_{10}$	logarithm to the base 10	
m	conductor of field extension	*73–79, 147–149, 171–175, 201–202, 287, 302–312*
$\mathcal{M}(u)$	set of conductors of field extensions used	*146–149*
$\mathcal{M}^+(l^e)$	set of suitable conductors for degree l^e extensions	*161–164*
mod	modulo	
n	integer to be tested for primality (see also above)	
N	upper bound on integers n to be tested for primality	*186, 198–208, 266*
N	norm	*24*
$\mathbf{N}$	ideal norm	*43*
$\mathrm{N}_{L/K}$	relative norm between number fields	*59, 70*
NP	complexity class of non-deterministic polynomial time problems	*6*
$o_p(t)$	number of prime factors p in t	*91, 146*
ord	the order of an element in a group	*57*
O_Δ	ring of integers of quadratic field $\mathbf{Q}(\sqrt{\Delta})$	*24–32, 47*
O_E	zero point on an elliptic curve	*43*
O_K (O_L)	ring of integers of the number field K (L)	*43, 59–87*
O_L/nO_L	residue class ring	*57*
$O_L/nO_L[\zeta_m]$	ring in which Gauss sums are defined	*85–90*
$o(g(x))$	complexity bound	*187*
$O(f(x))$	complexity bound $Cf(x)$	*5, 186*
p	prime number (see above under p^k)	*90, 114, 145–146, 157–180, 198–200, 209–211*
$\mathcal{P}$	set of primes from which t is built up	*145, 158, 163–167*
$\mathcal{P}_0$	set of primes from which t_0 is built up	*145, 199–200*
P	complexity class of polynomial time problems	*7*
q	prime conductor (see also above)	*90, 114, 157–180, 198–200*
$\mathbf{Q}$	field of rational numbers	
$\mathcal{Q}$	set of primes q from which s is built up	*145, 158, 163–167*
$\mathcal{Q}_0$	set of primes from which s_0 is built up	*145, 199–200*
$\mathbf{Q}(\sqrt{\Delta})$	quadratic field extension of $\mathbf{Q}$	*15–17, 24–25, 47*
r	divisor of n	*84–91, 93, 107, 181*
$\mathbf{R}$	field of real numbers	
s	modulus of definition of a character (see also above)	*65, 90, 122–129, 157–180, 198–200*
s_0	see above	*142, 145, 188*
s_1	see above	*90, 93, 181*

$s(u)$	size function of knapsack	*123, 125*
S	transition matrix	*147–149, 171–175, 288, 302–312*
S^*	inverse of transition matrix	*147–149, 171–175, 288, 302–312*
t	see above	*60, 90–92, 130–132*
t^∞	sufficiently large product of primes dividing t	*84, 123*
t_0	see above	*91, 130, 142, 145, 188, 198–201*
t_1	see above	*88–91, 93, 181*
t'	integer built up from the primes in t	*62, 85, 88*
$T(n)$	time needed to complete the primality test for n	*159–162, 189–192*
$\bar{T}$	upper bound on the time for completing the primality test	*157–161*
$\mathcal{T}$	set of triples representing Jacobi sum tests	*157–158, 163–167*
$t_d(a,b)$	time needed for division of integers of length a and b	*227–240*
$t_f(d)$	time needed for final trial division for integers of length d	*227–240*
$t_m(a,b)$	time needed for multiplication of integers of length a and b	*227–240*
$t_n(a)$	time needed for modular multiplication	*227–240*
$\mathrm{Tr}_{L/K}$	relative trace between number fields	*71*
u	extension degree (see also above)	*60, 77*
u_0	multiplicative order of n modulo t_0	*130, 156, 186*
$u_{p,k}$	multiplicative order of n modulo p^k	*131, 155*
v	the Lucas-Lehmer factors found	*93, 133–139, 154, 156*
$v(u)$	value function of knapsack	*123, 125*
v_ω	factor in extension of degree ω	*133–135, 154–156, 168–169*
w	extension degree for Lucas-Lehmer test (see also above)	*93, 133–139, 154, 156, 169*
W	upper bound for extension degrees w	*136–137, 158–161, 167–169, 241–246*
W_h	combination of Jacobi sum tests	*115–127*
X	set of characters, containing generators for $\mathrm{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle)$	*93, 97, 114–124*
$\bar{X}$	set of characters to be combined	*124–129*
Y	set of characters generating $\mathrm{Hom}((\mathbf{Z}/s\mathbf{Z})^*, \langle\zeta\rangle)$	*88, 90*
z	unit in O_L/nO_L	*86, 88, 90, 93, 100*
$\mathbf{Z}$	ring of rational integers	
$\mathbf{Z}_{\geq k}$	integers greater than or equal k	
$\mathbf{Z}[G]$	group ring	*71–72*
$\mathbf{Z}[i]$	ring of Gaussian integers	*44–45*
$\mathbf{Z}[\alpha]$	ring generated by $\mathbf{Z}$ and α	*25, 73*
$\mathbf{Z}/m\mathbf{Z}$	residue class ring of integers modulo m	
$(\mathbf{Z}/m\mathbf{Z})^*$	units in $\mathbf{Z}/m\mathbf{Z}$	
$\mathbf{Z}/n\mathbf{Z}[\sqrt{\Delta}]$	ring of n or n^2 elements	*25*
$\mathbf{Z}/n\mathbf{Z}[\eta]$	subring of O_L/nO_L, where $L = \mathbf{Q}(\eta)$	*72–77*

Greek symbols.

α_p	non-p-th power in O_L/nO_L	*78–80, 175–178*
β_p	p-th root of unity minus 1 in O_L/nO_L	*78–79, 175–178*
γ_p	p^k-th root of unity in O_L/nO_L	*78–79, 175–178, 181*
Δ	discriminant of a quadratic number field	*24–32, 47*
Δ_L	discriminant of a number field L	*60, 70*
$\Delta(f)$	discriminant of a minimal polynomial	*73*
ϵ	arbitrarily small postive real number	*8, 187*
ζ	root of unity; in particular in a cyclotomic constellation	*60, 64–69, 78, 80, 111*
ζ_m	primitive m-th root of unity	*38, 57*
η	generator of cyclic field	*72–79, 147–148, 172–174*
η_u	generator of cyclic field of degree u	*72–79, 148*
$\Theta(g(x))$	complexity bound	*187*
$\lambda(n)$	Carmichael lambda function	*19, 145, 198*
$\mu(n)$	Möbius function	*67*
μ	the lower bound on sv to complete the test for n is n^μ	*142, 159–165, 216, 241–246*
μ_χ	86	*88, 93*
π	prime in $\mathcal{J}$	*149–152, 203–207*
$\pi(x)$	prime counting function	*46, 189*
ρ	exponent of multiplication	*187*
ρ_0		*120*
σ	automorphism; in particular in a cyclotomic constellation	*24–31, 57, 60, 111*
ς	basis element	*148, 172–174*
$\tau(\chi)$	Gauss sum	*39, 66, 98–105, 149–152, 203–206*
$\tau^{n-\phi_n}$	(exponential notation for action on Gauss sum)	*85*
$\tau_a(\chi)$	Gauss sum	*66–69*
$\phi(n)$	Euler phi function	*19, 77, 143, 196*
ϕ_n	Artin symbol	*57, 61, 75, 78, 85, 90, 83*
Φ_t	t-th cyclotomic polynomial	*57, 157*
χ	character	*39, 64*
χ_p	component of a character	*65, 73*
X	set of characters	*70*
ω	extension degree for Lucas-Lehmer test	*133–135, 139, 155–157, 169*
Ω	set of extension degrees chosen for Lucas-Lehmer part of test	*133–135, 155–157, 170*
$\Omega(g(x))$	complexity bound	*187*
$\bar{\Omega}$	set of extension degrees allowed for Lucas-Lehmer tests	*133–135*

INDEX.

Slanted page numbers refer to definitions.

SAMENVATTING.

Primaliteit bewijzen met cyclotomie

Wanneer een positief geheel getal priem is, laat zich dat doorgaans met aan zekerheid grenzende waarschijnlijkheid vaststellen met behulp van een samengesteldheidstest. Echte zekerheid verkrijgt men echter pas door een bewijs voor primaliteit te geven, en daartoe past men een primaliteitstest toe: een algoritme waarmee primaliteitsbewijzen gegenereerd kunnen worden.

In 1981 hebben Adleman, Rumely en Pomerance een nieuwe primaliteitstest gepubliceerd, waarvan de berekeningscomplexiteit weliswaar niet polynomiaal in de lengte van de invoer is, maar die efficiënter was dan alle voorgaande methoden om primaliteit te bewijzen. Nadere analyse van de voorgestelde methode bracht H. Cohen en H. W. Lenstra, Jr., er toe theoretische verbeteringen te suggereren die later door H. Cohen en A. K. Lenstra zijn verwerkt in een computerprogramma. De resulterende Jacobisom-test is een algemene methode. Hiermee wordt bedoeld, dat de test niet afhankelijk is van speciale eigenschappen van het priemgetal voor het genereren van een bewijs. De benodigde rekentijd voor het verkrijgen van een primaliteitsbewijs is vrijwel uitsluitend afhankelijk van de orde van grootte van het priemgetal. Dit in tegenstelling tot de speciale methoden, zoals de tests van het Lucas-Lehmer type, die de bijzondere eigenschappen van een kleine klasse van priemgetallen zeer efficiënt uitbuiten.

In dit proefschrift is een groot aantal verbeteringen aangebracht aan bovengenoemde Jacobisom-test. Na het oplossen van een aantal, zowel vanuit het oogpunt van de algebraïsche getaltheorie als vanuit algoritmisch oogpunt, interessante problemen, werd een aanmerkelijk verbeterde methode verkregen, die gezien mag worden als de snelste, thans bekende, algemene primaliteitstest. Met deze methode is in de praktijk de primaliteit van getallen van meer dan 1000 decimalen vastgesteld. Asymptotisch is de complexiteit van de algoritme dezelfde als die van de vroegere Jacobisom-test. In het bijzonder betekent dit dat de methode (nog steeds) niet polynomiaal is in de lengte van de invoer.

Dit proefschrift bevat zeven hoofdstukken gevolgd door een appendix. Het eerste hoofdstuk geeft een historisch overzicht over primaliteitstests. Het tweede hoofdstuk behandelt de wiskundige theorie die aan de methode ten grondslag ligt. In het bijzonder wordt bewezen dat de condities waaraan een getal moet voldoen teneinde de test te doorstaan, primaliteit garanderen. De keuze van de parameters in de primaliteitstest beïnvloedt in

hoge mate de tijd die nodig is om het bewijs te voltooien. In het derde hoofdstuk wordt ingegaan op de problemen die optreden in dit optimalisatieprobleem, en worden de gekozen oplossingen gepresenteerd. Hoofdstuk vier bevat een gedetailleerde beschrijving van de algoritme. Op basis van deze beschrijving is een computerprogramma gemaakt, dat in staat is een primaliteitsbewijs te genereren. Het vijfde hoofdstuk bevat een analyse van de complexiteit van de algoritme, alsmede enige heuristieken die in deze analyse van belang zijn. Hoofdstuk zes geeft een overzicht van de resultaten die kunnen worden bereikt met de methode. Het zevende hoofdstuk bestaat uit een beknopte handleiding voor de installatie en het gebruik van het computerprogramma. De appendix bevat een drietal tabellen, die gebruikt worden door de algoritme. Bovendien is een lijst van symbolen, een index, en een uitgebreide bibliografie toegevoegd.