



Basic Research in Computer Science

Efficient Algorithms for gcd and Cubic Residuosity in the Ring of Eisenstein Integers

Ivan B. Damgård
Gudmund Skovbjerg Frandsen

BRICS Report Series

RS-03-8

ISSN 0909-0878

February 2003

Copyright © 2003,

**Ivan B. Damgård & Gudmund Skovbjerg
Frandsen.**

**BRICS, Department of Computer Science
University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK-8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

**BRICS publications are in general accessible through the World Wide
Web and anonymous FTP through these URLs:**

**<http://www.brics.dk>
<ftp://ftp.brics.dk>
This document in subdirectory RS/03/8/**

Efficient algorithms for gcd and cubic residuosity in the ring of Eisenstein integers *

Ivan Bjerre Damgård Gudmund Skovbjerg Frandsen

BRICS[†]

Department of Computer Science

University of Aarhus

Ny Munkegade

DK-8000 Aarhus C, Denmark

{ivan,gudmund}@brics.dk

February, 2003

Abstract

We present simple and efficient algorithms for computing gcd and cubic residuosity in the ring of Eisenstein integers, $\mathbf{Z}[\zeta]$, i.e. the integers extended with ζ , a complex primitive third root of unity. The algorithms are similar and may be seen as generalisations of the binary integer gcd and derived Jacobi symbol algorithms. Our algorithms take time $O(n^2)$ for n bit input. This is an improvement from the known results based on the Euclidean algorithm, and taking time $O(n \cdot M(n))$, where $M(n)$ denotes the complexity of multiplying n bit integers. The new algorithms have applications in practical primality tests and the implementation of cryptographic protocols. The technique underlying our algorithms can be used to obtain equally fast algorithms for gcd and quartic residuosity in the ring of Gaussian integers, $\mathbf{Z}[i]$.

*Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

[†]Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

1 Introduction

The Eisenstein integers, $\mathbf{Z}[\zeta] = \{a + b\zeta \mid a, b \in \mathbf{Z}\}$, is the ring of integers extended with a complex primitive third root of unity, i.e. ζ is root of $x^2 + x + 1$. Since the ring $\mathbf{Z}[\zeta]$ is a unique factorisation domain, a greatest common divisor (gcd) of two numbers is welldefined (up to multiplication by a unit). The gcd of two numbers may be found using the classic Euclidian algorithm, since $\mathbf{Z}[\zeta]$ is an Euclidian domain, i.e. there is a norm $N(\cdot) : \mathbf{Z}[\zeta] \setminus \{0\} \mapsto \mathbf{N}$ such that for $a, b \in \mathbf{Z}[\zeta] \setminus \{0\}$ there is $q, r \in \mathbf{Z}[\zeta]$ such that $a = qb + r$ with $r = 0$ or $N(r) < N(b)$.

When a gcd algorithm is directly based on the Euclidian property, it requires a subroutine for division with remainder. For integers there is a very efficient alternative in the form of the binary gcd, that only requires addition/subtraction and division by two [10]. A corresponding Jacobi symbol algorithm has been analysed as well [9].

It turns out that there is a natural generalisations of these binary algorithms over the integers to algorithms over the Eisenstein integers for computing the gcd and the cubic residuosity symbol. The role of 2 is taken by the number $1 - \zeta$, which is a prime of norm 3 in $\mathbf{Z}[\zeta]$.

We present and analyse these new algorithms. It turns out that they both have bit complexity $O(n^2)$, which is an improvement over the so far best known algorithms by Scheidler and Williams [7], Williams [11], Williams and Holte [12]. Their algorithms have complexity $O(nM(n))$, where $M(n)$ is complexity of integer multiplication.

1.1 Related work

Scheidler and Williams algorithms may be seen as generalisations of the classic Euclidian gcd and Jacobi algorithms rather than the binary algorithms. However, in the case of integer gcd and Jacobi symbol algorithms, both the Euclidian based and the binary versions have the same asymptotic complexity $O(n^2)$ (in their simple versions - there is an asymptotically faster version [8, 1]). This raises the question of why the Scheidler-Williams generalisation does not seem to achieve the complexity $O(n^2)$ as well.

For the classic Euclidian algorithm to be fast two observations about the division step (find remainder r from dividing a with b) are essential.

The number of iterations depends on finding a remainder r with a small norm in each step, preferably only a fraction of the dividend. Williams [11] obtained $N(r) < \frac{3}{4}N(b)$, whereas Scheidler and Williams

showed how to find r satisfying $N(r) < \frac{1}{3}N(b)$. This was the basis for their algorithms for gcd and cubic residuosity taking time corresponding to $O(\log N(ab))$ multiplications.

Each step in the Euclidian algorithm seems to require $O(1)$ multiplications (divisions). Over the integers, however, the combined complexity of these arithmetic operations over all iterations sum up to only $O(\log a)(\log b))$, due to properties of integer division in the standard binary radix representation of the integers [1]. If one could have an analogue radix representation of elements in $\mathbf{Z}[\zeta]$, a similar result might hold here. Both Knuth [3] and Penney [6] describe radix representations of the complex numbers (and hence the Gaussian integers, $\mathbf{Z}[i]$). It seems their representations could be generalised to $\mathbf{Z}[\zeta]$ (see also [5]), but Knuth comments on the arithmetic in his representation that *Examples of long division appear to require a stroke of genius when trial divisors are being picked.* I.e. so far it is not clear how to obtain gcd and cubic residuosity algorithms for $\mathbf{Z}[\zeta]$ that are based on the traditional Euclidian algorithm and yet obtains a time complexity comparable to our new algorithms.

1.2 Applications

Our algorithms may be used for the efficient computation of cubic residuosity in other rings than $\mathbf{Z}[\zeta]$ when using an appropriate homomorphism. As an example, consider the finite field $GF(p)$ for prime $p \equiv 1 \pmod{3}$. A number $z \in \{1, \dots, p-1\}$ is a cubic residue precisely when $z^{(p-1)/3} \equiv 1 \pmod{p}$, implying that (non)residuosity may be decided by a (slow) modular exponentiation. However, it is possible to decide cubic residuosity much faster provided we make some preprocessing depending only on p . The preprocessing consists in factoring p over $\mathbf{Z}[\zeta]$, i.e. finding a prime $\pi \in \mathbf{Z}[\zeta]$ such that $p = \pi\bar{\pi}$. A suitable π may be found as $\pi = \gcd(p, r - \zeta)$, where $r \in \mathbf{Z}$ is constructed as a solution to the quadratic equation $x^2 + x + 1 = 0 \pmod{p}$. Following this preprocessing cubic residuosity of any z is decided using that $z^{(p-1)/3} \equiv 1 \pmod{p}$ if and only if $[z/\pi] = 1$, where $[\cdot/\cdot]$ denotes the cubic residuosity symbol.

When the order of the multiplicative group in question is unknown, modular exponentiation cannot be used, but it may still be possible to identify some nonresidues by computing residuosity symbols. In particular, the primality test of Damgård and Frandsen [2] uses our algorithms for finding cubic nonresidues in a more general ring.

Computation of gcd and cubic residuosity is also used for the implementation of cryptosystems by Scheidler and Williams [7], and by

Williams [11].

Our fast algorithms are based on generalising the technique used in the binary gcd algorithm. The same generalisation does in fact also work for computing gcd and quartic residuosity in $\mathbf{Z}[i]$, the ring of Gaussian integers. The last section outlines this approach.

2 Preliminary facts about $\mathbf{Z}[\zeta]$

$\mathbf{Z}[\zeta]$ is the ring of integers extended with a primitive third root of unity ζ (complex root of $z^2 + z + 1$). We will be using the following definitions and facts [7].

Define the two conjugate mappings $\sigma_i : \mathbf{Z}[\zeta] \mapsto \mathbf{Z}[\zeta]$ by $\sigma_i(\zeta) = \zeta^i$ for $i = 1, 2$. The rational integer $N(\alpha) = \sigma_1(\alpha)\sigma_2(\alpha) \geq 0$ is called the norm of $\alpha \in \mathbf{Z}[\zeta]$, and $N(a + b\zeta) = a^2 + b^2 - ab$. (Note that $\sigma_2(\cdot)$ and $N(\cdot)$ coincides with complex conjugation and complex norm, respectively).

A unit in $\mathbf{Z}[\zeta]$ is an element of norm 1. There are 6 units in $\mathbf{Z}[\zeta]$: $\pm 1, \pm \zeta, \pm \zeta^2$. Two elements $\alpha, \beta \in \mathbf{Z}[\zeta]$ are said to be associates if there exists a unit ϵ such that $\alpha = \epsilon\beta$.

A prime π in $\mathbf{Z}[\zeta]$ is a non unit such that for any $\alpha, \beta \in \mathbf{Z}[\zeta]$, if $\pi|\alpha\beta$, then $\pi|\alpha$ or $\pi|\beta$.

$1 - \zeta$ is a prime in $\mathbf{Z}[\zeta]$ and $N(1 - \zeta) = 3$. A primary number has the form $1 + 3\beta$ for some $\beta \in \mathbf{Z}[\zeta]$. If $\alpha \in \mathbf{Z}[\zeta]$ is not divisible by $1 - \zeta$ then α is associated to a primary number. (This definition of *primary* is stronger than the usual one, which just requires the form $\pm 1 + 3\beta$, but our definition is more convenient in the present context).

If π is a prime in $\mathbf{Z}[\zeta]$ and π is not associated to $1 - \zeta$ then $N(\pi) \equiv 1 \pmod{3}$, and the cubic residue symbol is defined by

$$[\alpha/\pi] = \alpha^{(N(\pi)-1)/3} \pmod{\pi}.$$

3 Computing gcd in $\mathbf{Z}[\zeta]$

It turns out that the wellknown binary integer gcd algorithm has a natural generalisation to a gcd algorithm for the Eisenstein integers. The generalised algorithm is best understood by relating it to the binary algorithm which we start by recalling:

A slightly nonstandard version of the binary gcd is the following. Every integer can be represented as $(-1)^i \cdot 2^j \cdot (4m + 1)$. Without loss of generality, we may therefore assume that the numbers in question are

of the form $(4m+1)$. One iteration consists in replacing the numerically larger of the two numbers by their difference. If it is nonzero then the dividing 2-power (at least 2^2) may be removed without changing the gcd. If necessary the resulting odd number is multiplied with -1 to get a number of the form $4m + 1$ and we are ready for the next iteration. It is fairly obvious that the product of the numeric values of the two numbers decreases by a factor at least 2 in each step until the gcd is found, and hence the gcd of two numbers a, b can be computed in time $(\log^2 |ab|)$.

To make the analogue, we recall that any element of $\mathbf{Z}[\zeta]$ that is not divisible by $1 - \zeta$ is associated to a (unique) primary number, i.e. a number of the form $1 + 3\alpha$. This implies that any element in $\mathbf{Z}[\zeta] \setminus \{0\}$ has a (unique) representation on the form $(-\zeta)^i \cdot (1 - \zeta)^j \cdot (1 + 3\alpha)$ where $0 \leq i < 6$, $0 \leq j$ and $\alpha \in \mathbf{Z}[\zeta]$. In addition, the difference of two primary numbers is divisible by $(1 - \zeta)^2$, since $3 = -\zeta^2(1 - \zeta)^2$. Now a gcd algorithm for the Eisenstein integers may be formulated as an analogue to the binary integer gcd algorithm. We may assume without loss of generality that the two input numbers are primary. Replace the (normwise) larger of the two numbers with their difference. If it is nonzero, we may divide out any powers of $(1 - \zeta)$ that divide the difference (at least $(1 - \zeta)^2$) and convert the remaining factor to primary form by multiplying with a unit. We have again two primary numbers and the process may be continued. In each step we are required to identify the (normwise) larger of two numbers. Unfortunately it would be too costly to compute the relevant norm, but it suffices to choose the large number based on an approximation that we can afford to compute. By a slightly nontrivial argument one may prove that the product of the norms of the two numbers decreases by a factor at least 2 in each step until the gcd is found, and hence the gcd of two numbers α, β can be computed in time $O(\log^2 N(\alpha\beta))$.

Algorithm 1 describes the details including a start-up to bring the two numbers on primary form.

Theorem 1 *Algorithm 1 takes time $O(\log^2 N(\alpha\beta))$ to compute the gcd of α, β , or formulated alternatively, the algorithm has bit complexity $O(n^2)$.*

Proof. Let us assume that a number $\alpha = a + b\zeta \in \mathbf{Z}[\zeta]$ is represented by the integer pair (a, b) . Observe that since $N(\alpha) = a^2 + b^2 - ab$, we have that $\log |a| + \log |b| \leq \log N(\alpha) \leq 2(\log |a| + \log |b|)$ for $a, b \neq 0$, i.e. the logarithm of the norm is proportional to the number of bits in the representation of a number.

Algorithm 1 Compute gcd in $\mathbf{Z}[\zeta]$

Require: $\alpha, \beta \in \mathbf{Z}[\zeta] \setminus \{0\}$

Ensure: $g = \gcd(\alpha, \beta)$

- 1: Let primary $\gamma, \delta \in \mathbf{Z}[\zeta]$ be defined by $\alpha = (-\zeta)^{i_1} \cdot (1 - \zeta)^{j_1} \cdot \gamma$ and $\beta = (-\zeta)^{i_2} \cdot (1 - \zeta)^{j_2} \cdot \delta$.
 - 2: $g \leftarrow (1 - \zeta)^{\min\{j_1, j_2\}}$
 - 3: Replace α, β with γ, δ .
 - 4: **while** $\alpha \neq \beta$ **do**
 - 5: LOOP INVARIANT: α, β are primary
 - 6: Let primary γ be defined by $\alpha - \beta = (-\zeta)^i \cdot (1 - \zeta)^j \cdot \gamma$
 - 7: Replace “approximately” larger of α, β with γ .
 - 8: **end while**
 - 9: $g \leftarrow g \cdot \alpha$
-

We may do addition, subtraction on general numbers and multiplication by units in linear time. Since $(1 - \zeta)^{-1} = (2 + \zeta)/3$, division by (and check for divisibility by) $(1 - \zeta)$ may also be done in linear time.

Clearly, the startup part of the algorithm that brings the two numbers on primary form can be done in time $O(\log^2 N(\alpha\beta))$. Hence, we need only worry about the while loop.

We want to prove that the norm of the numbers decrease for each iteration. The challenge is to see that forming the number $\alpha - \beta$ does not increase the norm too much. In fact $N(\alpha - \beta) \leq 4 \cdot \max\{N(\alpha), N(\beta)\}$. This follows trivially from the equation $N(\alpha + \beta) + N(\alpha - \beta) = 2(N(\alpha) + N(\beta))$ that may be proven by an elementary computation when using that the norm is nonnegative. Hence, for the γ computed in the loop of the algorithm, we get $N(\gamma) = 3^{-j} N(\alpha - \beta) \leq 3^{-2} 4 \cdot \max\{N(\alpha), N(\beta)\}$. In each iteration, γ ideally replaces the one of α and β with the larger norm. However, we can not afford to actually compute the norms to find out which one is the larger. Fortunately, by Lemma 2, it is possible in linear time to compute an approximate norm that may be slightly smaller than the exact norm, namely up to a factor $9/8$. When γ replaces the one of α and β with the larger approximate norm, we know that $N(\alpha\beta)$ decreases by a factor at least $9/4 \cdot 8/9 = 2$ in each iteration, i.e. the total number of iterations is $O(\log N(\alpha\beta))$.

Each loop iteration takes time $O(\log N(\alpha\beta))$ except possibly for finding the exponent of $(1 - \zeta)$ that divides $\alpha - \beta$. Assume that $(1 - \zeta)^{t_i}$ is the maximal power of $(1 - \zeta)$ that divides $\alpha - \beta$ in the i th iteration. Then the combined time complexity of all loop iterations is $O((\sum_i t_i) \log N(\alpha\beta))$.

We also know that $N(\alpha\beta) \geq \prod_i(3^{t_i}/4)$. Since we already know that there is only $O(\log N(\alpha\beta))$ iterations it follows that $\prod_i 4 = N(\alpha\beta)^{O(1)}$ and hence $\sum_i t_i = O(\log N(\alpha\beta))$. ■

Lemma 2 *Given $\alpha = a + b\zeta \in \mathbf{Z}[\zeta]$ it is possible to compute an approximate norm $\tilde{N}(\alpha)$ such that*

$$\frac{8}{9}N(\alpha) \leq \tilde{N}(\alpha) \leq N(\alpha)$$

in linear time, i.e. in time $O(\log N(\alpha))$.

Proof. Note that

$$N(a + b\zeta) = \frac{(a - b)^2 + a^2 + b^2}{2}.$$

Given $\epsilon > 0$, we let \tilde{d} denote some approximation to integer d satisfying that $(1 - \epsilon)|d| \leq \tilde{d} \leq |d|$. Note that

$$(1 - \epsilon)^2 N(a + b\zeta) \leq \frac{(\widetilde{a - b})^2 + \tilde{a}^2 + \tilde{b}^2}{2} \leq N(a + b\zeta)$$

Since we may compute $a - b$ in linear time it suffices to compute \sim -approximations and square them in linear time for some $\epsilon < 1/18$. Given d in the usual binary representation, we take \tilde{d} to be $|d|$ with all but the 6 most significant bits replaced with zeroes, in which case

$$(1 - \frac{1}{32})|d| \leq \tilde{d} \leq |d|$$

and we can compute \tilde{d}^2 from d in linear time. ■

4 Computing cubic residuosity in $\mathbf{Z}[\zeta]$

Just as the usual integer gcd algorithms may be used for constructing algorithms for the Jacobi symbol, so can our earlier strategy for computing the gcd in $\mathbf{Z}[\zeta]$ be used as the basis for an algorithm for computing the cubic residuosity symbol.

We start by recalling the definition of the cubic residuosity symbol.

$$[\cdot/\cdot] : \mathbf{Z}[\zeta] \times (\mathbf{Z}[\zeta] - (1 - \zeta)\mathbf{Z}[\zeta]) \mapsto \{0, 1, \zeta, \zeta^{-1}\}$$

is defined as follows:

- For prime $\pi \in \mathbf{Z}[\zeta]$ where π is not associated to $1 - \zeta$:

$$[\alpha/\pi] = (\alpha^{\frac{N(\pi)-1}{3}}) \bmod \pi$$

- For number $\beta = \prod_{i=1}^t \pi_i^{m_i} \in \mathbf{Z}[\zeta]$ where β is not divisible by $1 - \zeta$:

$$[\alpha/\beta] = \prod_{i=1}^t [\alpha/\pi_i]^{m_i}$$

Note that these rules imply $[\alpha/\epsilon] = 1$ for a unit ϵ and $[\alpha/\beta] = 0$ when $\gcd(\alpha, \beta) \neq 1$. In addition, we will need the following laws satisfied by the cubic residuosity symbol (recall that β is primary when it has the form $\beta = 1 + 3\gamma$ for $\gamma \in \mathbf{Z}[\zeta]$) [4]:

- Modularity:

$$[\alpha/\beta] = [\alpha'/\beta], \quad \text{when } \alpha \equiv \alpha' \pmod{\beta}.$$

- Multiplicity:

$$[\alpha\alpha'/\beta] = [\alpha/\beta] \cdot [\alpha'/\beta].$$

- The cubic reciprocity law:

$$[\alpha/\beta] = [\beta/\alpha], \quad \text{when } \alpha \text{ and } \beta \text{ are both primary.}$$

- The complementary laws (for primary $\beta = 1 + 3(m + n\zeta)$, where $m, n \in \mathbf{Z}$)

$$\begin{aligned} [1 - \zeta/\beta] &= \zeta^m, \\ [\zeta/\beta] &= \zeta^{-(m+n)}, \\ [-1/\beta] &= 1. \end{aligned}$$

The cubic residuosity algorithm will follow the gcd algorithm closely. In each iteration we will assume the two numbers α, β to be primary with $\tilde{N}(\alpha) \geq \tilde{N}(\beta)$. We write their difference on the form $\alpha - \beta = (-\zeta)^i(1 - \zeta)^j\gamma$, for primary $\gamma = 1 + 3(m + n\zeta)$. By the above laws, $[\alpha/\beta] = \zeta^{mj-(m+n)i}[\gamma/\beta]$. If $\tilde{N}(\alpha) < \tilde{N}(\beta)$, we use the reciprocity law to swap γ and β before being ready to a new iteration. The algorithm stops, when the two primary numbers are identical. If the identical value (the gcd) is not 1 then the residuosity symbol evaluates to 0.

Algorithm 2 describes the entire procedure including a start-up to ensure that the numbers are primary.

Algorithm 2 Compute cubic residuosity in $\mathbf{Z}[\zeta]$

Require: $\alpha, \beta \in \mathbf{Z}[\zeta] \setminus \{0\}$, and β is not divisible by $(1 - \zeta)$

Ensure: $c = [\alpha/\beta]$

- 1: Let primary $\gamma, \delta \in \mathbf{Z}[\zeta]$ be defined by $\alpha = (-\zeta)^{i_1} \cdot (1 - \zeta)^{j_1} \cdot \gamma$ and $\beta = (-\zeta)^{i_2} \cdot \delta$.
 - 2: Let $m, n \in \mathbf{Z}$ be defined by $\delta = 1 + 3m + 3n\zeta$.
 - 3: $t \leftarrow mj_1 - (m+n)i_1 \bmod 3$
 - 4: Replace α, β by γ, δ .
 - 5: If $\tilde{N}(\alpha) < \tilde{N}(\beta)$ then interchange α, β .
 - 6: **while** $\alpha \neq \beta$ **do**
 - 7: LOOP INVARIANT: α, β are primary and $\tilde{N}(\alpha) \geq \tilde{N}(\beta)$
 - 8: Let primary γ be defined by $\alpha - \beta = (-\zeta)^i \cdot (1 - \zeta)^j \cdot \gamma$
 - 9: Let $m, n \in \mathbf{Z}$ be defined by $\beta = 1 + 3m + 3n\zeta$.
 - 10: $t \leftarrow t + mj - (m+n)i \bmod 3$
 - 11: Replace α with γ .
 - 12: If $\tilde{N}(\alpha) < \tilde{N}(\beta)$ then interchange α, β .
 - 13: **end while**
 - 14: If $\alpha \neq 1$ then $c \leftarrow 0$ else $c \leftarrow \zeta^t$
-

Theorem 3 Algorithm 2 takes time $O(\log^2 N(\alpha\beta))$ to compute $[\alpha/\beta]$, or formulated alternatively, the algorithm has bit complexity (n^2) .

Proof. The complexity analysis from the gcd algorithm carries over without essential changes. ■

5 Computing gcd and quartic residuosity in the ring of Gaussian integers

We may construct fast algorithms for gcd and quartic residuosity in the ring of Gaussian integers, $\mathbf{Z}[i] = \{a + bi \mid a, b \in \mathbf{Z}\}$, in a completely analogous way to the algorithms over the Eisenstein integers.

Here is a sketch of the necessary facts (see [4]). There are 4 units in $\mathbf{Z}[i]$: $\pm 1, \pm i$. $1+i$ is a prime in $\mathbf{Z}[i]$ and $N(1+i) = 2$. A primary number has the form $1 + (2+2i)\beta$ for some $\beta \in \mathbf{Z}[i]$. If $\alpha \in \mathbf{Z}[i]$ is not divisible by $1+i$ then α is associated to a primary number.

In particular, any element in $\mathbf{Z}[i] \setminus \{0\}$ has a (unique) representation on the form $i^j \cdot (1+i)^k \cdot (1+(2+2i)\alpha)$ where $0 \leq j < 4$, $0 \leq k$ and $\alpha \in \mathbf{Z}[i]$. In addition, the difference of two primary numbers is divisible by $(1+i)^3$,

since $(2 + 2i) = -i(1 + i)^3$. This is the basis for obtaining an algorithm for computing gcd over the Gaussian integers analogous to Algorithm 1. This new algorithm has also bit complexity $O(n^2)$ as one may prove when using that $N((1 + i)^3) = 8$ and $N(\alpha - \beta) \leq 4 \cdot \max\{N(\alpha), N(\beta)\}$.

For computing quartic residuosity, we need more facts [4]. If π is a prime in $\mathbf{Z}[i]$ and π is not associated to $1 + i$ then $N(\pi) \equiv 1 \pmod{4}$, and the quartic residue symbol $[\cdot/\cdot] : \mathbf{Z}[i] \times (\mathbf{Z}[i] - (1 + i)\mathbf{Z}[i]) \mapsto \{0, 1, -1, i, -i\}$ is defined as follows:

- For prime $\pi \in \mathbf{Z}[i]$ where π is not associated to $1 + i$:

$$[\alpha/\pi] = (\alpha^{\frac{N(\pi)-1}{4}}) \pmod{\pi}$$

- For number $\beta = \prod_{j=1}^t \pi_j^{m_j} \in \mathbf{Z}[i]$ where β is not divisible by $1 + i$:

$$[\alpha/\beta] = \prod_{j=1}^t [\alpha/\pi_j]^{m_j}$$

The quartic residuosity symbol satisfies in addition

- Modularity:

$$[\alpha/\beta] = [\alpha'/\beta], \quad \text{when } \alpha \equiv \alpha' \pmod{\beta}.$$

- Multiplicity:

$$[\alpha\alpha'/\beta] = [\alpha/\beta] \cdot [\alpha'/\beta].$$

- The quartic reciprocity law:

$$[\alpha/\beta] = [\beta/\alpha] \cdot (-1)^{\frac{N(\alpha)-1}{4} \cdot \frac{N(\beta)-1}{4}}, \quad \text{when } \alpha \text{ and } \beta \text{ are both primary.}$$

- The complementary laws (for primary $\beta = 1 + (2 + 2i)(m + ni)$, where $m, n \in \mathbf{Z}$)

$$\begin{aligned} [1 + i/\beta] &= i^{-n-(n+m)^2}, \\ [i/\beta] &= i^{n-m}. \end{aligned}$$

This is the basis for obtaining an algorithm for computing quartic residuosity analogous to Algorithm 2. This new algorithm has also bit complexity $O(n^2)$.

References

- [1] Eric Bach and Jeffrey Shallit. *Algorithmic number theory. Vol. 1.* Foundations of Computing Series. MIT Press, Cambridge, MA, 1996. Efficient algorithms.
- [2] Ivan B. Damgård and Gudmund Skovbjerg Frandsen. An extended quadratic Frobenius primality test with average and worst case error estimates. Research Series RS-03-9, BRICS, Department of Computer Science, University of Aarhus, February 2003.
- [3] Donald E. Knuth. An imaginary number system. *Comm. ACM* **3** (1960), 245–247.
- [4] Franz Lemmermeyer. *Reciprocity laws.* Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2000. From Euler to Eisenstein.
- [5] Asger Munk Nielsen and Peter Kornerup. Redundant radix representations of rings. *IEEE Trans. Comput.* **48**(11) (1999), 1153–1165.
- [6] Walter Penney. A “binary” system for complex numbers. *J. Assoc. Comput. Mach.* **12** (1965), 247–248.
- [7] Renate Scheidler and Hugh C. Williams. A public-key cryptosystem utilizing cyclotomic fields. *Des. Codes Cryptogr.* **6**(2) (1995), 117–131.
- [8] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informat.* **1** (1971), 139–144.
- [9] Jeffrey Shallit and Jonathan Sorenson. A binary algorithm for the jacobi symbol. *ACM SIGSAM Bull.* **27**(1) (1993), 4–11.
- [10] J. Stein. Computational problems associated with Racah algebra. *J. Comput. Phys.* **1** (1967), 397–405.
- [11] H. C. Williams. An M^3 public-key encryption scheme. In *Advances in cryptology—CRYPTO ’85 (Santa Barbara, Calif., 1985)*, Vol. 218 of *Lecture Notes in Comput. Sci.*, pp. 358–368. Springer, Berlin, 1986.
- [12] H. C. Williams and R. Holte. Computation of the solution of $x^3+Dy^3 = 1$. *Math. Comp.* **31**(139) (1977), 778–785.

Recent BRICS Report Series Publications

- RS-03-8 Ivan B. Damgård and Gudmund Skovbjerg Frandsen. *Efficient Algorithms for gcd and Cubic Residuosity in the Ring of Eisenstein Integers*. February 2003. 11 pp.
- RS-03-7 Claus Brabrand, Michael I. Schwartzbach, and Mads Vanggaard. *The METAFRONT System: Extensible Parsing and Transformation*. February 2003. 24 pp.
- RS-03-6 Giuseppe Milicia and Vladimiro Sassone. *Jeeg: Temporal Constraints for the Synchronization of Concurrent Objects*. February 2003. 41 pp. Short version appears in Fox and Getov, editors, *Joint ACM-ISCOPE Conference on Java Grande, JGI '02 Proceedings*, 2002, pages 212–221.
- RS-03-5 Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach. *Precise Analysis of String Expressions*. February 2003. 15 pp.
- RS-03-4 Marco Carbone and Mogens Nielsen. *Towards a Formal Model for Trust*. January 2003.
- RS-03-3 Claude Crépeau, Paul Dumais, Dominic Mayers, and Louis Salvail. *On the Computational Collapse of Quantum Information*. January 2003. 31 pp.
- RS-03-2 Olivier Danvy and Pablo E. Martínez López. *Tagging, Encoding, and Jones Optimality*. January 2003. To appear in Degano, editor, *Programming Languages and Systems: Twelfth European Symposium on Programming*, ESOP '03 Proceedings, LNCS, 2003.
- RS-03-1 Vladimiro Sassone and Paweł Sobociński. *Deriving Bisimulation Congruences: 2-Categories vs. Precategories*. January 2003. To appear in Gordon, editor, *Foundations of Software Science and Computation Structures*, FoSSaCS '03 Proceedings, LNCS, 2003.
- RS-02-52 Olivier Danvy. *A New One-Pass Transformation into Monadic Normal Form*. December 2002. 16 pp. To appear in Hedin, editor, *Compiler Construction, 12th International Conference*, CC '03 Proceedings, LNCS, 2003.