Primitive Sets of a Lattice and a Generalization of Euclidean Algorithm*

Spyros. S. Magliveras
Center for Cryptology and Information Security
Department of Mathematical Sciences
Florida Atlantic University
Boca Raton, FL 33431, U.S.A
spyros@fau.unl.edu

Tran van Trung
Institute for Experimental Mathematics
University of Duisburg-Essen
Essen, Germany
trung@iem.uni-due.de

Wandi Wei
Center for Cryptology and Information Security
Department of Mathematical Sciences
Florida Atlantic University
Boca Raton, FL 33431, U.S.A
wei@brain.math.fau.edu

Abstract

We present a generalization of the Euclidean algorithm, and apply it to give a solution to the following lattice problem. Let $\Lambda = \Lambda(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ be a lattice of rank n in $\mathbb{R}^m$ with the basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. Let $\mathbf{a} = a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \ldots + a_n\mathbf{b}_n \in \Lambda$ be a primitive vector. It has been proved that $\mathbf{a}$ can be extended to a basis. But there is no known formula for an extended basis of a primitive vector. Our generalization

*This work was partially supported by a Federal Earmark grant for *Research in Secure Telecommunication Networks* (2004-05)

of the Euclidean algorithm provides such a formula, whose entries can be computed very efficiently.

Key words. Euclidean algorithm, lattice, primitive vector, primitive set, basis

1 Introduction

Let $\mathbb{Z}$ denote the set of integers. Let $\mathbb{R}^m$ denote the m-dimensional real space, and let $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ be n vectors linearly independent over $\mathbb{R}$. The set

$$\Lambda = \Lambda(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \{z_1\mathbf{b}_1 + z_2\mathbf{b}_2 + \ldots + z_n\mathbf{b}_n : z_i \in \mathbb{Z}, 1 \le i \le n\}$$

is called a **lattice of rank** n **in** $\mathbb{R}^m$, and the set of vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ a **basis** of Λ.

A set C of k lattice vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k \in \Lambda$ is called a **primitive set** if these vectors are linearly independent over $\mathbb{R}$ and

$$\Lambda \cap \operatorname{span}_{\mathbb{R}}(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k) = \Lambda(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k).$$

When $k = 1$, the only vector in C is called a **primitive vector**. It is proved (see, for example, [4, 6]) that a lattice vector

$$\mathbf{a} = a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \ldots + a_n\mathbf{b}_n$$

is primitive if and only if

$$\gcd(a_1, a_2, \ldots, a_n) = 1.$$

It is also proved that any primitive set of vectors can be extended to a basis. The proof in [4] depends on the result that if $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$ ($k < n$) form a primitive set and

$$\mathbf{y} \in \Lambda \setminus \Lambda(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k),$$

then the $(k+1)$-dimensional parallelotope P spanned by $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k, \mathbf{y}$ contains a vector $\mathbf{y}_{k+1} \in P \cap \Lambda$ with a positive minimum distance to $\Lambda(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k)$, and the $k+1$ vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k, \mathbf{y}_{k+1}$ form a primitive set. The proof in [6] makes use of the fact that for any k linearly independent lattice vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$, the infimum

$$\begin{aligned} \inf\{t_k > 0 \quad : \quad & t_1\mathbf{y}_1 + t_2\mathbf{y}_2 + \ldots + t_k\mathbf{y}_k \in \Lambda \\ & t_i \in \mathbb{R}, t_i \ge 0 \ (1 \le i \le k)\} \end{aligned}$$

is actually attained for some vector $\mathbf{z} \in \Lambda$, and $\mathbf{z}$ is used in the construction of an extended basis. But there is no known polynomial algorithm for computing either $\mathbf{y}_{k+1}$ or $\mathbf{z}$ and therefore for constructing an extended basis, even for the special case of a single primitive vector.

In this article we study this problem from the viewpoint of the well-known Euclidean algorithm, treat the extended basis of a primitive vector of a lattice as a special case of a generalization of the Euclidean algorithm, and present a formula for an extended basis. The computations of the entries in the formula can be done very efficiently. We also present a formula for an extended basis of an arbitrary primitive set of size $n-1$.

In the next section, we state our generalization of the Euclidean algorithm, derive the above-mentioned formula, and analyze the time complexity of computing the entries in the formula. In §3, we apply this method to give an algorithmic solution to the construction of an extended basis of a primitive vector of a lattice. In §4, we discuss the case where the size of the primitive sets is $n-1$. Concluding remarks are drawn in the last section.

2 A Generalization of the Euclidean Algorithm

Let a, b be two integers not both zero, and $d = \gcd(a, b)$. By means of the Euclidean algorithm integers t, s can be determined so that

$$d = at + bs. \tag{2.1}$$

Here we consider a generalization.

Equality (2.1) can be rewritten as

$$d = \begin{vmatrix} a & b \\ -s & t \end{vmatrix}, \quad t, s \in \mathbb{Z}. \tag{2.2}$$

Let $\mathbb{Z}^n$ denote the set of n-dimensional integral vectors. Let $(a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$ be a nonzero vector, and $d = \gcd(a_1, a_2, \ldots, a_n)$. Then our generalization is to design an algorithm for finding an integral $(n-1) \times n$ matrix

$$(m_{ij}), \quad 2 \leq i \leq n, 1 \leq j \leq n$$

such that

$$d = \det\, (a_{ij}), \tag{2.3}$$

where

$$\begin{aligned} a_{1j} &= a_j, \quad j = 1, 2, \ldots, n \\ a_{ij} &= m_{ij}, \quad i = 2, 3, \ldots, n;\ j = 1, 2, \ldots, n \end{aligned}$$

For our purpose we can assume without loss of generality that $a_1 \neq 0$. Let

$$\begin{aligned} d_1 &= a_1, \\ d_i &= \gcd(a_1, a_2, \ldots, a_i), \quad 2 \leq i \leq n, \\ d &= d_n. \end{aligned}$$

Since $a_1 \neq 0$, all the d_i are well defined, and

$$d_i = \gcd(d_{i-1}, a_i), \quad 2 \leq i \leq n.$$

By the Euclidean algorithm, we can determine the values of $t_i,\ s_i,\ (2 \leq i \leq n)$ such that

$$d_i = t_{i-1}\, d_{i-1} + s_{i-1}\, a_i, \quad 2 \leq i \leq n.$$

We now prove

Theorem 2.1 *Let n be a positive integer greater than 1. Let $U = U_n$ be the $n \times n$ matrix*

$$U := U_n := (u_{ij}) = \begin{pmatrix} a_1 & a_2 & a_3 & \ldots & a_n \\ -s_1 & t_1 & 0 & \ldots & 0 \\ -\frac{a_1\, s_2}{d_2} & -\frac{a_2\, s_2}{d_2} & t_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ -\frac{a_1\, s_{n-1}}{d_{n-1}} & -\frac{a_2\, s_{n-1}}{d_{n-1}} & -\frac{a_3\, s_{n-1}}{d_{n-1}} & \ldots & t_{n-1} \end{pmatrix},$$

i.e.,

$$\begin{aligned} u_{1j} &= a_j, \quad 1 \leq j \leq n \\ u_{ij} &= -\frac{a_j\, s_{i-1}}{d_{i-1}}, \quad 1 \leq j \leq i-1,\ 2 \leq i \leq n \\ u_{ii} &= t_i \quad 2 \leq i \leq n \\ u_{ij} &= 0, \quad i+1 \leq j \leq n,\ 2 \leq i \leq n \end{aligned}$$

Then U_n is an integral matrix and

$$det(U_n) = d_n = d. \tag{2.4}$$

Proof. Since $d_{i-1} = \gcd(a_1, a_2, \ldots, a_{i-1})$, we know that $\frac{a_j\, s_{i-1}}{d_{i-1}}$ $(1 \le j \le i-1)$ are integers, and then all the numbers u_{ij} are integers.

We prove (2.4) by induction on $n \ge 2$. By (2.1) and (2.2) we know that the induction basis is true. Now we assume that (2.4) is true for $n = k$, i.e.,

$$\det(U_k) = d_k. \tag{2.5}$$

When $n = k+1$, we have

$$\det(U_{k+1}) = \begin{vmatrix} a_1 & a_2 & a_3 & \ldots & a_k & a_{k+1} \\ -s_1 & t_1 & 0 & \ldots & 0 & 0 \\ -\frac{a_1\, s_2}{d_2} & -\frac{a_2\, s_2}{d_2} & t_2 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ -\frac{a_1\, s_{k-1}}{d_{k-1}} & -\frac{a_2\, s_{k-1}}{d_{k-1}} & -\frac{a_3\, s_{k-1}}{d_{k-1}} & \ldots & t_{k-1} & 0 \\ -\frac{a_1\, s_k}{d_k} & -\frac{a_2\, s_k}{d_k} & -\frac{a_3\, s_k}{d_k} & \ldots & -\frac{a_k\, s_k}{d_k} & t_k \end{vmatrix}$$

Expanding it by its last column, we have

$$\begin{aligned} &\det(U_{k+1}) \\ = \quad & t_k \det(U_k) + (-1)^{k+2}\, a_{k+1} \begin{vmatrix} -s_1 & t_1 & 0 & \ldots & 0 \\ -\frac{a_1\, s_2}{d_2} & -\frac{a_2\, s_2}{d_2} & t_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ -\frac{a_1\, s_{k-1}}{d_{k-1}} & -\frac{a_2\, s_{k-1}}{d_{k-1}} & -\frac{a_3\, s_{k-1}}{d_{k-1}} & \ldots & t_{k-1} \\ -\frac{a_1\, s_k}{d_k} & -\frac{a_2\, s_k}{d_k} & -\frac{a_3\, s_k}{d_k} & \ldots & -\frac{a_k\, s_k}{d_k} \end{vmatrix} \end{aligned}$$

$$= \quad t_k\, d_k + (-1)^{k+3}\,\frac{s_k}{d_k}\,a_{k+1}\begin{vmatrix} -s_1 & t_1 & 0 & \dots & 0 \\ -\frac{a_1\,s_2}{d_2} & -\frac{a_2\,s_2}{d_2} & t_2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -\frac{a_1\,s_{k-1}}{d_{k-1}} & -\frac{a_2\,s_{k-1}}{d_{k-1}} & -\frac{a_3\,s_{k-1}}{d_{k-1}} & \dots & t_{k-1} \\ a_1 & a_2 & a_3 & \dots & a_k \end{vmatrix}$$

by induction hypothesis (2.5). Noting that the determinant in the last expression differs from $\det(U_k)$ only in the order of their rows, we have

$$\begin{aligned} & \det(U_{k+1}) \\ = \; & t_k\, d_k \\ & +(-1)^{k+3}\,\frac{s_k}{d_k}\,a_{k+1}\,(-1)^{k-1}\begin{vmatrix} a_1 & a_2 & a_3 & \dots & a_{k-1} & a_k \\ -s_1 & t_1 & 0 & \dots & 0 & 0 \\ -\frac{a_1\,s_2}{d_2} & -\frac{a_2\,s_2}{d_2} & t_2 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\frac{a_1\,s_{k-1}}{d_{k-1}} & -\frac{a_2\,s_{k-1}}{d_{k-1}} & -\frac{a_3\,s_{k-1}}{d_{k-1}} & \dots & -\frac{a_{k-1}\,s_{k-1}}{d_{k-1}} & t_{k-1} \end{vmatrix} \\ = \; & t_k\, d_k + \frac{s_k}{d_k}\det(B_k)\,a_{k+1} \\ = \; & t_k\, d_k + \frac{s_k}{d_k}\,d_k\,a_{k+1} \\ = \; & t_k\, d_k + s_k\,a_{k+1} \\ = \; & d_{k+1}, \end{aligned}$$

which completes the induction proof. □

Based on this theorem, one can easily compute matrix U as shown in the following algorithm.

An Algorithm for Computing Matrix U.

Step 1. Invoke the Euclidean Algorithm to compute the d_i $(2 \leq i \leq n)$ and the values of $s_i,\ t_i$ $(2 \leq i \leq n)$.

Step 2. We compute the integral values of the entries $\frac{a_i s_j}{d_j}$, $2 \leq i \leq j-1,\ 2 \leq j \leq n$.

For analyzing the time complexity of the algorithm, we need some lemmas.

Lemma 2.1 *Let $u, v \in \mathbb{Z}$ and $u \neq 0,\ v \neq 0$. Then there exist $s, t \in \mathbb{Z}$ such that $gcd(u,v) = su + tv$ and*

$$|s| \leq |v|,\ |t| \leq |u|.$$

Proof. Let d denote $\gcd(u,v)$, and $s_0,\ t_0 \in \mathbb{Z}$ any numbers such that $d = s_0 u + t_0 v$. Then for any $k \in \mathbb{Z}$,

$$s = s_0 + kv,\ t = t_0 - ku$$

satisfy $su + tv = d$. We have

$$\begin{aligned} d &\geq |s_0 + kv| \cdot |u| - |t_0 - ku| \cdot |v|, \\ d + |t_0 - ku| \cdot |v| &\geq |s_0 + kv| \cdot |u|. \end{aligned}$$

By the division algorithm, we can choose k such that $|t_0 - ku| < |u|$, i.e., $|t_0 - ku| \leq |u| - 1$. So

$$d + (|u| - 1) \cdot |v| \geq d + |t_0 - ku| \cdot |v| \geq |s_0 + kv| \cdot |u|,$$

and then

$$\begin{aligned} |s_0 + kv| &\leq \frac{d}{|u|} + (1 - \frac{1}{|u|})|v| \\ &= |v| - \frac{|v| - d}{|u|} \\ &\leq |v|. \end{aligned}$$

□

The following Lemmas are well known, and can be found in many books, for example, [1], [3], [5], etc.

Lemma 2.2 *Let $u, v \in \mathbb{Z}$ and $u \neq 0,\ v \neq 0$. Then $gcd(u,v)$ as well as s, t such that $gcd(u,v) = su + tv$ can be computed in $O((\log |u|)(\log |v|))$ bit operations.*

Lemma 2.3 *Let $u, v \in \mathbb{Z}$ and $u \neq 0,\ v \neq 0$. Then the product $u \cdot v$ can be computed in $O((\log\ |u|)(\log\ |v|))$ bit operations.*

Lemma 2.4 *Let $u, v \in \mathbb{Z}$ and $u \neq 0,\ v \neq 0$. Then the quotient $\frac{u}{v}$ can be computed in $O((\log\ |u|)(\log\ |v|))$ bit operations.*

We now analyze the time complexity of our algorithm. Let a_0 and a_0' be the two largest among all the absolute values $|a_i|$. The case where either a_0 or a_0' is 0 is trivial, so we now assume that both of them are nonzero.

Theorem 2.2 *The worst-case time complexity of the above algorithm is $O(n^2(\log\ a_0)(\log\ a_0'))$ bit operations.*

Proof. Step 1 of the algorithm can be carried out by invoking the Euclidean Algorithm $n-1$ times. By Lemma 2.2, this can be done in $(n-1) \cdot O((\log\ a_0)(\log\ a_0')) = O(n(\log\ a_0)(\log\ a_0'))$ bit operations.

The number of divisions and the number of multiplications in Step 2 is the same, which is

$$2 + 3 + \ldots + (n-1) = O(n^2).$$

By Lemma 2.1, the absolute values of all the numbers involved are bounded by a_0 and a_0'. Therefore, by Lemmas 2.3 and 2.4, all the integral values of the fractions dealt with in Step 2 can be computed in $O(n^2(\log\ a_0)(\log\ a_0'))$ bit operations. Therefore, the worst-case time complexity of the algorithm is

$$O(n(\log\ a_0)(\log\ a_0')) + O(n^2(\log\ a_0)(\log\ a_0')) = O(n^2(\log\ a_0)(\log\ a_0'))$$

bit operations. □

3 A Solution to the Construction of an Extended Basis of a Primitive Vector

Let $\Lambda = \Lambda(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ be a lattice with basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. Let $\mathbf{a} = a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \ldots + a_n\mathbf{b}_n \in \Lambda$ be a primitive vector. Then we have

$$\gcd(a_1, a_2, \ldots, a_n) = 1.$$

Theorem 2.1 asserts that $\det(U) = 1$, and then the row vectors of matrix $U = (u_{ij})$ with respect to basis $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ form a new basis of Λ, which is an extension of primitive vector $\mathbf{a}$.

Combining this and Theorem 2.2, we have

Theorem 3.1 *The primitive vector* $\mathbf{a}$ *together with the* $n-1$ *vectors*

$$\sum_{j=0}^{n} u_{ij}\mathbf{b}_j, \quad 2 \le i \le n$$

form an extended basis of $\mathbf{a}$*, where*

$$\begin{aligned} u_{ij} &= -\frac{a_j\, s_{i-1}}{d_{i-1}}, \quad 1 \le j \le i-1,\ 2 \le i \le n \\ u_{ii} &= t_i \quad 2 \le i \le n \\ u_{ij} &= 0. \quad i+1 \le j \le n,\ 2 \le i \le n \end{aligned}$$

The worst-case time complexity of computing the extended basis is $O(n^2(\log\, a_0)(\log\, a_0'))$ *bit operations.*

4 A Solution to the Construction of an Extended Basis of a Primitive Set of Size $n-1$

We adopt the notation about $a_i, d_i, s_i, t_i, \Lambda, \mathbf{b}_i$ introduced in §2.

We need two lemmas. The first can be proved by induction, and the second can be found in [2].

Lemma 4.1 *The* $\gcd(a_1, a_2, \ldots, a_n)$ *can be expressed as an integral linear combination of* a_i *as follows:*

$$\gcd(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} t_{i-1} s_i s_{i+1} s_{i+2} \cdots s_{n-1} \cdot a_i. \tag{4.1}$$

Lemma 4.2 *Let* $1 \le k < n$ *and*

$$\mathbf{a}_i = a_{i1}\mathbf{b}_1 + a_{i2}\mathbf{b}_2 + \ldots + a_{in}\mathbf{b}_n \in \Lambda, \quad 1 \le i \le k.$$

Let M *denote the* $k \times n$ *matrix* (a_{ij}) $(1 \le i \le k,\ 1 \le j \le n)$*. Then* $\{\mathbf{a}_i : 1 \le i \le k\}$ *is a primitive set if and only if the gcd of all the minors of order* k *of* M *is 1.*

We now consider the case when $k = n-1$. Suppose that the $n-1$ vectors

$$\mathbf{a}_i = a_{i1}\mathbf{b}_1 + a_{i2}\mathbf{b}_2 + \ldots + a_{in}\mathbf{b}_n \in \Lambda, \quad 1 \le i \le n-1$$

form a primitive set. Let $M = (a_{ij})$ be the $(n-1) \times n$ matrix of the coefficients of $\mathbf{a}_i$. Let A_i $(1 \leq i \leq n)$ be the n minors of M obtained by deleting the i^{th} column of M. Without loss of generality, we may assume that $A_1 \neq 0$. Then we can define

$$\begin{aligned} d'_1 &= A_1, \\ d'_i &= \gcd(A_1, A_2, \ldots, A_i), \quad 2 \leq i \leq n, \\ d' &= d'_n. \end{aligned}$$

By the Euclidean algorithm, we can determine the values of t'_i, s'_i, $(2 \leq i \leq n)$ such that

$$d'_i = t'_{i-1}\, d'_{i-1} + s'_{i-1}\, A_i, \quad 2 \leq i \leq n.$$

Then by Lemma 4.1 we have

$$\begin{aligned} \gcd(A_1, A_2, \ldots, A_n) &= \sum_{i=1}^{n} t'_{i-1} s'_i s'_{i+1} s'_{i+2} \cdots s'_{n-1} \cdot A_i \\ &= \sum_{i=1}^{n} (-1)^{n-i} [(-1)^{n-i} t'_{i-1} s'_i s'_{i+1} s'_{i+2} \cdots s'_{n-1}] \cdot A_i. \end{aligned}$$

Let

$$a_{ni} = (-1)^{n-i} t'_{i-1} s'_i s'_{i+1} s'_{i+2} \cdots s'_{n-1}, \quad 1 \leq i \leq n$$
$$A = (a_{ij}), \quad 1 \leq i, j \leq n,$$

and

$$\mathbf{a}_n = a_{n1}\mathbf{b}_1 + a_{n2}\mathbf{b}_2 + \ldots + a_{nn}\mathbf{b}_n.$$

Then $\mathbf{a}_n \in \Lambda$. By (4.2) and Lemma 4.2, we have

$$det(A) = \gcd(A_1, A_2, \ldots, A_n) = 1.$$

Therefore, $\mathbf{a}_i$ $(1 \leq i \leq n)$ form a basis.

Let us now discuss the time complexity of computing $\mathbf{a}_n$. There are many algorithms for computing integral determinants and many upper bounds for the absolute values of determinants. Suppose that the algorithm used for computing A_i has the worst-case complexity of $c(a_i, a'_i)$ bit operations, where a_i, a'_i are the two largest absolute values of the entries in A_i and $c(a_i, a'_i)$ is a function of a_i and a'_i. Let a_0, a'_0 denote the two largest absolute values of a_{ij} $(1 \leq i \leq n-1;\ 1 \leq j \leq n)$. Then the worst-case complexity of computing all A_i $(1 \leq i \leq n)$ is

$$O(n\, c(a_0, a'_0)).$$

Suppose that the upper bound used for $|A_i|$ is $w(a_i, a'_i)$, which is a function of a_i and a'_i. Then all $|A_i|$ $(1 \le i \le n)$ is upper bounded by $w(a_0, a'_0)$. Noting that there are $(n-1)-(i-1) = n-i$ multiplications in computing a_{ni}, by Lemmas 2.3 and 2.4 we know that the computations of all a_{ni} $(1 \le i \le n)$ need no more than

$$O(n\,c(a_0, a'_0)) + O(n\,w(a_0, a'_0)) = O(n\,(c(a_0, a'_0) + w(a_0, a'_0)))$$

bit operations.

In summary, we have

Theorem 4.1 *The vectors* $\mathbf{a}_i$ $(1 \le i \le n)$ *form an extended basis of primitive set* $\mathbf{a}_i$ $(1 \le i \le n-1)$*, and the worst-case complexity of computing this extended basis is* $O(n\,(c(a_0, a'_0) + w(a_0, a'_0)))$*.*

5 Concluding Remarks

We have employed the Euclidean algorithm and our generalization of it to provide formulas for an extended basis of an arbitrary primitive set of size $n-1$ or 1 of a lattice of rank n in $\mathbb{R}^m$. The entries in the formulas can be computed very efficiently. The results presented here shed some light on the general case where the size of the primitive set is between 2 and $n-2$, and we believe that the Euclidean algorithm will play a crucial role in the solution of the general case.

References

[1] Eric Bach and Jeffrey Shallit, Algorithmic Number Theory, Volume 1: Efficient Algorithms, The MIT Press, Cambridge, Massachusetts, 1997.

[2] J.W.S. Cassels, An Introduction to the Geometry of Numbers Springer, Berlin, 1997.

[3] Neal Koblitz, Algebraic Aspects of Cryptography, Springer, Berlin, 1998.

[4] C. G. Lekkerkerker and P. M. Gruber, Geometry of Numbers, Second edition North-Holland Mathematical Library, 37, North-Holland Publishing Co., 1987.

[5] Daniele Micciancio and Shafi Goldwasser, Complexity of Lattice Problems, A Cryptographic Perspective, Kluwer Academic Publishers, Boston, 2002

[6] Carl Ludwig Siegel, Lectures on the Geometry of Numbers, Springer-Verlag, 1989.