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 $\Lambda = \Lambda(b_1, b_2, \ldots, b_n)$ be a lattice of rank $n$ in $\mathbb{R}^m$ with the basis $b_1, b_2, \ldots, b_n$. Let $a = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n \in \Lambda$ be a primitive vector. It has been proved that $a$ 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 $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$ be $n$ vectors linearly independent over $\mathbb{R}$. The set

$$\Lambda = \Lambda(b_1, b_2, \ldots, b_n) = \{ z_1 b_1 + z_2 b_2 + \ldots + z_n b_n : z_i \in \mathbb{Z}, 1 \leq i \leq n \}$$

is called a **lattice of rank** $n$ in $\mathbb{R}^m$, and the set of vectors $b_1, b_2, \ldots, b_n$ a **basis** of $\Lambda$.

A set $C$ of $k$ lattice vectors $c_1, c_2, \ldots, c_k \in \Lambda$ is called a **primitive set** if these vectors are linearly independent over $\mathbb{R}$ and

$$\Lambda \cap \text{span}_\mathbb{R}(c_1, c_2, \ldots, c_k) = \Lambda(c_1, c_2, \ldots, c_k).$$

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

$$a = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

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 $y_1, y_2, \ldots, y_k$ ($k < n$) form a primitive set and

$$y \in \Lambda \setminus \Lambda(y_1, y_2, \ldots, y_k),$$

then the $(k + 1)$-dimensional parallelopiped $P$ spanned by $y_1, y_2, \ldots, y_k, y$ contains a vector $y_{k+1} \in P \cap \Lambda$ with a positive minimum distance to $\Lambda(y_1, y_2, \ldots, y_k)$, and the $k + 1$ vectors $y_1, y_2, \ldots, y_k, y_{k+1}$ form a primitive set. The proof in [6] makes use of the fact that for any $k$ linearly independent lattice vectors $y_1, y_2, \ldots, y_k$, the infimum

$$\inf\{ t_k > 0 : t_1 y_1 + t_2 y_2 + \ldots + t_k y_k \in \Lambda$$

$$t_i \in \mathbb{R}, t_i \geq 0 (1 \leq i \leq k) \}$$
is actually attained for some vector $z \in \Lambda$, and $z$ is used in the construction of an extended basis. But there is no known polynomial algorithm for computing either $y_{k+1}$ or $z$ 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

$$d = at + bs. \quad (2.1)$$

Here we consider a generalization.

Equality (2.1) can be rewritten as

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

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

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

where

$$a_{1j} = a_j, \quad j = 1, 2, \ldots, n$$

$$a_{ij} = m_{ij}, \quad i = 2, 3, \ldots, n; \quad j = 1, 2, \ldots, n$$
For our purpose we can assume without loss of generality that $a_1 \neq 0$. Let

\[ d_1 = a_1, \]
\[ d_i = \gcd(a_1, a_2, \ldots, a_i), \quad 2 \leq i \leq n, \]
\[ d = d_n. \]

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

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

i.e.,

\[
u_{1j} = a_j, \quad 1 \leq j \leq n
\]
\[
u_{ij} = -\frac{a_j s_{i-1}}{d_{i-1}}, \quad 1 \leq j \leq i - 1, \quad 2 \leq i \leq n
\]
\[
u_{ii} = t_i, \quad 2 \leq i \leq n
\]
\[
u_{ij} = 0, \quad i + 1 \leq j \leq n, \quad 2 \leq i \leq n
\]

Then $U_n$ is an integral matrix and

\[ \det(U_n) = d_n = d. \quad (2.4) \]
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 ≤ $j$ ≤ $i-1$) are integers, and then all the numbers $u_{ij}$ are integers.

We prove (2.4) by induction on $n \geq 2$. 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.,

$$\det(U_k) = d_k. \quad (2.5)$$

When $n = k + 1$, we have

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

Expanding it by its last column, we have

$$\det(U_{k+1}) = t_k \det(U_k) + (-1)^k a_{k+1}$$

\[ \begin{vmatrix}
-s_1 & t_1 & 0 & \ldots & 0 \\
-\frac{a_1 s_2}{d_2} & -\frac{a_2 s_2}{d_2} & t_2 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\frac{a_1 s_{k-1}}{d_{k-1}} & -\frac{a_2 s_{k-1}}{d_{k-1}} & -\frac{a_3 s_{k-1}}{d_{k-1}} & \ldots & t_{k-1} \\
-\frac{a_1 s_k}{d_k} & -\frac{a_2 s_k}{d_k} & -\frac{a_3 s_k}{d_k} & \ldots & -\frac{a_k s_k}{d_k}
\end{vmatrix} \]
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

\[
\det(U_{k+1}) = t_k d_k
\]

\[
+(-1)^{k+3} \frac{s_k}{d_k} a_{k+1} (-1)^{k-1}
\]

\[
= t_k d_k + \frac{s_k}{d_k} \det(B_k) a_{k+1}
\]

\[
= t_k d_k + \frac{s_k}{d_k} d_k a_{k+1}
\]

\[
= t_k d_k + s_k a_{k+1}
\]

\[
= d_{k+1},
\]

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 $\frac{a_i s_j}{s_j}$, $2 \leq i \leq j - 1$, $2 \leq j \leq n$.

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 $d = s_0u + t_0v$. Then for any $k \in \mathbb{Z}$,

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

satisfy $su + tv = d$. We have

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

$$d + |t_0 - ku| \cdot |v| \geq |s_0 + kv| \cdot |u|.$$  

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

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

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. \( \square \)

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

Let \( \Lambda = \Lambda(b_1, b_2, \ldots, b_n) \) be a lattice with basis \( b_1, b_2, \ldots, b_n \). Let \( a = a_1b_1 + a_2b_2 + \ldots + a_nb_n \in \Lambda \) 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 \( \{b_1, b_2, \ldots, b_n\} \) form a new basis of \( \Lambda \), which is an extension of primitive vector \( a \).

Combining this and Theorem 2.2, we have
Theorem 3.1 The primitive vector $a$ together with the $n - 1$ vectors
\[ \sum_{j=0}^{n} u_{ij} b_j, \quad 2 \leq i \leq n \]
form an extended basis of $a$, where
\[
\begin{align*}
    u_{ij} &= -\frac{a_j s_{i-1}}{d_{i-1}}, \quad 1 \leq j \leq i - 1, \quad 2 \leq i \leq n \\
    u_{ii} &= t_i, \quad 2 \leq i \leq n \\
    u_{ij} &= 0, \quad i + 1 \leq j \leq n, \quad 2 \leq i \leq n
\end{align*}
\]
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 $a_i, d_i, s_i, t_i, \Lambda, b_i$ 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:
\[
gcd(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} t_{i-1}s_is_{i+1}s_{i+2}\ldots s_{n-1}a_i. \quad (4.1)
\]

Lemma 4.2 Let $1 \leq k < n$ and
\[
a_i = a_{i1}b_1 + a_{i2}b_2 + \ldots + a_{in}b_n \in \Lambda, \quad 1 \leq i \leq k.
\]
Let $M$ denote the $k \times n$ matrix $(a_{ij})$ ($1 \leq i \leq k, \ 1 \leq j \leq n$). Then \( \{a_i : 1 \leq i \leq k\} \) 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
\[
a_i = a_{i1}b_1 + a_{i2}b_2 + \ldots + a_{in}b_n \in \Lambda, \quad 1 \leq i \leq n - 1
\]
form a primitive set. Let $M = (a_{ij})$ be the $(n - 1) \times n$ matrix of the coefficients of $a_i$. 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

$$d'_1 = A_1,$$
$$d'_i = \gcd(A_1, A_2, \ldots, A_i), \quad 2 \leq i \leq n,$$
$$d' = d'_n.$$

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

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

Then by Lemma 4.1 we have

$$\gcd(A_1, A_2, \ldots, A_n) = \sum_{i=1}^{n} t'_{i-1} s'_i s'_{i+1} s'_{i+2} \cdots s'_{n-1} A_i,$$

$$= \sum_{i=1}^{n} (-1)^{n-i} (-1)^{n-i} t'_{i-1} s'_i s'_{i+1} s'_{i+2} \cdots s'_{n-1} A_i.$$

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

$$a_n = a_{n1} b_1 + a_{n2} b_2 + \ldots + a_{nn} b_n.$$

Then $a_n \in \Lambda$. By (4.2) and Lemma 4.2, we have

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

Therefore, $a_i$ $(1 \leq i \leq n)$ form a basis.

Let us now discuss the time complexity of computing $a_n$. 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 \(|A_i|\) (\(1 \leq i \leq n\)) 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 \(a_{ni}\) (\(1 \leq i \leq n\)) 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 \(a_i\) (\(1 \leq i \leq n\)) form an extended basis of primitive set \(a_i\) (\(1 \leq i \leq n - 1\)), 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


