On the low hamming weight discrete logarithm problem for nonadjacent representations

J. A. Muir
Combinatorics and Optimization
University of Waterloo
Waterloo Ontario, N2L 3G1, Canada
jamuir@uwaterloo.ca

D. R. Stinson
School of Computer Science
University of Waterloo
Waterloo Ontario, N2L 3G1, Canada
dstinson@uwaterloo.ca

October 13, 2004

Abstract

So-called nonadjacent representations are commonly used in elliptic curve cryptography to facilitate computing a scalar multiple of a point on an elliptic curve. A nonadjacent representation having few non-zero coefficients would further speed up the computations. However, any attempt to use these techniques must also consider the impact on the security of the cryptosystem. The security is studied by examining a related discrete logarithm problem, the topic of this paper. We describe an algorithm to solve the relevant discrete logarithm problem in time that is approximately the square root of the search space. This algorithm is of the familiar "baby-step giant-step" type. In developing our algorithm we use two tools of independent interest; namely, a combinatorial set system called a "splitting system" and a new type of combinatorial Gray code.

1 Introduction

Let G be an abelian group, written multiplicatively. Let $\alpha \in G$, and denote the subgroup generated by α by $\langle \alpha \rangle$. Further, let $\mathsf{ord}(\alpha) = |\langle \alpha \rangle|$ (note that $\mathsf{ord}(\alpha)$ is called the *order* of α).

Suppose $\beta \in \langle \alpha \rangle$. The *discrete logarithm* $\log_\alpha \beta$ is the unique integer x such that $0 \leq x \leq \mathsf{ord}(\alpha) - 1$ and $\alpha^x = \beta$. The *discrete logarithm problem* is to compute $x = \log_\alpha \beta$, given α and β.

There are various ways that the integer x can be represented in the form $x = \sum_{i \geq 0} x_i 2^i$. Here are some common methods of doing this:

- In the usual binary representation (also called the $\{0, 1\}$-representation), $x_i \in \{0, 1\}$ for all i.
- In a $\{-1, 0, 1\}$-representation, $x_i \in \{-1, 0, 1\}$ for all i.
- In a $\{-1, 0, 1\}$-*nonadjacent representation*, the following two properties are satisfied:

1. $x_i \in \{-1, 0, 1\}$ for all i.
2. $x_i\, x_{i+1} = 0$ for all i.

Such a representation is also commonly called a *nonadjacent form* or *NAF*. Observe that the second property is stating that there do not exist two consecutive non-zero digits in a NAF.

All representations we consider in this paper will be finite, so there exists an integer n such that $x_i = 0$ for all $i \geq n$. The smallest such integer n is called the *length* of the representation. Observe that, under this definition, the integer 0 has a representation of length 0.

For future use, we record a basic, well-known fact about NAFs (see [11] for a proof).

Lemma 1.1. *Every integer x has a unique NAF, and the length of this representation is at most $1 + \lceil \log_2(|x| + 1) \rceil$.*

Let $\sum x_i 2^i$ be a representation, having length not exceeding n, of the integer x. Let the vector of coefficients of this representation be denoted $\mathbf{x}$; that is,

$$\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}).$$

From now on, the term "representation" will refer to this vector of coefficients. The (unique) NAF representation of the integer x will be denoted $\mathsf{NAF}(x)$.

The *hamming weight* of a representation $\mathbf{x}$, denoted $\mathsf{wt}(\mathbf{x})$, is the number of non-zero co-ordinates in the vector $\mathbf{x}$. Suppose two representations $\mathbf{x}$ and $\mathbf{y}$ both have length at most n. We define the *hamming distance* betwen $\mathbf{x}$ and $\mathbf{y}$, denoted $\mathsf{d}(\mathbf{x}, \mathbf{y})$, to be the number of co-ordinates i such that $x_i \neq y_i$.

In cryptographic protocols such as Diffie-Hellman key agreement (see, for example, [9]), it is often advantageous to choose an exponent x in such a way that α^x can be computed quickly. One way to do this is to choose x such that it has a representation $\mathbf{x}$ such that $\mathsf{wt}(\mathbf{x})$ is "small". In the context of elliptic curve cryptography, it is also useful to take $\mathbf{x}$ to be a NAF ([10]). Therefore, x might be chosen so that $\mathbf{x} = \mathsf{NAF}(x)$ has low hamming weight.

This motivates the following problem. Suppose t is a positive integer. Given α, β, and the integer t, the *fixed hamming weight NAF discrete logarithm problem* (*FHW-NAF-DLP*, for short) is to compute $x = \log_\alpha \beta$, given that $\mathsf{wt}(\mathbf{x}) = t$, where $\mathbf{x} = \mathsf{NAF}(x)$. We will also assume that an upper bound, say m, on the length of $\mathsf{NAF}(x)$ is given; if not, then we can assume without loss of generality that $m = 1 + \lceil \log_2 \mathsf{ord}(\alpha) \rceil$.

In this paper, we investigate algorithms for the FHW-NAF-DLP. The algorithms can be thought of as "baby-step giant-step algorithms" (see, e.g., [9, §3.6.2]). In general, we are attempting to design algorithms whose complexity is roughly proportional to the square root of the size of the search space. We adapt techniques from [4, 13] which were developed to solve the fixed hamming weight discrete logarithm problem (i.e., the corresponding problem for $\{0, 1\}$-representations).

The rest of this paper is organized as follows. In Section 2, we prove some preliminary lemmas that form the basis of our algorithms. Section 3 introduces splitting systems, and Section 4 presents the basic algorithm. In Section 5, we give a construction for a certain type of Gray code. Then, in Section 6, we show how these Gray codes enable an efficient implementation of the algorithm. Section 7 closes with some discussion and a conclusion.

2 Preliminaries

Given a vector $\mathbf{x} = (x_0, \ldots, x_{m-1}) \in \{-1, 0, 1\}^m$, we define the *support* of $\mathbf{x}$ to be the set

$$\mathsf{supp}(\mathbf{x}) = \{i : x_i \neq 0\}.$$

Observe that

$$\mathsf{wt}(\mathbf{x}) = |\mathsf{supp}(\mathbf{x})|.$$

Also, given a finite set $Y \subseteq \mathbb{Z}^+ \cup \{0\}$, define the *value* of Y to be the integer

$$\mathsf{val}(Y) = \sum_{i \in Y} 2^i.$$

Lemma 2.1. *Suppose that $x = \log_\alpha \beta$ is an integer such that $\mathbf{x} = \mathsf{NAF}(x)$ has length at most m and weight t. Suppose that $B \subseteq \{0, \ldots, m-1\}$ is such that*

$$|B \cap \mathsf{supp}(\mathbf{x})| = \left\lfloor \frac{t}{2} \right\rfloor.$$

Then there are disjoint subsets $Y_1^+, Y_1^- \subseteq B$ and disjoint subsets $Y_2^+, Y_2^- \subseteq \{0, \ldots, m-1\} \backslash B$ such that

$$|Y_1^+ \cup Y_1^-| = \left\lfloor \frac{t}{2} \right\rfloor, \qquad |Y_2^+ \cup Y_2^-| = \left\lceil \frac{t}{2} \right\rceil,$$

and

$$\alpha^{\mathsf{val}(Y_1^+) - \mathsf{val}(Y_1^-)} = \beta \left(\alpha^{\mathsf{val}(Y_2^+) - \mathsf{val}(Y_2^-)} \right)^{-1}.$$

Proof. Define

$$\begin{aligned} Y_1^+ &= \{i \in B : x_i = 1\}, \\ Y_1^- &= \{i \in B : x_i = -1\}, \\ Y_2^+ &= \{i \in \{0, \ldots, m-1\} \backslash B : x_i = 1\} \quad \text{and} \\ Y_2^- &= \{i \in \{0, \ldots, m-1\} \backslash B : x_i = -1\}. \end{aligned}$$

The desired properties are easily verified. □

Here is an example to illustrate.

Example 2.1. *Let $x = 229$; then*

$$\mathbf{x} = \mathsf{NAF}(x) = (1, 0, 1, 0, 0, -1, 0, 0, 1).$$

The length of this representation is $m = 9$, the weight is $t = 4$, and the non-zero co-ordinates are

$$x_0 = 1, \quad x_2 = 1, \quad x_5 = -1 \quad \text{and} \quad x_8 = 1.$$

Hence, $\mathsf{supp}(\mathbf{x}) = \{0, 2, 5, 8\}$. Suppose that $B = \{0, 1, 2, 6\}$; then $|B \cap \mathsf{supp}(\mathbf{x})| = 2 = \frac{t}{2}$. For this choice of B, we have

$$Y_1^+ = \{0, 2\}, \quad Y_1^- = \emptyset, \quad Y_2^+ = \{8\} \quad \text{and} \quad Y_2^- = \{5\}.$$

Then

$$\alpha^{\mathsf{val}(Y_1^+)-\mathsf{val}(Y_1^-)} = \alpha^{5-0} = \alpha^5$$

and

$$\alpha^{\mathsf{val}(Y_2^+)-\mathsf{val}(Y_2^-)} = \alpha^{256-32} = \alpha^{224}.$$

Hence, if $\beta = \alpha^x$, then we have

$$\beta\left(\alpha^{\mathsf{val}(Y_2^+)-\mathsf{val}(Y_2^-)}\right)^{-1} = \alpha^{229-224} = \alpha^5 = \alpha^{\mathsf{val}(Y_1^+)-\mathsf{val}(Y_1^-)},$$

as claimed in Lemma 2.1.

We require a slight variation of the previous lemma. A subset of $Y \subseteq \{0, \ldots, m-1\}$ is a *nonadjacent subset* provided that there do not exist two cyclically consecutive non-negative integers (modulo m), say i and $i+1 \bmod m$, that are both elements of Y. Observe that $\mathsf{supp}(\mathsf{NAF}(x))$ is a nonadjacent subset of $\{0, \ldots, m-1\}$ provided that $\mathsf{NAF}(x)$ has length not exceeding $m-1$.

Lemma 2.2. *Suppose that $x = \log_\alpha \beta$ is an integer such that $\mathbf{x} = \mathsf{NAF}(x)$ has length at most $m-1$ and weight t. Suppose that an interval $B = [\ell, r] \subseteq \{0, \ldots, m-1\}$ is such that*

$$|B \cap \mathsf{supp}(\mathbf{x})| = \left\lfloor \frac{t}{2} \right\rfloor.$$

Then there are disjoint subsets $Y_1^+, Y_1^- \subseteq B$ and disjoint subsets $Y_2^+, Y_2^- \subseteq \{0, \ldots, m-1\} \backslash B$ such that

$Y_1^+ \cup Y_1^- \subseteq B$ is a nonadjacent subset and $|Y_1^+ \cup Y_1^-| = \left\lfloor \frac{t}{2} \right\rfloor$,
$Y_2^+ \cup Y_2^- \subseteq \{0, \ldots, m-1\} \backslash B$ is a nonadjacent subset and $|Y_2^+ \cup Y_2^-| = \left\lceil \frac{t}{2} \right\rceil$, and
$\alpha^{\mathsf{val}(Y_1^+)-\mathsf{val}(Y_1^-)} = \beta\left(\alpha^{\mathsf{val}(Y_2^+)-\mathsf{val}(Y_2^-)}\right)^{-1}$.

Proof. The proof is the same as that of Lemma 2.1. □

3 Splitting systems

Our algorithms use a type of combinatorial set system, defined in [13], called a "splitting system". Suppose m and t are integers, $0 < t < m$. An (m,t)-*splitting system* is a pair $(X, \mathcal{B})$ that satisfies the following properties:

1. $|X| = m$, and $\mathcal{B}$ is a set of $\lfloor \frac{m}{2} \rfloor$-subsets of X, called *blocks*
2. for every $Y \subseteq X$ such that $|Y| = t$, there exists a block $B \in \mathcal{B}$ such that $|B \cap Y| = \lfloor \frac{t}{2} \rfloor$.

We will use the notation $(N; m, t)$-SS to denote an (m,t)-splitting system having N blocks.

Here is a simple construction for splitting systems (see [13]).

Lemma 3.1 (Coppersmith). *For all even integers m and t with $0 < t < m$, there exists an $(\frac{m}{2}; m, t)$-SS.*

Proof. Let $X = \{0, \ldots, m-1\}$ and define

$$B_i = \left\{ i + j \bmod m : 0 \leq j \leq \frac{m}{2} - 1 \right\}$$

for $i = 0, 1, \ldots$. Let $\mathcal{B} = \{B_i : 0 \leq i \leq \frac{m}{2} - 1\}$. Is is shown in [13] that $(X, \mathcal{B})$ is an (m, t)-splitting system for any even integer $t < m$. □

Example 3.1. The set of five blocks $\{0, 1, 2, 3, 4\}$, $\{1, 2, 3, 4, 5\}$, $\{2, 3, 4, 5, 6\}$, $\{3, 4, 5, 6, 7\}$, $\{4, 5, 6, 7, 8\}$ yields a $(5, 10, t)$-SS for $t = 2, 4, 6, 8$.

Splitting systems with $t = 4$ are investigated in [8], and systems with $t = 3$ are studied in [2]. Some asymptotic results for general t can be found in [13].

4 The algorithm

In Figure 1, we present a high-level view of our algorithm (this algorithm is a modification of the algorithm for the low hamming weight DLP presented in [13]). Some issues related to efficient implementation of the algorithm will be addressed in later sections. Our algorithm uses the splitting systems constructed in Lemma 3.1, so we will assume that m and t are even integers. In this algorithm, we also assume that the unknown representation $\mathbf{x} = \mathsf{NAF}(\log_\alpha \beta)$ has length not exceeding $m - 1$. Lemma 2.2 can be applied, because the blocks in the splitting systems from Lemma 3.1 are intervals; this proves the correctness of the algorithm.

The main implementation details relate to efficient generation of the subsets and partitions in steps $2, 4, 10$ and 12, as well as efficient updating of x and y in steps $5, 6, 13$ and 14. For the algorithm presented in [13], it was observed in [14] that a certain type of Gray code is needed to accomplish efficient updates in the algorithm. Here, we require a more complicated type of Gray code, which is developed in the next section.

5 A Gray code for NAFs

The term "Gray code" is commonly used to refer to an ordering of all the combinatorial objects of a given type in such a way that only a "minimal change" is required when one proceeds from one item in the list to the next item in the list.

A *binary Gray code of length n* gives an ordering of all 2^n n-tuples in $\{0, 1\}^n$, such that any two consecutive n-tuples have hamming distance equal to 1. For example, the list

$$[000, 001, 011, 010, 110, 111, 101, 100]$$

is a binary Gray code for $n = 3$. (Each element in this list is a binary 3-tuple, where we write "(a, b, c)" as "abc" for brevity.)

Next, we consider the set $\binom{Y}{k}$ of all $\binom{n}{k}$ k-subsets of the n-set $Y = \{0, \ldots, n-1\}$. Two k-subsets $Z, Z' \in \binom{Y}{k}$ are *neighbours* if $|Z \cap Z'| = k - 2$. A *k-combination Gray code* for $\binom{Y}{k}$ consists of an ordering of the k-subsets in $\binom{Y}{k}$ such that any two consecutive k-subsets in the ordering are neighbours.

Figure 1: Solving the FHW-NAF-DLP

INPUT: $\alpha, \beta \in G$ and even integers m, t

0. FOR $i \leftarrow 0, \ldots, \frac{m}{2} - 1$ DO
1. Initialize $\mathcal{L}_1$ to be an empty list
2. FOR ALL nonadjacent subsets $Y \subseteq \{0, \ldots, \frac{m}{2} - 1\}$ such that $|Y| = \frac{t}{2}$ DO
3. $Y_1 \leftarrow Y + i$
4. FOR ALL partitions $Y_1 = Y_1^+ \cup Y_1^-$ DO
5. $x \leftarrow \mathsf{val}(Y_1^+) - \mathsf{val}(Y_1^-)$
6. $g \leftarrow \alpha^x$
7. Append (x, g) to the list $\mathcal{L}_1$
8. Sort the list $\mathcal{L}_1$ in increasing order of second co-ordinates
9. Initialize $\mathcal{L}_2$ to be an empty list
10. FOR ALL nonadjacent subsets $Y \subseteq \{0, \ldots, \frac{m}{2} - 1\}$ such that $|Y| = \frac{t}{2}$ DO
11. $Y_2 \leftarrow Y + i + \frac{m}{2} \bmod m$
12. FOR ALL partitions $Y_2 = Y_2^+ \cup Y_2^-$ DO
13. $x \leftarrow \mathsf{val}(Y_2^+) - \mathsf{val}(Y_2^-)$
14. $g \leftarrow \beta\alpha^{-x}$
15. Append (x, g) to the list $\mathcal{L}_2$
16. Sort the list $\mathcal{L}_2$ in increasing order of second co-ordinates
17. If possible, find $(x_1, g_1) \in \mathcal{L}_1$ and $(x_2, g_2) \in \mathcal{L}_2$ such that $g_1 = g_2$
18. If step 17 was successful, output $\log_\alpha \beta = x_1 + x_2$ and QUIT
19. Otherwise, proceed to the next iteration of the FOR loop (step 0.)

Suppose $Z, Z' \in \binom{Y}{k}$ are two neighbouring k-sets. Write the elements in these k-subsets in increasing order, i.e.,

$$Z = \{a_0 < a_1 < \cdots < a_{k-1}\}$$

and

$$Z' = \{a'_0 < a'_1 < \cdots < a'_{k-1}\}.$$

Then we say that Z and Z' are *strong neighbours* if there is an index j such that $a_i = a'_i$ whenever $i \neq j$. A *k-combination strong Gray code* for $\binom{Y}{k}$ consists of an ordering of the k-subsets in $\binom{Y}{k}$ such that any two consecutive k-subsets in the ordering are strong neighbours.

For example $\{1, 2, 3\}$ and $\{2, 3, 4\}$ are neighbours, but they are not strong neighbours. The subsets $\{1, 2, 5\}$ and $\{1, 3, 5\}$ are strong neighbours.

Example 5.1. *We present a 3-combination strong Gray code for $n = 6$ on the set $Y = \{0, 1, 2, 3, 4, 5\}$:*

$$[345, 145, 045, 245, 235, 135, 035, 015, 025, 125,$$
$$123, 023, 013, 012, 014, 024, 124, 134, 034, 234].$$

(Each element in this list is a set of size three, where we write "$\{a, b, c\}$" as "abc" for brevity.)

Let $Y = \{0, \ldots, n-1\}$ and let $Z \in \binom{Y}{k}$. Recall from Section 2 that Z is a nonadjacent k-subset if there does not exist an integer i such that $\{i, i+1 \bmod n\} \subseteq Z$. The number of nonadjacent k-subsets of Y is $\binom{n-k+1}{k}$. In fact, it is easy to describe a bijection between the set of all nonadjacent k-subsets of Y and the set all k-subsets of $U = \{0, \ldots, n-k\}$. This can be done as follows. Suppose that

$$V = \{a_0 < a_1 < \cdots < a_{k-1}\}$$

is a k-subset in $\binom{U}{k}$. Define

$$Z = \mathsf{pad}(V) = \{a_0, a_1 + 1, a_2 + 2, \ldots, a_{k-1} + k - 1\}.$$

Then Z is a nonadjacent k-subset of Y. This correspondence

$$V \longleftrightarrow \mathsf{pad}(V)$$

is easily seen to be a bijection.

Lemma 5.1. *For all positive integers n and k such that $n \geq 2k-1$, there is a nonadjacent k-combination Gray code for $Y = \{0, \ldots, n-1\}$.*

Proof. Let $U = \{0, \ldots, n-k\}$ and denote $\ell = \binom{n-k+1}{k}$. Let

$$\left[V^0, V^1, \cdots, V^{\ell-1}\right]$$

be a k-combination strong Gray code for U (see [1, 3, 5]). Then construct the list

$$\left[\mathsf{pad}(V^0), \mathsf{pad}(V^1), \cdots, \mathsf{pad}(V^{\ell-1})\right].$$

This list forms the desired nonadjacent k-combination Gray code. (Note that it is necessary that the k-combination Gray code be strong in order for the result to be true.) □

Example 5.2. *We present a nonadjacent 3-combination Gray code for $n = 8$ on the set $\{0, 1, 2, 3, 4, 5, 6, 7\}$. This is easily done by starting with the 3-combination strong Gray code for $n = 6$ which was presented in Example 5.1, and then applying the technique described in Lemma 5.1. The following is obtained:*

$$[357, 157, 057, 257, 247, 147, 047, 027, 037, 137,$$
$$135, 035, 025, 024, 026, 036, 136, 146, 046, 246].$$

In Examples 5.1 and 5.2, a 3-subset is presented in the form *abc* (which designates the set $\{a, b, c\}$). It will be more useful for our purposes to represent a k-subset of an n-set by its characteristic vector. The *characteristic vector* of a subset $Z \subseteq \{0, \ldots, n-1\}$ is the n-tuple $\mathbf{z} = (z_0, \ldots, z_{n-1}) \in \{0, 1\}^n$ defined as follows:

$$z_i = \begin{cases} 1 & \text{if } i \in Z \\ 0 & \text{if } i \notin Z. \end{cases}$$

Example 5.3. *The nonadjacent 3-combination Gray code for $n = 8$ presented in Example 5.2 can be represented as a list of binary 8-tuples as follows:*

$$[00010101, 01000101, 10000101, 00100101, 00101001,$$
$$01001001, 10001001, 10100001, 10010001, 01010001,$$
$$01010100, 10010100, 10100100, 10101000, 10100010,$$
$$10010010, 01010010, 01001010, 10001010, 00101010].$$

Now we are in a position to describe a Gray code for nonadjacent forms. A (k, n)-*NAF Gray code* will be an ordering of all the NAFs in $\{-1, 0, 1\}^n$ having hamming weight equal to k, such that a certain condition holds. Suppose that that $\mathbf{x}^i$ and $\mathbf{x}^{i+1}$ are consecutive n-tuples in the ordering. Then we require that one of the following two properties is satisfied:

1. $\mathsf{supp}(\mathbf{x}^i) = \mathsf{supp}(\mathbf{x}^{i+1})$ and $\mathsf{d}(\mathbf{x}^i, \mathbf{x}^{i+1}) = 1$, or
2. $\mathsf{supp}(\mathbf{x}^i) \neq \mathsf{supp}(\mathbf{x}^{i+1})$ and $\mathsf{d}(\mathbf{x}^i, \mathbf{x}^{i+1}) = 2$.

The number of vectors in a (k, n)-NAF Gray code is determined in the following lemma.

Lemma 5.2. *Let $a_{n,k}$ denote the number of NAFs of length at most n and weight k. Then $a_{n,k} = 2^k \binom{n-k+1}{k}$.*

Proof. There are $\binom{n-k+1}{k}$ nonadjacent k-subsets of an n-set. For each such k-subset, there are 2^k ways to assign the values ± 1 to the nonzero entries. □

The fact that $a_{n,k} = 2^k \binom{n-k+1}{k}$ suggests that we might be able to construct a (k, n)-NAF Gray code by means of a certain "product" (i.e., a *composition*) of a nonadjacent k-combination Gray code on an n-set together with a binary Gray code of length k. We develop this idea now.

Suppose $\mathbf{x} \in \{0, 1\}^n$, $\mathsf{wt}(\mathbf{x}) = k$, and suppose that the nonzero coordinates of $\mathbf{x}$ are

$$i_0 < i_1 < \cdots < i_{k-1}.$$

Suppose also that $\mathbf{z} = (z_0, \ldots, z_{k-1}) \in \{-1, 1\}^k$. Define a *composition operation* as follows: $\mathbf{x} * \mathbf{z} = \mathbf{y}$, where

$$y_j = \begin{cases} 0 & \text{if } x_j = 0 \\ z_h & \text{if } j = i_h. \end{cases}$$

For example, we have

$$(0, 1, 1, 0, 1, 0) * (1, -1, -1) = (0, 1, -1, 0, -1, 0).$$

(The idea is that the vector $\mathbf{z}$ is embedded into the non-zero coordinates of $\mathbf{x}$.)

Next, let

$$\mathcal{G} = \left[\mathbf{z}^0, \ldots, \mathbf{z}^{2^k-1}\right]$$

be a binary Gray code of length k over the alphabet $\{-1, 1\}$. (For example, in a binary Gray code over $\{0, 1\}$, replace all occurrences of 0 by -1.) Let

$$\mathcal{G}^R = \left[\mathbf{z}^{2^k-1}, \ldots, \mathbf{z}^0\right],$$

i.e., $\mathcal{G}^R$ is derived from $\mathcal{G}$ by reversing the ordering of the vectors. Finally, let

$$\mathcal{C} = \left[\mathbf{x}^0, \ldots, \mathbf{x}^{\binom{n+k-1}{k}-1}\right]$$

be a nonadjacent k-combination Gray code.

Now, we can construct a certain composition of these Gray codes, which we denote by $\mathcal{C} * \mathcal{G}$. $\mathcal{C} * \mathcal{G}$ consists of all vectors of the form $\mathbf{x}^i * \mathbf{z}^j$, arranged in the following order:

$$\begin{bmatrix} \mathbf{x}^0 * \mathbf{z}^0, & \mathbf{x}^0 * \mathbf{z}^1, & \ldots, & \mathbf{x}^0 * \mathbf{z}^{2^k-1}, \\ \mathbf{x}^1 * \mathbf{z}^{2^k-1}, & \mathbf{x}1 * \mathbf{z}^{2^k-2}, & \ldots, & \mathbf{x}^1 * \mathbf{z}^0, \\ \mathbf{x}^2 * \mathbf{z}^0, & \mathbf{x}^2 * \mathbf{z}^1, & \ldots, & \mathbf{x}^2 * \mathbf{z}^{2^k-1}, \\ \mathbf{x}^3 * \mathbf{z}^{2^k-1}, & \mathbf{x}^3 * \mathbf{z}^{2^k-2}, & \ldots, & \mathbf{x}^3 * \mathbf{z}^0, \\ \vdots & \vdots & \ldots & \vdots \end{bmatrix}$$

Basically, we compose $\mathbf{x}^0$ with $\mathcal{G}$, then we compose $\mathbf{x}^1$ with $\mathcal{G}^R$, etc. $\mathbf{x}^i$ will be composed with $\mathcal{G}$ if i is even, and with $\mathcal{G}^R$ if i is odd.

It is straightforward to verify that $\mathcal{C} * \mathcal{G}$ is a (k, n)-NAF Gray code. Therefore we obtain the following theorem.

Theorem 5.3. *For all positive integers n and k such that $n \geq 2k-1$, there is a (k, n)-NAF Gray code.*

Example 5.4. *Suppose we have the following Gray code $\mathcal{G}$ on $\{-1, 1\}$ for $n = 3$, where "-" denotes "-1".*

[---, --1, -11, -1-, 11-, 111, 1-1, 1--].

If we compose the nonadjacent 3-combination Gray code for $n = 8$ (which was constructed in Example 5.3) with $\mathcal{G}$, then we get a $(3, 8)$-NAF Gray code:

[000-0-0-, 000-0-01, 000-0101, 000-010-, 0001010-, 00010101, 00010-01, 00010-0-,
01000-0-, 01000-01, 01000101, 0100010-, 0-00010-, 0-000101, 0-000-01, 0-000-0-,
⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
0010-0-0, 0010-010, 00101010, 001010-0, 00-010-0, 00-01010, 00-0-010, 00-0-0-0]

6 The algorithm, revisited

An efficient implementation of the algorithm in Figure 1 will replace the loops in steps 2 and 4, and the loops in steps 10 and 12, by a an efficient method of generating a $\left(\frac{t}{2}, \frac{m}{2}\right)$-NAF Gray code. It is well-known that binary Gray codes and k-combination Gray codes can be generated efficiently. More precisely, there exist *iterators* that will generate each vector in these Gray codes from the previous one in constant time (see, for example [1, 3, 5, 6, 7, 12]). By using the composition construction that we described in Section 5, we can obtain a constant-time iterator for a NAF Gray code.

Now, each transition from one vector in the NAF Gray code to the next requires that x and g be updated in a suitable way. There are two types of transitions, which were listed before Lemma 5.2. The required updates for steps 5 and 6 are as follows:

1. Suppose that $\mathsf{supp}(\mathbf{x}^i) = \mathsf{supp}(\mathbf{x}^{i+1})$ and $\mathsf{d}(\mathbf{x}^i, \mathbf{x}^{i+1}) = 1$. Here, one coordinate of $\mathbf{x}^i$, say coordinate j, changes from 1 to -1 or from -1 to 1. Then x is updated as

$$x \leftarrow x - 2^{j+1} \quad \text{or} \quad x \leftarrow x + 2^{j+1},$$

respectively. Also, g is updated as

$$g \leftarrow \frac{g}{\alpha^{2^{j+1}}} \quad \text{or} \quad g \leftarrow g\,\alpha^{2^{j+1}},$$

respectively.

2. Suppose that $\mathsf{supp}(\mathbf{x}^i) \neq \mathsf{supp}(\mathbf{x}^{i+1})$ and $\mathsf{d}(\mathbf{x}^i, \mathbf{x}^{i+1}) = 2$. Here, one coordinate of $\mathbf{x}^i$, say coordinate j, changes from 1 to 0 and another coordinate (coordinate j') changes from 0 to 1; or, coordinate j changes from -1 to 0 and coordinate j' changes from 0 to -1. Here the updating operations are

$$x \leftarrow x + 2^i - 2^j \quad \text{or} \quad x \leftarrow x + 2^j - 2^i;$$

and

$$g \leftarrow \frac{g\,\alpha^{2^i}}{\alpha^{2^j}} \quad \text{or} \quad g \leftarrow \frac{g\,\alpha^{2^j}}{\alpha^{2^i}},$$

respectively.

By precomputing all the possible values of $\alpha^{2^j}/\alpha^{2^i}$, the updates to g can be carried out in time $\Theta(1)$.

We can now perform an informal complexity analysis of the algorithm in Figure 1. The overall running time is proportional to

$$m\binom{m/2 - t/2 + 1}{t/2}2^{t/2-1}.$$

The size of the search space is

$$\binom{m-t}{t}2^t,$$

because we are assuming that the NAF has length not exceeding $m-1$. Therefore, the complexity of the algorithm is not too far from the square root of the size of the search space. The main difference is an extra factor of $m/2$ in the running time, which is due to the algorithm iterating over the $m/2$ blocks in the splitting system.

7 Discussion and Conclusion

To illustrate the effectiveness of our approach, let's consider a numerical example. Suppose that $m = 256$ and $t = 24$. Then

$$\binom{m-t}{t}2^t \approx 2^{131.8}$$

and

$$m\binom{m/2-t/2+1}{t/2}2^{t/2-1} \approx 2^{71.8}.$$

The algorithm's running time is a factor of (approximately) 2^6 larger than the square root of the size of the search space.

An alternative apporach would be to replace the specific splitting system we used in our algorithm by another (possibly smaller) splitting system. A "general" splitting system would not consist of blocks that are intervals, however. This means that we could not make use of Lemma 2.2 and instead we would have to rely on Lemma 2.1. The algorithm would then have to consider all the $\{0, \pm 1\}$-representations having hamming weight $t/2$ instead of just the NAFs.

Supposing that the splitting system has N blocks, the running time of the modified algorithm would be

$$N\binom{m/2-1}{t/2}2^{t/2}$$

given that the representation has length not exceeding $m-1$. In order for this to be an improvement, we would need a splitting system with

$$N < \frac{m\binom{m/2-t/2+1}{t/2}}{2\binom{m/2-1}{t/2}}.$$

In the example considered above, we would need a splitting system with fewer than 46 blocks; however, we do not know how to construct such a splitting system.

In general, due to a lack of knowledge about small splitting systems, it seems that the algorithm presented in Figure 1 would be preferred to this alternative approach.

Acknowledgements

Research of DRS is supported by the Natural Sciences and Engineering Research Council of Canada through the grant NSERC-RGPIN #203114-02.

References

[1] P. J. Chase. Combination generation and graylex ordering. *Congressus Numerantium* **69** (1989), 215–242.

[2] D. Deng, D. R. Stinson, P. C. Li, G. H. J. van Rees and R. Wei. Constructions and bounds for splitting systems. Submitted.

[3] P. Eades and B. McKay. An algorithm for generating subsets of fixed size with a strong minimal change property. *Information Processing Letters* **19** (1984), 131–133.

[4] R. Heiman. A note on discrete logarithms with special structure. *Lecture Notes in Computer Science* **658** (1993), 454–457 (Advances in Cryptology – EUROCRYPT '92).

[5] T. A. Jenkyns and D. McCarthy. Generating all k-subsets of $\{1, \ldots, n\}$ with minimal changes. *Ars Combinatoria* **40** (1995), 153–159.

[6] D. E. Knuth. *The Art of Computer Programming, Pre-fascicle 3A. A Draft of Section 7.2.1.3: Generating all Combinations.* Version of Sepetmber 2, 2004.

[7] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration and Search*, CRC Press, 1999.

[8] A. C. H. Ling, P. C. Li and G. H. J. van Rees. Splitting systems and separating systems. *Discrete Mathematics.* **279** (2004), 355–368.

[9] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.

[10] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *RAIRO Informatique Théorique et Applications* **24** (1990), 531–543.

[11] J. A. Muir and D. R. Stinson. Minimality and other properties of the width-w nonadjacent form. To appear in *Mathematics of Computation*.

[12] C. Savage. A survey of combinatorial Gray codes. *SIAM Review* **39** (1997), 605–629.

[13] D. R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. *Mathematics of Computation* **71** (2002), 379–391.

[14] E. Teske. Square-root algorithms for the discrete logarithm problem (a survey). In "Public-Key Cryptography and Computational Number Theory", Walter de Gruyter, 2001, pp. 283–301.