Practical Optimal Bucket Sets with Power-of-Two Radices for LFG-Based MSM Algorithms

Guiwen Luo

guiwen_luo@163.com

Abstract. Fixed point multi-scalar multiplication (MSM) is a major computational component of pairing-based SNARK proof generation. The Luo–Fu–Gong (LFG) framework is a variant of Pippenger's bucket method that uses a multiplier set M and precomputed point multiples to reduce the number of buckets and online point additions. For radix q and a positive multiplier set M, covering all digit residues requires a bucket set of size at least $q/(2|M|)$. Previous constructions approaching this bound based on primitive roots or power-residue classes impose radix restrictions that exclude $q = 2^c$, thereby requiring additional binary-to-radix conversion in software implementations.

We present two main bucket set constructions for power-of-two radices $q = 2^c$, using the multiplier sets $M = \{1, 2, 3\}$ and $M = \{1, 3\}$, respectively. Their cardinalities are $q/6 + O(\log q)$ and $q/4 + 2$, respectively, so both asymptotically attain the counting bound $q/(2|M|)$ while allowing efficient radix-digit extraction through shifts and masks.

We further give two generalizations. Using consecutive powers of 3 as multipliers yields bucket sets of size $q/(2\ell) + O(\log q)$ for every fixed positive integer ℓ. For $\ell = 2^d$ and $c \geq d + 2$, consecutive odd multipliers yield bucket sets of size $q/(2\ell) + d + 1$.

We also present scalar conversion and two strategies for handling the leading digit, allowing the bucket set constructions to be used within the LFG framework.

Keywords: Multi-scalar multiplication · elliptic curves · Pippenger method · SNARK · bucket sets

1 Introduction

Let $\mathbb{G}$ be an additive elliptic-curve group of prime order r. Given fixed points $P_1, P_2, \cdots, P_n \in \mathbb{G}$ and scalars $a_1, a_2, \cdots, a_n \in \mathbb{Z}_r$, where $0 \leq a_i < r$, fixed point multi-scalar multiplication (MSM) computes

$$S_{n,r} := \sum_{i=1}^{n} a_i P_i. \tag{1}$$

The term *fixed point* means that $P_1, P_2, \cdots, P_n$ remain unchanged across MSM instances and therefore may be precomputed, whereas the scalars $a_1, a_2, \cdots, a_n$ vary between instances.

Fixed point MSM is one of the principal computations in pairing-based SNARK proof generation. In many SNARK constructions, the points are fixed by a structured or universal reference string, whereas the scalars depend on the witness.

Precomputation can reduce the online cost of fixed point MSM. In the extreme case, one could store every scalar multiplication aP_i for $a \in \mathbb{Z}_r$, reducing the online computation to n table lookups and $n-1$ point additions. However, this requires approximately nr stored points and is infeasible for cryptographic size groups. Practical methods therefore store only structured subsets of point multiples [Pip76,BGMW93,LFG23,FKSX25]. Table 1 summarizes the precomputation storage and approximate online point addition counts of representative methods, including our constructions in this paper. Beyond algorithmic improvements, GPUs provide a widely used parallelization platform for accelerating MSM [LWY+23,ZHY+24].

1.1 The LFG Method

Luo, Fu and Gong (LFG) introduced a multiplier–bucket method for fixed point MSM that trades offline precomputation storage for fewer online point additions [LFG23].

Let $q = 2^c$ and $h = \lceil \log_q r \rceil$. We use the positive multiplier set

$$M = \{m_1, m_2, \cdots, m_{|M|}\}$$

and record the sign separately by $\varepsilon \in \{-1, 1\}$. This is equivalent to the signed multiplier set used in the original LFG paper [LFG23].

Suppose that a set of nonnegative integers B with $0 \in B$ permits every scalar $0 \le a_i < r$ to be converted into

$$a_i = \sum_{j=0}^{h-1} \varepsilon_{ij} m_{ij} b_{ij} q^j, \ \varepsilon_{ij} \in \{-1, 1\}, \ m_{ij} \in M, \ b_{ij} \in B. \tag{2}$$

Here B is called a bucket set. The MSM can then be rewritten as

$$\begin{aligned} S_{n,r} &= \sum_{i=1}^{n} a_i P_i \\ &= \sum_{i=1}^{n} \sum_{j=0}^{h-1} \varepsilon_{ij} m_{ij} b_{ij} q^j P_i \\ &= \sum_{i=1}^{n} \sum_{j=0}^{h-1} b_{ij} \left(\varepsilon_{ij} m_{ij} q^j P_i\right). \end{aligned} \tag{3}$$

LFG's method precomputes the following $nh|M|$ points

$$\left\{ m q^j P_i : 1 \le i \le n, \ 0 \le j < h, \ m \in M \right\}. \tag{4}$$

Table 1. Comparison of methods for computing $S_{n,r}$.

Method	Carry Handling	Stored Points	Online Additions
Pippenger [Pip76,BDLO12]	—	n	$h(n+q/2)$
Yrrid in ZPrize 2022 [Emm22]	—	$n\lceil h/2\rceil$	$nh+q$
BGMW method [BGMW93] Yrrid/Snarkify (ZPrize 2023) [YS23]	—	nh	$nh+q/2$
LFG method [LFG23]	Strategy I	$3nh$	$nh+0.21q$
LG method [LG24]	—	ℓnh	$nh+q/(2\ell)$
FKSX method [FKSX25]	Strategy II	$\ell n(h+1)$	$nh+s+q/(2\ell)$
This paper, $M=\{1,2,3\}$	Strategy I	$3nh$	$nh+q/6$
This paper, $M=\{1,2,3\}$	Strategy II	$3nh+n$	$nh+s+q/6$
This paper, $M=\{1,3\}$	Strategy I	$2nh$	$nh+q/4$
This paper, $M=\{1,3\}$	Strategy II	$2nh+n$	$nh+s+q/4$
This paper, Appendix B	Strategy I	ℓnh	$nh+q/(2\ell)$
This paper, Appendix B	Strategy II	$\ell nh+n$	$nh+s+q/(2\ell)$
This paper, Appendix C	Strategy I	ℓnh	$nh+q/(2\ell)$
This paper, Appendix C	Strategy II	$\ell nh+n$	$nh+s+q/(2\ell)$

1) n is the number of points, r is the group order, q is the radix, $h=\lceil \log_q r\rceil$ is the number of radix-q digits used to represent each scalar, and $\ell=|M|$ is the cardinality of the positive multiplier set. The quantity s $(0\le s\le n)$ denotes the number of scalars with a nonzero final carry.
2) In the carry handling column, the two strategies are described in Section 4.1. Strategy I denotes the strategy using a leading digit bucket set $B_{\text{lead}}(q,r)$, while Strategy II denotes the strategy using an additional digit. The Strategy I estimates assume that the additional buckets contributed by $B_{\text{lead}}(q,r)$ are negligible compared with q.
3) Stored points include the original n points. Online addition counts are approximate.
4) For Appendix B, the exponentially growing carry range makes the construction better suited to small ℓ. For Appendix C, $\ell=2^d$.
5) The LG and FKSX methods compared here do not support $q=2^c$ and require additional binary-to-radix conversion, whose implementation overhead is not negligible and not reflected in these point addition counts. Other methods support $q=2^c$.

Every point $\varepsilon_{ij} m_{ij} q^j P_i$ in (3) can thus be obtained by a table lookup, and followed by point negation if $\varepsilon_{ij} = -1$. $S_{n,r}$ is converted into an nh-scalar multiplication, where every scalar b_{ij} is in the bucket set B.

For each nonzero $b \in B$, denote

$$S_b = \sum_{1 \le i \le n,\ 0 \le j < h,\ b_{ij} = b} \varepsilon_{ij} m_{ij} q^j P_i, \tag{5}$$

then

$$S_{n,r} = \sum_{b \in B \setminus \{0\}} b S_b.$$

Elements in B may not be consecutive. Sort out B in ascending order as

$$B = \{0 = b_1 < b_2 < \cdots < b_{|B|}\},$$

and define the maximum difference between two neighboring elements as

$$d(B) = \max_{2 \le k \le |B|} (b_k - b_{k-1}). \tag{6}$$

$S_{n,r}$ can be evaluated using

$$S_{n,r} = \sum_{k=2}^{|B|} b_k S_{b_k} = \sum_{\ell=2}^{|B|} (b_\ell - b_{\ell-1}) \left(\sum_{k=\ell}^{|B|} S_{b_k} \right).$$

It was shown in [LFG23] that this can be evaluated by at most

$$nh + |B| + d(B) - 4 \tag{7}$$

point additions.

LFG has instantiated this method with $M = \{1, 2, 3\}$ and constructed a bucket set B such that $|B| \approx 0.21q$ and $d(B) \le 6$, which yields an algorithm to compute $S_{n,r}$ using approximately

$$nh + 0.21q \tag{8}$$

additions. Compared with Pippenger's method and BGMW method, LFG's construction reduces the cardinality of the bucket set from $q/2$ to $0.21q$.

1.2 Motivation and Contribution

An essential question in the LFG framework is how small the bucket set B can be for a given multiplier set M. A simple counting argument gives the necessary condition

$$2|M||B| \ge q$$

for covering all residues modulo q. This suggests that the optimal bucket set satisfies

$$|B| = \frac{q}{2|M|} + o(q).$$

Luo and Gong [LG24] construct such asymptotically optimal bucket sets by regularly selecting powers of 2 for radices q such that 2 is a primitive element of $\mathbb{F}_q$, using positive multiplier sets M consisting of consecutive powers of 2.

Fan, Kuchta, Sica and Xu [FKSX25] give a generalized construction based on power-residue classes modulo odd prime powers $q = p^c$, for multiplier sets $M = \{1, 2, \cdots, |M|\}$ satisfying certain algebraic conditions.

However these two constructions impose restrictions on the radix and do not apply to the binary radix $q = 2^c$. This matters in software implementations, where scalars are stored in binary. For $q = 2^c$, standard radix digits can be extracted directly using shifts and masks. Other radices require additional conversion arithmetic whose cost is not negligible. This motivates an explicit bucket construction that approaches the optimal bound while retaining a binary radix.

Contribution. For $q = 2^c$, we give explicit bucket constructions $B_{\mathrm{mod}}(q)$ for two multiplier sets. For $M = \{1, 2, 3\}$, the bucket set construction satisfies

$$|B_{\mathrm{mod}}(q)| = \left\lceil \frac{q}{6} \right\rceil + \left\lfloor \frac{c}{2} \right\rfloor .$$

For $M = \{1, 3\}$, we construct a set satisfying

$$|B_{\mathrm{mod}}(q)| = \frac{q}{4} + 2.$$

Both constructions asymptotically achieve size $q/(2|M|)$.

We also give two families of asymptotically optimal bucket set constructions for larger multiplier sets in Appendices B and C, using consecutive powers of 3 and consecutive odd multipliers, respectively. Since larger multiplier sets require more precomputed points, these extensions are primarily of theoretical interest and provide alternative choices when sufficient storage is available for the larger precomputation tables.

Notation. For an integer x and an integer $q \geq 2$, define

$$\langle x \rangle_q = \min\{x \bmod q,\ q - (x \bmod q)\},\ 0 \leq \langle x \rangle_q \leq q/2. \tag{9}$$

For a prime p and a positive integer x, let $\omega_p(x)$ denote the largest integer e such that p^e divides x. We use $B_{\mathrm{mod}}(q)$ for the bucket set constructed for the multiplier set M under consideration. We omit the dependence on M in the notation, and specify M explicitly when necessary.

2 A Bucket Set for $M = \{1, 2, 3\}$

Throughout this section, we fix the multiplier set

$$M = \{1, 2, 3\},$$

and record the sign of each digit separately. Let

$$G_q = \{1, 3, 5, \cdots, q/2 - 1\}$$

be the set of positive odd integers smaller than $q/2$.

Lemma 1. *Let $q = 2^c$ with $c \geq 4$, then*

$$G_q = \left\{ \langle 3^j \rangle_q : 0 \leq j < \frac{q}{4} \right\}. \tag{10}$$

Proof. Since every power of 3 is odd, $\langle 3^j \rangle_q$ is a positive odd integer smaller than $q/2$. Therefore,

$$\langle 3^j \rangle_q \in G_q \text{ for every } j \geq 0.$$

Since G_q contains exactly $q/4$ elements, it suffices to show that they are distinct. Suppose, to the contrary, that

$$\langle 3^i \rangle_q = \langle 3^j \rangle_q$$

for some $0 \leq i < j < q/4$, which means

$$3^j \equiv \pm 3^i \pmod{q}.$$

Consequently,

$$3^{j-i} \equiv \pm 1 \pmod{q}. \tag{11}$$

Denote $k = j - i$, then $0 < k < q/4$. We show that neither sign in (11) is possible.

First,

$$3^k \equiv \begin{cases} 3 \pmod 8, & k \text{ odd}, \\ 1 \pmod 8, & k \text{ even}. \end{cases}$$

This shows that $3^k \not\equiv -1 \pmod 8$, and thus $3^k \not\equiv -1 \pmod q$.

We next show $3^k \not\equiv 1 \pmod q$. We utilize the fact that for $s \geq 1$,

$$\omega_2\big(3^{2^s} - 1\big) = s + 2. \tag{12}$$

Indeed, the formula holds for $s = 1$, because $3^2 - 1 = 8$. If it holds for s, then

$$3^{2^{s+1}} - 1 = \big(3^{2^s} - 1\big)\big(3^{2^s} + 1\big).$$

Because $3^{2^s} \equiv 1 \pmod 4$, we have $4 \nmid 3^{2^s} + 1$ and thus $\omega_2\big(3^{2^s} + 1\big) = 1$. Therefore ω_2 increases by exactly one, which explains (12). Taking $s = c - 2$ shows that

$$3^{2^{c-2}} \equiv 1 \pmod{2^c}.$$

Since the order of 3 modulo q is the least positive exponent d such that $3^d \equiv 1 \pmod q$, the equation $3^{2^{c-2}} \equiv 1 \pmod{2^c}$ implies $\mathrm{ord}_q(3) \mid 2^{c-2}$, where $\mathrm{ord}_q(3)$ denotes the order of 3 modulo q. Every positive divisor of 2^{c-2} has the form 2^s for some $0 \leq s \leq c - 2$. However, for every $1 \leq s < c - 2$,

$$\omega_2\big(3^{2^s} - 1\big) = s + 2 < c,$$

and when $s = 0$ we have $3 \not\equiv 1 \pmod{2^c}$, hence

$$3^{2^s} \not\equiv 1 \pmod{2^c}.$$

Therefore, no proper divisor of 2^{c-2} is $\mathrm{ord}_q(3)$. It follows that

$$\mathrm{ord}_q(3) = 2^{c-2} = \frac{q}{4}.$$

Since $k < q/4 = \mathrm{ord}_q(3)$, we have $3^k \not\equiv 1 \pmod{q}$.

□

We select about half of G_q to construct an auxiliary set A_q and then define the set $B_{\mathrm{mod}}(q)$. Let $H = q/8$, and define

$$A_q = \{1\} \cup \left\{ \langle 3^{2i+1} \rangle_q : 0 \le i < \frac{H}{2} \right\} \cup \left\{ \langle 3^{H+2i} \rangle_q : 0 \le i < \frac{H}{2} \right\}. \tag{13}$$

Define the set $B_{\mathrm{mod}}(q)$ recursively by [1]

$$B_{\mathrm{mod}}(q) = \begin{cases} \{0,1,2,4,7\}, & q = 16, \\ \{0,1,3,4,5,7,8,13\}, & q = 32, \\ A_q \cup \{4b : b \in B_{\mathrm{mod}}(q/4)\}, & q \ge 64. \end{cases} \tag{14}$$

Every integer $x \in \{1, 2, \cdots, q/2\}$ with $\omega_2(x) = 0$ or $\omega_2(x) = 1$ can be written as $x = \langle mb \rangle_q$ for some $m \in M$ and $b \in A_q$. Every remaining nonzero integer in this range is divisible by 4 and is handled recursively using $\{4b : b \in B_{\mathrm{mod}}(q/4)\}$.

Theorem 1. *Let $q = 2^c$ with $c \ge 4$. The set $B_{\mathrm{mod}}(q)$ defined in (14) satisfies*

$$\{0, 1, \cdots, q/2\} \subseteq \left\{ \langle mb \rangle_q : m \in \{1,2,3\},\ b \in B_{\mathrm{mod}}(q) \right\}. \tag{15}$$

Proof. We proceed by induction on c. The two base cases $c = 4$ and $c = 5$ can be verified directly:

$$\left\{ \langle mb \rangle_{16} : m \in \{1,2,3\},\ b \in B_{\mathrm{mod}}(16) \right\} = \{0, 1, \cdots, 8\},$$

and

$$\left\{ \langle mb \rangle_{32} : m \in \{1,2,3\},\ b \in B_{\mathrm{mod}}(32) \right\} = \{0, 1, \cdots, 16\}.$$

Suppose now that $c \ge 6$, $q = 2^c \ge 64$, and assume that $B_{\mathrm{mod}}(q/4)$ satisfies (15) for modulus $q/4$. Let $x \in \{0, 1, \cdots, q/2\}$.

First suppose that $\omega_2(x) = 0$, so that x is odd. By (10), there is a unique integer j such that

$$x = \langle 3^j \rangle_q,\ 0 \le j < \frac{q}{4} = 2H.$$

[1] The base cases $B_{\mathrm{mod}}(16)$ and $B_{\mathrm{mod}}(32)$ are not unique. They may be replaced by sets of the same cardinalities that contain zero and satisfy the covering property in (15). Such replacements preserve the cardinality throughout the recursion, although the bucket values and maximum gaps may differ.

If $j = 0$, $j \in \{1, 3, \cdots, H-1\}$, or $j \in \{H, H+2, \cdots, 2H-2\}$, in other words j is one of the selected exponents in the definition of A_q, then $\langle 3^j \rangle_q \in A_q$, and x is expressed using multiplier 1. For every remaining value of j, the exponent $j-1$ is one of the selected exponents in the definition of A_q. Hence

$$b = \langle 3^{j-1} \rangle_q \in A_q$$

and

$$x = \langle 3b \rangle_q.$$

Next suppose that $\omega_2(x) = 1$. Write $x = 2y$, where y is odd and $1 \le y < q/4$. Applying (10) with modulus $q/2$, there is a unique integer j satisfying

$$y = \langle 3^j \rangle_{q/2}, \ 0 \le j < H.$$

It follows that

$$x = 2\langle 3^j \rangle_{q/2} = \langle 2 \cdot 3^j \rangle_q. \tag{16}$$

- If $j = 0$ or j is odd, then
$$\langle 3^j \rangle_q \in A_q,$$
and (16) represents x using multiplier 2.
- If $j > 0$ is even, then $\langle 3^j \rangle_q \notin A_q$. But by the definition of A_q, we have
$$\langle 3^{H+j} \rangle_q \in A_q.$$
The order of 3 modulo $q/2$ is
$$\frac{q/2}{4} = H.$$
Therefore,
$$3^{H+j} \equiv 3^j \pmod{q/2}.$$
Multiplying by 2 gives
$$2 \cdot 3^{H+j} \equiv 2 \cdot 3^j \pmod{q},$$
and consequently
$$\langle 2 \cdot 3^{H+j} \rangle_q = \langle 2 \cdot 3^j \rangle_q = x.$$

It remains to consider $x = 0$ or $\omega_2(x) \ge 2$. Since $4 \mid x$, write

$$x = 4y, \ 0 \le y \le q/8.$$

By the induction hypothesis, for $y \in \{0, 1, \cdots, q/8\}$ and modulus $q/4$, there exist

$$b \in B_{\mathrm{mod}}(q/4) \text{ and } m \in \{1, 2, 3\}$$

such that

$$y = \langle mb \rangle_{q/4}.$$

For every integer z, we have

$$4\langle z\rangle_{q/4} = \langle 4z\rangle_q.$$

Therefore,

$$x = 4y = 4\langle mb\rangle_{q/4} = \langle m(4b)\rangle_q,$$

where $4b \in B_{\mathrm{mod}}(q)$. □

Corollary 1. *For every integer $t \in [-1, q+1]$, there exist*

$$\varepsilon \in \{-1, 1\},\ m \in \{1, 2, 3\},\ b \in B_{\mathrm{mod}}(q),\ \alpha \in \{-1, 0, 1, 2\}$$

such that

$$t = \varepsilon mb + \alpha q. \tag{17}$$

Proof. Let $x = \langle t\rangle_q$. By Theorem 1, there exist $b \in B_{\mathrm{mod}}(q)$ and $m \in \{1, 2, 3\}$ such that

$$x = \langle mb\rangle_q.$$

By the definition of $\langle\cdot\rangle_q$, the equality $\langle t\rangle_q = \langle mb\rangle_q$ implies

$$t \equiv \varepsilon mb \pmod{q}$$

for some $\varepsilon \in \{-1, 1\}$. It follows that

$$\alpha = \frac{t - \varepsilon mb}{q}$$

is an integer and

$$t = \varepsilon mb + \alpha q.$$

Next we show $\alpha \in \{-1, 0, 1, 2\}$. Since

$$0 \le b \le q/2 \text{ and } |m| \le 3,$$

we have

$$-\frac{3q}{2} \le \varepsilon mb \le \frac{3q}{2}.$$

Together with $-1 \le t \le q+1$ and $q \ge 16$, this gives

$$\alpha = \frac{t - \varepsilon mb}{q} \ge \frac{-1 - 3q/2}{q} = -\frac{3}{2} - \frac{1}{q} > -2,$$

and

$$\alpha = \frac{t - \varepsilon mb}{q} \le \frac{q + 1 + 3q/2}{q} = \frac{5}{2} + \frac{1}{q} < 3.$$

Because α is an integer, these inequalities imply

$$\alpha \in \{-1, 0, 1, 2\}.$$

□

Theorem 2. *Let $q = 2^c$ with $c \geq 4$. Then $B_{\mathrm{mod}}(q)$ defined in (14) satisfies*

$$|B_{\mathrm{mod}}(q)| = \left\lceil \frac{q}{6} \right\rceil + \left\lfloor \frac{c}{2} \right\rfloor. \tag{18}$$

Proof. The selected exponents in the definition of A_q are distinct and lie in $\{0, 1, \cdots, q/4 - 1\}$. By (10), they give distinct elements. Therefore,

$$|A_q| = 1 + \frac{H}{2} + \frac{H}{2} = \frac{q}{8} + 1.$$

Every element of A_q is odd, whereas every element of

$$\{4b : b \in B_{\mathrm{mod}}(q/4)\}$$

is divisible by 4. These two sets are disjoint. The recursive definition thus gives

$$|B_{\mathrm{mod}}(q)| = \frac{q}{8} + 1 + |B_{\mathrm{mod}}(q/4)|, \ q \geq 64. \tag{19}$$

Denote

$$L = \left\lfloor \frac{c}{2} \right\rfloor - 2, \ q_0 = \frac{q}{4^L} \in \{16, 32\}.$$

Expanding the recurrence L times yields

$$\begin{aligned} |B_{\mathrm{mod}}(q)| &= \sum_{\ell=0}^{L-1} \left(\frac{q}{8 \cdot 4^\ell} + 1 \right) + |B_{\mathrm{mod}}(q_0)| \\ &= \frac{q}{6} \left(1 - 4^{-L}\right) + L + |B_{\mathrm{mod}}(q_0)| \\ &= \frac{q}{6} + L + |B_{\mathrm{mod}}(q_0)| - \frac{q_0}{6}. \end{aligned}$$

If c is even, then $q_0 = 16$ and

$$|B_{\mathrm{mod}}(q_0)| - \frac{q_0}{6} = 5 - \frac{16}{6} = \frac{7}{3}.$$

Since $q \equiv 4 \pmod 6$, we also have

$$\left\lceil \frac{q}{6} \right\rceil = \frac{q}{6} + \frac{1}{3}.$$

If c is odd, then $q_0 = 32$ and

$$|B_{\mathrm{mod}}(q_0)| - \frac{q_0}{6} = 8 - \frac{32}{6} = \frac{8}{3}.$$

In this case, $q \equiv 2 \pmod 6$, so

$$\left\lceil \frac{q}{6} \right\rceil = \frac{q}{6} + \frac{2}{3}.$$

Thus in both cases,

$$|B_{\mathrm{mod}}(q)| = \left\lceil \frac{q}{6} \right\rceil + \left\lfloor \frac{c}{2} \right\rfloor.$$

□

Property 1. For $q = 2^c$ $(10 \le c \le 30)$, the maximum difference between two neighboring elements of $B_{\mathrm{mod}}(q)$ defined in (14) is at most 47.

Proof. This is checked by computer program. The individual values of $d(B_{\mathrm{mod}}(q))$ are reported in Table 2 in the Appendix. □

3 A Bucket Set for $M = \{1, 3\}$

Throughout this section, we fix the multiplier set

$$M = \{1, 3\}.$$

For a positive integer b, write

$$b = 2^{\omega_2(b)} u,$$

where u is odd. Define

$$B_{\mathrm{mod}}(q) = \{0\} \cup \left\{ b : 1 \le b \le \frac{q}{2},\ \frac{b}{2^{\omega_2(b)}} \equiv \pm 1 \pmod 8 \right\}. \tag{20}$$

In other words, a positive integer belongs to the bucket set if and only if its odd part u is congruent to 1 or 7 modulo 8.

Theorem 3. *Let $q = 2^c$ with $c \ge 3$. Then the set $B_{\mathrm{mod}}(q)$ defined in (20) satisfies*

$$\{0, 1, \cdots, q/2\} \subseteq \{\langle mb \rangle_q : m \in \{1, 3\},\ b \in B_{\mathrm{mod}}(q)\}. \tag{21}$$

Proof. Let $x \in \{0, 1, \cdots, q/2\}$. The case $x = 0$ follows from $0 \in B_{\mathrm{mod}}(q)$. For $x > 0$, write

$$x = 2^v u, \text{ where } v = \omega_2(x) \text{ and } u \text{ is odd.}$$

If $u \equiv 1$ or $7 \pmod 8$, then $x \in B_{\mathrm{mod}}(q)$. Hence

$$x = \langle 1 \cdot x \rangle_q.$$

It remains to consider $u \equiv 3$ or $5 \pmod 8$. In this case $u \ge 3$. Define $k = c - v$. Since $x \le q/2$, we have $u = x/2^v \le 2^{c-1}/2^v = 2^{k-1}$, and therefore $k \ge 3$. Because 3 is invertible modulo 2^k, there is a unique residue $z \in \{0, 1, \cdots, 2^k - 1\}$ satisfying

$$3z \equiv u \pmod{2^k}.$$

The residue z is odd. Moreover, $3^{-1} \equiv 3 \pmod 8$, so

$$z \equiv 3^{-1} u \equiv 3u \equiv 1 \text{ or } 7 \pmod 8.$$

Define

$$b = 2^v \langle z \rangle_{2^k}.$$

The integer $\langle z \rangle_{2^k}$ is odd and is congruent to z or $-z$ modulo 8. Consequently, its residue modulo 8 belongs to $\{1, 7\}$, and

$$b \in B_{\mathrm{mod}}(q).$$

Since

$$\langle z \rangle_{2^k} \equiv \pm z \pmod{2^k} \text{ and } 3z \equiv u \pmod{2^k},$$

we have

$$3\langle z \rangle_{2^k} \equiv \pm u \pmod{2^k}.$$

Multiplying by 2^v and using $2^{v+k} = q$ gives

$$3b \equiv \pm 2^v u = \pm x \pmod{q}.$$

Since $0 < x \leq q/2$, this gives

$$x = \langle 3b \rangle_q.$$

□

Corollary 2. *For every integer $t \in [-1, q+1]$, there exist*

$$\varepsilon \in \{-1, 1\},\ m \in \{1, 3\},\ b \in B_{\mathrm{mod}}(q),\ \alpha \in \{-1, 0, 1, 2\}$$

such that

$$t = \varepsilon m b + \alpha q. \tag{22}$$

Proof. By Theorem 3, there exist $m \in \{1, 3\}$ and $b \in B_{\mathrm{mod}}(q)$ such that

$$\langle t \rangle_q = \langle mb \rangle_q.$$

Hence $t \equiv \varepsilon m b \pmod{q}$ for some $\varepsilon \in \{-1, 1\}$, and

$$\alpha = \frac{t - \varepsilon m b}{q}$$

is an integer. Since $0 \leq b \leq q/2$ and $m \leq 3$,

$$-\frac{3q}{2} \leq \varepsilon m b \leq \frac{3q}{2}.$$

Together with $-1 \leq t \leq q+1$, this implies

$$-\frac{3}{2} - \frac{1}{q} \leq \alpha \leq \frac{5}{2} + \frac{1}{q}.$$

For $q \geq 8$, the lower bound is greater than -2 and the upper bound is smaller than 3. Therefore

$$\alpha \in \{-1, 0, 1, 2\}.$$

□

Theorem 4. *Let $q = 2^c$ with $c \geq 3$. Then $B_{\mathrm{mod}}(q)$ defined in (20) satisfies*

$$|B_{\mathrm{mod}}(q)| = \frac{q}{4} + 2. \tag{23}$$

Proof. For $q \geq 16$, the interval $\{0, 1, \cdots, q/2 - 1\}$ consists of $q/16$ complete blocks of eight consecutive integers, i.e., 0 to 7, 8 to 15, $\cdots$, $q/2 - 8$ to $q/2 - 1$. In each block, exactly two odd integers are congruent to 1 or 7 modulo 8. These are precisely the odd elements selected by $B_{\mathrm{mod}}(q)$, so $B_{\mathrm{mod}}(q)$ contains exactly $q/8$ odd elements.

An even integer b belongs to $B_{\mathrm{mod}}(q)$ if and only if $b/2$ belongs to $B_{\mathrm{mod}}(q/2)$, since dividing by 2 preserves its odd part. This also holds for $b = 0$, which belongs to both sets. Consequently,

$$|B_{\mathrm{mod}}(q)| = \frac{q}{8} + |B_{\mathrm{mod}}(q/2)|.$$

The base case is

$$B_{\mathrm{mod}}(8) = \{0, 1, 2, 4\},\ |B_{\mathrm{mod}}(8)| = 4 = \frac{8}{4} + 2.$$

By the induction hypothesis,

$$|B_{\mathrm{mod}}(q/2)| = \frac{q/2}{4} + 2 = \frac{q}{8} + 2.$$

Therefore,

$$|B_{\mathrm{mod}}(q)| = \frac{q}{8} + \left(\frac{q}{8} + 2\right) = \frac{q}{4} + 2.$$

□

Property 2. For $q = 2^c$ $(10 \leq c \leq 30)$, the maximum difference between two neighboring elements of $B_{\mathrm{mod}}(q)$ defined in (20) is

$$d\big(B_{\mathrm{mod}}(q)\big) = 5. \tag{24}$$

Proof. This is checked by computer program. □

4 Scalar conversion

Throughout this section, we fix either $M = \{1, 2, 3\}$ or $M = \{1, 3\}$ and use the corresponding bucket set constructed in Sections 2 and 3, respectively. Both constructions admit the carry set $\{-1, 0, 1, 2\}$, so the same scalar conversion algorithm applies.

Let

$$h = \lceil \log_q r \rceil,\ r_{h-1} = \left\lfloor \frac{r}{q^{h-1}} \right\rfloor.$$

Algorithm 1 Scalar conversion with $B_{\mathrm{mod}}(q)$

Require: $\{a_j\}_{0\le j<h}$, where $0 \le a_j < q$ and $a = \sum_{j=0}^{h-1} a_j q^j$
Ensure: $\{(\varepsilon_j, m_j, b_j)\}_{0\le j<h}$ and α_{h-1} such that $a = \sum_{j=0}^{h-1} \varepsilon_j m_j b_j q^j + \alpha_{h-1} q^h$
1: **for** $j = 0$ to $h-1$ **do**
2: Obtain $\varepsilon_j, m_j, b_j, \alpha_j$ such that $a_j = \varepsilon_j m_j b_j + \alpha_j q$, $\varepsilon_j \in \{-1, 1\}$, $m_j \in M$, $b_j \in B_{\mathrm{mod}}(q)$, $\alpha_j \in \{-1, 0, 1, 2\}$
3: **if** $j < h-1$ **then**
4: $a_{j+1} \leftarrow a_{j+1} + \alpha_j$
5: **return** $\{(\varepsilon_j, m_j, b_j)\}_{0\le j<h}$, α_{h-1}

For a scalar $0 \le a < r$, write its standard radix-q expansion as

$$a = \sum_{j=0}^{h-1} a_j q^j, \ 0 \le a_j < q. \tag{25}$$

In particular, $a_{h-1} \le r_{h-1}$.

The scalar conversion algorithm is described in Algorithm 1.

The correctness of Algorithm 1 follows from Corollary 1 for $M = \{1, 2, 3\}$ and Corollary 2 for $M = \{1, 3\}$. The initial digit belongs to $[0, q-1]$. At every subsequent position, the original digit lies in $[0, q-1]$, and the incoming carry lies in $\{-1, 0, 1, 2\}$. Thus the updated digit belongs to $[-1, q+1]$, so the required decomposition always exists. At each j, replacing the current digit by $\varepsilon_j m_j b_j$ and adding α_j to the next digit preserves the represented integer, since

$$a_j q^j = \varepsilon_j m_j b_j q^j + \alpha_j q^{j+1}.$$

For $j = h-1$, the carry α_{h-1} is retained explicitly. Therefore, the output satisfies

$$a = \sum_{j=0}^{h-1} \varepsilon_j m_j b_j q^j + \alpha_{h-1} q^h. \tag{26}$$

For an efficient implementation, the decompositions used in Algorithm 1 can be precomputed. We can create a hash table $\mathcal{H}$ to store a valid decomposition for each integer $t \in [-1, q+1]$,

$$\mathcal{H}(t) = (\varepsilon, m, b, \alpha) \text{ such that } t = \varepsilon m b + \alpha q,$$

where

$$\varepsilon \in \{-1, 1\}, \ m \in M, \ b \in B_{\mathrm{mod}}(q), \ \alpha \in \{-1, 0, 1, 2\}.$$

The table is independent of the input scalar and can be stored in an array of $q+3$ entries. Since the algorithm adds each carry directly to the next digit, the decomposition at position j is obtained by looking up the current value of a_j. For zero input, we store $\mathcal{H}(0) = (1, 1, 0, 0)$. This replaces the online search for a decomposition by table lookup.

4.1 Handling the leading digit

The final carry α_{h-1} in Algorithm 1 need not be zero. We consider two strategies adapted from previous work to handle this situation.

Strategy I. Following LFG method [LFG23], we can add a leading digit bucket set $B_{\text{lead}}(q, r)$ and use

$$B = B_{\text{mod}}(q) \cup B_{\text{lead}}(q, r)$$

to terminate within h digits. $B_{\text{lead}}(q, r)$ is chosen for the selected multiplier set M, so that every integer

$$-1 \le t \le r_{h-1} + 2$$

has a direct representation $t = \varepsilon m b$ with $\varepsilon \in \{-1, 1\}$, $m \in M$, and $b \in B_{\text{lead}}(q, r)$. Using this direct representation at position $h-1$ ensures that no final carry α_{h-1} remains. This method requires $nh|M|$ precomputed points and has the online point-addition bound

$$T_{\text{lead}} \le nh + |B| + d(B) - 4. \tag{27}$$

When the number of additional buckets contributed by $B_{\text{lead}}(q, r)$ and the maximum gap term $d(B)$ are small relative to the main terms, the estimates are approximately $nh + q/6$ for $M = \{1, 2, 3\}$ and $nh + q/4$ for $M = \{1, 3\}$.

Strategy II. Alternatively, following FKSX method [FKSX25], we represent the final carry in an additional radix position and use

$$B = B_{\text{mod}}(q) \cup \{2\}.$$

Since $0, 1, 2 \in B$, every final carry can be represented using multiplier 1 and bucket value $|\alpha_{h-1}|$, with negative contributions handled by point negation. The additional radix position therefore requires the precomputation of

$$\{q^h P_i : 1 \le i \le n\}.$$

The total precomputation contains $nh|M| + n$ points, $3nh + n$ for $M = \{1, 2, 3\}$ and $2nh + n$ for $M = \{1, 3\}$. For the latter construction, bucket 2 is already in $B_{\text{mod}}(q)$, so $B = B_{\text{mod}}(q)$. Let s be the number of scalars with nonzero final carry α_{h-1}. Each such scalar contributes one additional point to bucket 1 or 2, while a zero final carry contributes nothing. The online point-addition bound is

$$T_{\text{extra}} \le nh + s + |B| + d(B) - 4, \ 0 \le s \le n. \tag{28}$$

When $d(B)$ is small, the estimates are approximately $nh+s+q/6$ for $M = \{1, 2, 3\}$ and $nh + s + q/4$ for $M = \{1, 3\}$. The worst case has $s = n$, but the actual count can be lower because not every scalar produces a nonzero final carry α_{h-1}.

The relative performance of the aforementioned two strategies depends on the curve's group order and the chosen parameters. Determining which is faster requires further analysis and experiments under specific settings.

5 Conclusion

In this paper, we presented two main bucket set constructions for the radix $q = 2^c$. For $M = \{1, 2, 3\}$, the construction satisfies

$$|B_{\mathrm{mod}}(q)| = \left\lceil \frac{q}{6} \right\rceil + \left\lfloor \frac{c}{2} \right\rfloor,$$

while for $M = \{1, 3\}$, it satisfies

$$|B_{\mathrm{mod}}(q)| = \frac{q}{4} + 2.$$

Both constructions asymptotically achieve the counting bound $q/(2|M|)$. Combined with the LFG framework and the leading digit handling strategies discussed in Section 4.1, these constructions yield $S_{n,r}$ computation algorithms whose online point addition counts are approximately $nh + q/6$ and $nh + q/4$ under Strategy I. The corresponding complexities are approximately $nh + s + q/6$ and $nh + s + q/4$ under Strategy II, where s is the number of scalars with a nonzero final carry.

The appendix bucket set constructions extend these results to larger multiplier sets. Larger multiplier sets offer diminishing returns. For fixed n, q and h, reducing the bucket term towards $q/(2|M|)$ does not reduce the main term nh. Moreover, the marginal reduction in $q/(2|M|)$ decreases as $|M|$ grows, while the required precomputation storage increases. The two small multiplier sets emphasized in the main text offer more practical choices by keeping precomputation storage modest, while the generalized constructions provide additional choices when more storage is available.

References

BDLO12. Daniel J. Bernstein, J. M. Doumen, Tanja Lange, and Jan-Jaap Oosterwijk. Faster batch forgery identification. In Steven Galbraith and Mridul Nandi, editors, *Progress in Cryptology—INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 454–473, Berlin, Heidelberg, 2012. Springer.

BGMW93. Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley, and David B. Wilson. Fast exponentiation with precomputation. In Rainer A. Rueppel, editor, *Advances in Cryptology—EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 200–207, Berlin, Heidelberg, 1993. Springer.

Emm22. Niall Emmart. Z-Prize MSM on the GPU Submission. https://github.com/yrrid/submission-msm-gpu, 2022. Yrrid Software. ZPrize 2022 GPU MSM submission.

FKSX25. Xinxin Fan, Veronika Kuchta, Francesco Sica, and Lei Xu. Speeding up multi-scalar multiplications for pairing-based zkSNARKs. *Journal of Cryptology*, 38(2):21, 2025.

LFG23. Guiwen Luo, Shihui Fu, and Guang Gong. Speeding up multi-scalar multiplication over fixed points towards efficient zkSNARKs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):358–380, 2023.

LG24. Guiwen Luo and Guang Gong. On the optimization of Pippenger's bucket method with precomputation. In Charles J. Colbourn and Jeffrey H. Dinitz, editors, *New Advances in Designs, Codes and Cryptography*, volume 86 of *Fields Institute Communications*, pages 327–343. Springer, Cham, 2024.

LWY+23. T. Lu, C. Wei, R. Yu, C. Chen, W. Fang, L. Wang, Z. Wang, and W. Chen. cuZK: Accelerating zero-knowledge proof with a faster parallel multi-scalar multiplication algorithm on GPUs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):194–220, 2023.

Pip76. Nicholas Pippenger. On the evaluation of powers and related problems (preliminary version). In *17th Annual Symposium on Foundations of Computer Science*, pages 258–263. IEEE Computer Society, 1976.

YS23. Yrrid Software and Snarkify. ZPrize 2023 GPU MSM submission. https://github.com/z-prize/2023-entries/tree/main/prize-1-fpga-gpu-proof/prize-1a-msm/gpu/Yrrid/zprize-2023-submission-gpu, 2023. Software submission archived in the ZPrize 2023 entries repository.

ZHY+24. X. Zhu, H. He, Z. Yang, Y. Deng, L. Zhao, and R. Hou. Elastic MSM: A fast, elastic and modular preprocessing technique for multi-scalar multiplication algorithm on GPUs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(4):258–284, 2024.

A Bucket Set Parameters

Table 2. Bucket sets $B_{\mathrm{mod}}(q)$ for $q = 2^c$ and $M = \{1, 2, 3\}$.

q	$\lvert B_{\mathrm{mod}}(q)\rvert$	$d(B_{\mathrm{mod}}(q))$	$\lvert B_{\mathrm{mod}}(q)\rvert/q$
2^{10}	176	16	0.171875
2^{11}	347	18	0.169434
2^{12}	689	16	0.168213
2^{13}	1372	21	0.167480
2^{14}	2738	15	0.167114
2^{15}	5469	19	0.166901
2^{16}	10931	22	0.166794
2^{17}	21854	31	0.166733
2^{18}	43700	24	0.166702
2^{19}	87391	27	0.166685
2^{20}	174773	32	0.166677
2^{21}	349536	27	0.166672
2^{22}	699062	29	0.166669
2^{23}	1398113	37	0.166668
2^{24}	2796215	31	0.166667
2^{25}	5592418	38	0.166667
2^{26}	11184824	35	0.166667
2^{27}	22369635	38	0.166667
2^{28}	44739257	39	0.166667
2^{29}	89478500	47	0.166667
2^{30}	178956986	40	0.166667

B Bucket Sets for Powers of Three Multipliers

Inspired by the bucket set construction of Luo and Gong [LG24], we adapt the consecutive powers of some element as multipliers.

Theorem 5. *Let $q = 2^c$ with $c \geq 3$, and let*

$$M = \{3^i : 0 \leq i \leq \ell - 1\},$$

where ℓ is a positive integer. Define

$$B_{\mathrm{mod}} = \left\{0, \frac{q}{4}, \frac{q}{2}\right\} \cup \bigcup_{k=3}^{c} \left\{2^{c-k}\langle 3^{\ell i}\rangle_{2^k} : 0 \leq i \leq \left\lceil \frac{2^{k-2}}{\ell} \right\rceil - 1\right\}. \tag{29}$$

Then

$$\{0, 1, \cdots, q/2\} \subseteq \{\langle mb\rangle_q : m \in M,\ b \in B_{\mathrm{mod}}\}, \tag{30}$$

and

$$|B_{\mathrm{mod}}| = 3 + \sum_{k=3}^{c} \left\lceil \frac{2^{k-2}}{\ell} \right\rceil = \frac{q}{2\ell} + O(\log q). \tag{31}$$

Proof. We first recall that, for every $k \geq 3$,

$$\mathrm{ord}_{2^k}(3) = 2^{k-2}.$$

Also, no power of 3 is congruent to -1 modulo 2^k, since powers of 3 are congruent to either 1 or 3 modulo 8. Consequently,

$$\{\langle 3^e\rangle_{2^k} : 0 \leq e \leq 2^{k-2} - 1\}$$

lists all positive odd integers smaller than 2^{k-1}.

We first prove (30). The values 0, $q/4$, and $q/2$ are covered using multiplier 1. Every other $x \in \{0, 1, \cdots, q/2\}$ can be written uniquely as $x = 2^{\omega_2(x)}y$, where $\omega_2(x)$ is the exponent of 2 in x, and y is odd. Since $x \notin \{0, q/4, q/2\}$, we have $0 \leq \omega_2(x) \leq c - 3$. Denote $k = c - \omega_2(x)$, then

$$x = 2^{c-k}y,\ 3 \leq k \leq c,\ 1 \leq y < 2^{k-1}.$$

By the preceding observation, there is a unique $0 \leq e < 2^{k-2}$ such that $y = \langle 3^e\rangle_{2^k}$. This gives x the unique representation

$$x = 2^{c-k}\langle 3^e\rangle_{2^k},\ 3 \leq k \leq c,\ 0 \leq e < 2^{k-2}.$$

Write $e = \ell i + s$, where $0 \leq s < \ell$, then

$$0 \leq i = \left\lfloor \frac{e}{\ell} \right\rfloor \leq \left\lfloor \frac{2^{k-2} - 1}{\ell} \right\rfloor = \left\lceil \frac{2^{k-2}}{\ell} \right\rceil - 1.$$

Choose

$$m = 3^s \in M,\ b = 2^{c-k}\langle 3^{\ell i}\rangle_{2^k} \in B_{\mathrm{mod}}.$$

Since $\langle 3^{\ell i}\rangle_{2^k} \equiv \pm 3^{\ell i} \pmod{2^k}$, we have

$$mb \equiv \pm 2^{c-k} 3^{\ell i+s} = \pm 2^{c-k} 3^e \equiv \pm x \pmod{q}.$$

Hence $\langle mb\rangle_q = x$.

We next prove (31). For each k, the selected exponents ℓi lie in $\{0, 1, \cdots, 2^{k-2}-1\}$, so the corresponding values $\langle 3^{\ell i}\rangle_{2^k}$ are distinct. Thus each k contributes exactly $\lceil 2^{k-2}/\ell \rceil$ buckets. These buckets are divisible by 2^{c-k} but not by 2^{c-k+1}, so contributions for different k are disjoint and exclude 0, $q/4$ and $q/2$. Therefore,

$$|B_{\mathrm{mod}}| = 3 + \sum_{k=3}^{c} \left\lceil \frac{2^{k-2}}{\ell} \right\rceil.$$

Since

$$\sum_{k=3}^{c} 2^{k-2} = \frac{q}{2} - 2,$$

and each ceiling function introduces an error smaller than 1,

$$|B_{\mathrm{mod}}| = \frac{q}{2\ell} + O(c) = \frac{q}{2\ell} + O(\log q).$$

□

Corollary 3. *Let M and B_{mod} be defined in Theorem 5, and put*

$$K = \frac{3^{\ell-1} - 1}{2}. \tag{32}$$

Suppose that $2K < q$. For every integer $t \in [-K, q+K]$, there exist

$$\varepsilon \in \{-1, 1\},\ m \in M,\ b \in B_{\mathrm{mod}},\ \alpha \in \{-K, -K+1, \cdots, K+1\}$$

such that

$$t = \varepsilon mb + \alpha q. \tag{33}$$

Handling the leading digit. Under Strategy II, which uses an additional digit, we take

$$B = B_{\mathrm{mod}} \cup \{0, 1, \cdots, K+1\}$$

so that it allows every nonzero final carry α_{h-1} to use multiplier 1, bucket $|\alpha_{h-1}|$, and the corresponding sign. Thus only the additional points $\{q^h P_i : 1 \le i \le n\}$ are needed, giving $\ell nh + n$ precomputed points.

Remark. Although the bucket set construction is mathematically valid for arbitrary ℓ, the scalar conversion and carry handling strategy require small ℓ.

C Bucket Sets for Consecutive Odd Multipliers

Because the maximum multiplier in Appendix B grows exponentially with ℓ, this appendix constructs bucket sets using odd multipliers for $\ell = 2^d$.

Theorem 6. *Let $\ell = 2^d$, where $d \geq 1$, and let $q = 2^c$ with $c \geq d+2$. Set*

$$M = \{1, 3, 5, \cdots, 2\ell - 1\}.$$

Define

$$\begin{aligned} B_{\mathrm{mod}} = \{0\} &\cup \{2^{c-k} : 1 \leq k \leq d+1\} \\ &\cup \bigcup_{k=d+2}^{c} \left\{2^{c-k} \cdot \langle 1 + 4\ell i \rangle_{2^k} : 0 \leq i \leq 2^{k-d-2} - 1\right\}. \end{aligned} \tag{34}$$

Then

$$\{0, 1, \cdots, q/2\} \subseteq \{\langle mb \rangle_q : m \in M,\ b \in B_{\mathrm{mod}}\}, \tag{35}$$

and

$$|B_{\mathrm{mod}}| = \frac{q}{2\ell} + d + 1. \tag{36}$$

Proof. Put $N = 4\ell = 2^{d+2}$. The elements of M, together with their negatives represent all odd residue classes modulo N exactly once. The value 0 is covered by bucket 0. Every $x \in \{1, 2, \cdots, q/2\}$ can be written uniquely as

$$x = 2^{c-k} y,\ 1 \leq k \leq c,\ y \text{ odd},\ 1 \leq y \leq 2^{k-1}.$$

- Suppose first that $k \geq d+2$. There exist $\varepsilon \in \{-1, 1\}$ and $m \in M$ such that

  $$y \equiv \varepsilon m \pmod{N}.$$

  Since m is odd, it is invertible modulo 2^k. Choose $u \in \{0, 1, \cdots, 2^k - 1\}$ satisfying

  $$u \equiv \varepsilon m^{-1} y \pmod{2^k}.$$

  Then $u \equiv 1 \pmod{N}$, so

  $$u = 1 + Ni,\ 0 \leq i < 2^k/N = 2^{k-d-2}.$$

  Taking

  $$b = 2^{c-k} \langle u \rangle_{2^k} \in B_{\mathrm{mod}}$$

  gives $mb \equiv \pm x \pmod{q}$, and hence $\langle mb \rangle_q = x$.
- If $1 \leq k \leq d+1$, the set $\pm M$ still covers all odd residue classes modulo 2^k, by reduction from modulo N. Thus $y \equiv \pm m \pmod{2^k}$ for some $m \in M$, and the bucket $b = 2^{c-k}$ covers x.

We next count the buckets. For fixed $k \geq d+2$, the selected values $1+Ni$ are distinct modulo 2^k. The values $\langle 1+Ni \rangle_{2^k}$ are also distinct. Otherwise, two distinct indices i and j would satisfy

$$1 + Ni \equiv -(1+Nj) \pmod{2^k}.$$

Reducing modulo N gives $1 \equiv -1 \pmod{N}$, which is impossible since $N \geq 4$. Consequently, each $k \geq d+2$ contributes exactly 2^{k-d-2} buckets.

For every $1 \leq k \leq c$, each corresponding bucket is divisible by 2^{c-k} but not by 2^{c-k+1}. Thus the sets corresponding to different values of k are disjoint, and none contains 0. It follows that

$$\begin{aligned} |B_{\mathrm{mod}}| &= 1 + (d+1) + \sum_{k=d+2}^{c} 2^{k-d-2} \\ &= d + 2 + \left(2^{c-d-1} - 1\right) \\ &= \frac{q}{2\ell} + d + 1. \end{aligned}$$

□

Corollary 4. *Let M and B_{mod} be as in Theorem 6. For every integer*

$$t \in [-\ell + 1, q + \ell - 1],$$

there exist

$$\varepsilon \in \{-1, 1\},\ m \in M,\ b \in B_{\mathrm{mod}},\ \alpha \in \{-\ell+1, -\ell+2, \cdots, \ell\}$$

such that

$$t = \varepsilon m b + \alpha q. \tag{37}$$

Handling the leading digit. Under Strategy II, we take

$$B = B_{\mathrm{mod}} \cup \{0, 1, \cdots, \ell\}.$$

Each nonzero final carry then uses multiplier 1, bucket $|\alpha_{h-1}|$, and its sign. Only the additional points

$$\{q^h P_i : 1 \leq i \leq n\}$$

are required. This strategy requires $\ell n h + n$ precomputed points.

Remark. The multipliers need not be consecutive odd integers. The same bucket set works for any set M of ℓ positive odd integers satisfying

$$\{\langle m \rangle_{4\ell} : m \in M\} = \{1, 3, \cdots, 2\ell - 1\}.$$

Its coverage, cardinality, and maximum gap $d(B_{\mathrm{mod}})$ remain unchanged, although the largest carry may depend on the largest multiplier in M.