Speeding Up Multi-Scalar Multiplications for Pairing-Based zkSNARKs

Xinxin Fan[1], Veronika Kuchta[2], Francesco Sica[2], and Lei Xu[3]

[1]IoTeX, Menlo Park, CA 94025
[2]Department of Mathematics and Statistics
Florida Atlantic University, Boca Raton, FL 33431
[3]Department of Computer Science
Kent State University, Kent, OH 44242

Abstract. Multi-scalar multiplication (MSM) is one of the core components of many zero-knowledge proof systems, and a primary performance bottleneck for proof generation in these schemes. One major strategy to accelerate MSM is utilizing precomputation. Several algorithms (e.g., Pippenger and BGMW) and their variants have been proposed in this direction. In this paper, we revisit the recent precomputation-based MSM calculation method proposed by Luo, Fu and Gong at CHES 2023 [10] and generalize their approach. In particular, we presented a general construction of optimal buckets. This improvement leads to significant performance improvements, which are verified by both theoretical analysis and experiments.

1 Introduction

The concept of zero-knowledge proof (ZKP) was introduced by Goldwasser, Micali, and Rackoff in 1985 [6]. Terms like ZKP or zero-knowledge arguments (ZKA) satisfy the three security properties, such as *correctness* - meaning that an honest prover will always convince a verifier of knowing a secret to a public statement, *soundness*—ensuring that a dishonest prover cannot prove a false statement—and *zero-knowledge*—indicating that the proof reveals no extra information beyond the truthfulness of the statement the prover aims to prove. While in ZKP systems the soundness property holds for provers with unbounded (statistical) capabilities, it is assumed that the prover is computationally bounded in ZKA systems.

There has been a surge of interest in putting ZKP into practice in the past few years, which was first triggered by the demands for privacy protection in the blockchain environment (e.g., Zerocash (zcash) [1]), and then more general applications such as verifiable computation.

An advanced version of ZKPs with short proofs and efficient verification is known as zkSNARKs (**z**ero-**k**nowledge **S**uccinct **N**on-interactive **AR**guments of

Author list in alphabetical order; see `https://www.ams.org/about-us/governance/committees/Statement_JointResearchanditsPublication.pdf`

Knowledge). They can be seen as a composition of the **N**on-**I**nteractive **Z**ero-**K**nowledge proofs (NIZKs) and succinct Arguments of Knowledge. NIZKs were introduced by Blum, Feldman and Micali [3] while Kilian [9] provided a definition on efficient zero-knowledge arguments. In a proof system, the prover's computational power may be unbounded, but in an argument system, it is assumed that the prover is computationally bounded.

While many zkSNARK schemes have been proposed since then, pairing-based zkSNARK is still one of the most attractive options in practice. While the verification in zkSNARKs is fast, the construction of such proof systems is usually time-consuming and hinders their wide adoption. In pairing-based zkSNARK constructions (e.g., [5, 7, 8, 11]), the proof consists of several points in an elliptic curve group which operate with each other within of this group.

The main computational bottleneck of such zkSNARK constructions lies in multiscalar multiplication (MSM). Let $S_{n,r}$ be the following n-scalar multiplication over fixed points $P_1, \ldots, P_n$,

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

where $a_i \in [0, r), i = 1, \ldots, n$ are integers. In the following, r will denote the order of the (elliptic curve) group where all these computations take place.

1.1 Existing MSM Computation Methods

In the last three decades, various methods have been proposed to accelerate MSM computation, and most of them utilize precomputation.

Straus Method To compute $S_{n,r}$, the Straus method this method precomputes 2^{nc} points

$$\Big\{ \sum_{i=1}^{n} b_i P_i | \forall b_i \in [0, 2^c - 1], i \in [1, n] \Big\}$$

where c is a small integer. Next, the algorithm divides each a_i from (1) into segments of length c, i.e.

$$a_i = a_{i,h-1} \| a_{i,h-2} \| \cdots \| a_{i,1} \| a_{i,0} = \sum_{j=0}^{h-1} a_{i,j} 2^{jc}, i \in [1, n]$$

where $h = \lceil \log_2(r)/c \rceil$ and $0 \leqslant a_{ij} < 2^c$ for $j \in [1, h-1]$ for $1 \leqslant j \leqslant h-1$. The algorithm retrieves the point

$$S_{n,2^c} = \sum_{i=1}^{n} a_{i,h-1} P_i$$

from the precomputation table, doubles it c times, adds the precomputed point $\sum_{i=1}^{n} a_{i,h-2}P_i$ to obtain

$$S_{n,2^{2c}} = \sum_{i=1}^{n} (a_{i,h-1} \| a_{i,h-2}) P_i.$$

After $h-1$ repetitions, we obtain

$$S_{n,2^{hc}} = \sum_{i=1}^{n} (a_{i,h-1} \| a_{i,h-2} \| \dots \| a_{i,0}) P_0 \tag{2}$$

Pippenger's Bucket Method This method proceeds in the same way as in Straus method except for computing

$$S_{n,2^c} = \sum_{i=1}^{n} a_{i,j} P_i$$

where $j \in [0, h-1], h = \lceil \log_2(r)/c \rceil$. First, the method sorts all points into $(2^c - 1)$ buckets with respect to their scalars. Let S_i denote the intermediate subsum of points corresponding to scalar i. The algorithm computes all S_i, for $i \in [1, 2^c - 1]$ and finally it computes $S_{n,2^c} = \sum_{i=1}^{2^c-1} i \cdot S_i$ using at most $2(2^c - 2)$ additions.

Luo-Fu-Gong (LFG) MSM Method [10] Let M be a set of integers, B a set of non-negative integers containing zero. Given scalars $a_i, 0 \leqslant a_i < r$, the LFG method first computes [10, Algorithm 6] a radix q representation

$$a_i = \sum_{j=0}^{h-1} a_{ij} q^j$$

where $h = \lceil \log_q r \rceil$, and for every $i \in [1, n], j \in [0, h-1]$,

$$a_{ij} = \epsilon_{ij} m_{ij} b_{ij}, \text{ where } \epsilon_{ij} \in \{\pm 1\}, m_{ij} \in M, b_{ij} \in B \ .$$

Then, $S_{n,r}$ can be computed as:

$$S_{n,r} = \sum_{i=0}^{n} a_i P_i = \sum_{i=1}^{n} \big(\sum_{j=0}^{h-1} a_{ij} q^j \big) P_i \tag{3}$$

$$= \sum_{i=1}^{n} \big(\sum_{j=0}^{h-1} \epsilon_{ij} m_{ij} b_{ij} q^j \big) P_i = \sum_{i=1}^{n} \sum_{j=0}^{h-1} b_{ij} \epsilon_{ij} m_{ij} q^j P_i$$

Let $P_{ij} = \epsilon_{ij} m_{ij} q^j P_i$. Then $S_{n,r}$ can be computed as follows.

$$S_{n,r} = \sum_{i=1}^{n} \sum_{j=0}^{h-1} b_{ij} P_{ij} = \sum_{i=1}^{n} \sum_{j=0}^{h-1} \big(\sum_{k \in B} k \cdot \sum_{i,j \text{ s.t. } b_{ij}=k} P_{ij} \big) \tag{4}$$

$$= \sum_{k \in B} k \cdot \big(\sum_{i=1}^{n} \sum_{j=0}^{h-1} \sum_{i,j \text{ s.t. } b_{ij}=k} P_{ij} \big).$$

Assume that there are $nh|M|$ such points which are defined as

$$\{mq^j P_i | 1 \leqslant i \leqslant n, 0 \leqslant j \leqslant h-1, m \in M\}$$

These points are precomputed (we don't need to precompute their opposites, see below). Then define intermediate subsums

$$S_k = \sum_{i=1}^{n} \sum_{j=0}^{h-1} \sum_{i,j \text{ s.t. } b_{ij}=k} P_{ij}, k \in B \ .$$

All S_k's can be computed with at most $nh-(|B|-1)$ additions and the remainder is computed by Algorithm 1 with at most $2(|B|-1)+d-3$ additions, where d is the maximum difference between the two neighboring elements in B. The total cost of computing $S_{n,r}$ is therefore at most

$$nh + |B| + d - 4 \tag{5}$$

elliptic curve additions.

Algorithm 1 Subsum accumulation algorithm [10]

Input: $1 \leqslant b_1 \leqslant b_2 \leqslant \ldots \leqslant b_{|B|}, S_1, S_2, \ldots, S_{|B|}$
Output: $S = b_1 S_1 + \cdots + b_{|B|} S_{|B|}$
1: Define a length-$(d+1)$ array $\mathsf{tmp} = [0] \times (d+1)$
2: **for** $i = |B|$ to 1 by -1 **do**
3: $\quad \mathsf{tmp}[0] = \mathsf{tmp}[0] + S_i$
4: $\quad k = b_i - b_{i-1}$
5: $\quad$ **if** $k \geqslant 1$ **then**
6: $\quad\quad \mathsf{tmp}[k] = \mathsf{tmp}[k] + \mathsf{tmp}[0]$
7: **return** $1 \cdot \mathsf{tmp}[1] + 2 \cdot \mathsf{tmp}[2] + \cdots + d \cdot \mathsf{tmp}[d]$

Let M denote a set of multipliers which are used to generate the precomputed points. Furthermore, the set B is called a bucket set which contains sorted points. In Algorithm 2 we recall the MSM algorithm of [10]. Note that Luo, Fu and Gong merge the multiplier set with the units ± 1, so that their M is in fact what for us will be $M \cup -M$.

1.2 Our contribution

In summary, our contribution in this paper is threefold:

(1) We point out a problem with Property 1 in the MSM method in [10] which proposes a new decomposition of scalars in base $q = 2^c$. The original property in [10] states the following: given a power of two integer $q = 2^c$ (for $10 \leqslant c \leqslant 31$), for all $0 \leqslant t \leqslant q$ there exists a value $b \in B$ and a multiplier $m \in \{1, 2, 3\}$ such that $t = mb$ or $q - t = mb$. This decomposition is then used in a

Algorithm 2 Multi-scalar multiplication over fixed points [10]

Input: Scalars $a_1, a_2, \ldots, a_n$, fixed points $P_1, P_2, \ldots, P_n$, radix q, scalar length h, multiplier set $M = \{m_0, m_1, \ldots, m_{|M|-1}\}$, bucket set $B = \{b_0, b_1, \ldots, b_{|B|-1}\}$.

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

1: Precompute a length-$nh|M|$ point array precomputation, s.t. precomputation $[|M|((i-1)h+j)+k] = m_k q_j P_i$.

2: Precompute a hash table mindex to record the index of every multiplier, s.t. $\mathsf{mindex}[m_k] = k$. Precompute a hash table bindex to record the index of every bucket, such that $\mathsf{bindex}[b_k] = k$.

3: Convert every a_i to its standard q-ary form, then convert it to $a_i = \sum_{j=0}^{h-1} m_{ij} b_{ij} q^j$.

4: Create a length-nh scalar array $\mathsf{scalars}$, s.t. $\mathsf{scalars}[(i-1)h+j] = b_{ij}$. Create a length-$nh$ array points recording the index of points, such that $\mathsf{points}[(i-1)h+j] = |M|((i-1)h+j) + \mathsf{mindex}[m_{ij}]$. n-scalar multiplication $S_{n,r}$ is equivalent to the following nh-scalar multiplication

$$\sum_{i=0}^{nh-1} \mathsf{scalars}[i] \cdot \mathsf{precomputation}[\mathsf{points}[i]],$$

where every scalar in $\mathsf{scalars}$ is from bucket set B.

5: Create a length-$|B|$ point array $\mathsf{buckets}$ to record the intermediate subsums, and initialize every point to infinity. For $0 \leqslant i \leqslant nh-1$, add point $\mathsf{precomputation}[\mathsf{points}[i]]$ to bucket $\mathsf{buckets}[\mathsf{bindex}[\mathsf{scalars}[i]]]$.

6: Invoke Algorithm 1 to compute $\sum_{i=0}^{|B|-1} b_i \cdot \mathsf{buckets}[i]$, return the result.

modified Pippenger bucket algorithm. The main idea behind this property is to remove redundant points from an initially defined set B_0 to obtain a new set B_1. This can be done for the purpose of saving computational costs, since in an elliptic curve group, $-P_i = (x_i, -y_i)$ can be determined on the fly from the computed points $P_i = (x_i, y_i)$. The algorithm provided by Luo, Fu, and Gong [10] for constructing B_1 from B_0 discards all elements of the form $q-2i$ and $q-3j$ for all values $i, j \in B_0$ and $q/4 \leqslant i < q/2$ and $q/6 \leqslant j < q/4$. We provide a counterexample that shows Property 1 of [10] (we will denote it the "LFG Property 1") does not always hold. Furthermore, we provide a general construction of counterexamples and prove that the LFG Property 1 is false for at least $q/(216) + O(1)$ integers. We also show that the LFG Property 1 is correct for all odd t, where $0 < t < q$.

(2) Our second contribution is to provide a method to fix the LFG Property 1. We present a new construction of the set B_1 by first removing elements of the form $q-2i$ and $q-3j$ from the initial set B_0 and then adding all elements of the form $q-6k \in B_0$ with $k \notin B_0$. Next, we show how to modify the LFG Property 1 to obtain a MSM algorithm with optimal runtime. The modification involves an element $\epsilon_t \in \{\pm 1\}$ such that $t \equiv \epsilon_t m_t b_t \mod q$ for a multiplier $m_t \in M$ and a bucket element $b_t \in B$. While the achieved running

time of the MSM algorithm in [10] is $(nh + 0.21q) \cdot \mathsf{Add}$, for $q = 2^c$ and $10 \leqslant c \leqslant 31$ and Add denoting the number of point additions on an elliptic curve, we stress out that the LFG Property 1 leaves out some values of t, therefore the time complexity of the LFG bucket method cannot be taken as a benchmark. In contrast to [10] the scalars decomposition property holds for all $0 < t < q$ and achieves the same space and running time complexity for $q = p^c$ and $4 \leqslant c \leqslant 11$ (see Table 1).

(3) In our third contribution we use efficient endomorphisms to achieve better running time complexity and to save storage space. We use an endomorphism ω such that $\omega^3(P) = P$ for all elliptic curve points P. Since it holds that $\omega(x, y) = (\zeta_3 x, y)$ for a complex cube root of unity $\zeta_3 \in \mathbb{F}_p$, i.e. $\zeta_3^3 = 1$, the computation of $\omega(P)$ can be done on the fly leading to significant savings of the storage cost of these points. The endomorphism ring of such elliptic curves is isomorphic to $\mathbb{Z}[\omega]$. With this in mind we update the scalar decomposition property to adapt it to the new setting, where $t \in \mathbb{Z}[w]$. We implement our idea with $p = 2 - \omega$, the multiplier set $M = \{1\}$ and $q = p^c$, where $p \mid 7$ in $\mathbb{Z}[\omega]$. With this approach we reduce the storage cost from $3nh + n$ to nh curve points and the time complexity is reduced from $(n(h + 1) + 0.21875q) \cdot \mathsf{Add}$ to $(nh + 0.167q + 20) \cdot \mathsf{Add}$.

Finally we confirm our result for the fixed LFG Property 1 and the new bucket set constructions by implementing the MSM algorithm in C++. To enable a fair comparison with the LFG approach we use the same elliptic curve BLS12-381. We measure the space complexity in terms of the number of stored elliptic curve points P. Therefore, the expression $nh \cdot P$ indicates that a total of nh curve points are stored.

Table 1. Comparison of Different MSM Algorithms, for $q = p^c,\ 4 \leqslant c \leqslant 11$.

Method	Space Complexity	Time Complexity (Worst Case)
Pippenger [2, 12]	$n \cdot P$	$h(n + 0.5q) \cdot \mathsf{Add}$
Pippenger variant [4]	$nh \cdot P$	$(nh + 0.5q) \cdot \mathsf{Add}$
LFG [10], $p = 2$[1]	$3nh \cdot P$	$(nh + 0.21q) \cdot \mathsf{Add}$
Repaired LFG Method, $p = 2$	$\boldsymbol{(3nh + n) \cdot P}$	$\boldsymbol{(n(h + 1) + 0.21875q) \cdot \mathsf{Add}}$
Our Method for a prime $p > 2$	$\boldsymbol{n(h + 1)\lvert M\rvert \cdot P}$	$\boldsymbol{(n(h + 1) + q/(2\lvert M\rvert) + p - 4) \cdot \mathsf{Add}}$
Our Method with endomorphisms for $\lvert M\rvert = \{1\}$	$\boldsymbol{nh \cdot P}$	$\boldsymbol{(nh + 0.167q + 20) \cdot \mathsf{Add}}$

2 Analysis of the LGF MSM Algorithm

In this section, we first point out an issue with the MSM method of [10], especially with their Property 1 on p. 369, and then propose a fix.

[1] Notice that the time and space complexities in [10] are provided for their incomplete bucket set B. Since the repaired bucket set B contains more points, the complexities would be comparable to our results.

2.1 Background of Property 1 in [10]

For a prime p and a positive integer n, define $\mathrm{ord}_p(n)$ to be the integer $e \geqslant 0$ such that $n = p^e k$ with $p \nmid k$. Luo, Fu and Gong define a new decomposition of scalars in base $q = 2^c$ for $c \in \mathbb{N}$ to which they can apply a modified Pippenger bucket algorithm.

They start by first defining the set B_0 as follows:

$$B_0 = \{0\} \cup \{b \in \mathbb{N} \colon 1 \leqslant b \leqslant q/2, \mathrm{ord}_2(b) + \mathrm{ord}_3(b) \equiv 0 \pmod 2\} \ .$$

Note that for any integer $1 \leqslant t \leqslant q/2$, there exists an $m \in \{1, 2, 3\}$ and $b \in B_0$ such that $t = mb$. Indeed, if $t \notin B_0$, then $2 \mid t$, in which case $b = t/2 \in B_0$ or $3 \mid t$, in which case $b = t/3 \in B_0$.

The authors want to take advantage of the fact that, in an elliptic curve group, opposites of points can be computed on the fly at virtually no cost, allowing for m to be chosen from $\{\pm 1, \pm 2, \pm 3\}$. This leads to the removal of redundant representations from B_0 as shown in Algorithm 3 by discarding from B_0 all elements of the form $q - 2i$ and $q - 3j$ for $i, j \in B_0$, with $q/4 \leqslant i < q/2$ and $q/6 \leqslant j < q/4$. The resulting set carved out of B_0 will be called B_1^{old} (B_1 in [10], but we will reserve this notation to our later fix), and the following property is claimed computationally for $B = B_1^{\mathrm{old}}$.

Algorithm 3 Construction of auxiliary set B_1^{old} in [10]

Input: B_0, q
Output: B_1^{old}
1: $B_1^{\mathrm{old}} = B_0$
2: **for** $i = \frac{q}{4}$ to $\frac{q}{2} - 1$ **do**
3: **if** $i \in B_0$ and $q - 2i \in B_0$ **then**
4: $B_1^{\mathrm{old}} = B_1^{\mathrm{old}} - \{q - 2i\}$
5: **for** $i = \lfloor \frac{q}{6} \rfloor$ to $\frac{q}{4} - 1$ **do**
6: **if** $i \in B_0$ and $q - 3i \in B_0$ **then**
7: $B_1^{\mathrm{old}} = B_1^{\mathrm{old}} - \{q - 3i\}$
8: **return** B_1^{old}

Property 1 (Property 1 in [10])**.** *Given $q = 2^c$ $(10 \leqslant c \leqslant 31)$, for all $0 \leqslant t \leqslant q$, there exist $b \in B$ and $m \in \{1, 2, 3\}$ such that*

$$t = mb \quad \textit{or} \quad q - t = mb \ . \tag{6}$$

We now provide counterexamples to this property, when $B = B_1^{\mathrm{old}}$. In fact, large families of counterexamples can be constructed for all such q. For instance, let $q = 2^{10} = 1024$ and $t = 292$. Note that

$$t = 2^2 \cdot 73 \in B_0 \quad \text{and} \quad q - t = 732 = 3 \cdot 2^2 \cdot 61 = 3j \ ,$$

with $j = 244 = 2^2 \cdot 61 \in B_0, 170 = \lfloor q/6 \rfloor < j < q/4 = 256$. Hence $t \in B_0$, so that $m = 1$ in $t = mb$ in (6), but $t \notin B_1^{\text{old}}$.

On the other hand,

$$j = 2^2 \cdot 61 = q - 2^2 \cdot 3 \cdot 5 \cdot 13 = q - 2i \ ,$$

with $i = 390 = 2 \cdot 3 \cdot 5 \cdot 13 \in B_0, 256 = q/4 \leqslant i < q/2 = 512$, hence $j \notin B_1^{\text{old}}$. Similarly,

$$q - t = 2i' \ ,$$

where $i' = 366 = 2 \cdot 3 \cdot 61 \in B_0$, and

$$i' = q - 2 \cdot 7 \cdot 47 = q - 2 \cdot i'' \ ,$$

with $i'' = 329 = 7 \cdot 47 \in B_0, 256 = q/4 \leqslant i'' < q/2 = 512$, hence $i' \notin B_1^{\text{old}}$. Since $q - t = mb$ in (6) implies that $m = 2$ (resp. 3), and $b = j$ (resp. $b = i'$) is not in B_1^{old}, we conclude that $t = 292$ doesn't satisfy Property 1 for $q = 1024$.

2.2 General Construction of Counterexamples

We show the general construction of counterexamples for Property 1 of [10].

Proposition 1. *Property 1 is false for at least* $\dfrac{q}{216} + O(1)$ *integers* $0 \leqslant t \leqslant q$.

Proof. As above, we let $q = 2^c$ for an integer $c \geqslant 10$. Pick any integer n that is coprime to 6, with $\frac{4q}{72} < n < \frac{5q}{72}$. Note that among six consecutive integers, at least two will be coprime to 6, i.e. the neighbors of a multiple of 6. Hence, there are at least

$$\frac{q}{216} + O(1)$$

such values of n. Define $j = 6n$ and $\theta = 3(q/2 - j)$. Note that

$$\frac{q}{3} < j < \frac{5q}{12} \quad \text{and} \quad \text{ord}_2(j) \equiv \text{ord}_3(j) \equiv 1 \pmod 2 \ ,$$

hence

$$\frac{q}{4} < \theta < \frac{q}{2} \quad \text{and} \quad \text{ord}_2(\theta) \equiv \text{ord}_3(\theta) \equiv 1 \pmod 2 \ ,$$

so that $j, \theta \in B_0$. Note that in particular, $3 \mid \theta$. Let $t = q - 2\theta$. We will show $0 < t < q/2$ does not satisfy Property 1.

First, $3 \nmid t$ and $\text{ord}_2(t) = \text{ord}_2(\theta) + 1 \equiv 0 \pmod 2$, therefore $t \in B_0$. However, since $\theta \in B_0$, $t \notin B_1^{\text{old}}$. Any expression $t = mb$ with $m \in \{1, 2, 3\}$ and $b \in B_1$ must therefore be excluded in Property 1. We now exclude that

$$q - t = mb, \quad \text{for some } b \in B_1^{\text{old}} \text{ and } m \in \{1, 2, 3\} \ .$$

Since $q - t > q/2$, $q - t \notin B_0$, hence $q - t \notin B_1^{\text{old}}$.

We have $q - t = 2\theta$. Write $\theta = q - 2i$ and note $3 \nmid i$, $\text{ord}_2(i) \equiv 0 \pmod 2$. Also $0 < i < q/2$, therefore $i \in B_0$ and $\theta \notin B_1^{\text{old}}$.

Finally, define $\psi = \frac{q-t}{3} = 2\theta/3$. Then $\mathrm{ord}_2(\psi) \equiv \mathrm{ord}_3(\psi) \equiv 0 \pmod 2$, and since trivially $0 < \psi < q/3$, we get $\psi \in B_0$. On the other hand, by definition of θ,

$$\psi = q - 2j \ ,$$

where we saw $j \in B_0$. Therefore $\psi \notin B_1^{\mathrm{old}}$ thus concluding our proof. □

2.3 Property 1 Holds for Odd $0 < t < q$

We can nevertheless show that the following is true.

Proposition 2. *Property 1 in [10] with $B = B_1^{\mathrm{old}}$ holds whenever $0 \leqslant t \leqslant q$ is odd.*

Proof. Replacing t by $q - t$ if necessary, we can suppose that $t < q/2$. There are several cases to consider.

$3 \nmid t$ and $3 \nmid q - t$: In this case, $t \in B_0$ and, since we can't write $t = q - 2n$ or $q - 3n$ with $n \in \mathbb{N}$, it follows that $t \in B_1^{\mathrm{old}}$.

$3 \nmid t$ and $3 \mid q - t$: Again, $t \in B_0$. If $t \notin B_1^{\mathrm{old}}$, then we could write

$$t = q - 3i_3 \ , \quad i_3 < \frac{q}{4} \ , i_3 \in B_0 \ .$$

Similarly, if $q - t = 3i_3$ and $i_3 \notin B_1^{\mathrm{old}}$, then

$$i_3 = q - 3i_3' \ , \quad i_3' < \frac{q}{4} \ , i_3' \in B_0 \ .$$

We reach a contradiction, since

$$q = i_3 + 3i_3' < \frac{q}{4} + \frac{3q}{4} = q \ .$$

$3 \mid t$ and $3 \nmid q - t$: This is the last possible case. Either $t \in B_0$ and then we can reason as in the first case to deduce that $t \in B_1^{\mathrm{old}}$; or

$$t = 3b \ , b \in B_0 \ .$$

If $b \notin B_1^{\mathrm{old}}$, then

$$b = q - 3j_3 \ , \quad j_3 < \frac{q}{4} \ , j_3 \in B_0 \ .$$

This is again impossible, since $b = t/3 < q/6$, resulting in

$$q = b + 3j_3 < \frac{q}{6} + \frac{3q}{4} < q \ .$$

□

3 Repairing Property 1 of [10]

We show in this section how to change the definition of B_1^{old} so that Property 1 holds.

We construct the new set B_1 in Algorithm 4, by first removing from B_0 as before all elements of the form $q - 2i > 0$ and $q - 3j$, for $i, j \in B_0$, and $j < q/4$. We then add back all elements $q - 6k \in B_0$ with $k \notin B_0$.

Proposition 3. *Property 1 with $B = B_1$ holds for all $0 \leqslant t \leqslant q$.*

Proof. Because Property 1 is symmetric in $t \leftrightarrow q - t$, we can suppose $3 \nmid t$. Also, in view of Proposition 2 we only need suppose that $0 \neq t$ is even, since $B_1^{\text{old}} \subseteq B_1$. We have two cases:

$\text{ord}_2(t)$ **even:** If $t \leqslant q/2$, then $t \in B_0$. If $t \notin B_1^{\text{old}}$, then either $t = q - 3j$ with $j \in B_0, j < q/4$, or $t = q - 2i$ with $i \in B_0$.
In the first case, we have $2 \mid j$, hence

$$t = q - 3j \Longrightarrow t = q - 2i \quad \text{with } i = \frac{3j}{2} \in B_0 \ .$$

We therefore only need focus on $t = q - 2i$, where $i \in B_0$. Since $\text{ord}_2(t) = \text{ord}_2(2i) = \text{ord}_2(i) + 1$ we deduce that $\text{ord}_3(i) \equiv 1 \pmod 2$, in particular that $3 \mid i$. Calling $k = i/3$, we have $k \notin B_0$ and $t = q - 6k$, therefore $t \in B_1$.
On the other hand, if $t > q/2$, then, since $q - t < q/2$ and $\text{ord}_2(q - t) = \text{ord}_2(t)$, if $\text{ord}_3(q - t)$ is even, then, reasoning as above with $q - t \in B_0$ in place of t, we find that $q - t \in B_1$.
If $t > q/2$ and $\text{ord}_3(q - t)$ is odd, then $q - t = 3b$ with $b \in B_0$. If $b \notin B_1^{\text{old}}$, then either $b = q - 3j$ with $j \in B_0, j < q/4$, or $b = q - 2i$ with $i \in B_0$. The former case is impossible, since we would get the contradiction

$$q = b + 3j < \frac{q}{6} + \frac{3q}{4} < q \ .$$

In the latter case,

$$b = q - 2i \quad \text{with } i \in B_0 \ .$$

As above $\text{ord}_2(i) + 1 = \text{ord}_2(b) = \text{ord}_2(q - t) = \text{ord}_2(t)$, therefore $\text{ord}_2(i)$ is odd and $\text{ord}_3(i)$ is odd; in particular $3 \mid i$. Writing $k = i/3$, we have $k \notin B_0$ and $b = q - 6k$, therefore $b \in B_1$.

$\text{ord}_2(t)$ **odd:** Then $t = 2b$, where $b \in B_0$. If $b \notin B_1^{\text{old}}$, then either $b = q - 3j$ with $j \in B_0, j < q/4$, or $b = q - 2i$ with $i \in B_0$. In the latter case, as before, $\text{ord}_2(i) \equiv 1 \pmod 2$ and therefore $3 \mid i$. Calling $k = i/3$, we have $k \notin B_0$ and $b = q - 6k$, therefore $b \in B_1$.
The former case is slightly more complicated, where we have

$$b = q - 3j \ , \quad j < \frac{q}{4} \ , j \in B_0 \ . \tag{7}$$

If $2 \mid j$, then, calling $k = j/2 \notin B_0$, we find that $b \in B_1$ as before. It may however be the case that $2 \nmid j$, that is $2 \nmid b$. By (7), $b \equiv q \pmod 3$,

hence $q \not\equiv t = 2b \pmod 3$. In this case, $\mathrm{ord}_2(q-t) = \mathrm{ord}_2(t) = 1$, hence $b' = (q-t)/2 \in B_0$. If $b' \notin B_1^{\mathrm{old}}$, then, noticing that $2 \nmid b'$,

$$b' = q - 3j' \ , \quad j' < \frac{q}{4} \ , j' \in B_0 \ . \tag{8}$$

Putting (7) and (8) together,

$$q - 3j' = b' = \frac{q-t}{2} = \frac{q}{2} - b = 3j - \frac{q}{2} \ ,$$

from which

$$3j + 3j' = \frac{3q}{2} \iff j + j' = \frac{q}{2} \ ,$$

which is impossible because $j, j' < q/4$.

□

We define the new $B = B_1$, which will allow us to prove a precise estimate of its cardinality, as now B no longer depends on the elliptic curve. Table 4 in the appendix lists our new bucket set constructions for $q = 2^c, 10 \leqslant c \leqslant 31$.

Algorithm 4 Construction of the new auxiliary set B_1

Input: B_0, q
Output: B_1

$B_1 = B_0$
for $\frac{q}{4} \leqslant i < \frac{q}{2}$ **do**
  if $i \in B_0$ and $q - 2i \in B_0$ **then**
    $B_1 = B_1.\mathrm{remove}(q - 2i)$
for $\frac{q}{6} \leqslant i < \frac{q}{4}$ **do**
  if $i \in B_0$ and $q - 3i \in B_0$ **then**
    $B_1 = B_1\mathrm{remove}(q - 3i)$
for $\frac{q}{12} \leqslant i < \frac{q}{6}$ **do**
  if $i \notin B_0$ and $q - 6i \in B_0$ **then**
    $B_1 = B_1.\mathrm{append}(q - 6i)$
return B_1

3.1 Analysis of the Size of B

The set $B = B_1$ can also be constructed by removing the following two subsets from B_0:

1. $B_2 = \{t = q - 2i \in B_0 \colon i \in B_0,\ i < q/2,\ 3 \nmid i\}$, and
2. $B_3 = \{\theta = q - 3j \in B_0 \colon j \in B_0,\ j < q/4,\ 2 \nmid j\}$.

The sets B_2 and B_3 are disjoint, since all elements of the former are even, while all elements of the latter are odd.

Lemma 1. *The cardinalities of the sets B_2 and B_3 (denoted as $|B_2|$ and $|B_3|$) satisfy*

$$|B_2| = \left|\left\{1 \leqslant t \leqslant \frac{q}{2} \colon \operatorname{ord}_2(t) \equiv \operatorname{ord}_3(t) \equiv 1 \pmod 2\right\}\right| \ ,$$

and

$$|B_3| = \left|\left\{\frac{q}{2} < u \leqslant \frac{3q}{4} \colon 2 \nmid u,\ \operatorname{ord}_3(u) \equiv 1 \pmod 2\right\}\right| \ .$$

Proof. If $t \in B_2$, $\operatorname{ord}_2(t) \equiv \operatorname{ord}_2(i) + 1 \equiv 1 \pmod 2$, hence $\operatorname{ord}_3(t) \equiv 1 \pmod 2$. Vice-versa, if $0 < t \leqslant q/2$ satisfies $\operatorname{ord}_2(t) \equiv \operatorname{ord}_3(t) \equiv 1 \pmod 2$, then $t \in B_2$. This shows that B_2 can in fact be described by the set on the right-hand side of the first equation of the lemma.

Regarding B_3, whenever $\theta = q - 3j \in B_0$ with $j \in B_0$, $j < q/4$ and j odd, then $q/4 < \theta \leqslant q/2$, and therefore $q/2 \leqslant u = q - \theta < 3q/4$, $\operatorname{ord}_2(u) = 0$ and $\operatorname{ord}_3(u) \equiv 1 \pmod 2$. Vice-versa, any odd $q/2 \leqslant u < 3q/4$ (note that u cannot equal any of those end values) such that $\operatorname{ord}_3(u)$ is odd will correspond to $\theta = q - u \in B_3$. □

Lemma 2. *Let $Q \in \mathbb{N}$, e, f be nonnegative integers. Define*

$$S_Q^{e,f} = \{1 \leqslant t \leqslant Q \colon \operatorname{ord}_2(t) = e,\ \operatorname{ord}_3(t) = f\},$$

then

$$|S_Q^{e,f}| = \frac{Q}{2^e 3^{f+1}} + O(1) \ .$$

Proof. By the inclusion-exclusion principle,

$$\begin{aligned} S_Q^{e,f} &= \{1 \leqslant t \leqslant Q \colon 2^e 3^f \mid t\} - \{1 \leqslant t \leqslant Q \colon 2^{e+1} 3^f \mid t\} \\ &\quad - \{1 \leqslant t \leqslant Q \colon 2^e 3^{f+1} \mid t\} \cup \{1 \leqslant t \leqslant Q \colon 2^{e+1} 3^{f+1} \mid t\} \ . \end{aligned}$$

Taking cardinalities,

$$\begin{aligned} |S_Q^{e,f}| &= \left\lfloor \frac{Q}{2^e 3^f} \right\rfloor - \left\lfloor \frac{Q}{2^{e+1} 3^f} \right\rfloor - \left\lfloor \frac{Q}{2^e 3^{f+1}} \right\rfloor + \left\lfloor \frac{Q}{2^{e+1} 3^{f+1}} \right\rfloor \\ &= \left(\frac{1}{2^e 3^f} - \frac{1}{2^{e+1} 3^f} - \frac{1}{2^e 3^{f+1}} + \frac{1}{2^{e+1} 3^{f+1}} \right) Q + O(1) \\ &= \frac{Q}{2^e 3^f} \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{3}\right) + O(1) \\ &= \frac{Q}{2^e 3^{f+1}} + O(1) \ . \end{aligned}$$

□

The Size of $B_0 \backslash B_2$ Applying Lemma 2 with $Q = q/2$, we compute

$$\begin{aligned} |B_0 \backslash B_2| &= \sum_{\substack{e \geqslant 0 \text{ even} \\ f \geqslant 0 \text{ even}}} |S^{e,f}_{q/2}| = \frac{q}{2} \sum_{\substack{0 \leqslant e \leqslant \log_2 q \\ e \text{ even}}} \sum_{\substack{0 \leqslant f \leqslant \log_3 q \\ f \text{ even}}} \frac{1}{2^e 3^{f+1}} + O(\log^2 q) \\ &= \frac{q}{6} \sum_{\epsilon \geqslant 0} \frac{1}{4^\epsilon} \sum_{\phi \geqslant 0} \frac{1}{9^\phi} + O(\log^2 q) = \frac{q}{4} + O(\log^2 q) \ . \end{aligned} \tag{9}$$

The Size of B_3 Similarly, by applying Lemma 2 with $Q = 3q/4, q/2$, and we have,

$$\begin{aligned} |B_3| &= \sum_{f \geqslant 0 \text{ odd}} |S^{0,f}_{3q/4}| - |S^{0,f}_{q/2}| \\ &= \left(\frac{3q}{4} - \frac{q}{2} \right) \sum_{f \geqslant 0 \text{ odd}} \frac{1}{3^{f+1}} + O(\log q) \\ &= \frac{q}{32} + O(\log q) \ . \end{aligned} \tag{10}$$

Computation of the Size of B Since $B = (B_0 \backslash B_2) \backslash B_3$ and $B_2 \cap B_3 = \varnothing$, using (9) and (10), we compute

$$|B| = |B_0 \backslash B_2| - |B_3| = \frac{q}{4} - \frac{q}{32} + O(\log^2 q) = \frac{7q}{32} + O(\log^2 q) \ ,$$

where $q = 2^c$. Note that $7/32 = 0.21875$.

3.2 The Maximum Difference Between Neighbors of B

Proposition 4. *Let $b_1 < b_2 < \cdots < b_{|B|}$ denote the elements of B, sorted in increasing order. Then for all $1 \leqslant r < |B|$,*

$$b_{r+1} - b_r \leqslant 6.$$

Proof. Consider the set

$$S = \{ m \leqslant q/2 \colon m \equiv \pm 1 \pmod 6 \} \ .$$

Then $S \subseteq B_0$, since integers in S are coprime to 6. Let $m \in S$. If $m \notin B$, then $m \in B_3$ so that $m \equiv q \pmod 3$. The neighbors $m_- < m < m_+$ of m in S are spaced in such a way that

$$\{ m - m_-, m_+ - m \} = \{2, 4\} \ .$$

But then $m_\pm \not\equiv q \pmod 3$, hence $m_\pm \notin B_3$ and therefore $m_\pm \in B$. This shows that consecutive elements of B are never more than 6 integers apart. □

3.3 On the Length of the Recoding

The original LFG method called for an additional set (called B_2 in [10]) to specifically force the scalar recoding to be of the same length as its q-ary expansion, see [10, Algorithm 6]. With our modification (Algorithm 5), the length can be one digit longer, namely $h+1$. However, this last digit a_h can only be 0 or 1, therefore, only points mq^jP_i and q^hP_i for $m \in M, 0 \leqslant j \leqslant h-1, 1 \leqslant i \leqslant n$ need to be precomputed, for a total of $3nh+n$ points.

Algorithm 5 Adjusted scalar recoding

Input: $\{a_j\}_{0\leqslant j\leqslant h-1}$, $0 \leqslant a_j < q$ such that $a = \sum_{j=0}^{h-1} a_j q^j$.
Output: $\{(\epsilon_j m_j, b_j)\}_{0\leqslant j\leqslant h}$, $\epsilon_j \in \{\pm 1\}$, $m_j \in M$, $b_j \in B$ such that $a = \sum_{j=0}^{h} \epsilon_j m_j b_j q^j$.
1: $a_h \leftarrow 0$
2: **for** $j = 0$ to $h-1$ **do**
3: Obtain $\epsilon_j \in \{\pm 1\}, m_j \in M, b_j \in B, \alpha_j \in \{0,1\}$ such that $a_j = \epsilon_j m_j b_j + \alpha_j q$ in (1)
4: $a_{j+1} = \alpha_j + a_{j+1}$ ▷ Note that $a_h = 0$ or 1
5: **return** $\{(\epsilon_j m_j, b_j)\}_{0\leqslant j\leqslant h}$

4 Construction of Optimal Bucket Sets for Efficient MSM Computation

We want to generalize the LFG construction of bucket sets B to provide examples of optimal-sized sets. We first start by generalizing Property 1.

Property 2. *Let p be a prime. Given $q = p^c$, for all $0 \leqslant t \leqslant q$, there exist $b \in B$ and $m \in M$ such that*

$$t = mb \quad or \quad q - t = mb \ .$$

We refer to B as the bucket set and M as the (unsigned) multiplier set. In the previous sections, $B = B_1$, $M = \{1,2,3\}$ and $p = 2$. In this context, a simple cardinality argument shows the following.

Theorem 1 (Lower bound on the bucket set size)**.** *If Property 2 holds, then*

$$2 \cdot |B| \cdot |M| \geqslant q \ .$$

Proof. We remark that there are at most $|B| \cdot |M|$ integers t of the form mb for $m \in M, b \in B$, and similarly for the t's of the form $q - mb$. The conclusion follows: if Property 2 holds, all $0 \leqslant t \leqslant q$ must be representable in one of these two ways. □

The set M determines the number of precomputed points[2], which, in the notation of LFG is $|M|nh$. Hence, if $|M| = 2$, then $|B| \geqslant q/4$ and if $|M| = 3$, $|B| \geqslant q/6$. The case $|M| = 1$ is Pippenger's variant, which is therefore optimal [10, Table 1]. We now describe an optimal bucket set B for $|M| = 2$, satisfying $|B| = q/4 + O(1)$.

4.1 Optimal Bucket Set for $|M| = 2$

Let p be an odd prime and $q = p^c$ for $c \in \mathbb{N}$. Define $M = \{1, 2\}$ and let B consist of 0 together with all integers $0 < b < q/2$ of the form $b = p^k\beta$, where $0 \leqslant k \leqslant c-1$ is an integer and β is a quadratic residue mod q (in particular, it is coprime to p). We show the following.

Theorem 2 (Optimal Bucket for 2-Multipliers)**.** *If p is an odd prime such that*

$$\left(\frac{-1}{p}\right) = -\left(\frac{2}{p}\right) = 1 \ ,$$

then Property 2 holds for $M = \{1, 2\}$ and B as described above.

Remark. $p = 5$ is the first such prime; the requirement of the theorem is equivalent to $p \equiv 5 \pmod 8$.

Remark. Note that $\beta \in \mathbb{Z}$ coprime to p is a quadratic residue mod p^c if and only if it is a quadratic residue mod p. This follows from Hensel's lemma, as any root of the equation $x^2 \equiv \beta \pmod p$ is simple, hence lifts to a unique root mod p^c.

Proof. We divide the proof into several cases. We first deal with $0 < t < q$ coprime to p.

$t < q/2$ **and** $(\frac{t}{p}) = 1$**:** $t \in B$, so there's nothing to show.
$t < q/2$ **even and** $(\frac{t}{p}) = -1$**:** $t/2 = b \in B$, so $t = 2b$.
$t < q/2$ **odd and** $(\frac{t}{p}) = -1$**:** $q-t$ is even and $(\frac{q-t}{p}) = -1$. Therefore, $(q-t)/2 = b \in B$ and $q - t = 2b$.
$t > q/2$ **even and** $(\frac{t}{p}) = 1$**:** in this case $q - t \in B$.
$t > q/2$ **even and** $(\frac{t}{p}) = -1$**:** here, $t/2 = b \in B$, so $t = 2b$.
$t > q/2$ **odd:** $q - t$ is even and either $q - t \in B$ or $(q - t)/2 \in B$.

Now to the general case (we suppose $t \neq 0, q$), when $t = p^k\tau$ $(0 \leqslant k < c)$, where $p \nmid \tau$. The condition

$$t = mb \quad \text{or} \quad q - t = mb$$

is equivalent to

$$p^k\tau = mb \quad \text{or} \quad p^c - p^k\tau = mb \ .$$

[2] Since precomputed points are of the form mq^jP_i, for $m \in M$, $0 \leqslant j \leqslant h-1$ and $1 \leqslant i \leqslant n$, it is not important to require that q be a power of 2.

Choosing $b < q/2$ of the form $p^k\beta$, with $p \nmid \beta$ and $\beta < p^{c-k}/2$ quadratic residue mod p^c (equivalently, mod p^{c-k}), the previous equation reads

$$\tau = m\beta \quad \text{or} \quad p^{c-k} - \tau = m\beta \ .$$

By our initial work, this condition is satisfied for some $\beta \in B$ when $p \equiv 5 \pmod 8$. □

We now show $|B| = q/4 + O(1)$.

Theorem 3. *The bucket set B in this section has cardinality*

$$|B| = \frac{q-1}{4} + 1 \ .$$

Proof. We begin with a lemma.

Lemma 3. *Let $p \equiv 1 \pmod 4$ be prime and $k \in \mathbb{N}$. The number of quadratic residues mod p^k less than $p^k/2$ is $p^{k-1}(p-1)/4$.*

Proof. The number of quadratic residues mod p^k is $p^{k-1}(p-1)/2$, since they form a cyclic group of index 2 inside the group of invertible classes mod p^k, of order $\varphi(p^k) = p^{k-1}(p-1)$. Also, the subset of those quadratic residues $< p^k/2$ is in bijection with its complement, via the map $t \mapsto p^k - t$, using the fact that -1 is a quadratic residue mod p^k. This leads to the result. □

Returning to the proof of the theorem, we partition B as

$$B = \{0\} \bigcup \bigcup_{k=0}^{c-1} \{t = p^k\beta \colon 0 < \beta < p^{c-k}/2 \text{ is a quadratic residue mod } p^{c-k}\} \ .$$

Taking cardinalities, and using the previous lemma, we find

$$|B| = 1 + \sum_{k=0}^{c-1} \frac{p^{c-k-1}(p-1)}{4} = 1 + \frac{p-1}{4} \cdot \frac{p^c - 1}{p-1} = 1 + \frac{p^c - 1}{4} \ .$$

□

Remark. Since b is a quadratic residue mod p^c if and only if it is a quadratic residue mod p, we deduce that elements of B are never more than p integers apart, generalizing Proposition 4 to this context (where we can take $p = 5$).

4.2 A General Construction of Optimal Buckets

Here we show the way to modify Property 1 (or 2) in order to obtain a scalar multiplication algorithm with a runtime of essentially $n(h+1) + \frac{q}{2|M|}$ point operations, by precomputing $|M|n(h+1)$ points.

We propose the following modification. As usual, we let $q = p^c$, where p is prime and $c \in \mathbb{N}$.

Property 3. *For all $t \in \mathbb{Z}$, there exist $\epsilon_t \in \{\pm 1\}$, $m_t \in M$, $b_t \in B$, such that*

$$t \equiv \epsilon_t m_t b_t \pmod{q} \ .$$

We now let $B \subseteq [0, q-1]$ be a bucket set such that $0 \in B$ and

$$M = \{1, 2, \ldots, |M|\} \ . \tag{11}$$

Assuming Property 3 holds in this case, we rewrite [10, Algorithm 6] to accommodate a recoding without *a priori* restricting α_j:

$$a_j = \epsilon_j m_j b_j + \alpha_j q \ , \quad \epsilon_j \in \{\pm 1\},\ m_j \in M,\ b_j \in B,\ \alpha_j \in \mathbb{Z} \ . \tag{12}$$

The result is the following new scalar recoding Algorithm 6.

Algorithm 6 New scalar recoding

Input: $\{a_j\}_{0 \leqslant j \leqslant h-1}$, $0 \leqslant a_j < q$ such that $a = \sum_{j=0}^{h-1} a_j q^j$.
Output: $\{(\epsilon_j m_j, b_j)\}_{0 \leqslant j \leqslant h}$, $\epsilon_j \in \{\pm 1\}$, $m_j \in M$, $b_j \in B$ such that $a = \sum_{j=0}^{h} \epsilon_j m_j b_j q^j$.
1: $a_h \leftarrow 0$
2: **for** $j = 0$ to $h-1$ **do**
3: Obtain $\epsilon_j, m_j, b_j, \alpha_j$ as in (12) such that $a_j = \epsilon_j m_j b_j + \alpha_j q$ ▷ $|\alpha_j| \leqslant |M|$
4: $a_{j+1} = \alpha_j + a_{j+1}$ ▷ Now $|a_{j+1}| < q + |M|$
5: Obtain ϵ_h, m_h, b_h such that $a_h = \epsilon_h m_h b_h$ ▷ $|a_h| \leqslant |M|, b_h = 0, 1, \alpha_h = 0$
6: **return** $\{(\epsilon_j m_j, b_j)\}_{0 \leqslant j \leqslant h}$

We need to show that Algorithm 6 terminates after Line 5. This is done by addressing the statements found in the comments.

Proposition 5. *In Algorithm 6, we have, for $-1 \leqslant j \leqslant h-1$ (where we define $\alpha_{-1} = 0$), after Line 4,*

$$\begin{cases} |\alpha_j| \leqslant |M| \ , \\ |a_{j+1}| < q + |M| \ , \\ |a_h| \leqslant |M| \ . \end{cases}$$

Proof. The first two statements are proved together by induction on $j \geqslant -1$. The base step is clear, since $\alpha_{-1} = 0$ and a_0 is not modified in Line 4. Supposing $|\alpha_j| \leqslant |M|$ and $|a_{j+1}| < q + |M|$, from Line 3 we deduce

$$|\alpha_{j+1}| \leqslant \frac{m_{j+1} b_{j+1}}{q} + \frac{|a_{j+1}|}{q} \leqslant |M| \left(1 - \frac{1}{q}\right) + 1 - \frac{1}{q} + \frac{|M|}{q} = |M| + 1 - \frac{1}{q} \ ,$$

and therefore, since α_{j+1} is an integer, $|\alpha_{j+1}| \leqslant |M|$. In addition, in Line 4, the new value of a_{j+2} is $a_{j+2} + \alpha_{j+1}$, therefore we can bound the updated value as

$$|a_{j+2}| < |M| + q \ .$$

This completes the inductive step. Finally, note that, as the initial value $a_h = 0$ is updated in Line 4 to $a_h + \alpha_{h-1} = \alpha_{h-1}$, we have the stricter bound $|a_h| = |\alpha_{h-1}| \leqslant |M|$. □

The new scalar recoding allows us to run Pippenger's algorithm as before [10, Algorithms 4 and 3] with at most

$$\big(n(h+1)+|B|+d-4\big)$$

curve additions – where d is the maximal distance between consecutive elements of B – and the help of

$$n(h+1)|M|$$

precomputed points. The main advantage of the recoding given by Algorithm 6 is that it allows us to use a bucket set B of optimal size $\frac{q}{2|M|}+O(1)$.

Theorem 4. *Let μ be a positive integer, $p>2$ be prime with $p\equiv 1 \pmod{2\mu}$. Assume $\{\pm 1,\cdots,\pm\mu\}$ form a complete set of representatives of $(\mathbb{Z}/p)^*$ modulo 2μ-th powers. Then, for any $c\in\mathbb{N}$, Property 3 holds for $q=p^c$, the multiplier set $M=\{1,2,\ldots,\mu\}$ and the bucket set*

$$B=\{0\}\\ \bigcup_{k=0}^{c-1}\Big\{0<b<q\colon b=p^k\beta, \text{where } 0<\beta<p^{c-k} \text{ is a } 2\mu\text{-th power modulo } p^{c-k}\Big\}.$$

Moreover, the maximal distance between consecutive integers in B is p and

$$|B|=\frac{q}{2\mu}+O(1)=\frac{q}{2\,|M|}+O(1)\ .$$

Proof. We claim that, for any $\kappa\in\mathbb{N}$, the set $S=\{\pm 1,\cdots,\pm\mu\}$ constitutes a complete set of representatives of $(\mathbb{Z}/p^\kappa)^\times$ (the invertible classes modulo p^κ) modulo 2μ-th powers. Indeed, since $2\mu \mid p^{\kappa-1}(p-1)$, knowing that $(\mathbb{Z}/p^\kappa)^\times$ is cyclic, the group

$$(\mathbb{Z}/p^\kappa)^\times/\left((\mathbb{Z}/p^\kappa)^\times\right)^{2\mu}$$

has order 2μ. Moreover, for $r,s\in S$, by Hensel's lemma, the equation

$$r\equiv sx^{2\mu} \pmod{p^\kappa} \text{ is solvable in } \mathbb{Z} \iff r\equiv sx^{2\mu} \pmod{p} \text{ is solvable in } \mathbb{Z}\ .$$

By assumption, this shows that if $r\neq s$, they represent different classes and thus proving our claim. In other words, we have a partition

$$(\mathbb{Z}/p^\kappa)^\times=\bigcup_{1\leqslant m\leqslant\mu}\Big(m\left((\mathbb{Z}/p^\kappa)^\times\right)^{2\mu}\bigcup -m\left((\mathbb{Z}/p^\kappa)^\times\right)^{2\mu}\Big)\ .$$

Let $t\in\mathbb{Z}$. As seen in the proof of Theorem 2, write $t=p^k\tau$ where $p\nmid\tau$. If $k\geqslant c$, then $t\equiv 0\equiv 1\cdot 1\cdot 0 \pmod q$. Otherwise, let $\kappa=c-k\in\mathbb{N}$. From our claim, solving in β (a 2μ-th power modulo p^κ) the equation

$$\tau\equiv\epsilon m\beta \pmod{p^\kappa}$$

for some $1 \leqslant m \leqslant \mu$ and $\epsilon \in \{\pm 1\}$ will yield, for $b = p^k\beta \in B$, an expression

$$t \equiv p^k\tau \equiv \epsilon m p^k \beta \equiv \epsilon m b \pmod{q} \ ,$$

thus showing Property 3.

To count the elements of B, as in Theorem 3 note that B is already defined as a disjoint union. Therefore

$$|B| = 1 + \sum_{\kappa=1}^{c} \frac{p^{\kappa-1}(p-1)}{2\mu} = 1 + \frac{p^c-1}{p-1} \cdot \frac{p-1}{2\mu} = 1 + \frac{p^c-1}{2\mu} = \frac{q}{2\mu} + O(1) \ .$$

Finally,

$$\left((\mathbb{Z}/p^c)^\times\right)^{2\mu} \subseteq B \ ,$$

where on the left we consider representatives in $[1, q-1]$, and we have seen via Hensel's lemma that the condition that b be a 2μ-th power mod q is equivalent to b being a 2μ-th power mod p. This proves the claim on the maximal distance of elements of B. □

Remark. A simple cardinality argument similar to Theorem 1 shows that any bucket set B satisfying Property 3 is such that $|B| \geqslant q/(2\,|M|)$. Therefore Theorem 4 is optimal.

We want to provide a criterion for finding primes p satisfying the hypotheses of Theorem 4.

Proposition 6. *Let $\mu \in \mathbb{N}$ and suppose that $p = 2\mu+1$ is prime. Then $\{\pm 1, \cdots, \pm\mu\}$ form a complete set of representatives of $(\mathbb{Z}/p)^\times / \left((\mathbb{Z}/p)^\times\right)^{2\mu}$.*

Proof. We have, by Fermat's little theorem,

$$\left((\mathbb{Z}/p)^\times\right)^{2\mu} = \{1\} \ ,$$

and $\mu = \frac{p-1}{2}$, so

$$\{\pm 1, \cdots, \pm\mu\} = \left\{\pm 1, \cdots, \pm\frac{p-1}{2}\right\} = (\mathbb{Z}/p)^\times \ .$$

□

Remark. The first few values of μ, namely $1, 2, 3, 5, 6, 8, 9, 11$, provide via Proposition 6 optimal bucket sets of cardinality $q/(2\mu) + O(1)$ in Theorem 4. Table 5 in the appendix lists our new bucket set constructions for $q = 7^c, 4 \leqslant c \leqslant 11$.

We will now show that, on a $j = 0$ elliptic curve (with equation $y^2 = x^3 + b$), our new property allows ideally to divide the storage requirement by 3. For instance, with nh stored points (the same as Pippenger's variant), one can execute a variant of Pippenger's algorithm in essentially $nh + q/6$ point operations. The result given in Section 5 has to be compared to this section's results with $\mu = 3$ and $p = 7$.

5 Combining Efficient Endomorphisms with Optimal Buckets for Efficient MSM Computation

Many families of pairing-friendly curves over $\mathbb{F}_p$ have j-invariant equal to zero. They have an equation $y^2 = x^3 + b$ for some $b \in \mathbb{F}_p$. Therefore, they are endowed with an endomorphism ω such that $\omega^3(P) = P$ for all P on the elliptic curve. We can write $\omega(x, y) = (\zeta_3 x, y)$, where $\zeta_3 \in \mathbb{F}_p$ such that $\zeta_3^3 = 1$. The computation of ω can therefore be done on the fly, and corresponding points do not need to be stored, which is now what we want to take advantage of.

The endomorphism ring of these curves is isomorphic to $\mathbb{Z}[\omega]$, where ω is a complex cube root of unity (using the same letter as for the fast endomorphism). We need to update the construction of Section 4.2 to work with a recoding where in Property 3, ϵ_t can be any unit in $\mathbb{Z}[\omega]$. We will replace that property with

Property 4. *For all $t \in \mathbb{Z}[\omega]$, there exist $\epsilon_t \in U = \{\pm 1, \pm\omega, \pm\omega^2\}$, $m_t \in M$, $b_t \in B$, such that*

$$t \equiv \epsilon_t m_t b_t \pmod{q} \ .$$

Let's consider the first implementation of this idea. Let $M = \{1\}$, $q = p^c$, where $p = 2 - \omega$. Note that $p \mid 7$ in $\mathbb{Z}[\omega]$. Also, $\mathbb{Z}[\omega]$ is a (norm-)Euclidean ring, and hence any $t \in \mathbb{Z}[\omega]$ has a representative in $\mathbb{Z}[\omega]/q$ of modulus less than $|q| = 7^{c/2}$. Finally, $|\mathbb{Z}[\omega]/q| = 7^c$. We now show that any $t \in \mathbb{Z}[\omega]$ has a base q expansion of length bounded by $\lfloor \frac{\log |t|}{\log |q|} \rfloor + 1$. The key point to achieve this is a controlled Euclidean algorithm.

Lemma 4. *Let $\alpha, \beta \in \mathbb{Z}[\omega]$ and $\beta \neq 0$. There exist $\delta, \rho \in \mathbb{Z}[\omega]$ such that*

$$\alpha = \beta\delta + \rho \ , \tag{13}$$

where $|\rho| < |\beta|$ and in addition

$$|\delta| \leqslant \left| \frac{\alpha}{\beta} \right| \ .$$

Proof. Rewrite (13) as

$$\tau = \delta + \varepsilon \ , \quad \delta \in \mathbb{Z}[\omega] \ , \quad |\varepsilon| < 1 \ ,$$

where $\tau = \alpha/\beta$ and $\varepsilon = \rho/\beta$. Multiplying by a unit in U, we can suppose that $0 \leqslant \arg(\tau) < \pi/3$. We analyze the two cases for τ as depicted in Fig.1. Here, we suppose without loss of generality that $\pi/6 \leqslant \arg(\delta_1) < \pi/3$.

It is easily seen that in this case, δ_2 is closer to the origin O than D, and that in any case, δ_1 is closer to O than both.

Case 1: $\tau = \tau_1$ lies inside the intersection of the fundamental parallelogram $\delta_1\delta_2 CD$ and the disk $\mathcal{D}$ centered at δ_1 passing through δ_2 and D (the boundary of that region is shown as a red dashed arc). In this case, we let $\delta = \delta_1$. Then $|\delta_1| \leqslant |\tau_1|$ and $|\varepsilon_1| = |\tau_1 - \delta_1| < 1$.

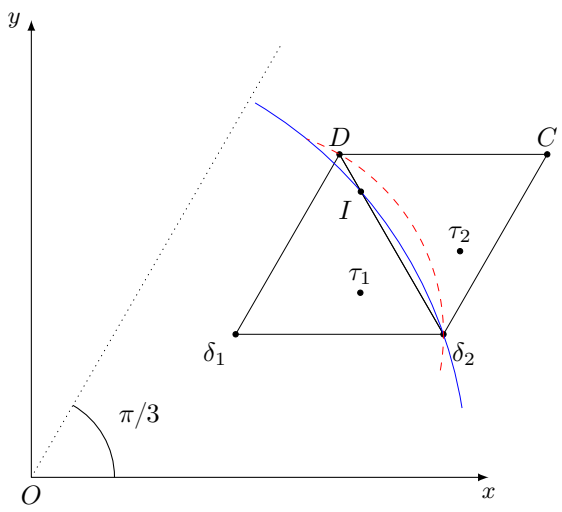

Fig. 1. Controlled Euclidean Division

Case 2: $\tau = \tau_2$ lies in the parallelogram $\delta_1\delta_2CD$ but outside $\mathcal{D}$ or on its dashed boundary. In this case, notice that the blue circle $\mathcal{C}$ centered at O and passing through δ_2 will intersect[3] the segment $[\delta_2, D]$ in a point I. This is because point D lies further away from O than δ_2 (or, if $\arg(\delta_1) = \pi/6$, they are both equidistant from O). Since the radius of $\mathcal{D}$ is smaller than that of $\mathcal{C}$, we deduce that the arc $\widehat{\delta_2 I}$ of $\mathcal{C}$ all lies strictly inside $\mathcal{D}$, except possibly at the endpoints.

This analysis shows that, if we let $\delta = \delta_2$, we have $|\delta_2| \leqslant |\tau_2|$ and $|\varepsilon_2| = |\tau_2 - \delta_2| < 1$. □

Remark. An algorithmic implementation of this idea is given in Algorithm 7, where a generic $\tau = \kappa/q$ is first transported to the first sector so that $0 \leqslant \arg(\tau) < \pi/3$ before choosing the right δ and finally transported back to the sector it came from.

Theorem 5. *We have*

$$(\mathbb{Z}[\omega]/p)^\times = \{\pm 1, \pm\omega, \pm\omega^2\} \ .$$

For any $c \in \mathbb{N}$, Property 4 holds for $q = p^c$, the multiplier set $M = \{1\}$ and the bucket set

$$B = \{0\} \cup \bigcup_{k=0}^{c-1} B_k \ ,$$

[3] It may happen, but only for δ_1 close to the origin, that $\widehat{D\delta_2 O} \geqslant \pi/2$, in which case $\mathcal{C}$ won't intersect the segment, but this is without consequence for the sequel, since the relevant arc of $\mathcal{C}$ all lies inside the triangle $\delta_1\delta_2 D$, hence inside the disk $\mathcal{D}$.

where

$$B_k = \left\{0 < |b| < |q| \colon b = p^k\beta, \textit{where } 0 < |\beta|^2 < 7^{c-k} \textit{ is a 6-th power modulo } p^{c-k}\right\} \ .$$

Moreover,

$$|B| = \frac{7^c}{6} + O(1) \ .$$

Proof. The proof is similar to the proof of Theorem 4 when $\mu = 3$. It will only be necessary to go over the differences. Note initially that,

$$(\mathbb{Z}[\omega]/p)^\times = \{\pm 1, \pm\omega, \pm\omega^2\} \ .$$

Indeed, by the considerations after the statement of Property 4, the cardinalities of the left- and right-hand sides of the previous equality match. Moreover, any two distinct elements of $\{\pm 1, \pm\omega, \pm\omega^2\}$ are not congruent modulo p, because their difference has algebraic norm either 1, 2 or 3, coprime to 7. Also, for any $\kappa \in \mathbb{N}$,

$$\mathcal{O}_\kappa = (\mathbb{Z}[\omega]/p^\kappa)^\times / \left((\mathbb{Z}[\omega]/p^\kappa)^\times\right)^6$$

is cyclic of order 6. A generalized version of Hensel's lemma will then show that two elements $r, s \in \mathbb{Z}[\omega]$ have distinct reductions in $\mathcal{O}_\kappa$ if and only if their reductions in $\mathcal{O}_1$ are distinct. Moreover, since

$$\mathcal{O}_1 = (\mathbb{Z}[\omega]/p)^\times \ ,$$

we obtain Property 4 for $t \in \mathbb{Z}[\omega]$ coprime to p. The general case is dealt in the same way as in Theorem 4 by introducing a suitable power of p to multiply the sixth powers mod p^{c-k}. Note also that an element is a sixth power mod p^{c-k} if and only if it is congruent to 1 mod p. Since $|\mathcal{O}_\kappa| = 7^{\kappa-1}$, a calculation similar to the proof of Theorem 4 shows that

$$|B| = 1 + \sum_{\kappa=1}^{c} 7^{\kappa-1} = 1 + \frac{7^c - 1}{6} = \frac{7^c}{6} + O(1) \ .$$

□

Algorithm 8 now replaces Algorithm 6 in computing a recoding amenable to the bucket algorithm, without any additional precomputation than the Pippenger variant.

Theorem 5 can be seen as a version of Theorem 4 when $\mu = 3$, with nh precomputed points instead of $3nh$. However, there is one fundamental difference, and we show how to deal with it.

Elliptic curve computations take place in a cyclic group $\mathbb{G}$ of prime order N. The parameter $h - 1$ is then defined – when the prime p is chosen in $\mathbb{Z}$ (for

Algorithm 7 Controlled Euclidean algorithm

Input: $\kappa = k_1 + k_2\omega$, $q = q_1 + q_2\omega$ with $k_1, k_2, q_1, q_2 \in \mathbb{Z}$.
Output: $\mathrm{CEA}(\kappa, q) = (\delta, \rho)$, where $\delta, \rho \in \mathbb{Z}[\omega]$ with $\kappa = q\delta + \rho$, $|\rho| < |q|$ and $|\delta| \leqslant |\kappa/q|$.

$u + v\omega \leftarrow \kappa/q$ ▷ $u, v \in \mathbb{Q}$
$u \leftarrow u - v$ ▷ In base $\{1, \theta = \omega + 1\}$
$c_0, c_1 \leftarrow 0$ ▷ To track rotation to first $\pi/3$ sector, where $u, v \geqslant 0$
if $v < 0$ **then**
    $v \leftarrow -v$, $c_0 \leftarrow 1$
while $u < 0$ **do** ▷ At most 2 iterations
    $u \leftarrow u + v$, $v \leftarrow v - u$, $c_1 \leftarrow c_1 + 1$
if $(u - \lfloor u \rfloor)^2 + (v - \lfloor v \rfloor)^2 + (u - \lfloor u \rfloor)(v - \lfloor v \rfloor) < 1$ **then** ▷ Case 1
    $d_1 \leftarrow \lfloor u \rfloor$, $d_2 \leftarrow \lfloor v \rfloor$
else
    if $\lfloor u \rfloor \leqslant \lfloor v \rfloor$ **then**
        $d_1 \leftarrow \lfloor u \rfloor + 1$, $d_2 \leftarrow \lfloor v \rfloor$
    else
        $d_1 \leftarrow \lfloor u \rfloor$, $d_2 \leftarrow \lfloor v \rfloor + 1$
while $c_1 > 0$ **do** ▷ Now rotate back
    $d_2 \leftarrow d_1 + d_2$, $d_1 \leftarrow d_1 - d_2$, $c_1 \leftarrow c_1 - 1$
if $c_0 > 0$ **then**
    $d_2 \leftarrow -d_2$
$d_1 \leftarrow d_1 + d_2$ ▷ Back in base $\{1, \omega\}$
$r_1 \leftarrow k_1 - q_1 d_1 + q_2 d_2$, $r_2 \leftarrow k_2 - q_2 d_1 - q_1 d_2 + q_2 d_2$ ▷ $\rho = \kappa - q\delta$
return $\delta = d_1 + d_2\omega$, $\rho = r_1 + r_2\omega$

instance $p = 7$ when $\mu = 3$) – as the exponent of the largest power of q not exceeding N, in other terms,

$$h = \left\lfloor \frac{\log N}{\log q} \right\rfloor + 1 \ .$$

This is because we must be able to represent any scalar multiplier ($\leqslant N$) in a base q expansion of length at most h. If we now let $p = 2 - \omega$, and let as before $q = p^c$, then, since $|q| = 7^{c/2}$, the corresponding h would double, necessitating twice as many precomputed points ($2nh$, instead of nh). Although this is below the $3nh$ provided by our refinement of the LFG method, we can do better.

Note that since N is large, there is no other copy isomorphic to $\mathbb{G}$ in the elliptic curve (over the field of definition of $\mathbb{G}$). Consequently, the endomorphism ω must act as an isomorphism of $\mathbb{G}$. Therefore, given a point $P \in \mathbb{G}$, $\omega P = \lambda P$ for some $\lambda \in \mathbb{Z}/N$ with $\lambda^3 \equiv 1 \pmod{N}$. This implies that $N \equiv 1 \pmod{3}$ splits in $\mathbb{Z}[\omega]$, so $N = \nu_1\nu_2$, with ν_1, ν_2 primes in $\mathbb{Z}[\omega]$. Since, denoting $\mathbf{0}$ the point at infinity,

$$\mathbf{0} = NP = \nu_1\nu_2 P \ ,$$

we deduce that either $\nu_1 P = \mathbf{0}$ or $\nu_2 P = \mathbf{0}$. Let ν represent the corresponding prime. Then, $|\nu| = \sqrt{N}$ and, if $\rho \equiv a \pmod{\nu}$, then $aP = \rho P$. The bottom line is that we can represent any scalar $a \leqslant N$, having an expansion of length h

Algorithm 8 Complex scalar recoding

Input: $a \in \mathbb{Z}[\omega]$, $q = p^c = (2-\omega)^c$ with $c \in \mathbb{N}$. ▷ $h = \lfloor \frac{\log |a|}{\log |q|} \rfloor + 1$

Output: $\{(\epsilon_j, b_j)\}_{0 \leqslant j \leqslant h-1}$, $\epsilon_j \in U$, $b_j \in B$ such that $a = \sum_{j=0}^{h-1} \epsilon_j b_j q^j$.

$\kappa \leftarrow a$
$h \leftarrow \lfloor \frac{\log |a|}{\log |q|} \rfloor + 1$
List $\leftarrow \{\}$
for $j = 0$ to $h-1$ **do**
    $(\rho, \delta) \leftarrow \mathrm{CEA}(\kappa, q)$
    if $\rho = 0$ **then**
        $\epsilon_j = 1$, $b_j = 0$
    else
        $r \leftarrow \rho$
        while $p \mid r$ **do**
            $r \leftarrow r/p$
        Define $\epsilon_j \in U$ so that $\epsilon_j \equiv r \pmod{p}$
        $b_j = \rho \epsilon_j^{-1}$ ▷ $b_j \in B$
    List $\leftarrow$ List.append$((\epsilon_j, b_j))$
    $\kappa \leftarrow \delta$
return: List

in base 7^c, by an equivalent expansion (of ρ) in base $q = (2-\omega)^c$ of the *same length*. In particular, it is sufficient to precompute the same h powers of q.

Proposition 7. *In Theorem 5, it is possible to label elements of B as $B = \{b_1, \dots b_{|B|}\} \subseteq \mathbb{Z}[\omega]$, in such a way that*

$$|b_{k+1} - b_k|^2 \leqslant 7 \ , \quad \text{for all } 1 \leqslant k \leqslant |B| - 1 \ .$$

Moreover, if $|b_{k+1} - b_k|^2 = 7$, then $b_{k+1} - b_k = \epsilon p$, where $\epsilon \in U = \{\pm 1, \pm\omega, \pm\omega^2\}$.

Proof. We first focus on the subset $B_0 \subseteq B$ (see Theorem 5 for the definition of B_0), normalized by dividing $B_0 - 1$ (B_0 translated by -1) by p so that all its points are in $\mathbb{Z}[\omega]$.

Consider a set S of points of $\mathbb{Z}[\omega]$ inside some (large) disk Δ. Informally, we say S is a *ziggurat* if, up to rotation, "horizontal layers away from the center of Δ are piled up on top of each other".

Formally, rotate Δ about its center, so that lines of points of S spaced by 1 are horizontal, then translate to move the center of Δ to 0. We will suppose henceforth that S is normalized in this fashion. A *layer* or *level* of S corresponds to points with the same imaginary part.

A *neighbor* of $z \in \mathbb{Z}[\omega]$ is one of the points $z + \epsilon$ where $\epsilon \in U$. A point on the *boundary* ∂S of S is a point of S which doesn't have all its neighbors in S. The points of $S \backslash \partial S$ are called *internal* points of S.

The set S^{top} is called a *top ziggurat* if $S^{\mathrm{top}} \subseteq \Delta \cap \{z \colon \Im z \geqslant 0\}$ and, for all $z \in S^{\mathrm{top}}$ with $\Re z \leqslant 0$ (resp. $\Re z \geqslant 0$), any $\zeta \in S^{\mathrm{top}}$ of z with $\Im \zeta > \Im z$ has $\Re \zeta \geqslant \Re z - 1/2$ (resp. $\Re \zeta \leqslant \Re z + 1/2$); in other words, in a top ziggurat, upper

layers are smaller and stacked on top of each other, like in a Hanoi tower game. In particular, if $z \in \partial S^{\text{top}}$ and $\Re z \leqslant 0$, then either there are no $\zeta \in S^{\text{top}}$ with $\Im\zeta > \Im z$ (z is in the top layer), or one of $z+\omega$ or $z+\omega+1$ is in ∂S^{top}.

A *bottom ziggurat* is symmetric with respect to the origin of a top ziggurat. A *ziggurat* is the union of a top and a bottom ziggurat. The ziggurat is *full* if there exist points of the bottom ziggurat that are neighbors of points of the top ziggurat.

It is relatively straightforward to see that the boundary ∂S of a full ziggurat S consists of a Hamiltonian cycle of neighboring points. Indeed, start from the first $z_1 \in S$ with smallest $\Im z_1 \geqslant 0$ and $\Re z_1 \leqslant 0$ (i.e. at the "bottom left" of the top ziggurat). Move "along the boundary" of S staying at the same horizontal level (adding $+1$) until you can jump up (by adding ω or $\omega+1$) in S. Repeat until you reach the top of the top ziggurat, then come down on the other side of the imaginary axis. Then move to the bottom ziggurat.

Additionally, $S\backslash\partial S$ is also a full ziggurat. This is because, when removing a point $z \in \partial S$ with $\Im z \geqslant 0$ and $\Re z \leqslant 0$ such that $z+1 \in S\backslash\partial S$, then $z+1 \in \partial(S\backslash\partial S)$ and, as seen above, one of $z+\omega$ or $z+\omega+1$ belongs to ∂S – call it z_ω, with preference given to the leftmost if both[4] belong to ∂S – so that $z_\omega+1 \in \partial(S\backslash\partial S)$ as soon as $z_\omega+1 \in S\backslash\partial S$. In any case, we get that $z+1 \in S\backslash\partial S$ implies $z+1 \in \partial(S\backslash\partial S)$ and there is no point in $S\backslash\partial S$ left of z_ω and at the same level; using symmetric arguments, we derive that $S\backslash\partial S$ is a ziggurat.

At this point, starting from z_0 with the smallest real part, remarking that $z_0 \in \partial S$, move along the boundary ∂S until reaching z, the last point before returning to z_0. From the previous discussion[5], $z+1 \in \partial(S\backslash\partial S)$; since $\Re z > \Re z_0$, we have $z_0 = z+\omega$ and $z+\omega+1 = z_0+1 \in \partial(S\backslash\partial S)$ is a neighbor of z. We can continue the path starting from z_0+1 and running along $\partial(S\backslash\partial S)$ and so forth, each time removing an outer layer of the ziggurat and moving towards the center, as a spider weaves its web.

This construction shows that B_0 has a Hamiltonian path (see Fig. 2). The extension to B is easy, because $B\backslash B_0$ is very sparse therefore, each time a lattice point of B_0 comes close to a point of $B\backslash B_0$, we can just divert our Hamiltonian path for B_0 onto this point before coming back to our original path without increasing the distance between consecutive points.

The final remark is a consequence of the initial normalization, where dividing by p gave us integer points. 🕷

Thanks to the previous proposition, using this relabeling on the bucket set B, one can replace Algorithm 1 in the LFG method with Algorithm 9.

Remark. Since the only primes (up to units) of $\mathbb{Z}[\omega]$ with norm at most 7 are $\omega-1$ (above 3) and $2-\omega$, (above 7, excluding the other prime $3+\omega$), there are at most $d=24$ nonzero integers $k \in \mathbb{Z}[\omega]$ such that $|k|^2 \leqslant 7$, which explains the first step in Algorithm 9.

[4] That can only happen if $\Im z$ is close to 0.

[5] That discussion also holds for layers close to the real axis, which is the case for the levels of z_0 and its neighbors.

Algorithm 9 Subsum accumulation algorithm (with endomorphisms)

Input: $B = \{b_1, b_2, \ldots, b_{|B|}\}$ as in Prop. 7, $S_1, S_2, \ldots, S_{|B|}$
Output: $S = b_1 S_2 + \cdots + b_{|B|} S_{|B|}$

Define a length 25 array $\mathsf{tmp} = [0] \times 25$
for $i = |B|$ to 1 by -1 **do**
    $\mathsf{tmp}[0] = \mathsf{tmp}[0] + S_i$
    $k = b_i - b_{i-1}$
    if $|k|^2 \geqslant 1$ **then**
        $\mathsf{tmp}[k] = \mathsf{tmp}[k] + \mathsf{tmp}[0]$
return $\sum_{\substack{|k|^2 \leqslant 7 \\ (3+\omega) \nmid k}} k \cdot \mathsf{tmp}[k]$

Counting Algorithm: Here we provide an algorithm presented in Algorithm 10 to count the points in B. The idea behind it is to start with the point b_i closest to the disk Δ and move on to the next point closest to b_i which lies within the disk and whose distance from the border of the disk is minimal. The counting path is shown in Figure 2.

Algorithm 10 Counting Algorithm for B

Input: $B = \{b_1, \ldots, b_{|B|}\}$ unsorted.
Output: $\hat{B} = \{\hat{b}_1, \ldots, \hat{b}_{|B|}\}$, s.t. $|\hat{b}_{j+1} - \hat{b}_j| \leqslant \sqrt{7}$

for $1 \leqslant k \leqslant |B|$ **do**
    Pick a point $b_k \in B$ with $|b_k| \leqslant r$ with $r - |b_k| = \min_{j \in \{1,\ldots,|B|\}}(r - |b_j|)$. Set $\hat{b}_1 := b_k$. Add $\hat{b}_1$ to $\hat{B}$.
for $1 \leqslant i \leqslant |B|$, $i \neq k$ **do**
    Take $z_\nu \in \{\pm 1, \pm\omega, \pm\omega^2\} \cdot \sqrt{7} = \{z_1, \ldots, z_6\} \cdot \sqrt{7}$, $\nu \in \{1, \ldots, 6\}$.
    Compute $\hat{b}_i + z_\nu$ with $r - |\hat{b}_i + z_\nu| = \min_{\mu \in \{1,\ldots,6\}}(r - |\hat{b}_i + z_\mu|)$
    Set $\hat{b}_{i+1} := \hat{b}_i + z_\nu$
    if $|\hat{b}_{i+1}| > r$ **then** discard $\hat{b}_{i+1}$
return: $\hat{B}$.

6 Performance Analysis and Implementation

In this section, we analyze the performance of the proposed approaches and present our implementation results.

To conduct evaluation and compare with LFG methd [10] in a fair way, we choose the BLS12-381 curve and the group order is

$$r = \texttt{0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001},$$

which determines the upper bound of scalars involved in $S_{n,r}$. BLS12-381 is a pairing-friendly curve with embedded degree 12 and defined by the equation

$$E(\mathbb{F}_p) : y^2 = x^3 + 4,$$

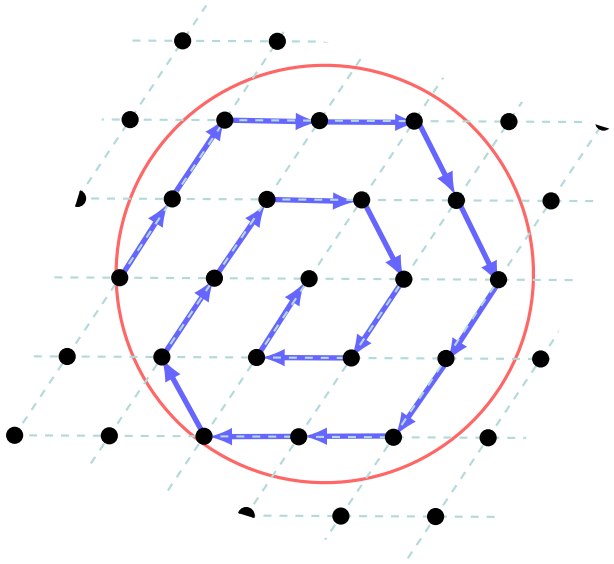

Fig. 2. Hamiltonian path for B_0

where p is the 381-bit field characteristic. Two additive rational point groups $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^2})$ over which bilinear pairings are defined have the same prime order r.

6.1 Theoretical Analysis

The storage cost of precomputation results and the computation complexity of computing MSM with precomputation of different algorithms are summarized in Table 2. From the summary given in Table 1, it is easy to see that the choice of h will affect the overall performance. To evaluate the MSM schemes, we first search for the optimal value of h to minimize the computation cost, and then calculate the corresponding storage[6] and computation cost. The repaired LFG algorithm ($p = 2$) and our algorithm ($p = 7$) have similar performance.

6.2 Implementation

We conducted the performance evaluation on an Apple MacBook Pro with 3.2 GHz M1 Max chip and 64GB memory and built our implementation base on the same code base[7] as that in [10]. Table 3 summarizes our implementation results of different methods for computing $S_{n,r}$ over $\mathbb{G}_1$. We implemented the repaired

[6] Using compression techniques, we need to store one coordinate and one bit.
[7] `https://github.com/LuoGuiwen/MSM_blst/tree/master`

Table 2. Comparison of Storage and Computation Cost of Computing $S_{n,r}$ over $\mathbb{G}_1$ with Different Methods

	Repaired LFG				Ours $p = 7$			
n	q	h	S	C	q	h	S	C
2^{10}	2^{13}	20	2.98 MB	2.33×10^4	7^5	19	2.93 MB	2.33×10^4
2^{11}	2^{14}	19	5.67 MB	4.45×10^4	7^5	19	5.87 MB	4.38×10^4
2^{12}	2^{14}	19	11.34 MB	8.55×10^4	7^5	19	11.74 MB	8.47×10^4
2^{13}	2^{16}	16	19.17 MB	1.54×10^5	7^6	16	19.95 MB	1.59×10^5
2^{14}	2^{16}	16	38.33 MB	2.93×10^5	7^6	16	39.9 MB	2.98×10^5
2^{15}	2^{16}	16	76.67 MB	5.71×10^5	7^6	16	79.8 MB	5.77×10^5
2^{16}	2^{19}	14	134.56 MB	1.10×10^6	7^7	13	131.43 MB	1.05×10^6
2^{17}	2^{20}	13	250.35 MB	2.06×10^6	7^7	13	262.86 MB	1.97×10^6
2^{18}	2^{20}	13	500.7 MB	3.90×10^6	7^7	13	525.73 MB	3.81×10^6
2^{19}	2^{20}	13	1 GB	7.56×10^6	7^7	13	1.05 GB	7.47×10^6
2^{20}	2^{22}	12	1.85 GB	1.45×10^7	7^8	12	1.95 GB	1.46×10^7
2^{21}	2^{22}	12	3.71 GB	2.82×10^7	7^8	12	3.91 GB	2.82×10^7

S: storage cost C: computation cost in the number `Add` operations

LFG method as well as our general optimal bucket construction with $p = 7$ and $M = \{1, 2, 3\}$. As shown in Table 3, the repaired LFG implementation with $p = 2$ and $M = \{1, 2, 3\}$ is about 15.8% to 40.6% faster than the Pippenger one in the blst library. In addition, the general optimal bucket construction with $p = 7$ can achieve the similar performance with the repaired LFG method with $p = 2$ and the same multiplier set $M = \{1, 2, 3\}$. For certain values of n our method with $p = 7$ can achieve a modest improvement of up to 4.4% such as for $n = 2^{17}$, when compared to the repaired LFG approach.

7 Conclusion

MSM is the major computation bottleneck for the proof generation of many pairing-based zkSNARK schemes. A major direction for MSM acceleration is making trade-offs between storage and computation. Both the popular Pippenger algorithm and the recent LFG algorithm follow this direction.

In this paper, we revised an important property proposed in the LGF algorithm and designed a more efficient MSM algorithm. The performance of the new algorithm is verified by both theoretical analysis and experiment. Furthermore, we proposed a method to find the optimal bucket size under the LGF framework.

We also introduced a bucket-amenable recoding using fast endomorphisms on $j = 0$ elliptic curves to divide the storage requirement by 3, at almost no performance penalty, compared to our LFG already optimized algorithm. It would be interesting to investigate if curve endomorphisms can be used not only for the benefit of storage, but also to boost the performance of MSM algorithms.

[8] `https://github.com/supranational/blst`

Table 3. Experimental Results for Computing $S_{n,r}$ over $\mathbb{G}_1$ with Different Methods

n	Pippenger Implementation in blst[8]	Repaired LFG $p = 2$ $M = \{1, 2, 3\}$	Our Method $p = 7$ $M = \{1, 2, 3\}$
2^{10}	15.28 ms	9.08 ms	9.07 ms
2^{11}	27.40 ms	17.70 ms	17.13 ms
2^{12}	49.15 ms	32.93 ms	32.78 ms
2^{13}	90.50 ms	61.34 ms	62.17 ms
2^{14}	166.07 ms	113.27 ms	116.27 ms
2^{15}	305.24 ms	217.49 ms	217.49 ms
2^{16}	556.75 ms	440.38 ms	423.15 ms
2^{17}	1.05 s	849.76 ms	809.67 ms
2^{18}	1.95 s	1.54 s	1.51 s
2^{19}	3.57 s	2.94 s	2.91 s
2^{20}	6.91 s	5.85 s	5.86 s
2^{21}	13.3 s	11.2 s	11.2 s

References

1. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: IEEE S&P 2014. pp. 459–474. IEEE (2014)
2. Bernstein, D.J., Doumen, J., Lange, T., Oosterwijk, J.J.: Faster batch forgery identification. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012. pp. 454–473. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
3. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 103–112. ACM (1988)
4. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast exponentiation with precomputation. In: Rueppel, R.A. (ed.) Advances in Cryptology — EUROCRYPT' 92. pp. 200–207. Springer Berlin Heidelberg, Berlin, Heidelberg (1993)
5. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Advances in Cryptology - EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer (2013)
6. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: ACM STOC 1985. pp. 291–304. ACM (1985)
7. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6477, pp. 321–340. Springer (2010). https://doi.org/10.1007/978-3-642-17373-8_19, `https://doi.org/10.1007/978-3-642-17373-8\_19`
8. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer

Science, vol. 9666, pp. 305–326. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_11, https://doi.org/10.1007/978-3-662-49896-5_11
9. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: ACM STOC 1992. pp. 723–732. ACM (1992)
10. Luo, G., Fu, S., Gong, G.: Speeding up multi-scalar multiplication over fixed points towards efficient zksnarks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(2), 358–380 (2023). https://doi.org/10.46586/TCHES.V2023.I2.358-380, https://doi.org/10.46586/tches.v2023.i2.358-380
11. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 2111–2128. ACM (2019). https://doi.org/10.1145/3319535.3339817, https://doi.org/10.1145/3319535.3339817
12. Pippenger, N.: On the evaluation of powers and related problems. In: 17th Annual Symposium on Foundations of Computer Science (sfcs 1976). pp. 258–263 (1976). https://doi.org/10.1109/SFCS.1976.21

Appendices

A The New Bucket Set Constructions for $q = 2^c, 10 \leqslant c \leqslant 31$

Table 4 lists our new bucket set constructions for $q = 2^c, 10 \leqslant c \leqslant 31$, where the maximum difference d between neighbors of $|B|$ is always equal to 6. When compared to the bucket sets constructed in [10], our bucket sets do not rely on a specific elliptic curve. Since the time complexity for computing $S_{n,r}$ is approximately $nh + |B|$, a smaller $|B|$ results in lower time complexity given the same h. Hence, radices 2^c for $c \in \{21, 23, 25, 27, 28, 30, 31\}$ are abandoned in the table.

Table 4. New Bucket Sets Constructions for $q = 2^c, 10 \leqslant c \leqslant 31$

q	h	$\|B\|$	$\|B\|/q$
2^{10}	26	226	0.22070
2^{11}	24	448	0.21875
2^{12}	22	897	0.21899
2^{13}	20	1791	0.21863
2^{14}	19	3587	0.21893
2^{15}	17	7167	0.21872
2^{16}	16	14340	0.21881
2^{17}	15	28672	0.21875
2^{18}	15	57346	0.21876
2^{19}	14	114686	0.21875
2^{20}	13	229380	0.21875
2^{21}	13	458750	0.21875
2^{22}	12	917508	0.21875
2^{23}	12	1835005	0.21875
2^{24}	11	3670018	0.21875
2^{25}	11	7340030	0.21875
2^{26}	10	14680067	0.21875
2^{27}	10	29360126	0.21875
2^{28}	10	58720261	0.21875
2^{29}	9	117440511	0.21875
2^{30}	9	234881027	0.21875
2^{31}	9	469762045	0.21875

B The New Bucket Set Constructions for $q = 7^c, 4 \leqslant c \leqslant 11$

Table 5 lists our new bucket set constructions for $q = 7^c, 4 \leqslant c \leqslant 11$, where the maximum difference d between neighbors of $|B|$ is always equal to 7.

Table 5. New Bucket Sets Constructions for $q = 7^c, 4 \leqslant c \leqslant 11$

q	h	$\lvert B\rvert$	$\lvert B\rvert/q$
7^4	23	401	0.167
7^5	19	2802	0.167
7^6	16	19609	0.167
7^7	13	137258	0.167
7^8	12	960801	0.167
7^9	11	6725602	0.167
7^{10}	10	47079209	0.167
7^{11}	9	329554458	0.167