Ed25519 Circuits

Saravanan Vijayakumaran

August 8, 2023

1 Ed25519 Elliptic Curve Equations

Let q be the prime $2^{255} - 19$. The ed25519 curve is given by the set

$$E = \left\{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q \mid -x^2 + y^2 = 1 + dx^2y^2\right\}, \tag{1}$$

where $d = -\frac{121665}{121666}$. The addition law is given by

$$(x_3, y_3) = (x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 + x_1x_2}{1 - dx_1x_2y_1y_2}\right). \tag{2}$$

The identity element is the point $(0, 1)$. The above addition law works even when the points being added are identical, i.e. $(x_1, y_1) = (x_2, y_2)$. So we do not require a different addition rule for point doubling.

2 Verifying Point Addition

In the context of SNARKs, we are concerned with verifying that three points $P, Q, R \in E$ satisfy $R = P + Q$ using an arithmetic circuit. The main challenge is the representation of the arithmetic over $\mathbb{F}_q$ using field elements from a different finite field $\mathbb{F}_n$. The field $\mathbb{F}_n$ is called the *native field* and the arithmetic over $\mathbb{F}_q$ is called *non-native arithmetic*.

A few popular choices for the native field have n either equal to a 254-bit prime or a 255-bit prime. The *capacity* of such fields is then atmost 253 bits, i.e. they can represent any integer in the range $\{0, 1, \ldots, 2^{253} - 1\}$. In this document, we exclusively focus on this case.

An element of $\mathbb{F}_q$ requires 255 bits and hence cannot be expressed a single element of $\mathbb{F}_n$. While two elements of $\mathbb{F}_n$ suffice to represent an element of $\mathbb{F}_q$, we will use more than two to allow the representations of products of $\mathbb{F}_q$ elements.

Suppose we use four elements of $\mathbb{F}_n$ to represent an element of $\mathbb{F}_q$. Let $a \in \mathbb{F}_q$ and $a_0, a_1, a_2, a_3 \in \mathbb{F}_n$ such that

$$a = \sum_{i=0}^{3} a_i 2^{64i}.$$

In the *reduced representation* of a, the a_i's are in the range $\{0, 1, 2, \ldots, 2^{64} - 1\}$. The elements a_i are called the *limbs* of a.

Arithmetic operations can lead to *unreduced representations*. Let $a = \sum_{i=0}^{3} a_i 2^{64i}, b = \sum_{i=0}^{3} b_i 2^{64i}$ for $a_i, b_i \in \mathbb{F}_r \cap \{0, 1, \ldots, 2^{64} - 1\}$. The product of a and b is given by

$$ab = \sum_{k=0}^{6} c_k 2^{64k} \qquad \text{where } c_k = \sum_{\substack{i=0 \\ i+j=k}}^{3} \sum_{j=0}^{3} a_i b_j.$$

Each product $a_i b_j$ occupies 128 bits and the sum c_k can occupy a maximum of $128 + 2 = 130$ bits. As $\mathbb{F}_n$ has a capacity of 253 bits, we can represent the product ab using seven $\mathbb{F}_n$ elements $c_0, c_1, \ldots, c_6$. This would be an unreduced representation of the product.

It may be possible to work with unreduced representations and delay reducing them to obtain smaller arithmetic circuits.[1] But we should be careful to avoid overflow in the limbs. For example, the unreduced representation of $\mathbb{F}_q$ elements using $\mathbb{F}_n$ limbs encounters overflow in a degree 4 product, i.e. an expression of the form $x_1x_2x_3x_4$ where each $x_i \in \mathbb{F}_q$. Each term of such a product will require at least 256 bits.

To verify that points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ satisfy the addition law in (2), we could check that

$$
\begin{aligned}
x_3(1 + dx_1x_2y_1y_2) &= x_1y_2 + x_2y_1, \\
y_3(1 - dx_1x_2y_1y_2) &= x_1x_2 + y_1y_2.
\end{aligned}
$$

The above equations involve degree 6 products. These equations use affine coordinates.

Using projective coordinates[2], the point addition formula is given by

$$
\begin{aligned}
C &= X_1X_2, \\
D &= Y_1Y_2, \\
E &= dCD, \\
X_3 &= (1 - E)((X_1 + Y_1)(X_2 + Y_2) - C - D), \\
Y_3 &= (1 + E)(D + C), \\
Z_3 &= 1 - E^2.
\end{aligned}
$$

These also involve a degree 6 product (in Z_3) and several degree 5 products.

Using extended coordinates[3], the point addition formula is given by

$$
\begin{aligned}
A &= (Y_1 - X_1)(Y_2 - X_2), \\
B &= (Y_1 + X_1)(Y_2 + X_2), \\
C &= kT_1T_2, \\
D &= 2Z_1Z_2, \\
E &= B - A, \\
F &= D - C, \\
G &= D + C, \\
H &= B + A, \\
X_3 &= EF, \\
Y_3 &= GH, \\
T_3 &= EH, \\
Z_3 &= FG,
\end{aligned}
$$

where $k = 2d$. These also involve a degree 6 product (in Z_3) and several degree 5 products. The inverted coordinates also involve degree 6 products.[4]

There does not seem any way to reduce the degree of the products in the point addition verification equations to 3 without resorting to intermediate reductions. So we cannot use 64-bit $\mathbb{F}_n$ limbs to represent $\mathbb{F}_q$ elements *if we want to restrict ourselves to a single unreduced to reduced representation conversion.*

The maximum limb bitwidth such that a product of six limbs fits in 253 bits is 42, as $6 \times 42 = 252$. But this will not accommodate carries.

[1] The `circom-ecdsa` project uses this approach. `https://github.com/0xPARC/circom-ecdsa`

[2] `https://hyperelliptic.org/EFD/g1p/auto-twisted-projective.html`

[3] `https://hyperelliptic.org/EFD/g1p/auto-twisted-extended-1.html`

[4] `https://hyperelliptic.org/EFD/g1p/auto-twisted-inverted.html`

In general, the product between $a = \sum_{i=0}^{k_a-1} a_i 2^{\eta i}$ and $b = \sum_{i=0}^{k_b-1} b_i 2^{\eta i}$ will have $k_a + k_b - 1$ limbs each having a maximum bitwidth of $m_a + m_b + \lceil \log_2 (\min (k_a, k_b)) \rceil$ where the a_i's have bitwidth m_a and the b_i's have bitwidth m_b.

We would need 7 limbs each having bitwidth equal to 42 to represent an $\mathbb{F}_q$ element (as $6 \times 42 = 252 < 255 < 294 = 7 \times 42$). Setting $k_a = k_b = 7$ shows that even a degree 2 product will require 3 bits to accomodate carries. A degree 6 product of $\mathbb{F}_q$ elements represented using 42-bit limbs from $\mathbb{F}_n$ will have overflow in its unreduced representation.

Let us now consider the case of 41-bit limbs. We again need 7 limbs to represent an $\mathbb{F}_q$ element as $255 < 287 = 7 \times 41$. A degree 6 product of 7-limbed terms with 41 bits per limb will need 261 bits to products and carries in the unreduced representation. The calculations are illustrated in Table 1.

a	b	m_a	m_b	k_a	k_b	$m_a + m_b + \lceil \log_2 (\min (k_a, k_b)) \rceil$
z_1	z_2	41	41	7	7	85
$z_1 z_2$	z_3	85	41	13	7	129
$z_1 z_2 z_3$	z_4	129	41	19	7	173
$z_1 z_2 z_3 z_4$	z_5	173	41	25	7	217
$z_1 z_2 z_3 z_4 z_5$	z_6	217	41	31	7	261

Table 1: Bitwidth growth for products of terms with 7 limbs of 41 bits each

A degree 6 product of 7-limbed terms with 40 bits per limb will need 255 bits to products and carries in the unreduced representation. The calculations are illustrated in Table 2.

a	b	m_a	m_b	k_a	k_b	$m_a + m_b + \lceil \log_2 (\min (k_a, k_b)) \rceil$
z_1	z_2	40	40	7	7	83
$z_1 z_2$	z_3	83	40	13	7	126
$z_1 z_2 z_3$	z_4	126	40	19	7	169
$z_1 z_2 z_3 z_4$	z_5	169	40	25	7	212
$z_1 z_2 z_3 z_4 z_5$	z_6	212	40	31	7	255

Table 2: Bitwidth growth for products of terms with 7 limbs of 40 bits each

A degree 6 product of 7-limbed terms with 39 bits per limb will need 256 bits to products and carries in the unreduced representation. The calculations are illustrated in Table 3.

a	b	m_a	m_b	k_a	k_b	$m_a + m_b + \lceil \log_2 (\min (k_a, k_b)) \rceil$
z_1	z_2	39	39	7	7	81
$z_1 z_2$	z_3	81	39	13	7	123
$z_1 z_2 z_3$	z_4	123	39	19	7	165
$z_1 z_2 z_3 z_4$	z_5	165	39	25	7	207
$z_1 z_2 z_3 z_4 z_5$	z_6	207	39	31	7	249

Table 3: Bitwidth growth for products of terms with 7 limbs of 39 bits each

While it is possible to use 39-bit limbs to safely calculate the unreduced representation of a degree 6 product, the circuit logic can be quite complex. It seems prudent to try a simpler approach first, even if it is inefficient in terms of arithmetic circuit size.

3 An Approach using Four 64-bit Limbs

Unreduced representations of cubic products using four 64-bit limbs can be safely calculated in fields with capacity 253 bits. The bitwidth growth of this case is illustrated in Table 4.

a	b	m_a	m_b	k_a	k_b	$m_a + m_b + \lceil \log_2 (\min (k_a, k_b)) \rceil$
z_1	z_2	64	64	4	4	130
$z_1 z_2$	z_3	130	64	7	4	196

Table 4: Bitwidth growth for products of terms with 4 limbs of 64 bits each

Recall that the affine point addition verification equations can be written as

$$x_3(1 + dx_1x_2y_1y_2) = x_1y_2 + x_2y_1, \tag{3}$$
$$y_3(1 - dx_1x_2y_1y_2) = x_1x_2 + y_1y_2. \tag{4}$$

We propose to try the following approach. It is inspired by the `circom-ecdsa` approach for verifying secp256k1 point addition.

- Calculate the reduced representation of the cubic dx_1x_2. Let this representation be given by $u = \sum_{i=0}^{3} u_i 2^{64i}$.
- Calculate the reduced representation of the cubic uy_1y_2. Let this representation be given by $v = \sum_{i=0}^{3} v_i 2^{64i}$. Note that v corresponds to the reduced representation of $dx_1x_2y_1y_2$.
- Check that the following quadratic equations hold.

$$x_1y_2 + x_2y_1 - x_3 - x_3v = 0,$$
$$x_1x_2 + y_1y_2 - y_3 + y_3v = 0.$$

3.1 Reducing a Cubic Product

Let a, b, c be three elements in $\mathbb{F}_q$ available in their reduced representations.

$$a = \sum_{i=0}^{3} a_i 2^{64i}, \quad b = \sum_{i=0}^{3} b_i 2^{64i}, \quad c = \sum_{i=0}^{3} c_i 2^{64i}.$$

Let $f = ab$. Then we have $f = \sum_{i=0}^{6} f_i 2^{64i}$ where

$$\begin{aligned}
f_0 &= a_0b_0, \\
f_1 &= a_0b_1 + a_1b_0, \\
f_2 &= a_0b_2 + a_1b_1 + a_2b_0, \\
f_3 &= a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0, \\
f_4 &= a_1b_3 + a_2b_2 + a_3b_1, \\
f_5 &= a_2b_3 + a_3b_2, \\
f_6 &= a_3b_3.
\end{aligned}$$

The limb f_3 occupies a maximum of 130 bits and the other limbs occupy 128 or 129 bits.

Let $g = fc$. Then we have $g = \sum_{i=0}^{9} g_i 2^{64i}$ where

$$\begin{aligned}
g_0 &= f_0c_0,\\
g_1 &= f_0c_1 + f_1c_0,\\
g_2 &= f_0c_2 + f_1c_1 + f_2c_0,\\
g_3 &= f_0c_3 + f_1c_2 + f_2c_1 + f_3c_0,\\
g_4 &= f_1c_3 + f_2c_2 + f_3c_1 + f_4c_0,\\
g_5 &= f_2c_3 + f_3c_2 + f_4c_1 + f_5c_0,\\
g_6 &= f_3c_3 + f_4c_2 + f_5c_1 + f_6c_0,\\
g_7 &= f_4c_3 + f_5c_2 + f_6c_1,\\
g_8 &= f_5c_3 + f_6c_2,\\
g_9 &= f_6c_3,
\end{aligned}$$

In terms of a_i, b_i, c_i, the limbs of g are given by

$$\begin{aligned}
g_0 &= a_0b_0c_0,\\
g_1 &= a_0b_0c_1 + a_0b_1c_0 + a_1b_0c_0,\\
g_2 &= a_0b_0c_2 + a_0b_1c_1 + a_1b_0c_1 + a_0b_2c_0 + a_1b_1c_0 + a_2b_0c_0,\\
g_3 &= a_0b_0c_3 + a_0b_1c_2 + a_1b_0c_2 + a_0b_2c_1 + a_1b_1c_1 + a_2b_0c_1 + a_0b_3c_0 + a_1b_2c_0 + a_2b_1c_0 + a_3b_0c_0,\\
g_4 &= a_0b_1c_3 + a_1b_0c_3 + a_0b_2c_2 + a_1b_1c_2 + a_2b_0c_2 + a_0b_3c_1 + a_1b_2c_1 + a_2b_1c_1 + a_3b_0c_1 + a_1b_3c_0 + a_2b_2c_0 + a_3b_1c_0,\\
g_5 &= a_0b_2c_3 + a_1b_1c_3 + a_2b_0c_3 + a_0b_3c_2 + a_1b_2c_2 + a_2b_1c_2 + a_3b_0c_2 + a_1b_3c_1 + a_2b_2c_1 + a_3b_1c_1 + a_2b_3c_0 + a_3b_2c_0,\\
g_6 &= a_0b_3c_3 + a_1b_2c_3 + a_2b_1c_3 + a_3b_0c_3 + a_1b_3c_2 + a_2b_2c_2 + a_3b_1c_2 + a_2b_3c_1 + a_3b_2c_1 + a_3b_3c_0,\\
g_7 &= a_1b_3c_3 + a_2b_2c_3 + a_3b_1c_3 + a_2b_3c_2 + a_3b_2c_2 + a_3b_3c_1,\\
g_8 &= a_2b_3c_3 + a_3b_2c_3 + a_3b_3c_2,\\
g_9 &= a_3b_3c_3,
\end{aligned}$$

The limbs g_4 and g_5 occupy a maximum of 196 bits, 192 bits for the products and 4 bits for the carry from the addition of 12 products. The other limbs occupy fewer than 196 bits.

Since $q = 2^{255} - 19$, we have $2^{256} = 38 \bmod q$ and $2^{512} = 38^2 = 1444 \bmod q$. We can rewrite g as

$$\begin{aligned}
g = \sum_{i=0}^{9} g_i 2^{64i} &= g_0 + g_1 2^{64} + g_2 2^{128} + g_3 2^{192} + g_4 2^{256} + g_5 2^{320} + g_6 2^{384} + g_7 2^{448} + g_8 2^{512} + g_9 2^{576}\\
&= g_0 + g_1 2^{64} + g_2 2^{128} + g_3 2^{192} + 2^{256}\left(g_4 + g_5 2^{64} + g_6 2^{128} + g_7 2^{192}\right) + 2^{512}\left(g_8 + g_9 2^{64}\right)\\
&= g_0 + g_1 2^{64} + g_2 2^{128} + g_3 2^{192} + 38\left(g_4 + g_5 2^{64} + g_6 2^{128} + g_7 2^{192}\right) + 1444\left(g_8 + g_9 2^{64}\right)\\
&= g_0 + 38g_4 + 1444g_8 + 2^{64}\left(g_1 + 38g_5 + 1444g_9\right) + 2^{128}\left(g_2 + 38g_6\right) + 2^{192}\left(g_3 + 38g_7\right)\\
&= h_0 + h_1 2^{64} + h_2 2^{128} + h_3 2^{192},
\end{aligned}$$

where

$$\begin{aligned}
h_0 &= g_0 + 38g_4 + 1444g_8,\\
h_1 &= g_1 + 38g_5 + 1444g_9,\\
h_2 &= g_2 + 38g_6,\\
h_3 &= g_3 + 38g_7.
\end{aligned}$$

We note the following:

- Each of the g_i's occupy a maximum of 196 bits.

- The numbers 38 and 1444 occupy 6 bits and 11 bits respectively.
- $h_0 = g_0 + 38g_4 + 1444g_8$ can occupy a maximum of 206 bits by the following observations.
  - g_0 occupies a maximum of 192 bits
  - g_4 occupies a maximum of 196 bits
  - $38g_4$ occupies a maximum of 202 bits
  - g_8 occupies a maximum of 194 bits
  - $1444g_8$ occupies a maximum of 205 bits
- $h_1 = g_1 + 38g_5 + 1444g_9$ can occupy a maximum of 204 bits by the following observations.
  - g_1 occupies a maximum of 194 bits
  - g_5 occupies a maximum of 196 bits
  - $38g_5$ occupies a maximum of 202 bits
  - g_9 occupies a maximum of 192 bits
  - $1444g_9$ occupies a maximum of 203 bits
- $h_2 = g_2 + 38g_6$ can occupy a maximum of 203 bits by the following observations.
  - g_2 occupies a maximum of 195 bits
  - g_6 occupies a maximum of 196 bits
  - $38g_6$ occupies a maximum of 202 bits
- $h_3 = g_3 + 38g_7$ can occupy a maximum of 202 bits by the following observations.
  - g_3 occupies a maximum of 196 bits
  - g_7 occupies a maximum of 195 bits
  - $38g_7$ occupies a maximum of 201 bits
- So all the h_i's can fit in $\mathbb{F}_n$ limbs.
- The maximum value of g is bounded by

$$2^{206} - 1 + \left(2^{204} - 1\right) 2^{64} + \left(2^{203} - 1\right) 2^{128} + \left(2^{202} - 1\right) 2^{192} < 2^{206} + 2^{268} + 2^{331} + 2^{394} < 2^{395}$$

- So the reduced representation of g will have at most seven 64-bit limbs. Let $g'_0, g'_1, \ldots, g'_6$ denote these limbs.

$$\begin{aligned} g = h_0 + h_1 2^{64} + h_2 2^{128} + h_3 2^{192} &= \sum_{i=0}^{6} g'_i 2^{64i} \\ &= g'_0 + g'_1 2^{64} + g'_2 2^{128} + g'_3 2^{192} + g'_4 2^{256} + g'_5 2^{320} + g'_6 2^{384} \end{aligned}$$

We want to find an $r \in \mathbb{F}_q$ in reduced representation such that

$$g = tq + r \tag{5}$$

for some quotient $t \in \mathbb{F}_q$. Here

$$r = r_0 + r_1 2^{64} + r_2 2^{128} + r_3 2^{192} \qquad \text{where } r_i \in \{0, 1, \ldots, 2^{64} - 1\}.$$

As $q > 2^{254}$, the maximum value of t required to satisfy this equation is strictly less than $2^{395-254} = 2^{141}$. So the quotient requires only three 64-bit limbs in its reduced representation.

$$t = t_0 + t_1 2^{64} + t_2 2^{128} \qquad \text{where } t_0, t_1 \in \{0, 1, \ldots, 2^{64} - 1\}, t_2 \in \{0, 1, \ldots, 2^{13} - 1\}.$$

The prime q will have four 64-bit limbs.

$$q = q_0 + q_1 2^{64} + q_2 2^{128} + q_3 2^{192} \qquad \text{where } q_i \in \{0, 1, \ldots, 2^{64} - 1\}.$$

The main challenge in checking (5) is the mismatch in the representations of the LHS and the RHS. On the LHS, g has an unreduced representation with four limbs each occupying upto 206 bits.

$$g = h_0 + h_1 2^{64} + h_2 2^{128} + h_3 2^{192}.$$

On the RHS, $tq + r$ has an unreduced representation with six limbs each occupying upto 130 bits.

$$\begin{aligned} tq + r = t_0 q_0 + r_0 &+ (t_0 q_1 + t_1 q_0 + r_1) 2^{64} + (t_0 q_2 + t_1 q_1 + t_2 q_0 + r_2) 2^{128} \\ &+ (t_0 q_3 + t_1 q_2 + t_2 q_1 + r_3) 2^{192} + (t_1 q_3 + t_2 q_2) 2^{256} + t_2 q_3 2^{320}. \end{aligned}$$

One way to check equality in (5) is to convert both g and $tq + r$ to their reduced representations and check that these representations are equal. This would require range checks on the limbs of g and $tq + r$. The number of boolean variables required for each range check will be equal to the bitwidth of the corresponding limb. Each boolean variable requires one constraint of the form $b(b - 1) = 0$ in the R1CS system.

We could reduce the number of range checks by checking that $g - tq - r$ equals zero. This is the approach used in `circom-ecdsa`. Consider the following argument assuming that $g - tq - r = 0$.

1. $h_0 - t_0 q_0 - r_0$ contains the 64 least significant bits of $g - tq - r$, i.e. bits 0 to 63. These bits must all be zero. So $h_0 - t_0 q_0 - r_0$ must be a multiple of 2^{64}.
2. Let $y_0 = \frac{h_0 - t_0 q_0 - r_0}{2^{64}}$. This represents the *carry* into the 2^{64} limb.
3. $y_0 + h_1 - t_0 q_1 - t_1 q_0 - r_1$ contains the *next* 64 least significant bits of $g - tq - r$, i.e. bits 64 to 127. These bits must also all be zero. So $y_0 + h_1 - t_0 q_1 - t_1 q_0 - r_1$ must be a multiple of 2^{64}
4. Let $y_1 = \frac{y_0 + h_1 - t_0 q_1 - t_1 q_0 - r_1}{2^{64}}$. This represents the carry into the 2^{128} limb.
5. $y_1 + h_2 - t_0 q_2 - t_1 q_1 - t_2 q_0 - r_2$ contains the next 64 least significant bits of $g - tq - r$, i.e. bits 128 to 191. These bits must also all be zero. So $y_1 + h_2 - t_0 q_2 - t_1 q_1 - t_2 q_0 - r_2$ must be a multiple of 2^{64}
6. Let $y_2 = \frac{y_1 + h_2 - t_0 q_2 - t_1 q_1 - t_2 q_0 - r_2}{2^{64}}$. This represents the carry into the 2^{192} limb.
7. $y_2 + h_3 - t_0 q_3 - t_1 q_2 - t_2 q_1 - r_3$ contains the next 64 least significant bits of $g - tq - r$, i.e. bits 192 to 255. These bits must also all be zero. So $y_2 + h_3 - t_0 q_3 - t_1 q_2 - t_2 q_1 - r_3$ must be a multiple of 2^{64}
8. Let $y_3 = \frac{y_2 + h_3 - t_0 q_3 - t_1 q_2 - t_2 q_1 - r_3}{2^{64}}$. This represents the carry into the 2^{256} limb.
9. $y_3 - t_1 q_3 - t_2 q_2$ contains the next 64 least significant bits of $g - tq - r$, i.e. bits 256 to 319. These bits must also all be zero. So $y_3 - t_1 q_3 - t_2 q_2$ must be a multiple of 2^{64}
10. Let $y_4 = \frac{y_3 - t_1 q_3 - t_2 q_2}{2^{64}}$. This represents the carry into the 2^{320} limb.
11. $y_4 - t_2 q_3$ contains the remaining 75 least significant bits of $g - tq - r$, i.e. bits 320 to 394. Recall that g is bounded by 2^{395}. These bits must also all be zero. So $y_4 - t_2 q_3$ *must be zero.*

Note that the limbs of $g-tq-r$ can have negative values, i.e. they can experience underflows during the subtraction operation. We use the convention that $x \in \mathbb{F}_n$ is *negative* if $x > \frac{n-1}{2}$. For example, $h_0 - t_0q_0 - r_0$ can be a negative multiple of 2^{64}.

As h_0, h_1, h_2, h_3 have maximum bitwidths of 206, 204, 203, and 202, $t_0, t_1, q_0, q_1, q_2, q_3, r_0, r_1, r_2, r_3$ have maximum bitwidths of 64, and t_2 has a maximum bitwidth of 13, the following terms (the unreduced limbs of $g-tq-r$) lie in the range indicated next to them.

$$\begin{aligned}
&h_0 - t_0q_0 - r_0 && \in \{-2^{129}+1, 2^{206}-1\},\\
&h_1 - t_0q_1 - t_1q_0 - r_1 && \in \{-2^{130}+1, 2^{204}-1\},\\
&h_2 - t_0q_2 - t_1q_1 - t_2q_0 - r_2 && \in \{-2^{130}+1, 2^{203}-1\},\\
&h_3 - t_0q_3 - t_1q_2 - t_2q_1 - r_3 && \in \{-2^{130}+1, 2^{202}-1\},\\
&-t_1q_3 - t_2q_2 && \in \{-2^{129}+1, 0\},\\
&-t_2q_3 && \in \{-2^{77}+1, 0\}.
\end{aligned}$$

The procedure for checking $g-tq-r=0$ involves the addition of multiple terms some of which can be negative. Furthermore, the addition will be performed in $\mathbb{F}_n$. A set of terms sum to zero in $\mathbb{F}_n$ may not sum to zero in $\mathbb{F}_q$. To ensure that they do sum to zero in $\mathbb{F}_q$, we should ensure that the bitwidths of the partial sums does not exceed the capacity of $\mathbb{F}_n$.

The bitwidths of the terms h_i, q_i, t_i, r_i will be known due to range checks. The carries $y_0, y_1, \ldots, y_4$ will be provided as non-deterministic advice to the arithmetic circuit. Instead of calculating y_0 as $\frac{h_0-t_0q_0-r_0}{2^{64}}$, we will check that $2^{64}y_0 = h_0 - t_0q_0 - r_0$ in the field $\mathbb{F}_n$. We need to apply range checks on the y_i's to ensure that adding them will not exceed the capacity of $\mathbb{F}_n$.

- As $h_0 - t_0q_0 - r_0$ is in the range $\{-2^{129}+1, \ldots, 2^{206}-1\}$, y_0 can be checked to be in the range $\{-2^{65}+1, \ldots, 2^{142}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_0 + 2^{65}$ is in the range $\{0, 1, \ldots, 2^{143}-1\}$.

- $y_0 + h_1 - t_0q_1 - t_1q_0 - r_1$ is in the range $\{-2^{131}+1, \ldots, 2^{205}-1\}$. Since $y_1 = \frac{y_0+h_1-t_0q_1-t_1q_0-r_1}{2^{64}}$, we can check that y_1 is in the range $\{-2^{67}+1, \ldots, 2^{141}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_1 + 2^{67}$ is in the range $\{0, 1, \ldots, 2^{142}-1\}$.

- $y_1+h_2-t_0q_2-t_1q_1-t_2q_0-r_2$ is in the range $\{-2^{131}+1, \ldots, 2^{204}-1\}$. Since $y_2 = \frac{y_1+h_2-t_0q_2-t_1q_1-t_2q_0-r_2}{2^{64}}$, we can check that y_2 is in the range $\{-2^{67}+1, \ldots, 2^{140}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_2 + 2^{67}$ is in the range $\{0, 1, \ldots, 2^{141}-1\}$.

- $y_2+h_3-t_0q_3-t_1q_2-t_2q_1-r_3$ is in the range $\{-2^{131}+1, \ldots, 2^{203}-1\}$. Since $y_3 = \frac{y_2+h_3-t_0q_3-t_1q_2-t_2q_1-r_3}{2^{64}}$, we can check that y_3 is in the range $\{-2^{67}+1, \ldots, 2^{139}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_3 + 2^{67}$ is in the range $\{0, 1, \ldots, 2^{140}-1\}$.

- $y_3 - t_1q_3 - t_2q_2$ is in the range $\{-2^{130}+1, \ldots, 2^{139}-1\}$. Since $y_4 = \frac{y_3-t_1q_3-t_2q_2}{2^{64}}$, we can check that y_4 is in the range $\{-2^{66}+1, \ldots, 2^{75}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_4 + 2^{66}$ is in the range $\{0, 1, \ldots, 2^{76}-1\}$.

A range check for an n-bit range costs n R1CS constraints. In the above list, there are 5 range checks costing a total of $143+142+141+140+76 = 642$ constraints. The above check has to be done two times, once for dx_1x_2 and once for uy_1y_2. So the range checks for carries in cubic reductions alone would cost 1284 constraints.

3.2 Reducing a Quadratic Expression

In Section 3.1, the cubic products being reduced had positive values due to the approach chosen in Section 3. Recall our convention that $x \in \mathbb{F}_n$ is positive if $x \le \frac{n-1}{2}$. Even after the folding shown in Section 3.1, the values remain positive. Consequently, the quotient t could be bounded in magnitude by 2^{140} and represented using three 64-bit limbs.

On the other hand, the quadratic expressions we plan to verify can have negative values as they have the form

$$x_1 y_2 + x_2 y_1 - x_3 - x_3 v = 0,$$
$$x_1 x_2 + y_1 y_2 - y_3 + y_3 v = 0.$$

Note that we need to check that the LHS is equal to zero modulo q in the above two equations. This can be done by showing that the expressions are a multiple of q. Negative values for the LHS will require negative quotients which will require more limbs to represent. For example, a quotient $-1 \in \mathbb{F}_q$ will require four limbs.

Let a, b be two elements in $\mathbb{F}_q$ available in their reduced representations.

$$a = \sum_{i=0}^{3} a_i 2^{64i}, \qquad b = \sum_{i=0}^{3} b_i 2^{64i}.$$

Let $f = ab$. Then we have $f = \sum_{i=0}^{6} f_i 2^{64i}$ where

$$\begin{aligned}
f_0 &= a_0 b_0, \\
f_1 &= a_0 b_1 + a_1 b_0, \\
f_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0, \\
f_3 &= a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0, \\
f_4 &= a_1 b_3 + a_2 b_2 + a_3 b_1, \\
f_5 &= a_2 b_3 + a_3 b_2, \\
f_6 &= a_3 b_3.
\end{aligned}$$

Since $q = 2^{255} - 19$, we have $2^{256} = 38 \bmod q$. We can rewrite f as

$$\begin{aligned}
f = \sum_{i=0}^{6} f_i 2^{64i} &= f_0 + f_1 2^{64} + f_2 2^{128} + f_3 2^{192} + f_4 2^{256} + f_5 2^{320} + f_6 2^{384} \\
&= f_0 + f_1 2^{64} + f_2 2^{128} + f_3 2^{192} + 2^{256} \left(f_4 + f_5 2^{64} + f_6 2^{128} \right) \\
&= f_0 + f_1 2^{64} + f_2 2^{128} + f_3 2^{192} + 38 \left(f_4 + f_5 2^{64} + f_6 2^{128} \right) \\
&= f_0 + 38 f_4 + 2^{64} (f_1 + 38 f_5) + 2^{128} (f_2 + 38 f_6) + f_3 2^{192} \\
&= h_0 + h_1 2^{64} + h_2 2^{128} + h_3 2^{192},
\end{aligned}$$

where

$$\begin{aligned}
h_0 &= f_0 + 38 f_4, \\
h_1 &= f_1 + 38 f_5, \\
h_2 &= f_2 + 38 f_6, \\
h_3 &= f_3.
\end{aligned}$$

We note the following:

- Each of the f_i's occupy a maximum of 130 bits.
- The number 38 occupies 6 bits.
- $h_0 = f_0 + 38 f_4$ can occupy a maximum of 137 bits by the following observations.
  - f_0 occupies a maximum of 128 bits
  - f_4 occupies a maximum of 130 bits

- $38f_4$ occupies a maximum of 136 bits

- $h_1 = f_1 + 38f_5$ can occupy a maximum of 136 bits by the following observations.
  - f_1 occupies a maximum of 129 bits
  - f_5 occupies a maximum of 129 bits
  - $38f_5$ occupies a maximum of 135 bits
- $h_2 = f_2 + 38f_6$ can occupy a maximum of 135 bits by the following observations.
  - f_2 occupies a maximum of 130 bits
  - f_6 occupies a maximum of 128 bits
  - $38f_6$ occupies a maximum of 134 bits
- $h_3 = f_3$ occupies a maximum of 130 bits.
- So all the h_i's can fit in $\mathbb{F}_n$ limbs.
- The maximum value of f is bounded by

$$2^{137} - 1 + \left(2^{136} - 1\right) 2^{64} + \left(2^{135} - 1\right) 2^{128} + \left(2^{130} - 1\right) 2^{192} < 2^{137} + 2^{200} + 2^{263} + 2^{322} < 2^{323}$$

We can then conclude that the expression $x_1y_2 + x_2y_1 - x_3 - x_3v$ will (conservatively) be in the range $\{-2^{325} + 1, \ldots, 2^{325} - 1\}$. The same conclusion holds for the expression $x_1x_2 + y_1y_2 - y_3 + y_3v$.

As $q > 2^{254}$, adding $2^{71}q$ to these two expressions will make the corresponding sums positive. Note that the individual limbs of $x_1y_2 + x_2y_1 - x_3 - x_3v + 2^{71}q$ and $x_1x_2 + y_1y_2 - y_3 + y_3v + 2^{71}q$ can be negative but the integers themselves will be positive. Moreover each limb will have magnitude less than 2^{139}. This is because each product in $\{x_1y_2, x_2y_1, x_3v, x_1x_2, y_1y_2, x_3v\}$ has limbs with magnitude less than 2^{137}. The four limbs of $2^{71}q$ occupy a maximum of 135 bits.

Let g be equal to $x_1y_2 + x_2y_1 - x_3 - x_3v + 2^{71}q$ or $x_1x_2 + y_1y_2 - y_3 + y_3v + 2^{71}q$. Let the unreduced representation of g be

$$g = g_0 + g_1 2^{64} + g_2 2^{128} + g_3 2^{192},$$

where each $g_i \in \{0, \ldots, 2^{138} - 1\}$. The value of g is less than 2^{326}.

To show that $g = 0 \bmod q$, we show the existence of a quotient t such that $g - tq = 0$. As $q > 2^{254}$ and $g < 2^{326}$, the maximum value of t required to satisfy this equation is strictly less than $2^{326-254} = 2^{72}$. So the quotient requires only two 64-bit limbs in its reduced representation.

$$t = t_0 + t_1 2^{64} \qquad \text{where } t_0 \in \{0, 1, \ldots, 2^{64} - 1\}, t_1 \in \{0, 1, \ldots, 2^8 - 1\}.$$

The prime q will have four 64-bit limbs.

$$q = q_0 + q_1 2^{64} + q_2 2^{128} + q_3 2^{192} \qquad \text{where } q_i \in \{0, 1, \ldots, 2^{64} - 1\}.$$

The product tq has an unreduced representation with five limbs each occupying upto 129 bits.

$$tq = t_0q_0 + (t_0q_1 + t_1q_0)2^{64} + (t_0q_2 + t_1q_1)2^{128} + (t_0q_3 + t_1q_2)2^{192} + t_1q_3 2^{256}.$$

Consider the following argument to check that $g - tq = 0$.

1. $g_0 - t_0q_0$ contains the 64 least significant bits of $g - tq$, i.e. bits 0 to 63. These bits must all be zero. So $g_0 - t_0q_0$ must be a multiple of 2^{64}.
2. Let $y_0 = \frac{g_0 - t_0q_0}{2^{64}}$. This represents the *carry* into the 2^{64} limb.
3. $y_0 + g_1 - t_0q_1 - t_1q_0$ contains the *next* 64 least significant bits of $g - tq$, i.e. bits 64 to 127. These bits must also all be zero. So $y_0 + g_1 - t_0q_1 - t_1q_0$ must be a multiple of 2^{64}.

4. Let $y_1 = \frac{y_0+g_1-t_0q_1-t_1q_0}{2^{64}}$. This represents the carry into the 2^{128} limb.

5. $y_1 + g_2 - t_0q_2 - t_1q_1$ contains the next 64 least significant bits of $g - tq$, i.e. bits 128 to 191. These bits must also all be zero. So $y_1 + g_2 - t_0q_2 - t_1q_1$ must be a multiple of 2^{64}.

6. Let $y_2 = \frac{y_1+g_2-t_0q_2-t_1q_1}{2^{64}}$. This represents the carry into the 2^{192} limb.

7. $y_2 + g_3 - t_0q_3 - t_1q_2$ contains the next 64 least significant bits of $g - tq$, i.e. bits 192 to 255. These bits must also all be zero. So $y_2 + g_3 - t_0q_3 - t_1q_2$ must be a multiple of 2^{64}.

8. Let $y_3 = \frac{y_2+g_3-t_0q_3-t_1q_2}{2^{64}}$. This represents the carry into the 2^{256} limb.

9. $y_3 - t_1q_3$ contains the remaining 70 least significant bits of $g - tq$, i.e. bits 256 to 325. Recall that g is bounded by 2^{326}. These bits must also all be zero. So $y_3 - t_1q_3$ *must be zero.*

As the g_i's have a maximum bitwidth of 139, $t_0, q_0, q_1, q_2, q_3, r_0, r_1, r_2, r_3$ have maximum bitwidths of 64, and t_1 has a maximum bitwidth of 8, the following terms (the unreduced limbs of $g - tq - r$) lie in the range indicated next to them. As the g_i's have a maximum bitwidth of 139, the following terms (the unreduced limbs of $g - tq$) lie in the range $\{-2^{139}+1, \dots, 2^{139}-1\}$.

$$
\begin{aligned}
&g_0 - t_0q_0 && \in \{-2^{128}+1, 2^{139}-1\},\\
&g_1 - t_0q_1 - t_1q_0 && \in \{-2^{129}+1, 2^{139}-1\},\\
&g_2 - t_0q_2 - t_1q_1 && \in \{-2^{129}+1, 2^{139}-1\},\\
&g_3 - t_0q_3 - t_1q_2 && \in \{-2^{129}+1, 2^{139}-1\},\\
&-t_1q_3 && \in \{-2^{72}+1, 0\}.
\end{aligned}
$$

The carries y_0, y_1, y_2, y_3 will be provided as non-deterministic advice to the arithmetic circuit. Instead of calculating y_0 as $\frac{g_0-t_0q_0}{2^{64}}$, we will check that $2^{64}y_0 = g_0 - t_0q_0$ in the field $\mathbb{F}_n$. We need to apply range checks on the y_i's to ensure that adding them will not exceed the capacity of $\mathbb{F}_n$.

- As $g_0 - t_0q_0$ is in the range $\{-2^{128}+1, \dots, 2^{139}-1\}$, y_0 can be checked to be in the range $\{-2^{64}+1, \dots, 2^{75}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_0 + 2^{64}$ is in the range $\{0, 1, \dots, 2^{76}-1\}$.

- $y_0 + g_1 - t_0q_1 - t_1q_0$ is in the range $\{-2^{130}+1, \dots, 2^{140}-1\}$. Since $y_1 = \frac{y_0+g_1-t_0q_1-t_1q_0}{2^{64}}$, we can check that y_1 is in the range $\{-2^{66}+1, \dots, 2^{76}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_1 + 2^{66}$ is in the range $\{0, 1, \dots, 2^{77}-1\}$.

- $y_1 + g_2 - t_0q_2 - t_1q_1$ is in the range $\{-2^{130}+1, \dots, 2^{140}-1\}$. Since $y_2 = \frac{y_1+g_2-t_0q_2-t_1q_1}{2^{64}}$, we can check that y_2 is in the range $\{-2^{66}+1, \dots, 2^{76}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_2 + 2^{66}$ is in the range $\{0, 1, \dots, 2^{77}-1\}$.

- $y_2 + g_3 - t_0q_3 - t_1q_2$ is in the range $\{-2^{130}+1, \dots, 2^{140}-1\}$. Since $y_3 = \frac{y_2+g_3-t_0q_3-t_1q_2}{2^{64}}$, we can check that y_3 is in the range $\{-2^{66}+1, \dots, 2^{76}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_3 + 2^{66}$ is in the range $\{0, 1, \dots, 2^{77}-1\}$.

A range check for an n-bit range costs n R1CS constraints. In the above list, there are 4 range checks costing a total of $76 + 3 \times 77 = 307$ constraints. The above check has to be done two times, once each for the two quadratic expressions. So the range checks for carries in quadratic expression reductions alone would cost 614 constraints.

The range checks for carries in cubic reductions cost 1284 constraints. So the total range checks would cost 1898 constraints.

a	b	m_a	m_b	k_a	k_b	$m_a + m_b + \lceil\log_2(\min(k_a, k_b))\rceil$
z_1	z_2	32	32	8	8	67
z_1z_2	z_3	67	32	15	8	102
$z_1z_2z_3$	z_4	102	32	22	8	137
$z_1z_2z_3z_4$	z_5	137	32	29	8	172
$z_1z_2z_3z_4z_5$	z_6	172	32	36	8	207

Table 5: Bitwidth growth for products of terms with 8 limbs of 32 bits each

4 An Approach using Eight 32-bit Limbs

Unreduced representations of sextic products using eight 32-bit limbs can be safely calculated in fields with capacity 253 bits. The bitwidth growth of this case is illustrated in Table 5. But we also have to account for the bitwidth growth due to the folding operations. These will be considered in the subsequent sections.

Recall that the affine point addition verification equations can be written as

$$x_3(1 + dx_1x_2y_1y_2) = x_1y_2 + x_2y_1, \tag{6}$$
$$y_3(1 - dx_1x_2y_1y_2) = x_1x_2 + y_1y_2. \tag{7}$$

We propose to try the following approach. Given the affine points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, check that the following sextic equations hold.

$$dx_1x_2x_3y_1y_2 + x_3 - x_1y_2 - x_2y_1 = 0 \bmod q, \tag{8}$$
$$dx_1x_2y_1y_2y_3 - y_3 + x_1x_2 + y_1y_2 = 0 \bmod q. \tag{9}$$

In (9), y_3 is the only term with a negative sign. So it is the only term that can reduce the sum. Our goal is to have a non-negative value in the unreduced representation of the LHS in (9) after folding. We can rewrite this equation as

$$dx_1x_2y_1y_2y_3 + (q - y_3) + x_1x_2 + y_1y_2 = 0 \bmod q, \tag{10}$$

and ensure that the terms in the LHS sum to a non-negative value.

In (8), the term $-x_1y_2 - x_2y_1$ can contribute negative values. We can rewrite this equation as

$$dx_1x_2x_3y_1y_2 + x_3 + (q^2 - x_1y_2) + (q^2 - x_2y_1) = 0 \bmod q, \tag{11}$$

and ensure that the terms in the LHS sum to a non-negative value. But it turns out that we do not need to add $2q^2$ to the LHS. We can add a much smaller number. As shown in the next section, once we fold products like x_1y_2 their magnitude is bounded by 2^{292}. As $q > 2^{254}$, $2^{38}q - x_1y_2$ is guaranteed to be non-negative. So we can rewrite (8) as

$$dx_1x_2x_3y_1y_2 + x_3 + 2^{38}q - x_1y_2 - x_2y_1 = 0 \bmod q, \tag{12}$$

and ensure that the terms in the LHS sum to a non-negative value.

4.1 Folding a Quadratic Product

Let a, b be two elements in $\mathbb{F}_q$ available in their reduced representations.

$$a = \sum_{i=0}^{7} a_i 2^{32i}, \qquad b = \sum_{i=0}^{7} b_i 2^{32i},$$

where $a_i, b_i \in \{0, 1, \ldots, 2^{32}-1\}$. Let $f = ab$. Then we have $f = \sum_{i=0}^{14} f_i 2^{32i}$ where

$$\begin{aligned}
f_0 &= a_0b_0,\\
f_1 &= a_0b_1 + a_1b_0,\\
f_2 &= a_0b_2 + a_1b_1 + a_2b_0,\\
f_3 &= a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0,\\
f_4 &= a_0b_4 + a_1b_3 + a_2b_2 + a_3b_1 + a_4b_0,\\
f_5 &= a_0b_5 + a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1 + a_5b_0,\\
f_6 &= a_0b_6 + a_1b_5 + a_2b_4 + a_3b_3 + a_4b_2 + a_5b_1 + a_6b_0,\\
f_7 &= a_0b_7 + a_1b_6 + a_2b_5 + a_3b_4 + a_4b_3 + a_5b_2 + a_6b_1 + a_7b_0,\\
f_8 &= a_1b_7 + a_2b_6 + a_3b_5 + a_4b_4 + a_5b_3 + a_6b_2 + a_7b_1,\\
f_9 &= a_2b_7 + a_3b_6 + a_4b_5 + a_5b_4 + a_6b_3 + a_7b_2,\\
f_{10} &= a_3b_7 + a_4b_6 + a_5b_5 + a_6b_4 + a_7b_3,\\
f_{11} &= a_4b_7 + a_5b_6 + a_6b_5 + a_7b_4,\\
f_{12} &= a_5b_7 + a_6b_6 + a_7b_5,\\
f_{13} &= a_6b_7 + a_7b_6,\\
f_{14} &= a_7b_7.
\end{aligned}$$

Since $q = 2^{255} - 19$, we have $2^{256} = 38 \bmod q$. We can rewrite f (modulo q) as

$$\begin{aligned}
f = \sum_{i=0}^{14} f_i 2^{32i} &= f_0 + f_1 2^{32} + f_2 2^{64} + f_3 2^{96} + f_4 2^{128} + f_5 2^{160} + f_6 2^{192} + f_7 2^{224}\\
&\quad + f_8 2^{256} + f_9 2^{288} + f_{10} 2^{320} + f_{11} 2^{352} + f_{12} 2^{384} + f_{13} 2^{416} + f_{14} 2^{448}\\
&= f_0 + f_1 2^{32} + f_2 2^{64} + f_3 2^{96} + f_4 2^{128} + f_5 2^{160} + f_6 2^{192} + f_7 2^{224}\\
&\quad + 2^{256} \left(f_8 + f_9 2^{32} + f_{10} 2^{64} + f_{11} 2^{96} + f_{12} 2^{128} + f_{13} 2^{160} + f_{14} 2^{192} \right)\\
&= f_0 + 38 f_8 + 2^{32} (f_1 + 38 f_9) + 2^{64} (f_2 + 38 f_{10}) + 2^{96} (f_3 + 38 f_{11})\\
&\quad + 2^{128} (f_4 + 38 f_{12}) + 2^{160} (f_5 + 38 f_{13}) + 2^{192} (f_6 + 38 f_{14}) + f_7 2^{224}\\
&= h_0 + h_1 2^{32} + h_2 2^{64} + h_3 2^{96} + h_4 2^{128} + h_5 2^{160} + h_6 2^{192} + h_7 2^{224},
\end{aligned}$$

where

$$\begin{aligned}
h_0 &= f_0 + 38 f_8,\\
h_1 &= f_1 + 38 f_9,\\
h_2 &= f_2 + 38 f_{10},\\
h_3 &= f_3 + 38 f_{11},\\
h_4 &= f_4 + 38 f_{12},\\
h_5 &= f_5 + 38 f_{13},\\
h_6 &= f_6 + 38 f_{14},\\
h_7 &= f_7.
\end{aligned}$$

We note the following:

- Each of the f_i's occupy a maximum of 67 bits.
- The number 38 occupies 6 bits.
- $h_0 = f_0 + 38 f_8$ can occupy a maximum of 73 bits by the following observations.
  - f_0 occupies a maximum of 64 bits

- f_8 occupies a maximum of 66 bits, being a sum of seven 64-bit products
- $38f_4$ occupies a maximum of 72 bits

- $h_1 = f_1 + 38f_9$ can occupy a maximum of 73 bits by the following observations.
  - f_1 occupies a maximum of 65 bits
  - f_9 occupies a maximum of 66 bits, being a sum of six 64-bit products
  - $38f_9$ occupies a maximum of 72 bits
- $h_2 = f_2 + 38f_{10}$ can occupy a maximum of 73 bits by the following observations.
  - f_2 occupies a maximum of 65 bits
  - f_{10} occupies a maximum of 66 bits, being a sum of five 64-bit products
  - $38f_{10}$ occupies a maximum of 72 bits
- $h_3 = f_3 + 38f_{11}$ can occupy a maximum of 73 bits by the following observations.
  - f_3 occupies a maximum of 66 bits
  - f_{11} occupies a maximum of 66 bits, being a sum of four 64-bit products
  - $38f_{11}$ occupies a maximum of 72 bits
- $h_4 = f_4 + 38f_{12}$ can occupy a maximum of 72 bits by the following observations.
  - f_4 occupies a maximum of 66 bits
  - f_{12} occupies a maximum of 65 bits, being a sum of three 64-bit products
  - $38f_{12}$ occupies a maximum of 71 bits
- $h_5 = f_5 + 38f_{13}$ can occupy a maximum of 72 bits by the following observations.
  - f_5 occupies a maximum of 66 bits
  - f_{13} occupies a maximum of 65 bits, being a sum of two 64-bit products
  - $38f_{13}$ occupies a maximum of 71 bits
- $h_6 = f_6 + 38f_{14}$ can occupy a maximum of 71 bits by the following observations.
  - f_6 occupies a maximum of 66 bits
  - f_{14} occupies a maximum of 64 bits
  - $38f_{14}$ occupies a maximum of 70 bits
- $h_7 = f_7$ can occupy a maximum of 67 bits, being the sum of eight 64-bit products
- So all the h_i's can fit in $\mathbb{F}_n$ limbs.
- The maximum value of f is bounded by

$$
\begin{aligned}
&h_0 + h_1 2^{32} + h_2 2^{64} + h_3 2^{96} + h_4 2^{128} + h_5 2^{160} + h_6 2^{192} + h_7 2^{224} \\
&\leq 2^{73} - 1 + \left(2^{73} - 1\right) 2^{32} + \left(2^{73} - 1\right) 2^{64} + \left(2^{73} - 1\right) 2^{96} + \left(2^{72} - 1\right) 2^{128} + \left(2^{72} - 1\right) 2^{160} \\
&\quad + \left(2^{71} - 1\right) 2^{192} + \left(2^{67} - 1\right) 2^{224} \\
&< 2^{73} + 2^{105} + 2^{137} + 2^{169} + 2^{200} + 2^{232} + 2^{263} + 2^{291} \\
&< 2^{292}
\end{aligned}
$$

4.2 Folding a Sextic Product

Let a, b, c, d, e, f be six elements in $\mathbb{F}_q$ available in their reduced representations.

$$a = \sum_{i=0}^{7} a_i 2^{32i}, \quad b = \sum_{i=0}^{7} b_i 2^{32i}, \quad c = \sum_{i=0}^{7} c_i 2^{32i}, \quad d = \sum_{i=0}^{7} d_i 2^{32i}, \quad e = \sum_{i=0}^{7} e_i 2^{32i}, \quad f = \sum_{i=0}^{7} f_i 2^{32i},$$

where $a_i, b_i, c_i, d_i, e_i, f_i \in \{0, 1, \ldots, 2^{32} - 1\}$. Let $g = abcdef$. Then we have $g = \sum_{l=0}^{42} g_l 2^{32l}$ where

$$g_l = \sum_{\substack{i=0 \\ i+j+k+u+v+w=l}}^{7} \sum_{j=0}^{7} \sum_{k=0}^{7} \sum_{u=0}^{7} \sum_{v=0}^{7} \sum_{w=0}^{7} a_i b_j c_k d_u e_v f_w.$$

Since $q = 2^{255} - 19$, we have

$$\begin{aligned}
2^{256} &= 38 \bmod q, \\
2^{512} &= 38^2 \bmod q, \\
2^{768} &= 38^3 \bmod q, \\
2^{1024} &= 38^4 \bmod q, \\
2^{1280} &= 38^5 \bmod q.
\end{aligned}$$

We do not need to consider higher powers of 2^{256}, since the highest power of 2 that appears in g is $32 \times 42 = 1344$.

We can rewrite g (modulo q) as

$$\begin{aligned}
g = \sum_{l=0}^{42} g_l 2^{32l} &= \sum_{l=0}^{7} g_l 2^{32l} + 2^{256} \sum_{l=8}^{15} g_l 2^{32(l-8)} + 2^{512} \sum_{l=16}^{23} g_l 2^{32(l-16)} + 2^{768} \sum_{l=24}^{31} g_l 2^{32(l-24)} \\
&\quad + 2^{1024} \sum_{l=32}^{39} g_l 2^{32(l-32)} + 2^{1280} \sum_{l=40}^{42} g_l 2^{32(l-40)} \\
&= \sum_{l=0}^{7} g_l 2^{32l} + 38 \sum_{l=8}^{15} g_l 2^{32(l-8)} + 38^2 \sum_{l=16}^{23} g_l 2^{32(l-16)} + 38^3 \sum_{l=24}^{31} g_l 2^{32(l-24)} \\
&\quad + 38^4 \sum_{l=32}^{39} g_l 2^{32(l-32)} + 38^5 \sum_{l=40}^{42} g_l 2^{32(l-40)} \\
&= g_0 + 38 g_8 + 38^2 g_{16} + 38^3 g_{24} + 38^4 g_{32} + 38^5 g_{40} \\
&\quad + 2^{32} \left(g_1 + 38 g_9 + 38^2 g_{17} + 38^3 g_{25} + 38^4 g_{33} + 38^5 g_{41}\right) \\
&\quad + 2^{64} \left(g_2 + 38 g_{10} + 38^2 g_{18} + 38^3 g_{26} + 38^4 g_{34} + 38^5 g_{42}\right) \\
&\quad + 2^{96} \left(g_3 + 38 g_{11} + 38^2 g_{19} + 38^3 g_{27} + 38^4 g_{35}\right) \\
&\quad + 2^{128} \left(g_4 + 38 g_{12} + 38^2 g_{20} + 38^3 g_{28} + 38^4 g_{36}\right) \\
&\quad + 2^{160} \left(g_5 + 38 g_{13} + 38^2 g_{21} + 38^3 g_{29} + 38^4 g_{37}\right) \\
&\quad + 2^{192} \left(g_6 + 38 g_{14} + 38^2 g_{22} + 38^3 g_{30} + 38^4 g_{38}\right) \\
&\quad + 2^{224} \left(g_7 + 38 g_{15} + 38^2 g_{23} + 38^3 g_{31} + 38^4 g_{39}\right)
\end{aligned}$$

Let $h_0, h_1, \ldots, h_7$ denote the limbs of the folded g, i.e.

$$\begin{aligned}
h_0 &= g_0 + 38g_8 + 38^2 g_{16} + 38^3 g_{24} + 38^4 g_{32} + 38^5 g_{40}, \\
h_1 &= g_1 + 38g_9 + 38^2 g_{17} + 38^3 g_{25} + 38^4 g_{33} + 38^5 g_{41}, \\
h_2 &= g_2 + 38g_{10} + 38^2 g_{18} + 38^3 g_{26} + 38^4 g_{34} + 38^5 g_{42}, \\
h_3 &= g_3 + 38g_{11} + 38^2 g_{19} + 38^3 g_{27} + 38^4 g_{35}, \\
h_4 &= g_4 + 38g_{12} + 38^2 g_{20} + 38^3 g_{28} + 38^4 g_{36}, \\
h_5 &= g_5 + 38g_{13} + 38^2 g_{21} + 38^3 g_{29} + 38^4 g_{37}, \\
h_6 &= g_6 + 38g_{14} + 38^2 g_{22} + 38^3 g_{30} + 38^4 g_{38}, \\
h_7 &= g_7 + 38g_{15} + 38^2 g_{23} + 38^3 g_{31} + 38^4 g_{39}.
\end{aligned}$$

As illustrated in Table 5, each g_l can occupy a maximum of 215 bits. Some g_l's will fit in fewer bits. For example, $g_0 = a_0 b_0 c_0 d_0 e_0 f_0$ will occupy a maximum of 192 bits. For ease of analysis, we will consider the worst-case bitwidth of 215 for all g_l's. A more accurate analysis can lead to fewer constraints in the circuit. This is left for future work. The bitwidths of the powers of 38 are as shown in Table 6.

Number	Bitwidth
38	6
38^2	11
38^3	16
38^4	21
38^5	27

Table 6: Bitwidths of the powers of 38

Based on these bitwidths, we note the following:

- $h_0 = g_0 + 38g_8 + 38^2 g_{16} + 38^3 g_{24} + 38^4 g_{32} + 38^5 g_{40}$ can occupy a maximum of 243 bits by the following observations.
  - g_0 occupies a maximum of 215 bits
  - $38g_8$ occupies a maximum of 221 bits
  - $38^2 g_{16}$ occupies a maximum of 226 bits
  - $38^3 g_{24}$ occupies a maximum of 231 bits
  - $38^4 g_{32}$ occupies a maximum of 236 bits
  - $38^5 g_{40}$ occupies a maximum of 242 bits
- Similarly, h_1 and h_2 can occupy a maximum of 243 bits.
- $h_3 = g_3 + 38g_{11} + 38^2 g_{19} + 38^3 g_{27} + 38^4 g_{35}$ can occupy a maximum of 237 bits by the following observations.
  - g_3 occupies a maximum of 215 bits
  - $38g_{11}$ occupies a maximum of 221 bits
  - $38^2 g_{19}$ occupies a maximum of 226 bits
  - $38^3 g_{27}$ occupies a maximum of 231 bits
  - $38^4 g_{35}$ occupies a maximum of 236 bits
- Similarly, h_3, h_4, h_5, h_6, and h_7 can occupy a maximum of 237 bits.

- So all the h_i's can fit in $\mathbb{F}_n$ limbs.
- The maximum value of h is bounded by

$$
\begin{aligned}
& h_0 + h_1 2^{32} + h_2 2^{64} + h_3 2^{96} + h_4 2^{128} + h_5 2^{160} + h_6 2^{192} + h_7 2^{224} \\
& \leq 2^{243} - 1 + \left(2^{243} - 1\right) 2^{32} + \left(2^{243} - 1\right) 2^{64} + \left(2^{237} - 1\right) 2^{96} + \left(2^{237} - 1\right) 2^{128} + \left(2^{237} - 1\right) 2^{160} \\
& \quad + \left(2^{237} - 1\right) 2^{192} + \left(2^{237} - 1\right) 2^{224} \\
& < 2^{243} + 2^{275} + 2^{307} + 2^{333} + 2^{365} + 2^{397} + 2^{429} + 2^{461} \\
& < 2^{462}
\end{aligned}
$$

4.3 Verifying the Point Addition using Sextic Equations

Recall the point verification equations from (10) and (12).

$$dx_1x_2y_1y_2y_3 + (q - y_3) + x_1x_2 + y_1y_2 = 0 \bmod q, \tag{13}$$
$$dx_1x_2x_3y_1y_2 + x_3 + 2^{38}q - x_1y_2 - x_2y_1 = 0 \bmod q. \tag{14}$$

In (13), the products x_1x_2 and y_1y_2 will be bounded by 2^{292} after folding (see end of Section 4.1). The term $q - y_3$ will be bounded by 2^{255}. And finally, the sextic term $dx_1x_2y_1y_2y_3$ will be bounded by 2^{462} after folding (see end of Section 4.2). So the LHS in (13) will be bounded by 2^{463}.

In (14), once the folded versions of x_1y_2 and x_2y_1 are deducted from $2^{38}q$ the remaining value is non-negative. The term $2^{38}q - x_1y_2 - x_2y_1$ will be bounded by 2^{293}. The term x_3 will be bounded by 2^{255}. And finally, the sextic term $dx_1x_2x_3y_1y_2$ will be bounded by 2^{462} after folding (see end of Section 4.2). So the LHS in (13) will be bounded by 2^{463}.

Let s denote either of the LHS terms of (13) and (14) after folding. Let the unreduced representation of s be

$$s = s_0 + s_1 2^{32} + s_2 2^{64} + s_3 2^{96} + s_4 2^{128} + s_5 2^{160} + s_6 2^{192} + s_7 2^{224}.$$

Each limb s_i will be (conservatively) in the range $\{-2^{74} + 1, 2^{244} - 1\}$. This is because of the following observations.

- The limbs of the sextic product terms $dx_1x_2y_1y_2y_3$ and $dx_1x_2x_3y_1y_2$ after folding are in the range $\{0, \ldots, 2^{243} - 1\}$
- The limbs of x_3 and q are in the range $\{0, 1, \ldots, 2^{64} - 1\}$.
- The limbs of $-y_3$ are in the range $\{-2^{64} + 1, \ldots, 0\}$.
- The limbs of x_1x_2 and y_1y_2 after folding are in the range $\{0, 1, \ldots, 2^{73} - 1\}$. So their sum will have limbs in the range $\{0, 1, \ldots, 2^{74} - 2\}$.
- The limbs of x_1y_2 and x_2y_1 after folding are in the range $\{0, 1, \ldots, 2^{73} - 1\}$. So their sum will have limbs in the range $\{0, 1, \ldots, 2^{74} - 2\}$. And their negation $-x_1y_2 - x_2y_1$ will have limbs in the range $\{-2^{74} + 2, \ldots, 0\}$
- The limbs of $2^{38}q$ will be in the range $\{0, 1, \ldots, 2^{102} - 1\}$.

To show that $s = 0 \bmod q$, we show the existence of a quotient t such that $s - tq = 0$. As $q > 2^{254}$ and $s < 2^{463}$, the maximum value of t required to satisfy this equation is $2^{463-254} = 2^{209}$. So the quotient requires only seven 32-bit limbs in its reduced representation.

$$t = t_0 + t_1 2^{32} + t_2 2^{64} + t_3 2^{96} + t_4 2^{128} + t_5 2^{160} + t_6 2^{192} \qquad \text{where } t_i \in \{0, 1, \ldots, 2^{32} - 1\}.$$

The prime q will have eight 32-bit limbs.

$$q = q_0 + q_1 2^{32} + q_2 2^{64} + q_3 2^{96} + q_4 2^{128} + q_5 2^{160} + q_6 2^{192} + q_7 2^{224} \qquad \text{where } q_i \in \{0, 1, \ldots, 2^{64} - 1\}.$$

The product tq has an unreduced representation with 14 limbs each occupying upto 66 bits.

$$
\begin{aligned}
tq = & \; t_0q_0 \\
& + (t_0q_1 + t_1q_0)2^{32} \\
& + (t_0q_2 + t_1q_1 + t_2q_0)2^{64} \\
& + (t_0q_3 + t_1q_2 + t_2q_1 + t_3q_0)2^{96} \\
& + (t_0q_4 + t_1q_3 + t_2q_2 + t_3q_1 + t_4q_0)2^{128} \\
& + (t_0q_5 + t_1q_4 + t_2q_3 + t_3q_2 + t_4q_1 + t_5q_0)2^{160} \\
& + (t_0q_6 + t_1q_5 + t_2q_4 + t_3q_3 + t_4q_2 + t_5q_1 + t_6q_0)2^{192} \\
& + (t_0q_7 + t_1q_6 + t_2q_5 + t_3q_4 + t_4q_3 + t_5q_2 + t_6q_1)2^{224} \\
& + (t_1q_7 + t_2q_6 + t_3q_5 + t_4q_4 + t_5q_3 + t_6q_2)2^{256} \\
& + (t_2q_7 + t_3q_6 + t_4q_5 + t_5q_4 + t_6q_3)2^{288} \\
& + (t_3q_7 + t_4q_6 + t_5q_5 + t_6q_4)2^{320} \\
& + (t_4q_7 + t_5q_6 + t_6q_5)2^{352} \\
& + (t_5q_7 + t_6q_6)2^{384} \\
& + t_6q_72^{416}
\end{aligned}
$$

Consider the following argument to check that $s - tq = 0$.

1. $s_0 - t_0q_0$ contains the 32 least significant bits of $s - tq$, i.e. bits 0 to 31. These bits must all be zero. So $s_0 - t_0q_0$ must be a multiple of 2^{32}.
2. Let $y_0 = \frac{s_0 - t_0q_0}{2^{32}}$. This represents the *carry* into the 2^{32} limb.
3. $y_0 + s_1 - t_0q_1 - t_1q_0$ contains the *next* 32 least significant bits of $s - tq$, i.e. bits 32 to 63. These bits must also all be zero. So $y_0 + s_1 - t_0q_1 - t_1q_0$ must be a multiple of 2^{32}.
4. Let $y_1 = \frac{y_0 + s_1 - t_0q_1 - t_1q_0}{2^{32}}$. This represents the carry into the 2^{64} limb.
5. $y_1 + s_2 - t_0q_2 - t_1q_1 - t_2q_0$ contains the next 32 least significant bits of $s - tq$, i.e. bits 64 to 95. These bits must also all be zero. So $y_1 + s_2 - t_0q_2 - t_1q_1 - t_2q_0$ must be a multiple of 2^{32}.
6. Let $y_2 = \frac{y_1 + s_2 - t_0q_2 - t_1q_1 - t_2q_0}{2^{32}}$. This represents the carry into the 2^{96} limb.
7. $y_2 + s_3 - t_0q_3 - t_1q_2 - t_2q_1 - t_3q_0$ contains the next 32 least significant bits of $s - tq$, i.e. bits 96 to 127. These bits must also all be zero. So $y_2 + s_3 - t_0q_3 - t_1q_2 - t_2q_1 - t_3q_0$ must be a multiple of 2^{32}.
8. Let $y_3 = \frac{y_2 + s_3 - t_0q_3 - t_1q_2 - t_2q_1 - t_3q_0}{2^{32}}$. This represents the carry into the 2^{128} limb.
9. $y_3 + s_4 - t_0q_4 - t_1q_3 - t_2q_2 - t_3q_1 - t_4q_0$ contains the next 32 least significant bits of $s - tq$, i.e. bits 128 to 159. These bits must also all be zero. So $y_3 + s_4 - t_0q_4 - t_1q_3 - t_2q_2 - t_3q_1 - t_4q_0$ must be a multiple of 2^{32}.
10. Let $y_4 = \frac{y_3 + s_4 - t_0q_4 - t_1q_3 - t_2q_2 - t_3q_1 - t_4q_0}{2^{32}}$. This represents the carry into the 2^{160} limb.
11. $y_4 + s_5 - t_0q_5 - t_1q_4 - t_2q_3 - t_3q_2 - t_4q_1 - t_5q_0$ contains the next 32 least significant bits of $s - tq$, i.e. bits 160 to 191. These bits must also all be zero. So $y_4 + s_5 - t_0q_5 - t_1q_4 - t_2q_3 - t_3q_2 - t_4q_1 - t_5q_0$ must be a multiple of 2^{32}.
12. Let $y_5 = \frac{y_4 + s_5 - t_0q_5 - t_1q_4 - t_2q_3 - t_3q_2 - t_4q_1 - t_5q_0}{2^{32}}$. This represents the carry into the 2^{192} limb.
13. $y_5 + s_6 - t_0q_6 - t_1q_5 - t_2q_4 - t_3q_3 - t_4q_2 - t_5q_1 - t_6q_0$ contains the next 32 least significant bits of $s - tq$, i.e. bits 192 to 223. These bits must also all be zero. So $y_5 + s_6 - t_0q_6 - t_1q_5 - t_2q_4 - t_3q_3 - t_4q_2 - t_5q_1 - t_6q_0$ must be a multiple of 2^{32}.

14. Let $y_6 = \frac{y_5+s_6-t_0q_6-t_1q_5-t_2q_4-t_3q_3-t_4q_2-t_5q_1-t_6q_0}{2^{32}}$. This represents the carry into the 2^{224} limb.

15. $y_6 + s_7 - t_0q_7 - t_1q_6 - t_2q_5 - t_3q_4 - t_4q_3 - t_5q_2 - t_6q_1$ contains the next 32 least significant bits of $s - tq$, i.e. bits 224 to 255. These bits must also all be zero. So $y_6 + s_7 - t_0q_7 - t_1q_6 - t_2q_5 - t_3q_4 - t_4q_3 - t_5q_2 - t_6q_1$ must be a multiple of 2^{32}.

16. Let $y_7 = \frac{y_6+s_7-t_0q_7-t_1q_6-t_2q_5-t_3q_4-t_4q_3-t_5q_2-t_6q_1}{2^{32}}$. This represents the carry into the 2^{256} limb.

17. $y_7 - t_1q_7 - t_2q_6 - t_3q_5 - t_4q_4 - t_5q_3 - t_6q_2$ contains the next 32 least significant bits of $s - tq$, i.e. bits 256 to 287. These bits must also all be zero. So $y_7 - t_1q_7 - t_2q_6 - t_3q_5 - t_4q_4 - t_5q_3 - t_6q_2$ must be a multiple of 2^{32}.

18. Let $y_8 = \frac{y_7-t_1q_7-t_2q_6-t_3q_5-t_4q_4-t_5q_3-t_6q_2}{2^{32}}$. This represents the carry into the 2^{288} limb.

19. $y_8 - t_2q_7 - t_3q_6$ contains the next 32 least significant bits of $s - tq$, i.e. bits 288 to 319. These bits must also all be zero. So $y_8 - t_2q_7 - t_3q_6 - t_4q_5 - t_5q_4 - t_6q_3$ must be a multiple of 2^{32}.

20. Let $y_9 = \frac{y_8-t_2q_7-t_3q_6-t_4q_5-t_5q_4-t_6q_3}{2^{32}}$. This represents the carry into the 2^{320} limb.

21. $y_9 - t_3q_7 - t_4q_6 - t_5q_5 - t_6q_4$ contains the next 32 least significant bits of $s - tq$, i.e. bits 320 to 351. These bits must also all be zero. So $y_9 - t_3q_7 - t_4q_6 - t_5q_5 - t_6q_4$ must be a multiple of 2^{32}.

22. Let $y_{10} = \frac{y_9-t_3q_7-t_4q_6-t_5q_5-t_6q_4}{2^{32}}$. This represents the carry into the 2^{352} limb.

23. $y_{10} - t_4q_7 - t_5q_6 - t_6q_5$ contains the next 32 least significant bits of $s - tq$, i.e. bits 352 to 383. These bits must also all be zero. So $y_{10} - t_4q_7 - t_5q_6 - t_6q_5$ must be a multiple of 2^{32}.

24. Let $y_{11} = \frac{y_{10}-t_4q_7-t_5q_6-t_6q_5}{2^{32}}$. This represents the carry into the 2^{384} limb.

25. $y_{11} - t_5q_7 - t_6q_6$ contains the next 32 least significant bits of $s - tq$, i.e. bits 384 to 415. These bits must also all be zero. So $y_{11} - t_5q_7 - t_6q_6$ must be a multiple of 2^{32}.

26. Let $y_{12} = \frac{y_{11}-t_5q_7-t_6q_6}{2^{32}}$. This represents the carry into the 2^{416} limb.

27. $y_{12} - t_6q_7$ contains the remaining 47 least significant bits of $s - tq$, i.e. bits 416 to 462. Recall that s is bounded by 2^{463}. These bits must also all be zero. So $y_{12} - t_6q_7$ *must be zero.*

As the s_i's lie in the range $\{-2^{74}+1, \ldots, 2^{244}-1\}$ and the limbs of tq occupy a maximum of 66 bits, the following terms (the unreduced limbs of $s - tq$) lie in the range $\{-2^{75}+1, \ldots, 2^{244}-1\}$

$$
\begin{aligned}
&s_0 - t_0q_0,\\
&s_1 - t_0q_1 - t_1q_0,\\
&s_2 - t_0q_2 - t_1q_1 - t_2q_0,\\
&s_3 - t_0q_3 - t_1q_2 - t_2q_1 - t_3q_0,\\
&s_4 - t_0q_4 - t_1q_3 - t_2q_2 - t_3q_1 - t_4q_0,\\
&s_5 - t_0q_5 - t_1q_4 - t_2q_3 - t_3q_2 - t_4q_1 - t_5q_0,\\
&s_6 - t_0q_6 - t_1q_5 - t_2q_4 - t_3q_3 - t_4q_2 - t_5q_1 - t_6q_0,\\
&s_7 - t_0q_7 - t_1q_6 - t_2q_5 - t_3q_4 - t_4q_3 - t_5q_2 - t_6q_1,\\
&- t_1q_7 - t_2q_6 - t_3q_5 - t_4q_4 - t_5q_3 - t_6q_2,\\
&- t_2q_7 - t_3q_6 - t_4q_5 - t_5q_4 - t_6q_3,\\
&- t_3q_7 - t_4q_6 - t_5q_5 - t_6q_4,\\
&- t_4q_7 - t_5q_6 - t_6q_5,\\
&- t_5q_7 - t_6q_6,\\
&- t_6q_7.
\end{aligned}
$$

Both the upper and lower ends of the ranges are conservative but we keep these values for convenience.

The carries $y_0, y_1, y_2, \ldots, y_{12}$ will be provided as non-deterministic advice to the arithmetic circuit. Instead of calculating y_0 as $\frac{g_0 - t_0 q_0}{2^{64}}$, we will check that $2^{64} y_0 = g_0 - t_0 q_0$ in the field $\mathbb{F}_n$. We need to apply range checks on the y_i's to ensure that adding them will not exceed the capacity of $\mathbb{F}_n$.

- As $s_0 - t_0 q_0$ is in the range $\{-2^{75}+1, \ldots, 2^{244}-1\}$, y_0 can be checked to be in the range $\{-2^{43}+1, \ldots, 2^{212}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_0 + 2^{43}$ is in the range $\{0, 1, \ldots, 2^{213}-1\}$.
- $y_0 + s_1 - t_0 q_1 - t_1 q_0$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_1 = \frac{y_0 + s_1 - t_0 q_1 - t_1 q_0}{2^{32}}$, we can check that y_1 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_1 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_1 + s_2 - t_0 q_2 - t_1 q_1 - t_2 q_0$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_2 = \frac{y_1 + s_2 - t_0 q_2 - t_1 q_1 - t_2 q_0}{2^{32}}$, we can check that y_2 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_2 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_2 + s_3 - t_0 q_3 - t_1 q_2 - t_2 q_1 - t_3 q_0$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_3 = \frac{y_2 + s_3 - t_0 q_3 - t_1 q_2 - t_2 q_1 - t_3 q_0}{2^{32}}$, we can check that y_3 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_3 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_3 + s_4 - t_0 q_4 - t_1 q_3 - t_2 q_2 - t_3 q_1 - t_4 q_0$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_4 = \frac{y_3 + s_4 - t_0 q_4 - t_1 q_3 - t_2 q_2 - t_3 q_1 - t_4 q_0}{2^{32}}$, we can check that y_4 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_4 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_4 + s_5 - t_0 q_5 - t_1 q_4 - t_2 q_3 - t_3 q_2 - t_4 q_1 - t_5 q_0$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_5 = \frac{y_4 + s_5 - t_0 q_5 - t_1 q_4 - t_2 q_3 - t_3 q_2 - t_4 q_1 - t_5 q_0}{2^{32}}$, we can check that y_5 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_5 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_5 + s_6 - t_0 q_6 - t_1 q_5 - t_2 q_4 - t_3 q_3 - t_4 q_2 - t_5 q_1 - t_6 q_0$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_6 = \frac{y_5 + s_6 - t_0 q_6 - t_1 q_5 - t_2 q_4 - t_3 q_3 - t_4 q_2 - t_5 q_1 - t_6 q_0}{2^{32}}$, we can check that y_6 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_6 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_6 + s_7 - t_0 q_7 - t_1 q_6 - t_2 q_5 - t_3 q_4 - t_4 q_3 - t_5 q_2 - t_6 q_1$ is in the range $\{-2^{76}+1, \ldots, 2^{245}-1\}$. Since $y_7 = \frac{y_6 + s_7 - t_0 q_7 - t_1 q_6 - t_2 q_5 - t_3 q_4 - t_4 q_3 - t_5 q_2 - t_6 q_1}{2^{32}}$, we can check that y_7 is in the range $\{-2^{44}+1, \ldots, 2^{213}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_7 + 2^{44}$ is in the range $\{0, 1, \ldots, 2^{214}-1\}$.
- $y_7 - t_1 q_7 - t_2 q_6 - t_3 q_5 - t_4 q_4 - t_5 q_3 - t_6 q_2$ is in the range $\{-2^{67}+1, \ldots, 2^{213}-1\}$. Since $y_8 = \frac{y_7 - t_1 q_7 - t_2 q_6 - t_3 q_5 - t_4 q_4 - t_5 q_3 - t_6 q_2}{2^{32}}$, we can check that y_8 is in the range $\{-2^{35}+1, \ldots, 2^{181}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_8 + 2^{35}$ is in the range $\{0, 1, \ldots, 2^{182}-1\}$.
- $y_8 - t_2 q_7 - t_3 q_6 - t_4 q_5 - t_5 q_4 - t_6 q_3$ is in the range $\{-2^{67}+1, \ldots, 2^{213}-1\}$. Since $y_9 = \frac{y_8 - t_2 q_7 - t_3 q_6 - t_4 q_5 - t_5 q_4 - t_6 q_3}{2^{32}}$, we can check that y_9 is in the range $\{-2^{35}+1, \ldots, 2^{181}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_9 + 2^{35}$ is in the range $\{0, 1, \ldots, 2^{182}-1\}$.
- $y_9 - t_3 q_7 - t_4 q_6 - t_5 q_5 - t_6 q_4$ is in the range $\{-2^{67}+1, \ldots, 2^{213}-1\}$. Since $y_{10} = \frac{y_9 - t_3 q_7 - t_4 q_6 - t_5 q_5 - t_6 q_4}{2^{32}}$, we can check that y_{10} is in the range $\{-2^{35}+1, \ldots, 2^{181}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_{10} + 2^{35}$ is in the range $\{0, 1, \ldots, 2^{182}-1\}$.
- $y_{10} - t_4 q_7 - t_5 q_6 - t_6 q_5$ is in the range $\{-2^{67}+1, \ldots, 2^{213}-1\}$. Since $y_{11} = \frac{y_{10} - t_4 q_7 - t_5 q_6 - t_6 q_5}{2^{32}}$, we can check that y_{11} is in the range $\{-2^{35}+1, \ldots, 2^{181}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_{11} + 2^{35}$ is in the range $\{0, 1, \ldots, 2^{182}-1\}$.
- $y_{11} - t_5 q_7 - t_6 q_6$ is in the range $\{-2^{67}+1, \ldots, 2^{213}-1\}$. Since $y_{12} = \frac{y_{11} - t_5 q_7 - t_6 q_6}{2^{32}}$, we can check that y_{12} is in the range $\{-2^{35}+1, \ldots, 2^{181}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_{12} + 2^{35}$ is in the range $\{0, 1, \ldots, 2^{182}-1\}$.

A range check for an n-bit range costs n R1CS constraints. In the above list, there are 13 range checks costing a total of $213 + 7 \times 214 + 5 \times 182 = 2621$ constraints. The above check has to be done twice, once each (13) and (14). Just the range checks alone would cost 5242 constraints. With the 64-bit limb approach, the point addition verification is possible in about 4000 constraints. So the 32-bit limb approach is more expensive.

5 An Approach using Three 85-bit Limbs

The previous approach demonstrated that the number of limbs can affect the circuit size via the number and size of range checks required at each stage. It seems worthwhile to investigate the case of using three 85-bit limbs to represent a ed25519 base field element. This representation was used by Electron Labs.[5] However, they did not use the lazy reduction approach of `circom-ecdsa`.

As the cubic product 85-bit limbs will require 255 bits, the point verification can only involve quadratic terms in unreduced form. Recall that the affine point addition verification equations can be written as

$$x_3(1 + dx_1x_2y_1y_2) = x_1y_2 + x_2y_1, \tag{15}$$
$$y_3(1 - dx_1x_2y_1y_2) = x_1x_2 + y_1y_2. \tag{16}$$

We propose to try the following approach.

- Calculate the reduced representation of the cubic x_1x_2. Let this representation be given by $u = \sum_{i=0}^{2} u_i 2^{85i}$.
- Calculate the reduced representation of the cubic y_1y_2. Let this representation be given by $v = \sum_{i=0}^{2} v_i 2^{85i}$.
- Calculate the reduced representation of the cubic uv. Let this representation be given by $w = \sum_{i=0}^{2} w_i 2^{85i}$.
- Calculate the reduced representation of the cubic dw. Let this representation be given by $z = \sum_{i=0}^{2} z_i 2^{85i}$.
- Check that the following quadratic equations hold.

$$x_1y_2 + x_2y_1 - x_3 - x_3z = 0,$$
$$x_1x_2 + y_1y_2 - y_3 + y_3z = 0.$$

5.1 Reducing a Quadratic Product

Let a, b be two elements in $\mathbb{F}_q$ available in their reduced representations.

$$a = \sum_{i=0}^{2} a_i 2^{85i}, \qquad b = \sum_{i=0}^{2} b_i 2^{85i}.$$

Let $g = ab$. Then we have $g = \sum_{i=0}^{4} g_i 2^{85i}$ where

$$\begin{aligned} g_0 &= a_0b_0, \\ g_1 &= a_0b_1 + a_1b_0, \\ g_2 &= a_0b_2 + a_1b_1 + a_2b_0, \\ g_3 &= a_1b_2 + a_2b_1, \\ g_4 &= a_2b_2. \end{aligned}$$

[5] `https://github.com/Electron-Labs/ed25519-circom`

The limbs g_1, g_2, g_3 occupy a maximum of 171 bits and the other limbs occupy 170 bits.

Since $q = 2^{255} - 19$, we have $2^{255} = 19 \bmod q$. We can rewrite g (modulo q) as

$$\begin{aligned} g = \sum_{i=0}^{4} g_i 2^{85i} &= g_0 + g_1 2^{85} + g_2 2^{170} + g_3 2^{255} + g_4 2^{340} \\ &= g_0 + g_1 2^{85} + g_2 2^{170} + 2^{255} \left(g_3 + g_4 2^{85}\right) \\ &= g_0 + 19 g_3 + (g_1 + 19 g_4) 2^{85} + g_2 2^{170} \\ &= h_0 + h_1 2^{85} + h_2 2^{170}, \end{aligned}$$

where

$$\begin{aligned} h_0 &= g_0 + 19 g_3, \\ h_1 &= g_1 + 19 g_4, \\ h_2 &= g_2. \end{aligned}$$

We note the following:

- Each of the g_i's occupy a maximum of 172 bits.
- The number 19 occupies 5 bits.
- $h_0 = g_0 + 19 g_3$ can occupy a maximum of 177 bits by the following observations.
  - g_0 occupies a maximum of 170 bits
  - g_3 occupies a maximum of 171 bits
  - $19 g_3$ occupies a maximum of 176 bits
- $h_1 = g_1 + 19 g_4$ can occupy a maximum of 176 bits by the following observations.
  - g_1 occupies a maximum of 171 bits
  - g_4 occupies a maximum of 170 bits
  - $19 g_4$ occupies a maximum of 175 bits
- $h_2 = g_2$ can occupy a maximum of 172 bits
- So all the h_i's can fit in $\mathbb{F}_n$ limbs.
- The maximum value of g is bounded by

$$2^{177} - 1 + \left(2^{176} - 1\right) 2^{85} + \left(2^{172} - 1\right) 2^{170} < 2^{177} + 2^{261} + 2^{342} < 2^{343}$$

We want to find an $r \in \mathbb{F}_q$ in reduced representation such that

$$g = tq + r \tag{17}$$

for some quotient $t \in \mathbb{F}_q$. Here

$$r = r_0 + r_1 2^{85} + r_2 2^{170} \qquad \text{where } r_i \in \{0, 1, \ldots, 2^{85} - 1\}.$$

As $q > 2^{254}$, the maximum value of t required to satisfy this equation is strictly less than $2^{343-254} = 2^{89}$. So the quotient requires only two 85-bit limbs in its reduced representation.

$$t = t_0 + t_1 2^{85} \qquad \text{where } t_0 \in \{0, 1, \ldots, 2^{85} - 1\}, t_1 \in \{0, 1, .., 2^4 - 1\}.$$

The prime q will have three 85-bit limbs.

$$q = q_0 + q_1 2^{85} + q_2 2^{170} \qquad \text{where } q_i \in \{0, 1, \ldots, 2^{64} - 1\}.$$

The main challenge in checking (17) is the mismatch in the representations of the LHS and the RHS. On the LHS, g has an unreduced representation with three limbs each occupying upto 177 bits.

$$g = h_0 + h_1 2^{85} + h_2 2^{170}.$$

On the RHS, $tq + r$ has an unreduced representation with six limbs each occupying upto 130 bits.

$$tq + r = t_0 q_0 + r_0 + (t_0 q_1 + t_1 q_0 + r_1) 2^{85} + (t_0 q_2 + t_1 q_1 + r_2) 2^{170} + t_1 q_2 2^{255}.$$

One way to check equality in (17) is to convert both g and $tq + r$ to their reduced representations and check that these representations are equal. This would require range checks on the limbs of g and $tq + r$. The number of boolean variables required for each range check will be equal to the bitwidth of the corresponding limb. Each boolean variable requires one constraint of the form $b(b - 1) = 0$ in the R1CS system.

We could reduce the number of range checks by checking that $g - tq - r$ equals zero. This is the approach used in `circom-ecdsa`. Consider the following argument assuming that $g - tq - r = 0$.

1. $h_0 - t_0 q_0 - r_0$ contains the 85 least significant bits of $g - tq - r$, i.e. bits 0 to 84. These bits must all be zero. So $h_0 - t_0 q_0 - r_0$ must be a multiple of 2^{85}.
2. Let $y_0 = \frac{h_0 - t_0 q_0 - r_0}{2^{85}}$. This represents the *carry* into the 2^{85} limb.
3. $y_0 + h_1 - t_0 q_1 - t_1 q_0 - r_1$ contains the *next* 85 least significant bits of $g - tq - r$, i.e. bits 85 to 169. These bits must also all be zero. So $y_0 + h_1 - t_0 q_1 - t_1 q_0 - r_1$ must be a multiple of 2^{85}
4. Let $y_1 = \frac{y_0 + h_1 - t_0 q_1 - t_1 q_0 - r_1}{2^{85}}$. This represents the carry into the 2^{170} limb.
5. $y_1 + h_2 - t_0 q_2 - t_1 q_1 - r_2$ contains the next 85 least significant bits of $g - tq - r$, i.e. bits 170 to 254. These bits must also all be zero. So $y_1 + h_2 - t_0 q_2 - t_1 q_1 - r_2$ must be a multiple of 2^{85}.
6. Let $y_2 = \frac{y_1 + h_2 - t_0 q_2 - t_1 q_1 - r_2}{2^{85}}$. This represents the carry into the 2^{255} limb.
7. $y_2 - t_1 q_2$ contains the remaining 87 least significant bits of $g - tq - r$, i.e. bits 255 to 342. Recall that g is bounded by 2^{343}. These bits must also all be zero. So $y_2 - t_1 q_2$ *must be zero.*

Note that the limbs of $g - tq - r$ can have negative values, i.e. they can experience underflows during the subtraction operation. We use the convention that $x \in \mathbb{F}_n$ is *negative* if $x > \frac{n-1}{2}$. For example, $h_0 - t_0 q_0 - r_0$ can be a negative multiple of 2^{85}.

As h_0, h_1, h_2 have maximum bitwidths of 177, 176, and 172, $t_0, q_0, q_1, q_2, r_0, r_1, r_2$ have maximum bitwidths of 85, and t_1 has a maximum bitwidth of 4, the following terms (the unreduced limbs of $g - tq - r$) lie in the range indicated next to them.

$$\begin{aligned}
&h_0 - t_0 q_0 - r_0 && \in \{-2^{171} + 1, 2^{177} - 1\},\\
&h_1 - t_0 q_1 - t_1 q_0 - r_1 && \in \{-2^{171} + 1, 2^{176} - 1\},\\
&h_2 - t_0 q_2 - t_1 q_1 - r_2 && \in \{-2^{171} + 1, 2^{172} - 1\},\\
&- t_1 q_2 && \in \{-2^{89} + 1, 0\}.
\end{aligned}$$

The procedure for checking $g - tq - r = 0$ involves the addition of multiple terms some of which can be negative. Furthermore, the addition will be performed in $\mathbb{F}_n$. A set of terms sum to zero in $\mathbb{F}_n$ may not sum to zero in $\mathbb{F}_q$. To ensure that they do sum to zero in $\mathbb{F}_q$, we should ensure that the bitwidths of the partial sums does not exceed the capacity of $\mathbb{F}_n$.

The bitwidths of the terms h_i, q_i, t_i, r_i will be known due to range checks. The carries y_0, y_1, y_2 will be provided as non-deterministic advice to the arithmetic circuit. Instead of calculating y_0 as $\frac{h_0 - t_0 q_0 - r_0}{2^{85}}$, we will check that $2^{85} y_0 = h_0 - t_0 q_0 - r_0$ in the field $\mathbb{F}_n$. We need to apply range checks on the y_i's to ensure that adding them will not exceed the capacity of $\mathbb{F}_n$.

- As $h_0 - t_0q_0 - r_0$ is in the range $\{-2^{171}+1, \ldots, 2^{177}-1\}$, y_0 can be checked to be in the range $\{-2^{86}+1, \ldots, 2^{92}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_0 + 2^{86}$ is in the range $\{0, 1, \ldots, 2^{93}-1\}$.

- $y_0 + h_1 - t_0q_1 - t_1q_0 - r_1$ is in the range $\{-2^{172}+1, \ldots, 2^{177}-1\}$. Since $y_1 = \frac{y_0+h_1-t_0q_1-t_1q_0-r_1}{2^{85}}$, we can check that y_1 is in the range $\{-2^{87}+1, \ldots, 2^{92}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_1 + 2^{87}$ is in the range $\{0, 1, \ldots, 2^{93}-1\}$.

- $y_1+h_2-t_0q_2-t_1q_1-t_2q_0-r_2$ is in the range $\{-2^{172}+1, \ldots, 2^{173}-1\}$. Since $y_2 = \frac{y_1+h_2-t_0q_2-t_1q_1-t_2q_0-r_2}{2^{85}}$, we can check that y_2 is in the range $\{-2^{87}+1, \ldots, 2^{88}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_2 + 2^{87}$ is in the range $\{0, 1, \ldots, 2^{89}-1\}$.

- $y_2 - t_1q_2$ is in the range $\{-2^{90}+1, \ldots, 2^{88}-1\}$. Since $y_4 = \frac{y_2-t_1q_2}{2^{85}}$, we can check that y_4 is in the range $\{-2^{5}+1, \ldots, 2^{3}-1\}$. In the arithmetic circuit, this is accomplished by checking that $y_4 + 2^5$ is in the range $\{0, 1, \ldots, 2^6-1\}$.

A range check for an n-bit range costs n R1CS constraints. In the above list, there are 4 range checks costing a total of $93 \times 2 + 89 + 6 = 281$ constraints. The above check has to be done **twelve** times, six times each for (15) and (16). Just the carry range checks alone would cost 3372 constraints. With the 64-bit limb approach, the point addition verification is possible in about 4000 constraints, of which the carry range checks were 2034 constraints. As the total number of constraints is roughly double the carry range checks, the 85-bit limb approach is *likely* to be more expensive.