Zero Knowledge Proofs of Elliptic Curve Inner Products from Principal Divisors and Weil Reciprocity

Liam Eagen
liameagen@protonmail.com

May 15, 2022

Abstract

Zero Knowledge proofs of Elliptic Curve Inner Products (ECIPs) and elliptic curve operations more generally are an increasingly important part of zero knowledge protocols and a significant bottle neck in recursive proof composition over amicable cycles of elliptic curves. To prove ECIPs more efficiently, I represent a collection of points that sum to zero using a polynomial element of the function field and evaluate this function at a random principal divisor. By Weil reciprocity, this is equal to the function interpolating the random divisor evaluated at the original points. Taking the logarithmic derivative of both expressions allows the prover to use a similar technique to the Bulletproofs++ permutation argument and take linear combinations logarithmic derivatives of divisor witnesses and collect terms for the same basis point by adding the multiplicities. The linear combination can be random or can be structured to cancel intermediate points in computing the sum. Since the multiplicities are field elements, this system can prove ECIP relations in zero knowledge with respect to the linear combination, the curve points, or both. Compared to existing techniques, the witness size is reduced by up to a factor of 10 and the number of multiplications by a factor of about 100 with significantly more flexibility in the organization of the protocol. The specific improvement will depend on the instantiating proof system, number of curve points, and which information is zero knowledge. This technique also works, with small modification, for proving multiexponentiations in the multiplicative group of the field.

1 Introduction

Proofs of elliptic curve inner products are an important part of many proof systems, like JubJub [3] in ZCash [4] and proof systems based on amicable cycles of curves [5] such as [6] used in CODA [7] and Halo [8]. While the techniques of this paper may not be suitable as a drop in replacement for existing techniques in all proof systems, as they require an additional round of interaction between the prover and verifier, when possible they achieve the far smaller witness sizes and many fewer multiplications. This is possible by representing collections of points that sum to zero by elements of the function field of the curve E, taking the logarithmic derivative at a random point, and then taking linear combinations of the resulting rational functions to achieve maximum cancellation of intermediate points. Using complex multiplication, the size can be further reduced.

The techniques of this paper are flexible enough to handle all three configurations, that is proofs that are zero knowledge in the points, proofs that are zero knowledge in the scalars, and proofs that are zero knowledge in both. It can also handle multiple configurations for different points in the same circuit, being zero knowledge in some of the multiplicities and some of the points, while others are public. Depending on which data are public, certain optimizations are available. Of the three possible configurations, proofs zero knowledge in the points but not the scalars generate the samllest witnesses and proofs zero knowledge in both generate the largest, although the difference is not particularly large.

This paper does not discuss implementations of the ECIP proof techniques in existing zero knowledge proof systems like SNARKs [9] or Bulletproofs [10]. However, any system capable of encoding arithmetic circuits plus one additional round of interaction between the prover and verifier is powerful enough to use the techniques of this paper. This requirement is necessary because the prover must perform divisions in response to some random data chosen by the verifier.

This paper will show the soundness of the protocol, specifically given challenges sampled honestly by the verifier that the probability of an invalid witness satisfying the final checks is negligible. Other properties of zero knowledge proof systems are the responsibility of the implementing proof system. In

practice, challenges will be sampled by a random oracle hash function, which is why the honest verifier assumption is reasonable in practice. The use of such a hash function is typically necessary to make protocols non-interactive via the Fiat-Shamir heuristic.

1.1 Notation

Let E be an elliptic curve in Weierstrass form $E : y^2 = x^3 + Ax + B$ over a prime order field $\mathbb{F}$. Note that every curve can be put in Weierstrass form if it is not already. I will use E to refer both to the curve and the group of points of the curve. The points of the curve will be denoted by capital letters $P, Q, ...$ and scalars by lower case letters $x, y, ...$ The group operation will be written additively, the point at infinity by O, and scalar multiplication by juxtaposition. I will write $Q = \sum_{i=1}^{n} e_i P_i$ for the inner product of the scalar vector of e_i with the point vector of P_i. The affine coordinates of a point will be written as $x(P)$ and $y(P)$, so $y(P)^2 = x(P)^3 + Ax(P) + B$ by definition. Any polynomial in x and y will inherit this notation for evaluation at a point, so if $z = y - \lambda x$ for constant λ one can write $z(P)$ as a shorthand for $y(P) - \lambda x(P)$.

It is important to keep in mind that the elliptic curve E is *not* the elliptic curve used by the underlying proof system to prove the protocols of this paper, although it may be closely related. The most common use cases are either curves like JubJub, which are used for hashing and have no special relationship to the underlying proof system apart from the agreement of the base field of the curve and the scalar field of the proving curve, and amicable cycles of curves. The latter case is used by Halo and other proof systems for recursive proof composition. Each curve proves statements about the other, since the base fields of each agree with the curve order of the other. Since this paper does not deal with the underlying proof system, and for maximum generality, this paper will make no assumptions about E.

2 The Picard Group of E

Many discussions of elliptic curves avoid delving into the details of how the group law is actually derived, preferring instead to define the group law in terms of an opaque collection of axioms. It is not apparent why these axioms define a group, but for the purposes of programmers implementing elliptic curves or cryptographers using elliptic curves in a black box manner for the hardness of the discrete log problem, it is typically not necessary to have a deeper understanding. This computationally oriented view of the elliptic curve group law is the starting point for how zero knowledge proofs of elliptic curve operations have tended to be defined. Curves like JubJub were selected, at least in part, specifically because the curve operations of Edwards curves are easier to compute than those of Weierstrass curves [11].

However, using the Picard group, one can show that a collection of curve points sum to zero by demonstrating the existence of an element of the function field of the curve that intersects exactly these points. This is not sufficient to prove anything by itself, but this element of the function field is remarkably compact, requiring only a single scalar per elliptic curve point and no special cases for duplicate, inverse, or identity curve points. This representation of the group law is not useful for computing elliptic curve operations, but it is extremely useful for checking that elliptic curve operations were carried out correctly. This asymmetry between what is necessary to compute and what is necessary to verify a computation was carried out correctly is fundamental to computer science, and reframing other problems in this way may lead to more performant zero knowledge proof systems.

2.1 Divisors

Define an elliptic curve "divisor" to be a formal sum of curve points, including the point at infinity. This forms a group $\mathrm{Div}(E)$ naturally via pointwise addition and negation of the multiplicities of each point

$$D = \sum_{P \in E} v_P(D) P \qquad v_P(D_0 + D_1) = v_P(D_0) + v_P(D_1)$$

Define the degree of a divisor to be the sum of its multiplicities

$$\deg(D) = \sum_{P \in E} v_P(D)$$

And note that the degree of the sum of two divisors is the sum of their degrees, $\deg(D_0 + D_1) = \deg(D_0) + \deg(D_1)$. The kernel of this group homomorphism, the divisors where $\deg(D) = 0$, forms a subgroup of the group of divisors denoted $\mathrm{Div}_0(E)$.

2.2 Function Field

Define the function field of the elliptic curve $\mathbb{F}(E)$ to be the quotient of the field of rational functions in x and y by the curve

$$\mathbb{F}(E) = \mathbb{F}(x, y)/(y^2 - x^3 - Ax - B)$$

One can think about this field as extending the field $\mathbb{F}(x)$ by y a "square root" of the polynomial $x^3 + Ax + B$ in a manner analogous to how a quadratic number field extends a base field. With this in mind, one can similarly eliminate higher powers of y by substitution of $x^3 + Ax + B$ for y^2, define the conjugate of an element via $y \mapsto -y$, and define the norm of element as its product with its conjugate. Concretely, for any $f(x, y) \in \mathbb{F}(x, y)$ there exist rational functions $a(x)$ and $b(x)$ such that

$$f(x, y) \equiv a(x) - yb(x) \mod y^2 - x^3 - Ax - B$$

And the norm of $f(x, y)$ is the product of it with its conjugate

$$N(f) = f(x, y)f(x, -y) = a(x)^2 - (x^3 + Ax + B)b(x)^2 = \prod_{i=1}^{n}(x - x_i)^{m_i}$$

Note that each root of the norm $x_i = x(P_i)$ where $f(P_i) = 0$ over a suitable extension. To compute the correct y coordinate, one can substitute $x(P_i)$ into f and solve $f(x(P_i), y) = 0$ to find

$$y(P_i) = \frac{a(x(P_i))}{b(x(P_i))}$$

This associates to each element of the function field the set of points where it intersects the elliptic curve with multiplicity, that is a divisor $\mathrm{div}(f)$. Any poles of f will appear as points of negative multiplicity in the divisor. Considering the zero or pole at infinity of the polynomial $N(f)$ and assigning this multiplicity to the point at infinity, every divisor of an element of the function field actually has degree zero and therefore $\mathrm{div}(f) \in \mathrm{Div}_0(E)$.

This set of degree zero divisors associated to elements of the function field is the set of principal divisors $\mathrm{Prin}(E)$. Since the function field is closed under multiplication, the group of principal divisors forms a group. As a subgroup of the group of degree zero divisors, and since both groups are abelian, the quotient $\mathrm{Pic}(E) = \mathrm{Div}_0(E)/\mathrm{Prin}(E)$ is well defined and is isomorphic to the group of points of an elliptic curve $\mathrm{Pic}(E) \simeq E$. Note the important corollary that all these relationships go both ways, so any collection of points that sums to zero is a principal divisor and has an associated element of the function field.

3 Proving Sums of Point

From the previous section, $\sum_{i=1}^{n} P_i = 0$ iff there is a divisor interpolating exactly the points P_i and no other points, apart from the point at infinity. That is, for $n = 2k + r$ with $r \in \{0, 1\}$, there exists $\deg(a(x)) = k$ and $\deg(b(x)) = k - 2 + r$ so that $D(x, y) = a(x) - yb(x)$ satisfies $D(P_i) = 0$ iff the points sum to zero. These degree constraints ensure that $\deg(N(D)) = \max(2k, 2(k + r) - 1) = n$. For the purposes of this section, I will mostly ignore the point at infinity and refer to the function field element as the divisor witness for its associated divisor. All divisor witnesses are assumed to be polynomials and all points to have positive multiplicities, except the point at infinity. The number of non-identity points in the divisor of polynomial will be hereafter referred to as

$$n(D) = -v_O(\mathrm{div}(D))$$

3.1 Interpolating the Divisor Witness

There are several different ways to construct the witness interpolating interpolating a principal divisor over P_i. It is outside the scope of this paper to determine which is the most efficient in practice, but for completeness I will describe two of them. The first constructs the divisor witness incrementally by multiplying sub-divisors together and factoring out intermediate values. The second constructs the Mumford representation directly and then uses a half-GCD algorithm for polynomials.

3.1.1 Incremental construction

Given the sequence of points P_i, assuming that they do form a principal divisor, to find a witness of this fact in the form of an element of the function field of the elliptic curve, the prover will first construct lines interpolating adjacent points and their negated sums Q_i as

$$\text{div}(L_i) = P_{2i} + P_{2i+1} + Q_i - 3O$$

The prover will then repeat the procedure on the points $-Q_i$ to produce negated sums Q'_i so that

$$\text{div}(L'_i) = (-Q_{2i}) + (-Q_{2i+1}) + Q'_i - 3O$$

To each point Q'_i the prover will associate the new polynomials

$$\text{div}\left(\frac{L'_i L_{2i} L_{2i+1}}{(x - x(Q_{2i}))(x - x(Q_{2i+1}))}\right) = P_{4i} + P_{4i+1} + P_{4i+2} + P_{4i+3} + Q'_i - 5O$$

Since the numerator intersects both $\pm Q_{2i}$ and $\pm Q_{2i+1}$, the denominator will factor cleanly out of the product, and the result will be a polynomial that intersects four of the basis points. Repeating this procedure will terminate with a polynomial D intersecting all of the points P_i, and if they sum to zero, no other points. If at any level the number of points is odd, it is sufficient to carry the point and its associated polynomial up to the next level. This procedure performs a linear number of polynomial multiplication of degree at most linear in the input, so the procedure is quadratic up to a polylogarithmic factor. In reality, the procedure is quasilinear since the number of multiplications and the length of the polynomials to be multiplied maintain a fixed ratio. In practice, the most efficient algorithm for polynomial multiplication will depend on the underlying field and the number of points.

3.1.2 Mumford Representation and Half-GCD

To find D interpolating P_i, it is sufficient to start with the Mumford representation of the divisor: two polynomials $u(x(P_i)) = 0$ and $v(x(P_i)) = y(P_i)$. For points of higher multiplicity, the second property generalizes to $u(x) \mid v(x)^2 - (x^3 + Ax + B)$. The polynomial $u(x)$ is easy to construct, and the polynomial $v(x)$ can be constructed via a combination of Hermite interpolation and Hensel lifting in quasilinear time. To find $a(x)$ and $b(x)$ one can use the extended Euclidean algorithm for polynomials or a more efficient half-GCD [12] algorithm to compute a small solution to

$$c(x)u(x) + b(x)v(x) = a(x)$$

From which one can extract $a(x)$ and $b(x)$ to form D. This solution is valid because, taking the above equation $\mod u(x)$

$$a(x) \equiv b(x)v(x) \mod u(x)$$

Squaring both sides and substituting $x^3 + Ax + B$ for $v(x)^2$ using the second property of the Mumford representation

$$a(x)^2 - (x^3 + Ax + B)b(x)^2 \equiv 0 \mod u(x)$$

If the $a(x)$ and $b(x)$ meet the degree constraints, then the degree of the left hand side of the equivalence is equal to that of $u(x)$, and thus it must be a constant multiple. One can equivalently consider the problem in terms of trying to find a short rational representation of $v(x) \mod u(x)$ since $v(x) \equiv a(x)/b(x) \mod u(x)$.

If the collection of points includes any pairs of a point and its inverse, it is important to choose whichever point or its inverse has higher multiplicity, or neither if they have equal multiplicity, for the interpolation step, as $v(x(\pm P_i))$ can only have one value. The $(x - x(P))^m$ factor can be multiplied back onto both $a(x)$ and $b(x)$ after running the previous algorithm.

3.2 Proving Interpolation

Once the divisor witness D is constructed, the prover must somehow show that it actually interpolates the correct points. One natural way to do this would be to consider the divisor norm, choose a random P, and verify that the norm equals the product of the point factors. Correctness would follow from the Schwartz-Zippel lemma.

$$D(P)D(-P) = \prod_{i=1}^{n} (x(P) - x(P_i))$$

Unfortunately, this cannot distinguish D that interpolates P_i from D that interpolates $-P_i$. The prover could instead try to generalize the norm construction and find the norm with respect to y, since the curve can just as well be defined as a as a cubic extension of the rational function field $\mathbb{F}(y)$ by x by writing the defining equation as $x^3 + Ax + (B - y^2) = 0$. Unfortunately, this has a similar problem in that it cannot distinguish points that share an x coordinate. For non-supersingular curves with $A = 0$ all points share and x coordinate with 2 other points.

In both instances, the issue is that the prover can tell ahead of time what points will map to the same roots of the norm. If the prover could generalize the norm, and compute it with respect to an arbitrary function of the elliptic curve Z, rather than just x or y, the verifier would be able to select such a function at random. The prover would not be able to anticipate other points where the norm evaluates to the same result, with overwhelming probability, and so the proof would be sound. Such a generalized norm can be given in terms of the elliptic resultant, where $\mathrm{lc}(D)$ is the coefficient of the highest degree term where the weight of x is 2 and y is 3.

$$N_Z(D, z) = \mathrm{Res}_E(D, Z - z) \qquad D, Z \in \mathbb{F}[E] \quad z \in \mathbb{F}$$

$$\begin{aligned}\mathrm{Res}_E(D_0, D_1) &= \mathrm{lc}(D_1)^{n(D_0)} \prod_{D_1(P)=0} D_0(P)^{v_P(D_1)} \\ &= (-1)^{n(D_0)n(D_1)}\mathrm{lc}(D_0)^{n(D_1)} \prod_{D_0(P)=0} D_1(P)^{v_P(D_0)}\end{aligned}$$

Setting $Z = x$ or $Z = y$ and $z = Z(P)$ for the challenge point P gives the x and y norms respectively. The fact that the resultant, which can be defined in terms of the polynomial resultant for the purposes of computation, is expressible as a product over the divisors of both curves is actually an expression of a deeper statement about elliptic curves and is equivalent to a limited form of Weil reciprocity for polynomials with disjoint support everywhere except the point at infinity. To express using the polynomial resultant

$$\mathrm{Res}_E(a_0(x) - b_0(x)y, a_1(x) - b_1(x)y) = \mathrm{Res}_x\left(a_0(x) - \frac{a_1(x)}{b_1(x)}b_0(x), N(a_1(x) - b_1(x)y)\right)$$

For any choice of $Z \in \mathbb{F}[E]$ the above norm is actually a polynomial in z. The right hand side is clearly, as it is a product over $(Z - z)(Q)$ for $D(Q) = 0$. The left hand side is also, since the product is symmetric in the points of $\mathrm{div}(Z - z)$, and so is symmetric in the roots of the x norm of $Z - z$. By elementary properties of symmetric functions, the coefficients of the norm of $Z - z$, which are polynomials in z, are sufficient to represent any symmetric function in the roots of the norm of Z.

Thankfully, the general case is avoidable so long as the verifier chooses a Z and z such that $\mathrm{div}(Z - z) \subset E/\mathbb{F}$, i.e. does not have points in an extension. In that case, it is sufficient to evaluate D at the roots directly. For simplicity, the verifier will select two points A_0 and A_1 and let $A_2 = -(A_0 + A_1)$ and the line $y - \lambda x - \mu$ interpolate these points. Setting $Z = y - \lambda x$ and $z = \mu$ yields the desired norm. The prover would then show, assuming the leading coefficient of D is 1

$$D(A_0)D(A_1)D(A_2) = \prod_{i=1}^{n}(\mu - Z(P_i))$$

The prover can do better. Since both expressions of the elliptic resultant are polynomials in z, the prover can take the logarithmic derivative of both sides of the equation. This requires some care when handling the z derivative of A_i. These points depend on z, so their z derivatives are not merely free parameters. Differentiating the equations $Z(A_i) = z$ and substituting $dx/dy = (3x^2 + A)/(2y)$, the prover finds the solutions

$$\left(\frac{3x(A_i)^2 + A}{2y(A_i)} - \lambda\right)\frac{dx(A_i)}{dz} = 1$$

To check that these derivatives are valid, it is sufficient to verify that the z derivative of the additional expression for λ^2 holds.

$$\lambda^2 = x(A_0) + x(A_1) + x(A_2) \qquad 0 = \sum_{i=0}^{2}\frac{dx(A_i)}{dz}$$

This is left as an exercise for the reader. Substituting these derivatives into the logarithmic derivative of the product equation, the prover can show

$$D' = \frac{d}{dx}D = a'(x) - \frac{3x^2 + A}{2y}b(x) - yb'(x)$$

$$\sum_{i=0}^{2} \frac{D'(A_i)}{D(A_i)} \frac{dx(A_i)}{dz} = \sum_{i=1}^{n} \frac{1}{\mu - Z(P_i)} \tag{1}$$

3.2.1 Correctness

In an instantiation of this zero knowledge proof protocol, the prover will commit to the points and divisors and then the verifier will choose the challenge parameters λ and μ. There are three ways that a malicious prover could make the verifier accept an invalid witness: by making one of the denominators zero, by having any of the points in the divisor and any of the points P_i share a z coordinate while still being distinct, or by making the combined numerator zero. The number of points is negligible by assumption and the prover will show that D is not identically zero by normalizing so at least one coefficient is one.

The first kind of invalid proof has negligible probability because the probability that one of the challenge points is a zero of the divisor is at most $3\deg(N(D))/\#E$ and the probability that one of the P_i lies on the line $\mu - y + \lambda x$ is also negligible, by the following reasoning. There are at most $\#E - 1$ lines passing through a fixed $\mathbb{F}^2$ point and an elliptic curve point, one for each curve point except the point at infinity. To select the line, the verifier chooses a curve point and some other point not equal to the original point or its inverse. Since there are three ways to choose each such line, there are

$$\frac{(\#E - 1)(\#E - 3)}{6}$$

Possible lines, one of which is chosen uniformly at random. The probability that this line is one of the lines that passes through P_i is at most

$$\frac{6(\#E - 1)}{(\#E - 1)(\#E - 3)} = \frac{6}{\#E - 3}$$

Which is negligible. The probability of z coordinate collisions is also negligible since each pair of points, one chosen from the divisor $D(Q) = 0$ and one chosen from P_i, have a fixed slope, which is precisely the value of λ where their z coordinates are the same

$$\lambda(Q, P_i) = \frac{y(Q) - y(P_i)}{x(Q) - x(P_i)}$$

The number of lines with a particular value for λ is at most $p = \#\mathbb{F}$ or maximum number of μ, so the probability of selecting one of these lines is p divided by the number of total lines, which was determined earlier to be quadratic in $\#E$. By the Hasse bound, $\#E \geq (\sqrt{p} - 1)^2$. So, the probability of choosing a particular λ is

$$\frac{6p}{(\#E - 1)(\#E - 3)} = \frac{6p}{(\#E)^2 - 4\#E + 3} \leq \frac{6p}{(\sqrt{p} - 1)^4 - 4(\sqrt{p} - 1)^2 + 3} = O\left(\frac{1}{p}\right)$$

Which is negligible. There are at most $n\deg(N(D))$ distinct slopes between the various Q and P_i, thus the probability of picking one of these slopes is negligible out of all the possible slopes is also negligible.

The final case, when the combined numerator is zero, is a straightforward application of the Schwartz-Zippel lemma and is zero with probability proportional to the sum of the degrees. This is also negligible, and so the overall probability of the verifier accepting an invalid witness is negligible.

3.3 Higher Multiplicity Challenge

If $A_0 = A_1$ then the divisor evaluations at this point would be the same and the trace would only need two terms. Unfortunately, the previous method fails on points of higher multiplicity because

$$\frac{dz}{dx} = \frac{dy}{dx} - \lambda$$

Is zero when the tangent slope is λ, which is exactly the case at points of higher multiplicity. This causes the z derivatives of the higher multiplicity point to diverge. To avoid the singularity, the prover can treat λ as a formal variable with non zero z derivative. Expanding, the new relation between the derivatives becomes

$$\frac{dx}{dz}\frac{dy}{dx} - x\frac{d\lambda}{dz} - \lambda\frac{dx}{dz} = 1$$

Which, when evaluated at A_0 where λ equals the tangent, gives the necessary value of the λ derivative

$$x(A_0)\frac{d\lambda}{dz} = -1$$

To translate this back to the logarithmic derivative equation, the prover must modify the numerators of the point reciprocal terms to account for the non-zero λ derivative and account for the fact that differentiating $x(A_i)$ with respect to z depends on λ

$$\sum_{D(P)=0} \frac{1 + x(P)\frac{d\lambda}{dz}}{z - y(P) + \lambda x(P)} = \sum_{i=0}^{2} \frac{D'(A_i)}{D(A_i)}\frac{dx(A_i)}{dz}$$

For A_2, the prover can find the z derivative by differentiating the line equation and solving

$$\frac{dx(A_2)}{dz} = \frac{1 + x(A_2)\frac{d\lambda}{dz}}{\frac{dy}{dx}\big|_{A_2} - \lambda} = \frac{1}{x(A_0)}\frac{2y(A_2)(x(A_0) - x(A_2))}{3x(A_2)^2 + A - 2\lambda y(A_2)}$$

This does not work for the derivative of $x(A_0)$ as both the numerator and denominator vanish. The prover can work around this by instead differentiating the equation $\lambda^2 = 2x(A_0) + x(A_2)$ to find

$$\frac{dx(A_0)}{dz} = \lambda\frac{d\lambda}{dz} - \frac{1}{2}\frac{dx(A_2)}{dz}$$

Substituting all the z derivatives appropriately, the prover will find that the equation holds at $z = \mu$

$$c_2 = \frac{2y(A_2)(x(A_0) - x(A_2))}{3x(A_2)^2 + A - 2\lambda y(A_2)}$$

$$\sum_{D(P)=0} \frac{x(A_0) - x(P)}{\mu - y(P) + \lambda x(P)} = c_2\frac{D'(A_2)}{D(A_2)} - (c_2 + 2\lambda)\frac{D'(A_0)}{D(A_0)} \tag{2}$$

Note that setting the λ derivative to be non-zero in the case of distinct $A_0 \neq A_1$ also works. The point derivatives need to be modified, but there is a remaining degree of freedom. This allows the prover to set one of the point derivatives to be zero and achieve the same number of evaluations as the $A_0 = A_1$ case. The tradeoff, as in the higher multiplicity case, is that the numerators of the point reciprocals are no longer 1.

This procedure also generalizes to larger divisors. While it is not relevant for the protocols of this paper, to generalize one would take the norm of the larger divisor and then differentiate the coefficients. Writing each coefficient as a symmetric function of the point coordinates will produce a system of n equations in $2n - 1$ unknowns, the derivatives of the coefficients and the derivatives of the points. The procedure here implicitly normalized one of the coefficient derivatives to 1 by differentiating with respect to z. Since the equations are symmetric in the x coordinates, any points of higher multiplicity reduce the rank of the system.

3.3.1 Correctness

In the case where $A_0 = A_1$, it remains the case that the probability of one of the challenge points being a zero of the divisor is negligible. The probability that one of the P_i lies on the line is slightly different, but also still negligible. All the tangent lines that can pass through a particular point are must satisfy

$$\lambda(Q, P_i) = \frac{y(Q) - y(P_i)}{x(Q) - x(P_i)} = \frac{3x(Q)^2 + A}{2y(Q)}$$

$$2y(Q)(y(Q) - y(P_i)) = (3x(Q)^2 + A)(x(Q) - x(P_i))$$

$$-2y(Q)y(P_i) = x(Q)^3 - Ax(Q) - 2B - (3x(Q)^2 + A)x(P_i)$$
$$4y(P_i)^2(x(Q)^3 + Ax(Q) + B) = (x(Q)^3 - Ax(Q) - 2B - (3x(Q)^2 + A)x(P_i))^2$$

Since this is a degree six polynomial in $x(Q)$, the point P_i can lie on the tangent lines of at most 6 curve points. The probability of choosing one of these curve points for one of the committed P_i is at most $6n/(\#E - 1)$, which is negligible. The probability that the z coordinates of one one of the divisor points and one of the P_i are equal is also still negligible because each λ is slope of at most 4 tangents

$$(3x^2 + A)^2 = (2\lambda)^2(x^3 + Ax + B)$$

Therefore, choosing A_0 at random will produce a particular λ with probability at most $4/(\#E - 1)$, and $4n \deg(D)/(\#E - 1)$ is negligible. Since the sum of negligible functions is negligible, the prover can make the verifier accept an invalid witness with negligible probability.

3.4 Protocol

To show that a principal divisor proof is satisfied, the prover will commit to the coefficients of the normalized principal divisor witness $a(x)$ and $b(x)$, the coordinates of all the points, and the quotients for each term following the verifier's choice of A_0. The divisor witness should be normalized so that the lowest order coefficient is one, i.e. $a(0) = 1$. This will make representing certain divisors impossible, but the probability of selecting one of them is negligible. If this is unacceptable, the prover can normalize the highest order coefficient to be 1. The advantage of using the lower order coefficient is that divisors with fewer points are still representable.

Following the selection of A_0 and commitment to the evaluation of the witnesses and reciprocals, the prover can use any proof system for arithmetic circuits to show that the constraints are satisfied. The proof is quite simple, each numerator and denominator of each rational function is expressible as a linear combination of the committed values, in the cases where either the multiplicities or the points are public. In the case both are private, the prover will need two multiplications per basis point. The rest of the proof consists of checking that all the evaluations sum to zero. The exact costs of multiplications versus summations will vary by proof system.

4 Proving Elliptic Curve Inner Products

Suppose that the prover wishes to show in that for points P_i and scalars e_i that some other point Q is the linear combination

$$Q = \sum_{i=1}^{n} e_i P_i$$

If the prover were to compute this inner product, it would be possible to use Shamir's trick to combine all the doubling operations for all the scalar multiplications. That is, break down each e_i into bits, sum all the bits in each "row" and add twice the result of the previous round. For this protocol, rather than using base 2 it is more convenient to use a negative base like -2 or even -3 with digits $d_{ij} \in \{-1, 0, 1\}$ so that all the intermediate results have the same sign. In that case

$$e_i = \sum_{j=0}^{l} d_{ij}(-3)^j$$

$$Q_j = -3Q_{j+1} + \sum_{i=1}^{n} d_{ij} P_i \qquad Q_l = 0$$

With the final sum given by $Q = Q_0$. The prover can prove each of these row computations using the divisor technique of the previous section. What's more, since each of the rows takes the result from the previous row and multiplies it by three, it is possible to sum the trace operations in just the right way to make all the intermediate points cancel and sum up the multiplicities on the reciprocals. Letting the function $T(P)$ vary based on whether $A_0 = A_1$

$$T(P) = \frac{1}{\mu - Z(P)} \begin{cases} 1 & \text{if } A_0 \neq A_1 \\ x(A_0) - x(P) & \text{if } A_0 = A_1 \end{cases}$$

As well as using the appropriate logarithmic derivatives depending on whether $A_0 = A_1$. In the following, I will write $\mathrm{Tr}_Z(F)$ to denote the sum of F over the points of $\mathrm{div}(Z)$ in the same way the norm is the product over these points.

$$\sum_{j=0}^{l}(-3)^j \mathrm{Tr}_Z\left(\frac{d}{dz}\log D_j\right) = T(Q) + \sum_{i=1}^{n} a_i T(P_i) + b_i T(-P_i) \tag{3}$$

Where a_i is all the 1 digits of e_i and $-b_i$ is all the -1 digits of the scalars, so that $e_i = a_i - b_i$. As written, this technique can be unsound if it is possible for either a_i or b_i to be divisible by the field characteristic but not be zero. To avoid this, the prover can choose scalars such that it is impossible by construction for this to occur, for example if the scalars are drawn from a small subset of the field. If this is not possible, the prover can alternatively split the scalars in half by digits and sum both halves using this technique independently treating the output from the upper half as a new basis point with scalar multiplication by 1. Finally, sometimes the prover can a larger basis of points to split all the scalars into smaller scalars that cannot be divisible by the field characteristic, for example using complex multiplication.

4.1 Zero Knowledge in Scalars

The prior case allows proving that a public linear combination of points another point in zero knowledge with respect to the point. The structure of the protocol is flexible enough to be made zero knowledge in the scalars as well, potentially using public points if required. The multiplicities for each basis point can be committed within the witness and incorporated into the reciprocals in zero knowledge. So long as the divisors are not capable or expressing multiplicities that exceed the field characteristic, soundness will follow.

There are some additional caveats in the private scalar case. To avoid leaking any information about the structure of the scalars, the prover will need to allocate enough space for the divisor witnesses so that each digit could be non-zero, even if it is not. There may be ways to improve on this, elaborated in the improvements section.

In the case that the points are public and the scalars are secret, there is another optimization available to the prover, since the denominators of the point reciprocals do not need to be computed in zero knowledge. The verifier can perform the necessary divisions directly over the public points directly, reducing the complexity of the proof. Unfortunately, the evaluation of the divisor witnesses must still be done in zero knowledge so the extra round is still necessary. This does change the optimal structure of the proof, especially in the non-CM case since the scalars can just be split directly into lower and higher order bits and the prover can a large multiple of the basis points computed publicly.

4.2 Correctness

The correctness of the equality of the weighted sum of divisor logarithms and reciprocal terms follows straightforwardly from adding up the terms of each separate divisor equation, as well as via simplifying the sum of logarithms to be the logarithm of the product of the exponentiated divisor

$$\sum_{j=0}^{l}(-3)^j \mathrm{Tr}_Z\left(\frac{d}{dz}\log D_j\right) = \mathrm{Tr}_Z\left(\frac{d}{dz}\log \prod_{j=0}^{l} D_j^{(-3)^j}\right)$$

Note that it is possible for the row polynomials to have roots that are not from among the basis points but cancel in the final sum. For example, if divisor D_i has point X with multiplicity 3 and divisor D_{i+1} has point X with multiplicity 1 then the points will cancel.

If the multiplicities actually sum to 0 as integers, then the point X will not affect the value of the sum. However, the proof can only measure whether the multiplicity of X sums to 0 $\mod p$. Non-zero integer multiplicities that are divisible by p will cause the reciprocal to vanish in the verification equation but will not cause the point X to vanish in the sum. For this reason, the prover must ensure that it is not possible for the combined inner product divisor to represent multiplicities that are divisible by p other than 0. When this is the case, all the points whose reciprocals vanish must also net to zero in the sum.

4.3 Complex Multiplication

The number of elliptic curves used in practice is fairly small, and many of those curves have complex multiplication where either $B = 0$ or $A = 0$. In those cases, for example SECP256k1 the curve used by Bitcoin or BLS381 the pairing friendly curve used in ZCash, there is a nontrivial endomorphism of the curve group expressible as a linear function of the coordinates of the points. For $B = 0$ the endomorphism is $(x, y) \mapsto (-x, iy)$ where i is a primitive fourth root of unity, and for $A = 0$ it is $(x, y) \mapsto (\omega x, y)$ where ω is a primitive third root of unity. It turns out that since these operations have order 4 and 3 respectively and since they respect the group operation of the elliptic curve, they typically correspond to multiplication by a fourth or third root of unity in the scalar field of the curve.

That is, in the $A = 0$ case, if $E : y^2 = x^3 + B$, $\#E = q$, $\#\mathbb{F} = p$ and $p = 1 \mod 3$ then given suitable $\omega_p \in \mathbb{F}$ and $\omega_q \in \mathbb{F}_q$ both primitive third roots of unity $x(\omega_q P) = \omega_p x(P)$ and $y(\omega_q P) = y(P)$. This property has been used [13] to reduce the complexity of elliptic curve scalar multiplication by writing the scalar $e \in \mathbb{F}_q$ as a linear combination of two smaller values $e = u + \omega_q v$. This transformation can be efficiently computed given the decomposition $q = N(a + \omega b) = a^2 - ab + b^2$ over the integers, which also allows canonically defining the third root of unity as $\omega_q = -a/b$.

4.4 Inner Product Proofs using CM

Given the inner product $\sum_{i=1}^{n} e_i P_i = Q$ to prove, the prover can first split each of the scalars into $e_i = u_i + \omega_q v_i$ and then increase the number of scalar multiplications by a factor of two, including both P_i and $\omega_q P_i$ to show

$$\sum_{i=1}^{n} u_i P_i + v_i(\omega_q P_i) = Q$$

Using the base -3 digits as in the previous section, most of the digit patterns occur as "conjugate" points given by $(-\omega_q)^i P$. Specifically, all the values except $\pm(1 - \omega_q)$, which can be computed using two additions of $(-\omega_q)^i$ multiples. To represent these points in the inner product, the prover can construct lines interpolating them as a linear combination of existing basis elements. Since the points are negations of each other, it is sufficient to do this once.

$$B_i = y - \lambda_i x - \mu_i \qquad \operatorname{div}(B_i) = P_i + (-\omega_q P_i) + (\omega_q P_i - P_i) - 3O$$

Then, the multiplicities for $(1 - \omega_q)P_i$ and $(\omega_q - 1)P_i$ will scale the traces of $B_i(x, y)$ and $B_i(x, -y)$ and be subtracted from the basis present in the respective lines.

$$\begin{aligned}\sum_{j=0}^{l/2}(-3)^j \mathrm{Tr}_Z\left(\frac{d}{dz}\log D_j\right) = \sum_{i=1}^{n} a_{i6}\mathrm{Tr}_Z\left(\frac{d}{dz}\log B_i(x,y)\right) + a_{i7}\mathrm{Tr}_Z\left(\frac{d}{dz}\log B_i(x,-y)\right)\\ + (a_{i0} - a_{i6})T(P_i) + (a_{i1} - a_{i6})T(-\omega_q P_i) + a_{i2}T(\omega_q^2 P_i)\\ + (a_{i3} - a_{i7})T(-P_i) + (a_{i4} - a_{i7})T(\omega_q P_i) + a_{i5}T(-\omega_q^2 P_i)\end{aligned}$$

Here the multiplicities a_{ik} for $k = 0..7$ are defined in the same manner as the a_i and b_i from the original inner product proof but for the corresponding symbol values. Since each of the points can be computed as a linear combination of the coordinates of the points P_i or the divisor B_i, the prover only needs to commit to these values. Using this technique also eliminates the potential for any of these multiplicities to be divisible by the characteristic as they are at most half the length.

4.4.1 Witness Size

For n public scalar, private point multiplications using base -3 in a curve with $j(E) = 0$, the size of the witness can be estimated subject to the assumption that each digit is chosen uniformly at random. For most practical cases, this is reasonable since the values themselves are drawn uniformly at random or are some function of uniformly random values. For these calculations to hold, especially as $n \to \infty$, it is only necessary that the weaker form of the digit distribution hold: that the expected number of non-zero digits in each row is at most $8n/9$ and that the variance goes to zero, which is true.

For a 256 bit field, the number of base -3 digits after splitting via the endomorphism is $(\log_3 2^{128}) \approx 81$ and therefore the number of non-zero digits per scalar is 72. There are 4 scalars per point necessary for the basis, two for the coordinates of the point and 2 for the B_i line. Each row requires 2 committed values for the evaluations at A_0 and A_2, and each point requires 6 evaluations at the conjugate points and

2 for each of the non-conjugate points, for a total of 10. Split into three rounds, one for the information independent of the linear combination, one for the divisor witnesses, and one for the evaluations, the round witness sizes (w_0, w_1, w_2) and number of multiplications m for n points are

$$w_0 = 4n \qquad w_1 = 243 + 72n \qquad w_2 = m = 162 + 10n$$

$$w = w_0 + w_1 + w_2 = 405 + 86n$$

4.4.2 Zero Knowledge in Scalars

This technique also translates naturally to the private scalar cases. In fact, when the points are public the R_i points can simply be computed in public so it is not necessary to commit to B_i or their evaluations. However, the prover must commit to the multiplicities for all the basis points, so the proof cost per scalar multiplication roughly balances out. Rather than splitting the scalars using ω_q and having $8 = 3^2 - 1$ basis points, in the public point case the prover can split the scalars into a linear combination of the $3^0, (-3)^{53}, (-3)^{106}$ each of 54 digits and use $3^3 - 1 = 26$ basis points per scalar multiplication. This breaks down as $26 + 54 = 80$ scalars per scalar multiplication plus the cost of tripling each row. In the case where both points and scalars are private, this is not possible and the prover will just use private point technique and commit to the private scalars.

4.5 Protocol

The ECIP protocol is essentially just the parallel composition of the divisor protocol for each row. All the divisor witnesses and points will be committed, the verifier will choose the challenge point, and the prover will commit to the evaluations. As already described, the multiplicities for the same point in different rows will be summed so that each point needs only to be evaluated once. The resulting arithmetic circuit has essentially the same structure as the individual divisor proofs, where each rational function is verified and all the evaluations are summed and the result is verified to be zero.

5 Applications

The proofs of this paper achieve significantly better performance than other proof systems for elliptic curve operations. State of the art operationally oriented proof systems, i.e. proof systems that translate elliptic curve operations into the proof system directly, achieve approximately 6 [14] multiplications for secret points and 2 [14] for secret scalars. Estimating the number of scalars in the witness is challenging without fixing a particular proof system, but it is seems to be about 10 per bit of the input. The proofs of this paper can achieve substantially fewer, although exactly how many fewer depends on the context.

In the best case, using $j(E) = 0$ CM, the proof of this paper require only $72 + 10 + 4 = 84$ scalars per public 256 bit scalar on average, for approximately $1/3$ scalar per bit of input, plus a constant overhead of 243 scalars for the spine, and 10 multiplications in total for each scalar, or $5/128 \approx 1/25$ multiplications per bit of input plus a constant 162. As the number of scalars increases, asymptotically this is about an order of magnitude reduction in witness size and a reduction of two orders of magnitude in the number of multiplications.

This proof system also features, effectively, a built in permutation argument. The argument shares certain structural properties with the permutation argument of Bulletproofs++ [1], specifically logarithmically differentiating a polynomial to produce a sum of "reciprocals." This is essential to the savings in witness size and number of multiplications, but also grants significantly more flexibility in structuring the proof. It is difficult to compare this with operational alternatives without specifying more of the proof system, and in practice the prover would probably either use the permutation argument of Bulletproofs++ or would design the protocol so that a permutation argument was not necessary.

There is a significant limitation to this proof system: it requires an interaction between the prover and verifier to evaluate the divisor witnesses. Since this involves division, it cannot be represented directly using a polynomial commitment scheme which makes it awkward to integrate into many existing proof systems. For example, integrating into SNARKs would require, in all likelihood, an additional published commitment, some sort of recursive proof composition, or a way to generate the challenges inside the SNARK. On the other hand, Bulletproof based protocols should not have much trouble with an additional round, and since they tend to be larger the relative increase in proof size should be minimal. When possible, the improvements in witness size and number of multiplications make this approach clearly preferable, but in order to integrate it many existing proof systems will likely need to be modified.

5.1 Pedersen Hashing

ZCash uses an elliptic curve called JubJub, an elliptic curve defined over the scalar field of the underlying elliptic curve used by the SNARKs, for hashing. Within ZCash, the hash takes a string of bits and, for a fixed set of points, computes

$$h(\mathbf{b}) = \sum_i b_i G_i$$

This is highly amenable to the proof techniques of this paper. One way to encode this using a principal divisor proof would construct a divisor of degree equal to the length of the bit string, commit to the resulting sum and then use the binary digits as the multiplicities for the basis points G_i. This would require one multiplication per bit and two multiplications to evaluate the divisor. Each multiplication requires committing to one scalar.

This proof system also immediately extends to proving multiple hashes at the same time by taking a random linear combination of the divisors, and a random linear combination of the bits. Each additional hash adds only two multiplications and one scalar. Unfortunately, this proof system cannot be directly used within SNARKs without some modifications to the enclosing proof system. A variant that did not take the logarithmic derivative might be more feasible.

5.2 Recursive Proof Composition

There has been an increasing interest in recursive proof composition, specifically using amicable primes [5], also called 2-cycles of primes, for an elliptic curve. These primes p and q for an elliptic curve E satisfy

$$\#E(\mathbb{F}_p) = q \qquad \#E(\mathbb{F}_q) = p$$

Recursive SNARKs [6] use a SNARK over each curve, both of which must be pairing friendly, while Halo [8] does not require pairings but is not succinct. The reason amicable curves are used for such proof systems is that it is especially efficient to prove arithmetic operations about the base field of one curve in the other. More specifically, it is efficient to prove statements about elliptic curve operations over each curve in the other.

Existing techniques [14] have encoded elliptic curve computations directly into the proof system. Using the techniques of this paper it is possible to achieve much smaller witness sizes and enormously smaller multiplication counts than an equivalent proof of elliptic curve computations. While operational proofs measure the number of multiplications as a multiple of total mixed additions, these proofs have two to six, depending on CM, multiplications *per point* and two multiplications per digit. Conservatively this is nearly two orders of magnitude fewer multiplications.

5.3 Multiplicative Group of a Finite Field

These techniques can be generalized from elliptic curve point additions to also prove multiplication of field elements. The prover will substitute the group of principal divisors of the elliptic curve with the group of polynomials with leading and constant coefficient equal to 1. Each such polynomial encodes a multiset of field elements that multiply to 1.

$$f(x) = \sum_{i=0}^{n} c_i x^i = \prod_{i=1}^{n} (x + r_i) \qquad c_n = c_0 = 1$$

Taking the logarithmic derivative of such a polynomial and evaluating the result at a random point gives a sum of reciprocals at the roots in the same manner as a divisor witness for an elliptic curve. The prover can take a linear combination of these rational functions structured to encode an ECIP using Shamir's trick, or any other vector addition sequence, which will cancel the intermediate values in the same manner. To compute the "inverse" of a divisor, the prover can reverse the order of the coefficients in the polynomial

$$\sum_{i=0}^{n} c_{n-i} x^i = x^n f(1/x) = \prod_{i=1}^{n} (x + 1/r_i)$$

Since there is no analogous operation to complex multiplication in a prime field, it is especially important to make sure that the multiplicities cannot represent a value divisible by the field characteristic. This means the prover will likely need to split the ECIP into halves, with the lower and higher order

digits treated separately. In the case of a prime power field, it would be possible to use the frobenius endomorphism in a way analogous to complex multiplication, although representing multiplication of these field elements would likely pose other challenges.

The advantage of this technique over directly encoding of the computation is smaller than the advantage in the elliptic curve version. However, the number of multiplications is still about two orders of magnitude smaller and the built in permutation argument remains useful. Generally, transforming circuits to have fewer multiplications in favor of larger linear combinations produces more efficient zero knowledge proofs, but this will depend on the underlying proof system.

6 Improvements

6.1 Shorter Proofs for Private Scalars

The private scalar proof presented earlier uses one scalar per digit of the private scalar multiple. Since the ECIP proof is zero knowledge with respect to which digits in each row are set, only requiring publication of the maximum total number of non-zero digits, for a large number of private scalars the prover can do better. Assuming each digit is uniformly random, the prover can determine the distribution of possible non-zero digits for each row. Setting some threshold such that the probability of the scalars producing a row of length greater than the threshold is acceptably low, the prover can use rows of length equal to this threshold.

To smooth out results between rows, the prover can compensate for particularly dense rows by moving some of the additions from one row into another. In base -3 this is challenging since it would require 3 more scalars in the lower order row for each addition in the higher row, but other bases or digit representations offer more flexibility. For example, using base -2 non-adjacent form the prover can elect to insert adjacent digits to smooth out the number of digits per row.

At a higher level, if the scalars are distributed randomly according to a challenge in the proof, and not fixed externally, the prover can abort proving if the rows cannot fit and retry with a new challenge. This might occur if the prover wants to take a linear combination of commitments and then open the result entirely in zero knowledge. There are significant implications for information leakage if the prover can take a variable amount of time dependent on the data, so instead the prover could attempt to construct multiple proofs in parallel with different challenges. By choosing the thresholds and number of parallel trials carefully, the probability of all the proofs failing can be made negligible.

The performance implications of this may actually be relatively small, since determining if any of the rows for a particular challenge are unacceptably dense likely does not require any elliptic curve computations. To preserve soundness, the protocol will need to account for the fact that the prover gets multiple choices of challenge value. However, since the probability of dense rows decreases exponentially in the number of choices, adding more challenge candidates to a well designed protocol should not give the prover a non-negligible advantage in constructing an unsound proof.

6.2 Common Digits

In the ECIP circuit, the prover can factor out common sets of digits to reduce the witness size. For simplicity, each digit can be treated as one of nine values and the similarity of rows can be defined as the number of places where two rows agree and they are not zero

$$\mathrm{sim}(a, b) = \mathrm{len}(a : d_{aj} = d_{bj}, d_{aj} \neq 0)$$

The prover can factor out common digit sets from multiple rows into new divisor proofs. This will increase the number of multiplications, but has the potential to reduce the number of witness elements. Computing the optimal refactoring of rows is probably too difficult, but considering pairwise relations and refactoring all the common digits from rows of similarity greater than some threshold is likely sufficient to reduce the witness size.

The procedure can also be generalized to more rows simultaneously. Triples or perhaps even quadruples of rows may be worth considering for a sufficiently large number of points, although since the number of digits is fixed configurations in arbitrary numbers of rows are extremely unlikely to ever occur. To handle commonality among a large number of rows, this approach could also be applied to pairs of basis points instead. It may be possible to use these techniques in zero knowledge with respect to the scalars, using the techniques of the previous section.

6.3 Vector Addition Chains

An addition chain [2] is a sequence of variables, each defined as the sum of two previous variables with the initial variable is canonically set to 1. A vector addition chain is a generalization of a regular addition chain where each variable represents a vector and the first n represent linearly independent basis vectors. Both are special cases of straight line programs , which can be defined over any group, over addition of scalars or vectors respectively without inversion. When inverses are allowed, the chain is sometimes referred to as an addition subtraction chain.

Shamir's trick produces a vector addition chain, as will any other algorithms for computing ECIPs. General addition chains have been investigated for their ability to speed up RSA as well as elliptic curve inner products/multiexponentiations. Computing the optimal addition chain for a given vector is NP complete, but it may be sufficient to find an approximate solution if it produces a smaller circuit.

The methods of this paper for proving ECIPs using Shamir's trick generalize to arbitrary addition chains, so it may be possible to achieve smaller proof sizes using existing techniques for finding addition chains. Intermediate points can be cancelled by summing the divisor proofs, although for a general addition chain this is much more difficult to do in a way that preserves the zero knowledge property with respect to secret scalars. Alternatively, all intermediate points could remain and the permutation argument inside the divisor proofs would make it trivial to preserve the zero knowledge with respect to scalars, but this would likely increase proof size far beyond the Shamir's trick version.

When encoding an addition chain into a zero knowledge proof, it is more efficient to generalize each new variable to be an arbitrary length sum of earlier variables, and eliminate variables that occur only once. Equivalently, a vector addition chain can be represented as a matrix M such that

$$M \in \mathbb{Z}^{N \times N+n} \qquad M_{ij} = 0 \text{ if } j + n > i \qquad M_{ij} = -1 \text{ if } j + n = i$$

Note that this allows subtractions, as they are cheap for elliptic curves. Each row of the matrix can be proven using a single divisor witness, and intermediate variables can be eliminated using the same technique as the ECIP proof previous described. If the sign of the variable in its defining row and the sign of the variable in future rows do not agree, then it will be necessary for the prover to evaluate its associated divisor twice. This is transitive, so any variable that references a variable transitively with a different sign will require the defining divisor to be evaluated twice. This is essentially the same principle as the B_i polynomials and is the reason why negative bases are actually preferable in the Shamir's trick version.

Encoding this matrix into a zero knowledge proof, each row corresponds to a divisor witness and will require two multiplications to evaluate if only one sign is used or four if both signs are used. Each divisor witness will use a number of scalars equal to the "weight" of the row or the sum of the absolute value of the entries. The weight of the matrix is then

$$w(M) = \sum_{i<j} |M_{ij}|$$

Incorporating complex multiplication works the same way as before, but increases the number of times intermediate divisors must be evaluated to six in general. This can be avoided if the "diagonal" entries of M are set to 1 and all the non-basis entries of M are restricted to positive integers. In that case, only the basis points need to be evaluated six times.

References

[1] Liam Eagen. *Bulletproofs++*. Cryptology ePrint Archive, Report 2022/510. https://ia.cr/2022/510. 2022.

[2] Jurjen Bos and Matthijs Coster. "Addition chain heuristics". In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 400–407.

[3] Daira Hopwood et al. "Zcash protocol specification". In: *GitHub: San Francisco, CA, USA* (2016), p. 68.

[4] Daira Hopwood et al. "Zcash protocol specification". In: *GitHub: San Francisco, CA, USA* (2016).

[5] Joseph H. Silverman and Katherine E. Stange. "Amicable Pairs and Aliquot Cycles for Elliptic Curves". In: *Experimental Mathematics* 20.3 (2011), pp. 329–357. DOI: 10.1080/10586458.2011.565253. URL: https://doi.org/10.1080%2F10586458.2011.565253.

[6] Eli Ben-Sasson et al. "Scalable zero knowledge via cycles of elliptic curves". In: *Algorithmica* 79.4 (2017), pp. 1102–1160.

[7] Joseph Bonneau et al. *Coda: Decentralized Cryptocurrency at Scale*. Cryptology ePrint Archive, Report 2020/352. https://ia.cr/2020/352. 2020.

[8] Sean Bowe, Jack Grigg, and Daira Hopwood. *Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. https://ia.cr/2019/1021. 2019.

[9] Jens Groth. *On the Size of Pairing-based Non-interactive Arguments*. Cryptology ePrint Archive, Report 2016/260. https://ia.cr/2016/260. 2016.

[10] Benedikt Bünz et al. *Bulletproofs: Short Proofs for Confidential Transactions and More*. Cryptology ePrint Archive, Report 2017/1066. https://ia.cr/2017/1066. 2017.

[11] *What is JubJub?* URL: https://z.cash/technology/jubjub/.

[12] Klaus Thull and Chee Yap. "A Unified Approach to HGCD Algorithms for polynomials and integers". In: (Aug. 1998).

[13] Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms". In: *Advances in Cryptology — CRYPTO 2001*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 190–200. ISBN: 978-3-540-44647-7.

[14] Daira Hopwood. *Faster variable-base scalar multiplication in zk-SNARK circuits*. URL: https://github.com/zcash/zcash/issues/3924.

7 Appendix

7.1 Mumford Representation for Higher Multiplicities

To interpolate the Mumford representative for a single point of multiplicity larger than one, say nP, one can incrementally construct the polynomial via Hensel lifting. That is, compute a sequence of $v_0(x)..v_k(x)$ values until $2^{k-1} < n \leq 2^k$ as follows

$$v_0(x) = y(P) \qquad r_0(x) = \frac{y(P)^2 - (x^3 + Ax + B)}{x - x(P)}$$

$$q_i(x) = \frac{-2r_i(x)}{v_i(x)} \mod (x - x(P))^{2^i}$$

$$v_{i+1}(x) = v_i(x) + (x - x(P))^{2^i} q_i(x)$$

$$r_{i+1}(x) = \frac{2v_i(x)q_i(x) + r_i(x)}{(x - x(P))^{2^i}} + q_i(x)^2$$

At each stage, $(x - x(P))^{2^i} \mid v_i(x)^2 - (x^3 + Ax + B)$. The final result is obtained via $v(x) = v_k(x) \mod (x - x(P))^n$. To find the rest of the Mumford representation, the prover can use an ordinary polynomial interpolation algorithm.

7.2 Weil Reciprocity

Given an element of the function field of E, $f \in \mathbb{F}(E)$, and a divisor $D \in \mathrm{Div}(E)$ the evaluation of f at the divisor D can be defined in the following manner

$$f(D) = \prod_{P \in D} f(P)^{v_P(D)}$$

Letting $\mathrm{div}(f)$ denote the principal divisor associated to an element of the function field, Weil reciprocity states that

$$f(\mathrm{div}(g)) = g(\mathrm{div}(f))$$

Note that this is true for arbitrary elements of the function field, but if the divisors have any common support both sides will vanish or diverge. Note also that the divisors here include the point at infinity. It is possible to generalize to arbitrary pairs of divisors with common support by first defining

$$\langle f, g \rangle_P = (-1)^{v_P(\mathrm{div}(f))v_P(\mathrm{div}(g))} \frac{f(P)^{v_P(\mathrm{div}(g))}}{g(P)^{v_P(\mathrm{div}(f))}}$$

This evaluation should be interpreted as a limit or equivalently that any common zeros or poles cancel. Note also that when one input has a zero, the exponent of the input with a zero or pole is zero, and the value should be interpreted as depending only on the other input. That is, for $P \in \mathrm{div}(g)$ but not in $\mathrm{div}(f)$

$$\langle f, g \rangle_P = f(P)^{v_P(\mathrm{div}(g))}$$

Which is the same factor as the first form. Weil reciprocity is the statement that the product of these symbols over all the points of the curve equals unity.

$$\prod_{P \in E} \langle f, g \rangle_P = 1$$

The elliptic resultant is a limited form of this equality for divisors that have disjoint support everywhere except for the point at infinity, as all nontrivial polynomial elements of the function field have a pole at infinity. The symbol at the point at infinity produces exactly the leading coefficients up to sign that appear in the elliptic resultant. Thankfully, under logarithmic differentiation, these leading coefficients cancel, so it ultimately makes no difference to the final protocol.