Bulletproofs++

Liam Eagen
liameagen@protonmail.com

March 18, 2022

Abstract

Building on Bulletproofs [1] and Bulletproofs+ [2], I describe several new range proofs that achieve both shorter proof sizes and witness lengths as well as a new confidential transaction protocol for multiple types of currency. The first section describes how to modify the (weighted) inner product protocol to prove a norm relation, i.e. self inner product, while only committing to the vector once. In the second section, this is used to construct a binary digit range proof of half the witness length. Using a novel permutation argument, which is essentially the logarithmic derivative of [3], and the norm argument, I then construct a family of range proofs for arbitrary bases. In the case of 64 bit range proofs, using 16 base 16 digits, the reciprocal range proof achieves a proof size of 10 curve points and 3 scalars, 416 bytes in Curve25519 and 418 in SECP256k1, and witness length of 23 scalars. This proof size is approximately 27% smaller than Bulletproofs+ and 38% smaller than Bulletproofs. The witness length, which is proportional to verification complexity, is reduced by a factor of roughly 6, which asymptotically approaches 8 as the number of ranges increases. Finally, I use the permutation argument to construct a zero knowledge confidential transaction protocol for multiple types of currency. This uses one multiplication per input and per output and supports multiparty proving, substantially improving the state of the art [4].

1 Introduction

The popularity of cryptocurrencies has led to an unprecedented level of interest in and increase in the practical applications for zero knowledge proofs. Unlike many other applications of zero knowledge proofs that typically need to compete with trusted, centralized alternatives, cryptocurrencies and blockchains are an unusual environment in which these centralized alternatives either cannot exist or are categorically rejected by the users. This hostile environment has created the perfect set of constraints to incentivize the research and development of zero knowledge proof protocols.

Different proof systems offer a variety of tradeoffs among proof size, proving time, verification time, security assumptions, and power. In the context of blockchains specifically, proof size and verification time of prime importance, as the consensus algorithm is primarily constrained by the amount of data communicated and how long it takes to verify the data. Proving time is also critically important, especially in zero knowledge proofs for privacy. The privacy of the system grows and diminishes in proportion to the number of users of the privacy technology, so proof systems that are impractical to prove tend not be used, reducing the degree of privacy, thus creating a negative feedback loop.

Among proof systems in practical use for privacy preserving transactions, the primary tradeoff is between power and security assumptions, especially the use of a trusted setup. SNARKs [5] can be used to construct proofs for circuits of arbitrary complexity, and proofs of exceptionally small size, but require a trusted setup. This assumption is fundamentally at odds with the essential ethos of public blockchains, which is to eliminate central points of trust or, at the very least, make them transparent to the users of the system.

1.1 Bulletproofs

Bulletproofs [1] are on the other side of the power, assumptions spectrum. They do not require a trusted setup or even any additional cryptographic assumptions beyond the hardness of the discrete log problem. They are provably secure in the random oracle model and compose well with existing technologies, for example elliptic curve digital signatures, already in use by cryptocurrencies. Proof size increases logarithmically in the size of the witness, and is small enough in practice that they have found applications in projects like Monero [6], MimbleWimble [7] protocols like [8] and [9], and many

others. Unfortunately, proving time and verification time increase linearly in the size of the witness. This severely, compared to SNARKs, limits the size of the circuits that are practical to prove.

Thankfully, many common uses for zero knowledge proofs do not require large circuits. In particular, range proofs are small enough to efficiently prove and verify and are, more or less, sufficient to build confidential transaction protocols. That is, transactions that hide the amount of money involved in the transaction, as well as other transaction data like the types of currency involved. This increases privacy and offers practical protection against targeted theft attempts.

Bulletproofs use a recursive structure to prove that a Pedersen commitment to a pair of vectors is equal to a committed scalar. At each step in the proof, the prover is able to transform the committed vector into a vector of half the length using a constant amount of communication with the verifier, resulting in an overall proof size logarithmic in the length of witness. This builds on the earlier work of Bootle et al [10] reducing the proof size by a factor of three and accepting inputs via Pedersen commitments.

Bulletproofs are further improved upon in Bulletproofs+ [2] to support a weighted inner product structure. Most inner product based protocols use a system of random weights to transform independent multiplicative constraints into a single inner product constraint by randomizing each sub product. When the weights are powers of some value q, Bulletproofs+ integrate these weights into the recursive reduction structure by factoring the weight vector into a tensor product of dimension 2 vectors

$$\bigoplus_{i=0}^{2^n-1} q^i = \bigotimes_{j=0}^{n-1} 1 \oplus q^{2^i}$$

1.2 Contributions

I continue to build on this work by modifying the recursive structure of the (weighted) inner product argument to show that a single committed vector satisfies a norm relation, that is a self inner product. This can be used to construct a smaller binary (digit) range proof by phrasing the binary digit constraint as a difference of squares instead of a product of each digit with its complement.

Next, I introduce a modification of the permutation argument in [3] to construct range proofs using larger bases. Encoding the digits of a number as roots of a polynomial, this polynomial is equal to a product of linear factors with roots at each public digit symbol raised to a secret power encoding the multiplicity of that symbol. Taking the logarithmic derivative of both of these representations, the equality becomes a sum of poles at the negated digits equals a multiplicity weighted sum of the poles corresponding to the symbols. While this adds at least one extra round, in many cases the reduction in the number of digits more than compensates for the increase in proof size.

In many cases ranges are naturally expressible using larger bases, e.g. $2^{64} = 16^{16}$ which reduces the number of digits by a factor 4, but in other cases this requires modifications to handle arbitrary range and bases. For proofs of many ranges, it becomes economical to use even larger bases, e.g. $N > 256$ the proof can use $2^{64} = 256^8$ which reduces the commitment length by a factor of 16 compared to the binary range proof. Compared to an inner product based range proof that commits to all the digits and their complement, the reduction asymptotically approaches 16 as N grows.

Finally, I use the same reciprocal permutation argument to construct a confidential transaction protocol for multiple types of currency that is zero knowledge with respect to: the types, which inputs and outputs are of the same type, and how many types there are, beyond what information can be inferred from the number of inputs and outputs. In this case, the types are in the denominator and the signed amounts are in the numerator. The marginal cost of this protocol compared to a range proof is very small and this could be used to add transactions with multiple types of currency to protocols like Grin and Monero at essentially no additional cost in terms of proof size, proving time, verification time, or security assumptions.

2 Preliminaries

Bulletproofs++ use essentially the same model as Bulletproofs(+). The only important differences are either superficial, i.e. using additive vs multiplicative notation for the group operation or the manner in which vectors are decomposed, or in the case of the reciprocal argument a weakening from perfect completeness to statistical completeness.

2.1 Cryptography

Bulletproofs++ use the same cryptographic assumptions as Bulletproofs and Bulletproofs+, the hardness of the discrete log problem over curve points, and are provably secure in the random oracle model.

2.1.1 Discrete Log Problem

The discrete log problem can be equivalently phrase in several different ways. For the purposes of this paper, the discrete log problem is most usefully stated as: given a uniformly random vector $\mathbf{G}$ of curve points, it is computationally infeasible to find any vector of scalars $\mathbf{x}$ such that

$$\langle \mathbf{x}, \mathbf{G} \rangle = 0$$

The best known algorithms for the elliptic curve discrete log problem in general elliptic curves, i.e. non-anomolous, with high embedding degree, is proportional to the square root of the group size, so an elliptic curve group with order $2^{2\lambda}$ requires $O(2^{\lambda})$ time to compute a random discrete log relation and is therefore said to have a security level of λ.

2.1.2 Zero Knowledge Proofs

Zero knowledge proofs satisfy three properties: completeness, soundness, and zero knowledge. Completeness requires that a prover can construct a proof for any valid witness, soundness requires that a prover not be able to construct a proof for an invalid witness, and zero knowledge requires that the verifier not learn anything about the witness from the proof.

These properties can themselves either hold perfectly, meaning that they hold unconditionally, computationally, meaning that they hold only under computational assumptions about the prover or verifier, or statistically, meaning that they hold with overwhelming probability. For example, soundness will only hold computationally, since a computationally unbound prover could just compute a discrete log relation to construct a proof for an invalid witness. Traditionally, zero knowledge protocols with computational soundness are referred to as "arguments" and those with statistical soundness are referred to as "proofs," although informally I will refer to both as zero knowledge proofs.

On the other hand, completeness typically holds perfectly meaning that the prover can always construct a proof for a valid witness. In the case of the reciprocal argument, this is actually not true since there is a negligible probability that the challenge might cause the denominator to vanish. In that case it is not possible to construct a proof, but since the case occurs with negligible probability, the proof still has statistical completeness.

2.1.3 Non-Interactive Zero Knowledge Proofs

To make zero knowledge proofs practical in certain applications like blockchains, it must be possible for the prover to construct the proof without interacting with "the" verifier. This is accomplished by replacing all interactions with the verifier by calls to a hash function, which is modeled as a random oracle for the purposes of proving correctness. The random oracle will choice response data uniformly at random, independently of the communication from the verifier. This limits what sorts of data the prover can request from the verifier.

2.1.4 Computational Witness Extended Emulation

As in Bulletproofs(+) to prove that a protocol has computational soundness in the random oracle model, one uses an emulator to rewind the protocol and sample multiple challenges. Given multiple challenges for the same witness, it is possible to solve for the witness, of a well formed protocol, using linear algebra. The ability to extract the entire witness is demonstrates that the prover must know the entire witness in order to construct the proof.

2.1.5 Special Honest Verifier Zero Knowledge

The counterpart to CWEE is SHVZK, where one shows that it is not possible to learn the witness from the public commitments and openings of the prover. The way I will do this is to show that there are enough free linear degrees of freedom in the blinding of all the commitments such that fixing a witness and a transcript it is possible to solve for blinding values that produce the transcript for the witness. The "special honest verifier" refers to the requirement that the verifier choose the challenge values uniformly

at random. In non-interactive settings in the random oracle model, the verifier does not actually have any freedom in choosing the challenges, so this is automatically the case.

2.1.6 Schwartz-Zippel Lemma

To show soundness, ZKPs typically reduce to the Schwartz-Zippel lemma, which states that for a non-zero polynomial F of total degree d and uniform distribution D of size N

$$\frac{d}{D} = P(F(\mathbf{e}) = 0 \mid F \in \mathbb{F}_p[\mathbf{x}], \mathbf{e} \in D^n)$$

Informally, the probability of randomly choosing a zero of a polynomial is inversely proportional to the size of the distribution per argument, and proportional to the total degree. The immediate corollary is that if the polynomial is zero at the challenge point, the polynomial must be identically zero.

2.1.7 Commitments

Commitments are binding and hiding, either computationally or perfectly. A commitment cannot be both perfectly hiding and binding, so a tradeoff must be made. Bulletproofs++ use Pedersen commitments, which are computationally binding up to the hardness of the discrete log problem. In the case of the inputs the protocols, they are also perfectly hiding and are distributed uniformly at random. Blinded Pedersen commitments will be denoted

$$\mathrm{Com}\left(v; \mathbf{x}, \mathbf{G}\right); \mathbf{y}, \mathbf{H} = sH + vG + \langle \mathbf{x}, \mathbf{G} \rangle + \langle \mathbf{y}, \mathbf{H} \rangle$$

In the zero knowledge protocols, commitments are also computationally binding, but instead of being individually perfectly hiding I will show that the entire protocol is SHVZK. In a sense this is equivalent, as in both cases no information about the witness is revealed, but these commitments will not be explicitly blinded. These commitments will be denoted

$$\mathrm{Com}_{BP}\left(v; \mathbf{x}, \mathbf{G}; \mathbf{y}, \mathbf{H}\right) = vG + \langle \mathbf{x}, \mathbf{G} \rangle + \langle \mathbf{y}, \mathbf{H} \rangle$$

2.2 Notation

Fix an elliptic curve E and a prime order cyclic subgroup of E of order p. While the protocol is valid over all elliptic curve groups, certain sections of the paper deal with optimizations for elliptic curves with complex multiplication, in which case E is not super singular and the subgroup of points is closed under complex multiplication. This is essentially always the case in practice.

Elements of this group will be called both "group elements" or "curve points" and denoted with capital letters. Values in the finite field $\mathbb{F}_p$ will be denoted by lowercase letters and called either "finite field elements" or "scalars." The group operation will be written additively and scalar multiplication via juxtaposition so that

$$(x + y)(G + H) = xG + yG + xH + yH$$

Vectors of both curve points and scalars will be written using bold letters. All vectors will be zero indexed and padded on the right with zeros as necessary. The i element of a vector $\mathbf{v}$ will be written v_i. The inner product of two vectors is defined when the components can be multiplied and is written

$$\langle \mathbf{x}, \mathbf{G} \rangle = \sum_{i=0}^{n} x_i G_i$$

Where n is the maximum of the lengths of the vectors. Equivalently, since all vectors are padded with zeros and of finite length, the sum can be over all indices. For a scalar q the (power) weighted inner product starts at q^1 and will be written

$$\langle \mathbf{x}, \mathbf{y} \rangle_q = \sum_{i=0}^{n} x_i y_i q^{i+1}$$

Norms of vectors will be defined to be the (weighted) inner product of a vector with itself. This may conflict with other definitions of norms, but in this paper I will refer to it as "the norm" unambiguously

$$|\mathbf{x}|_q^2 = \langle \mathbf{x}, \mathbf{x} \rangle_q$$

Matrices will also be denoted by capital letters. In particular, the matrix $D(x)$ is the diagonal matrix of powers of x such that $D(x)_{ii} = x^i$ using zero based indexing, and the matrix $Q = qD(q)$ which starts at the first power of q, which can be inferred from context. It will generally be clear from context whether I am referring to a group element or a matrix as matrices almost always appear multiplying a vector. The tensor product of two vectors is used mostly as a notational convenience and is defined to be

$$\mathbf{a} \otimes (b_0, b_1, ...) = b_0\mathbf{a} \oplus b_1\mathbf{a} \oplus ...$$

In the inner product and norm arguments, each round splits a pair of vectors into nearly equal halves. For a vector $\mathbf{v}$ the halves will be denoted $\mathbf{v}_0$ and $\mathbf{v}_1$ such that

$$\text{len}(\mathbf{v}_0) = \left\lceil \frac{\text{len}(\mathbf{v})}{2} \right\rceil \text{ and } \text{len}(\mathbf{v}_1) = \left\lfloor \frac{\text{len}(\mathbf{v})}{2} \right\rfloor$$

In this paper, these halves are defined to be interleaved rather than concatenated. That is

$$v_{0,i} = v_{2i} \qquad v_{1,i} = v_{2i+1}$$

This is distinct from Bulletproofs(+) which use concatenated halves for the purposes of the inner product argument. There is no theoretical difference between the two, and any of the protocols can be made to work with either concatenated or interleaved halves. I will use interleaved halves due to personal preference and because all of the computations happen "locally," in a certain sense.

2.2.1 Norm Arguments

The norm arguments are not zero knowledge, which is a point where Bulletproofs and Bulletproofs+ differ. This is because in the protocols of this paper all the witnesses will be blinded before invoking the arguments in a manner that is tightly integrated into the protocols themselves. This is more like the original Bulletproof inner product argument in structure. It is therefore only necessary to show that the norm argument is sound and CWEE, not that it is SHVZK.

3 Norm Argument

There are two different norm arguments in this section. The first is a reduction from a $-r^2$ weighted norm argument to the weighted inner product argument by factoring adjacent difference of squares into products. The second uses a different response structure to directly prove that a, potentially weighted, sum of squares equals a committed value. Depending on the length of the vectors, the latter argument can result in a slightly smaller proof size.

Then, both arguments are augmented by a linear argument, where the prover takes the inner product of a committed vector with a public vector. In the former case, this is equivalent to concatenating the witness and constraint onto vectors in the inner product after the transformation, but in the latter case it is not possible to directly express without squaring the witness. In both cases, the response structure is unchanged.

This linear argument has advantages in multiparty proofs and in the generic blinding protocol. All the high level protocols in the paper will use this protocol to blind the witness and the "error terms" from the blinding polynomial in a single round. In many cases, this avoids the need for a dedicated blinding component in the vectors as the blinding from this protocol is sufficient. In case it is not, the linear argument also supports moving the blinding value into the linear vector with associated coefficient 0. This protocol is responsible for a substantial portion of the reduction in proof size in small cases, like the 64 bit range proof.

3.1 Inner Product Arguments

Bulletproofs use a recursive technique to transform a commitment to two vectors and their inner product

$$C = \text{Com}_{BP}(\langle \mathbf{x}, \mathbf{y} \rangle; \mathbf{x}, \mathbf{G}; \mathbf{y}, \mathbf{H})$$

Into a commitment to two vectors of half the length and their inner product

$$C' = \text{Com}_{BP}(\langle \mathbf{x}', \mathbf{y}' \rangle; \mathbf{x}', \mathbf{G}'; \mathbf{y}', \mathbf{H}')$$

Such that if $v' = \langle \mathbf{x}', \mathbf{y}' \rangle$ then $v = \langle \mathbf{x}, \mathbf{y} \rangle$ with overwhelming probability. The proof structure, as well as completeness, follows from the following identity of polynomials in e evaluated at a random point of the verifier's choosing and the Schwartz-Zippel lemma

$$\left\langle e^{-1}\mathbf{x}_0 + e\mathbf{x}_1, e\mathbf{y}_0 + e^{-1}\mathbf{y}_1 \right\rangle = e^{-2} \langle \mathbf{x}_0, \mathbf{y}_1 \rangle + \langle \mathbf{x}, \mathbf{y} \rangle + e^2 \langle \mathbf{x}_1, \mathbf{y}_0 \rangle$$

Applying this identity to all three inner products in the commitment, $\langle \mathbf{x}, \mathbf{y} \rangle$, $\langle \mathbf{x}, \mathbf{G} \rangle$, and $\langle \mathbf{y}, \mathbf{H} \rangle$, the prover needs only send two additional commitments L and R from which the verifier can evaluate the identity for all the inner products at a random point.

$$L = \mathrm{Com}_{BP}\left(\langle \mathbf{x}_0, \mathbf{y}_1 \rangle ; \mathbf{x}_0, \mathbf{G}_1; \mathbf{y}_1, \mathbf{H}_0\right)$$

$$R = \mathrm{Com}_{BP}\left(\langle \mathbf{x}_1, \mathbf{y}_0 \rangle ; \mathbf{x}_1, \mathbf{G}_0; \mathbf{y}_0, \mathbf{H}_1\right)$$

To do so, after selecting e, the verifier will compute

$$\mathbf{G}' = e\mathbf{G}_0 + e^{-1}\mathbf{G}_1 \qquad \mathbf{H}' = e^{-1}\mathbf{H}_0 + e\mathbf{H}_1$$

$$\mathbf{x}' = e^{-1}\mathbf{x}_0 + e\mathbf{x}_1 \qquad \mathbf{y}' = e\mathbf{y}_0 + e^{-1}\mathbf{y}_1$$

So that

$$C' = C + e^{-2}L + e^2 R = \mathrm{Com}_{BP}\left(\langle \mathbf{x}', \mathbf{y}' \rangle ; \mathbf{x}', \mathbf{G}'; \mathbf{y}', \mathbf{H}'\right)$$

This argument can be applied recursively to generate a sequence of responses (L_i, R_i) and commitments C_i to vectors of the of sizes $n_i = \lceil n_{i-1}/2 \rceil$ such that $C_{i-1} = C_i + e_i^{-2}L_i + e_i^2 R_i$. Eventually, the prover will be left with the commitment $C_0 = s_0 H + vG + xG' + yH'$ which cannot be further reduced.

3.2 Weighted Inner Product

This technique is modified in Bulletproofs+ to evaluate the inner product $\langle \mathbf{x}, \mathbf{y} \rangle_q$ directly, which Bulletproofs do by separately scaling the vector and then using the inner product argument. This works by decomposing the Q power matrix as a tensor product and combining the q scaling into the inner product relation. In Bulletproofs+ this is done "top down" starting with the largest q^{2^i} and working down, since the subvectors concatenate to form the larger vector, but in Bulletproofs++ I do this "bottom up" starting with the smallest q^{2^i} and working up since the vectors are interleaved. Writing the weighted inner product with interleaved vectors

$$\mathbf{x}' = (eq)^{-1}\mathbf{x}_0 + e\mathbf{x}_1 \qquad \mathbf{G}' = qe\mathbf{G}_0 + e^{-1}\mathbf{G}_1$$

$$\langle \mathbf{x}', \mathbf{y}' \rangle_{q^2} = e^{-2}q^{-1} \langle \mathbf{x}_0, \mathbf{y}_1 \rangle_{q^2} + \langle \mathbf{x}, \mathbf{y} \rangle_q + e^2 \langle \mathbf{x}_1, \mathbf{y}_0 \rangle_{q^2}$$

$$\langle \mathbf{x}', \mathbf{G}' \rangle = e^{-2}q^{-1} \langle \mathbf{x}_0, \mathbf{G}_1 \rangle + \langle \mathbf{x}, \mathbf{G} \rangle + e^2 \langle \mathbf{x}_1, \mathbf{G}_0 \rangle$$

Where the $\mathbf{y}'$ and $\mathbf{H}'$ are unchanged. Substituting these definitions into the definitions of L and R, the polynomial relation is unchanged. In the next round, the prover will use q^2 instead of q.

3.3 Norm Argument

The inner product argument requires committing to both vectors in order to show that the inner product relation is satisfied. In some cases committing to both vectors includes redundant information. For example, in range a proof the prover might want to show that a sequence of committed values are either zero or one. To do this in Bulletproofs(+) using the inner product argument, the prover will typically commit to the vector of bits in $\mathbf{x}$ and the vector of binary complements in $\mathbf{y}$ and show

$$x_i y_i = 0 \qquad x_i + y_i = 1$$

Which is equivalent to showing that $x_i(x_i - 1) = 0$. The vector of binary complements is redundant in the sense that there is an algebraically equivalent constraint, given by completing the square, that does not require committing to two separate values. Given a norm argument, the square constraint one requires committing to one value per bit, reducing the witness length by half.

$$x_i(x_i - 1) = (x_i - 1/2)^2 - 1/4$$

3.3.1 Reduction to Inner Product

In the case of a weighted inner product, which is what the higher level protocols will ultimately use, there are some weights that directly can be reduced to a weighted inner product argument. If $q = -r^2 \mod p$ the weighted norm equals

$$|\mathbf{x}|_q^2 = \sum_{i=0}^{2n-1} x_i^2 q^{i+1} = q^{-1} \sum_{i=0}^{n-1} (x_{2i} - r x_{2i+1})(x_{2i} + r x_{2i+1}) q^{2(i+1)} = \left\langle r^{-1}\mathbf{x}_0 - \mathbf{x}_1, r^{-1}\mathbf{x}_0 + \mathbf{x}_1 \right\rangle_{q^2}$$

Given a commitment $C = \text{Com}_{BP}\left(|\mathbf{x}|_q^2 ; \mathbf{x}, \mathbf{G}\right)$, there exists a different basis such that C commits to to the inner product witness. It is important to note that, since the basis transformation is invertible, the curve points remain linearly independent and therefore valid for the purposes of constructing a Bulletproof.

$$\mathbf{x}' = r^{-1}\mathbf{x}_0 - \mathbf{x}_1 \qquad \mathbf{y}' = r^{-1}\mathbf{x}_0 + \mathbf{x}_1$$
$$\mathbf{G}' = (r/2)\mathbf{G}_0 - (1/2)\mathbf{G}_1 \qquad \mathbf{H}' = (r/2)\mathbf{G}_0 + (1/2)\mathbf{G}_1$$
$$C = \text{Com}\left(\langle \mathbf{x}', \mathbf{y}' \rangle_{q^2} ; \mathbf{x}', \mathbf{G}'; \mathbf{y}', \mathbf{H}'\right)$$

If the field characteristic satisfies $p = 1 \mod 4$ then -1 is a quadratic residue, and the constraint is equivalent to requiring q be a quadratic residue. This further implies that the q weighted inner product is equivalent, up to a linear transformation of $\mathbf{x}$ to an unweighted sum of squares.

In the case $p = 3 \mod 4$, this is not true. However, it is still possible to reduce an unweighted sum of squares to an inner product of vectors of nearly half the length in such a field as a consequence of the Chevalley-Warning theorem. In fact, this is true of any quadratic form over any finite field. This is not used in any of the zero knowledge proof protocols but is described in the Appendix.

3.3.2 Modified Norm Argument

The inner product argument reduces $\mathbf{x}$ and $\mathbf{y}$ differently, with the first half of $\mathbf{x}$ scaled by $1/e$ and the first half of $\mathbf{y}$ scaled by e. To create a norm argument directly, without reduction to the inner product argument, the prover must use a polynomial relation that treats both vectors symmetrically such as

$$\langle \mathbf{x}_0 + e\mathbf{x}_1, \mathbf{y}_0 + e\mathbf{y}_1 \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + e\left(\langle \mathbf{x}_0, \mathbf{y}_1 \rangle + \langle \mathbf{x}_1, \mathbf{y}_0 \rangle\right) + (e^2 - 1)\langle \mathbf{x}_1, \mathbf{y}_1 \rangle$$

The norm argument follows from this relation by applying it to both inner products, $|\mathbf{x}|^2$ and $\langle \mathbf{x}, \mathbf{G} \rangle$, and grouping like terms in the same manner as the inner product argument. Soundness follows from the fact that the polynomials 1, e, and $e^2 - 1$ are linearly independent. That is, given

$$C_n = \text{Com}_{BP}\left(|\mathbf{x}|^2 ; \mathbf{x}, \mathbf{G}\right)$$

The prover will commit to

$$X_n = \text{Com}_{BP}\left(2\langle \mathbf{x}_0, \mathbf{x}_1 \rangle ; \mathbf{x}_0, \mathbf{G}_1; \mathbf{x}_1, \mathbf{G}_0\right) \qquad R_n = \text{Com}_{BP}\left(|\mathbf{x}_1|^2 ; \mathbf{x}_1, \mathbf{G}_1\right)$$

Which the verifier will use these to compute the next commitment as

$$\mathbf{G}' = \mathbf{G}_0 + e\mathbf{G}_1 \qquad \mathbf{x}' = \mathbf{x}_0 + e\mathbf{x}_1$$
$$C_{n-1} = C_n + eX_n + (e^2 - 1)R_n = \text{Com}_{BP}\left(|\mathbf{x}'|^2 ; \mathbf{x}', \mathbf{G}'\right)$$

After recursively applying this argument, the protocol will terminate with a commitment C_0 to a value and its square. However, the final round of the protocol reduces the length of the committed vector by 1 at the cost of two additional group elements. The prover will instead halt the proof at C_1 or even C_2 to achieve the smallest possible proof size. This is the primary advantage of this technique over the reduction to the inner product proof, as it is sometimes possible achieve a slightly smaller proof size using the specialized norm argument.

For example, a sum of 10 squares will split to an inner product of two vectors of length 5, which after 2 rounds will both be of length 2 for a total of four group elements and four scalars. The norm argument after two rounds will have 3 scalars as $10 \to 5 \to 3$ and four group elements, which saves one scalar. On the other hand, committing to X requires evaluating asymptotically twice as many elliptic curve operations so different use cases may prefer different sides of the trade off.

3.3.3 q-Power Weighted Norm

To scale the norm by q powers, the norm argument will adapt the technique of Bulletproofs+. As in the reduction to the inner product argument, the powers must lie in a particular quadratic residuosity class, in this case $q = r^2$.

$$\mathbf{x}' = r^{-1}\mathbf{x}_0 + e\mathbf{x}_1 \qquad \mathbf{G}' = r\mathbf{G}_0 + e\mathbf{G}_1$$

$$|\mathbf{x}'|_{q^2}^2 = \left|r^{-1}\mathbf{x}_0 + e\mathbf{x}_1\right|_{q^2}^2 = |\mathbf{x}|_q^2 + 2er^{-1}\left\langle \mathbf{x}_0, \mathbf{x}_1\right\rangle_{q^2} + (e^2-1)\left|\mathbf{x}_1\right|_{q^2}^2$$

$$\left\langle \mathbf{x}', \mathbf{G}'\right\rangle = \left\langle \mathbf{x}, \mathbf{G}\right\rangle + e\left(r^{-1}\left\langle \mathbf{x}_0, \mathbf{G}_1\right\rangle + r\left\langle \mathbf{x}_1, \mathbf{G}_0\right\rangle\right) + (e^2-1)\left\langle \mathbf{x}_1, \mathbf{G}_1\right\rangle$$

The modified X_n and R_n commitments then become

$$X = \mathrm{Com}_{BP}\left(2r^{-1}\left\langle \mathbf{x}_0, \mathbf{x}_1\right\rangle_{q^2}; r\mathbf{x}_1, \mathbf{G}_0; r^{-1}\mathbf{x}_0, \mathbf{G}_1\right) \qquad R = \mathrm{Com}_{BP}\left(\left|\mathbf{x}_1\right|_{q^2}^2; \mathbf{x}_1, \mathbf{G}_1\right)$$

With the next commitment given by the same linear combination as the unweighted norm argument

$$C' = C + eX + (e^2-1)R$$

3.4 Linear Argument

In both the inner product and the norm arguments, the new vectors are constructed using the same polynomial relation, which means the prover can reduce two different vectors at the same time. The prover can show, given a public vector $\mathbf{c}$ of constraints, two committed vectors $\mathbf{l}$ and $\mathbf{n}$ and a scalar v satisfy

$$v = \left\langle \mathbf{c}, \mathbf{l}\right\rangle + |\mathbf{n}|^2$$

3.4.1 Reduction to Inner Product

When reduced to an inner product argument, the exact same argument and commitment structure can be used for the linear vector $\mathbf{l}$ but instead of committing to $\mathbf{c}$ the prover and verifier will carry out all of the reduction computations in public. The similarity in structure allows the prover to actually concatenate the linear vector onto one of the arguments to the inner product and carry out the argument. When the sizes of the vectors satisfy

$$s2^k < \mathrm{len}\left(\mathbf{l}\right) + \mathrm{len}\left(\mathbf{n}\right) \leq (s+1)2^k \qquad s2^k < \mathrm{len}\left(\mathbf{n}\right)$$

For $s = 2, 3$ concatenating the linear vector onto one of the vectors for the inner product argument will reduce the proof size. This is because the number of rounds to reduce the vectors is unchanged, and the final scalar for the reduced $\mathbf{l}$ is eliminated. In fact, it is possible to achieve smaller proof sizes even than the direct norm argument.

3.4.2 Multiparty Proving

In a multiparty proof, it is easy for multiple provers to compute the sum of secret values, as Pedersen commitments are linear. It is also easy to construct zero knowledge proofs involving linear combinations of secret values since each party can compute the linear combination over their secret data and provers can then sum the results publicly. Computing products of secret data, while certainly possible, is much more expensive.

Throughout all of the protocols, at no point will secret values from different ranges by multiplied. The only multiplications occur within a single component of the norm vector and between elements during the norm argument, at which point all the components will be blinded and the products can be computed publicly. Since the components of the linear vector are never multiplied before they are blinded, the provers can all linearly contribute to them.

3.4.3 Norm Argument

For a public vector $\mathbf{c}$, committed vectors $\mathbf{l}$ and $\mathbf{n}$, and $v = \left\langle \mathbf{c}, \mathbf{l}\right\rangle + |\mathbf{n}|^2$ the prover will initially commit to

$$C = \mathrm{Com}_{BP}\left(v; \mathbf{l}, \mathbf{H}; \mathbf{n}, \mathbf{G}\right)$$

And then commit to the modified responses

$$X = \text{Com}_{BP}\left(\langle \mathbf{c}_0, \mathbf{l}_1\rangle + \langle \mathbf{c}_1, \mathbf{l}_0\rangle + 2\langle \mathbf{n}_0, \mathbf{n}_1\rangle; \mathbf{l}_1, \mathbf{H}_0; \mathbf{l}_0, \mathbf{H}_1; \mathbf{n}_1, \mathbf{G}_0; \mathbf{n}_0, \mathbf{G}_1\right)$$

$$R = \text{Com}_{BP}\left(\langle \mathbf{c}_1, \mathbf{l}_1\rangle + |\mathbf{n}_1|^2; \mathbf{l}_1, \mathbf{H}_1; \mathbf{n}_1, \mathbf{G}_1\right)$$

The verifier will choose the challenge value and the prover will compute the next commitment and updated vectors as

$$C' = C + eX + (e^2 - 1)R$$

$$\mathbf{G}' = \mathbf{G}_0 + e\mathbf{G}_1 \qquad \mathbf{H}' = \mathbf{H}_0 + e\mathbf{H}_1$$

$$\mathbf{c}' = \mathbf{c}_0 + e\mathbf{c}_1 \qquad \mathbf{l}' = \mathbf{l}_0 + e\mathbf{l}_1 \qquad \mathbf{n}' = \mathbf{n}_0 + e\mathbf{n}_1$$

In general $\mathbf{l}$ and $\mathbf{n}$ will not be the same length and the optimal number of rounds for overall proof size will depend on how much larger one is than the other. The optimal number of rounds k can be defined such that increasing the number of rounds removes two or fewer total scalars from the final opening. This only becomes an issue when the lengths of the vectors are similar in size and one occurs in the "upper half" of the range given by its number of bits. Stated precisely the optimal number of rounds k is

$$r(\mathbf{v}) = \lceil \log_2 \text{len}(\mathbf{v}) \rceil - 2$$

$$k = \begin{cases} \max(r(\mathbf{l}), r(\mathbf{n})) + 1 & \text{if } |k_l - k_n| \leq 1 \text{ and } \max\left(\frac{\text{len}(\mathbf{l})}{2^{r(\mathbf{l})}}, \frac{\text{len}(\mathbf{n})}{2^{r(\mathbf{n})}}\right) = 4 \\ \max(r(\mathbf{l}), r(\mathbf{n})) & \text{otherwise} \end{cases}$$

3.5 Fast Scalar Multiplication

In proving the norm and linear arguments, as well as the original Bulletproof inner product argument, the prover must compute the updated basis points $\mathbf{G}'$ at each round of the protocol. This is fairly expensive, even compared to the elliptic curve inner product computations, since updating the basis does not benefit from optimizations as much as large elliptic curve inner product computations.

The amount of time spent on computing the updated bases can be substantially reduced by taking advantage of the fact that each updated basis element is the same linear combination of different basis points. That is, the updated basis points are

$$\mathbf{G}' = s\mathbf{G}_0 + t\mathbf{G}_1$$

Where s and t will depend on the kind of argument, but are each pair of points in each vector being reduced. For a fixed linear relation, there exists a shortest vector in the lattice

$$\{(a, b, q) : tb - sa + pq = 0\}$$

This vector can be found using a general algorithm like LLL or for this lattice the extended euclidean algorithm. Optimal performance will likely be achieved by a specialized half-gcd algorithm. Each a and b will be about half the length of the field and given such a solution

$$bs^{-1}\mathbf{G}' = b\mathbf{G}_0 + a\mathbf{G}_1$$

The factor of bs^{-1} can be deferred until the end of the proof and linearly factors through subsequent rounds, while the multiplications of a and b are amenable to Shamir's trick.

3.5.1 Complex Multiplication

In elliptic curves with complex multiplication, basis updates can be performed even more efficiently by exploiting the fact that for some quadratic integer α computing αP is very efficient. Assuming α embeds into $\mathbb{F}_p$, which is always the case if the curve is not supersingular and the prime subgroup is not unusually small, the basis update can be performed using the minimal vector in the lattice

$$\{(a, b, c, d, q) : a + b\alpha - ec - ed\alpha + pq = 0\}$$

Which is the set of solutions to

$$e = \frac{a + b\alpha}{c + d\alpha} \mod p$$

These scalars will each be about one fourth the length of the field. Each updated basis element is now is a linear combination of 4 values, two of which are α multiples of the other two. In elliptic curve with $j = 0, 1728$ these complex multiples are extremely efficient to compute, a linear transformation of the coefficients, and the resulting basis computations can be even faster than in the first case. In these cases, since the associated CM fields are euclidean domains, it is possible to use the same kind of half gcd technique to find the a, b, c, d representation of e. See Appendix for details.

3.6 Combined q Weighted Norm Linear Argument

To integrate the fast scalar multiplications and the q power scaling into the prover, each round of the norm linear argument will accept four normalization factors (s_s, s_l, s_n, q) where q is not, except for the first round, the same as the initial q input. As described earlier, these normalizations allow for shorter basis update computations without affecting the security properties of the protocol. At the beginning and end of each round, the commitment

$$\mathbf{C} = sH + vG + \langle \mathbf{l}, \mathbf{H} \rangle + \langle \mathbf{n}, \mathbf{G} \rangle$$

Will satisfy, given the normalization at that point

$$v = s_s \langle \mathbf{c}, \mathbf{l} \rangle + s_s s_n^2 |\mathbf{n}|_q^2 \qquad q = r^2$$

Algorithm 1 q-Weighted Norm Linear Argument

Require: $v = s_s s_l v_l + s_s s_n^2 v_n$
  function NormArgument$(s_s, s_l, s_n, q, r, \mathbf{c}, C = \mathrm{Com}\,(v; \mathbf{l}, \mathbf{H}; \mathbf{n}, \mathbf{G}))$
    if $\mathrm{len}\,(\mathbf{n}) + \mathrm{len}\,(\mathbf{l}) < 6$ **then**
      return$(s_n, s_l, \mathbf{c}, C)$
    end if
    $v_x = s_s s_l(\langle \mathbf{c}_0, \mathbf{l}_1 \rangle + \langle \mathbf{c}_1, \mathbf{l}_0 \rangle) + 2 s_s s_n^2 r^{-1} \langle \mathbf{x}_0, \mathbf{x}_1 \rangle_{q^2}$
    $v_r = s_s s_l \langle \mathbf{c}_1, \mathbf{l}_1 \rangle + s_s s_n^2 |\mathbf{x}_1|_{q^2}^2$
    commit $X = \mathrm{Com}_{BP}\left(v_x; \mathbf{l}_1, \mathbf{H}_0; \mathbf{l}_0, \mathbf{H}_1; r\mathbf{n}_1, \mathbf{G}_0; r^{-1}\mathbf{n}_0, \mathbf{G}_1\right)$
    commit $R = \mathrm{Com}_{BP}\,(v_r; \mathbf{l}_1, \mathbf{H}_1; \mathbf{n}_1, \mathbf{G}_1)$
    query $e \leftarrow \mathbb{F}_p$
    $(a_l, b_l) =$ RationalRep(e)
    $(a_n, b_n) =$ RationalRep(e/r)
    $\mathbf{c}' = b_l \mathbf{c}_0 + a_l \mathbf{c}_1$
    $\mathbf{l}' = (1/b_l)\mathbf{l}_0 + (e/b_l)\mathbf{l}_1 \quad \mathbf{n}' = (1/b_n)\mathbf{n}_0 + (er/b_n)\mathbf{n}_1$
    $\mathbf{H}' = b_l \mathbf{H}_0 + a_l \mathbf{H}_1 \quad \mathbf{G}' = b_n \mathbf{G}_0 + a_n \mathbf{G}_1$
    $C' = C + eX + (e^2 - 1)R = \mathrm{Com}_{BP}\,(v'; \mathbf{l}', \mathbf{H}'; \mathbf{n}', \mathbf{G}')$
    NormArgument$(s_s, s_l b_l, s_n b_n r^{-1}, q^2, r^2, C')$
  end function

For the first round, the prover will initialize $s_l = 1$ and $s_n = 1$. At the final round of the protocol, the prover will output the final q value q_f and the remaining vectors scaled by their respective normalization factors, $\mathbf{l}_f = s_l \mathbf{l}$ and $\mathbf{n}_f = s_n \mathbf{n}$. For verification, the verifier will receive the initial commitment C, the responses (X_i, R_i), the final scalar v, and the final vectors $\mathbf{l}_f$ and $\mathbf{n}_f$. The verifier will expand out the challenges into two vectors

$$\mathbf{e}_l = \bigotimes_{i=0}^{k} 1 \oplus e_i \qquad \mathbf{e}_n = \bigotimes_{i=0}^{k} r^{2^i} \oplus e_i$$

And verification will consist of one field computation and one elliptic curve inner product. Keeping in mind that the tensor products may be longer than the basis vectors and that vectors should be padded with identity elements as necessary, these are

$$v = s_s \langle \mathbf{c}, \mathbf{e}_l \otimes \mathbf{l}_f \rangle + s_s |\mathbf{n}_f|_{q_f}^2$$

$$C + \sum_{i=0}^{k} e_i X_i + (e_i^2 - 1) R_i = sH + vG + \langle \mathbf{e}_l \otimes \mathbf{l}_f, \mathbf{H} \rangle + \langle \mathbf{e}_n \otimes \mathbf{n}_f, \mathbf{G} \rangle$$

As I have already mentioned, it is possible to modify this protocol to break the vectors into concatenated, rather than interleaved, halves. To do this, the prover can reverse the order of the e_i values in the tensor and keep the r powers in the same order.

3.7 Blinding

All the range proof protocols, as well as the arithmetic circuit protocol, will use the linear argument to blind the entire proof in a single round. To do this, consider a generic argument of the following form.

$$C_i = \text{Com}_{BP}(\mathbf{x}_i) \qquad \sum_{i=1}^{n} \langle \mathbf{c}_i, \mathbf{x}_i \rangle = v \qquad \langle \mathbf{x}_n, \mathbf{x}_{n-1} \rangle_q = 0$$

In the case where the final multiplication needs to be a square, the system works out in more or less the same way. For simplicity, I will just consider this case. Given an additional blinding commitment $B = \text{Com}_{BP}(\mathbf{b})$, the commitments naturally form a polynomial

$$\mathbf{p}(t) = \mathbf{b} + \sum_{i=1}^{n} t^i \mathbf{x}_i + t^{2n-1-i} Q^{-1} \mathbf{c}_{n-i+1}$$

Where the central coefficient encodes our constraints

$$[t^{2n-1}] \, |\mathbf{p}(t)|_q^2 = C + \langle \mathbf{x}_n, \mathbf{x}_{n-1} \rangle_q + \sum_{i=1}^{n} 2 \langle \mathbf{c}_i, \mathbf{x}_i \rangle = 0$$

The rest of the terms of this polynomial are "error terms" that do not encode the constraint directly, but are necessary to prove knowledge of the norm of $\mathbf{p}(t)$. The error terms can be partitioned into three groups. For degrees greater than $3n-1$, the error terms are computable from public constants and q. Error terms of degree $2n+1$ through $3n-2$ are sums of inner products of witnesses with constraints, which causes all the q terms to cancel. The rest of the error terms of degree 1 through $2n-2$ and $2n$ depend on the witness and on q.

For each of the third group of error terms, the protocol will add linear components to all the vectors corresponding to these error terms of degree t^0 through t^{2n-2} and t^{2n}. The updated commitments then becomes

$$B = \text{Com}_{BP}(\mathbf{e}, \mathbf{b}) \qquad C_i = \text{Com}_{BP}(\mathbf{u}_i, \mathbf{x}_i)$$

Given the linear combination of the commitments corresponding to the polynomial

$$C' = B + \sum_{i=1}^{n} t^i C_i = \text{Com}_{BP}(\mathbf{e}', \mathbf{x}')$$

The openings of the error term components are

$$e'_j = e_j + \sum_{i=1}^{n} t^i u_{ij}$$

The linear combination of the error term components with their associated t power monomial, assuming $u_{ij} = 0$ for values that are not present in a commitment, is then

$$\sum_j e'_j t^{f(j)} = \sum_j t^{f(j)} \left(e_j + \sum_{i=1}^{n} t^i u_{ij} \right) = t^{2n-1} \left(\sum_{i+f(k)=2n-1} u_{ik} \right) + \sum_j t^{f(j)} \left(e_j + \sum_{f(j)=f(k)+i} u_{ik} \right)$$

Where the function $f(j)$ injectively maps the indices in the error components of the linear vector to the error monomial degree. If the prover can show that all the diagonal components, u_{ik} where $i + f(k) = 2n-1$, are zero and choose the e_j values appropriately then all the u_{ij} values will cancel. Let the error terms be

$$|\mathbf{p}(t)|_q^2 = [t^{2n-1}] \, |\mathbf{p}(t)|_q^2 + \sum_j \epsilon_j t^{f(j)}$$

Then the prover will commit to C_1 through C_{n-1} and the verifier will choose a random value r_0. Let the set A be

$$A = \{a : 2n+1 \leq f(i) \leq 3n-2, f(a) = f(i) - n\}$$

These are the components of the error term vector in C_n which map to the q independent error terms. The prover will set $u_{nj} = (1/r_0)\epsilon_j$ for $j \in A$. Following the commitment to C_n, the verifier will choose a second challenge r_1 and q. The prover will commit to the blinding with $\mathbf{e}$

$$e_j = r_1^{\delta_A(j)} \left((r_0 r_1)^{-1} \epsilon_j - \sum_{f(j)=f(k)+i} u_{ik} \right)$$

Where δ_A is the indicator function on the set. Evaluating the linear combination of the opening $\mathbf{e}'$ with the modified monomials then leaves only the error terms

$$\sum_j e'_j r_0 r_1^{1-\delta_A(j)} t^{f(j)} = \sum_j \epsilon_j t^{f(j)}$$

Note that for all commitments besides C_n, any nonzero diagonal components will be scaled by r_0 or $r_0 r_1$ and added to the value term. This will cause the proof to fail with overwhelming probability. For C_n, the diagonal term is $n-1$ which is not present in A. Therefore, it will be scaled by r_1 which will have the same effect.

For two or more commitments, the u_{ij} values are sufficient to blind all the commitments, including the blinding. See the proofs section for details. In the case that the multiplication should be an inner product, it is sufficient to modify the error terms to include $2n-1$ and exclude $2n$. For a proof with one commitment, the prover can add a single additional blinding component to the commitment C_1 with linear coefficient 0. The t^0 error term can be mapped to the scalar component of the bulletproof commitments.

3.7.1 Inputs

Inputs to the proof will be handled separately and scaled by t^{2n-1} so that their scalar component is added to the value term directly. The base that they use for blinding can be set to the t^1 error term and their blinding values subtracted off of the $2n$ error term. Except for the inputs, it is sufficient to show that this protocol is SHVZK and CWEE to show that all the range proofs and the arithmetic circuit protocols are as well.

3.7.2 Linear Error Terms

Linear components to the proof are handled using the same structure. Treating the t power of each commitment as fixed with respect to the norm blinding procedure, and letting $\mathbf{l}_i$ be the linear witness and $\mathbf{c}_{l,i}$ be the linear coefficients for commitment i the combined linear witness and constraint are

$$\mathbf{l}(t) = \sum_{i=1}^{n} t^i \mathbf{l}_i \qquad \mathbf{c}_l(t) = \sum_{i=1}^{n} t^{2n-i-1} \mathbf{c}_{l,i}$$

Has the same structure as the constraints for the norm. The modifications to the error terms come from the expansion of the unweighted inner product added to the old error terms

$$|\mathbf{p}(t)|_q^2 + \langle \mathbf{c}_l(t), \mathbf{l}(t) \rangle = [t^{2n-1}] \, |\mathbf{p}(t)|_q^2 + \sum_j \epsilon_j t^{f(j)}$$

3.7.3 Interface

All of this functionality is self contained, and the rest of the protocols can interface with it using the functions

$$\textsc{BlindNormTerm}(n, k, C_k) \qquad \textsc{BlindIPTerm}(n, k, C_k)$$

Which add the correct number of random values to the commitments depending on their position in the polynomial. For the final term, protocols can also call

$$\textsc{BlindFinalIPTerm}(n, \mathbf{e}, C_{n-1})$$

With the q independent error terms. The final blinding term can then be constructed, given the rest of the error terms, the inputs, and the previous commitments and openings via

$$\textsc{NormBlinding}(n, \mathbf{e}, \mathbf{C}) \qquad \textsc{IPBlinding}(n, \mathbf{e}, \mathbf{C})$$

4 Binary Range Proof

The binary digit range proof is a straightforward extension of the q-power norm argument and with witness half the size of an equivalent Bulletproof(+) range proof using the technique mentioned in the Norm Argument section. That is, by completing the square of the standard binary digit constraint $d_i(d_i - 1) = 0$, the prover can instead check

$$\left(d_i - \frac{1}{2}\right)^2 - \frac{1}{4} = 0$$

Which is suitable for a norm argument directly and reduces the witness length by half compared to the inner product argument. Taking a random linear combination of these constraints is sufficient to show that all the values $d_i \in \{0, 1\}$ and therefore that they are valid binary digits. To complete the proof, the prover must show that the appropriate 2 power linear combination of these bits equals the value.

Instead of just using powers of 2, the prover can vary the coefficients to support arbitrary ranges. To show that $A \leq v < B$ for any integers $A < B$ such that $B - A < p$, the prover will instead show that $0 \leq v - A < B - A$ by construct the base vector $\mathbf{b}$ as

$$n = \lceil \log_2(B - A) \rceil \qquad b_i = 2^i \text{ for } i = 0..n-2 \qquad b_{n-1} = (B - A) - 2^{n-1}$$

To see why this is valid, it suffices to check that the minimum binary linear combination of this base vector is 0 and the maximum is

$$b_{n-1} + \sum_{i=0}^{n-2} b_{n-2} = (B - A - 2^{n-1}) + (2^{n-1} - 1) = B - A - 1$$

This is possible because certain elements of the ranges where $B - A$ is not a power of two are representable using multiple bit patterns. Given the base vector and after committing to the value v and the digit vector $\mathbf{d}$, the verifier will select a challenge value q and the prover will show that

$$\begin{aligned}\left|\left(\mathbf{d} - \frac{1}{2}\mathbf{1}\right) + Q^{-1}\mathbf{b}\right|_q^2 &= \left|\left(\mathbf{d} - \frac{1}{2}\mathbf{1}\right)\right|_q^2 + 2v - \langle \mathbf{1}, \mathbf{b}\rangle + \left|Q^{-1}\mathbf{b}\right|_q^2 \\ &= 2v + \left|\frac{1}{2}\mathbf{1} - Q^{-1}\mathbf{b}\right|_q^2\end{aligned}$$

The final norm term is independent of any committed values, and the equality holds only if the digits are valid and $\langle \mathbf{b}, \mathbf{d}\rangle = v$. To extend this to multiple values, the prover will concatenate the digit and base vectors for each value and define a new diagonal matrix $U(x)$ which scales the digits of value i by x^{i+1}. Scaling the base vector by this matrix

$$\langle U(x)\mathbf{b}, \mathbf{d}\rangle = \sum_{i=0}^{n} x^{i+1} v_i$$

Finally, to blind the proof, the prover will use the norm version of the blinding protocol with one term in the polynomial. This requires adding an additional blinding component to the linear portions of the vectors with coefficient 0. Given the public values

$$P(x, q) = \left(\left|\frac{1}{2}\mathbf{1} - Q^{-1}\mathbf{b}\right|_q^2 - \sum_{i=0}^{n} x^{i+1} A_i\right) G + \left\langle -\frac{1}{2}\mathbf{1} + Q^{-1}U(x)\mathbf{b}, \mathbf{G}\right\rangle$$

The range proof protocol is

Algorithm 2 Binary Range Proof

Require: $A_i \leq v_i < B_i$

function BinaryProver($(A_i, B_i, C_i = \mathrm{Com}\,(v_i)) : i$)

commit $D = \textsc{BlindNormTerm}(1, 1, \mathrm{Com}_{BP}\,(\mathbf{d}, \mathbf{G}))$

query $x, q = q_s^2, r \leftarrow \mathbb{F}_p$

commit $B = \textsc{NormBlinding}(1, D, \mathbf{C})$

query $t \leftarrow \mathbb{F}_p$

$\mathbf{l}_c = (0, rt)$

$C' = P(x, q, t) + B + tD + 2t^2 \sum_{i=0}^{n} x^{i+1} C_i$

$\textsc{NormArgument}(1, 1, 1, q, q_s, \mathbf{l}_c, C')$

end function

5 Reciprocal Range Proofs

Reciprocal range proofs have many more degrees of freedom than binary range proofs. The first choice is how to represent the digits and multiplicities for each value. These can either be represented "inline" for each value or they can be "shared" between multiple values. These offer different tradeoffs. The inline case has an optimal base for a particular range $[A, B)$ given by

$$b^{b+1} = B - A$$

Some common ranges are expressible very nearly optimally, like $2^{64} = 16^{16}$, while many others are not. For this reason, it is especially important that the protocol be able to prove arbitrary ranges for arbitrary bases.

The other kind of range proof shares a single base among multiple values. In this protocol, the base can increase past the optimal value for any individual range, but the witness size is also increased by an amount equal to the size of the shared bases relative to the inline case. This kind of range proof is only more efficient for large numbers of values.

5.1 Reciprocal Argument

Naively trying to extend the binary digit check to multiple bases, one might try to simply use a larger polynomial. The check for binary digits is $x(x-1)$ which naturally generalizes to

$$d_b(x) = \prod_{i=0}^{b-1} (x - i)$$

Unfortunately, this uses $(b-1)/\log b$ multiplications which is increasing for $b \geq 2$. One can do slightly better using balanced digit representations

$$d_{2k+1} = x \prod_{i=0}^{k-1} (x^2 - i^2)$$

But this still has more multiplications per value than a binary range proof. Other more exotic techniques like using complex, i.e. $x^{2^n} - x$, bases still do not achieve a lower number of multiplications per value and are impractical for other reasons.

Rather than showing each digit is the root of some polynomial, the prover will reverse the representation and construct a polynomial that encodes the digits of our value, or their negation, as its roots. Given the digits d_i of some value in base b the polynomial $f(e)$ encoding its roots satisfies

$$f(e) = \prod_{i=0}^{n} (e + d_i) = \prod_{j=0}^{b-1} (e + j)^{m_j}$$

Where m_j is the multiplicity of the symbol j among the digits. Taking the logarithmic derivative of this equation, keeping in mind that this reduces the multiplicities mod p, the equation becomes an equality of sums of rational functions

$$\frac{f'(e)}{f(e)} = \sum_{i=0}^{n} \frac{1}{e - d_i} = \sum_{j=0}^{b-1} \frac{m_j}{e - j}$$

This moves all of the interesting data about the value, namely its digits by position and their multiplicities, into the field and suggests a zero knowledge proof protocol: commit to the digits d_i and the symbol multiplicities m_j, choose a challenge value for e, and then commit to values

$$r_i = \frac{1}{e - d_i}$$

Then, to show the values all belong to the set of valid base b symbols, prove

$$(e + d_i)r_i = 1 \quad \text{and} \quad \sum_{i=0}^{n} r_i = \sum_{j=0}^{b-1} \frac{m_j}{e + j}$$

5.1.1 Multiparty Reciprocal Argument

Notice that, unlike the polynomial representation, the reciprocal argument involves no multiplications between digits, only between a particular digit and its reciprocal. This makes the reciprocal argument especially well suited to multiparty proofs. Regardless of how the proof is organized, the sum of reciprocals can be taken over all the set membership proofs for a particular set, i.e. a particular set of digits, regardless of which ranges are known to which provers. At each point in the protocol, the provers will sum their commitments and submit the result to the random oracle. The resulting proof will be indistinguishable from a single party proof.

5.2 Arbitrary Ranges

While the arbitrary range condition is convenient for binary range proofs, it is absolutely necessary for arbitrary bases. Many ranges are fixed externally to the proof system and often do not have nice integer solutions like $4^4 = 2^8$ or $16^{16} = 2^{64}$ that allow expressing them efficiently as a power of a larger base, e.g.$10^9 < 2^{32} < 10^{10}$. Without loss of generality with respect to the lower bound, for a range $0 \leq v < B$ in base b there are three cases.

5.2.1 $b - 1 \mid B - 1$

This is the natural generalization of the binary case, in the sense that $2 - 1 = 1$ divides every range. For arbitrary bases this is obviously not the case, but when it is the prover will use the base vector $\mathbf{b}$

$$b_i = b^i \quad \text{for} \quad i < n - 1 \qquad b_{n-1} = \frac{B - b^{n-1}}{b - 1}$$

Expanding out the maximum representable value to check soundness

$$\sum_{i=0}^{n-1} (b - 1)b_i = (b^{n-1} - 1) + (B - b^{n-1}) = B - 1$$

5.2.2 $b^{n-1} < B \leq 2b^{n-1}$

When the range is not divisible by $b-1$ but is sufficiently close to the lower power of b the prover can still use n digits by making the last digit binary. To see why this works, observe that the subrange spanned by the first $n - 1$ digits is $[0, b^{n-1})$. Since this is at least half the distance to B, the prover can overlay the same range shifted up by $B - b^{n-1}$ and cover the whole range $[0, B)$. Thus in this case the base vector is

$$b_i = b^i \quad \text{for} \quad i < n - 1 \qquad b_{n-1} = B - b^{n-1}$$

5.2.3 $2b^{n-1} < B < b^n$

In the final case, the prover must add an additional digit increasing the length of $\mathbf{b}$ to $n+1$ but cannot let $b_{n-1} = b^{n-1}$, as this would allow representing values outside the range. Instead, the prover will choose some value so that the maximum representable value with the first n digits is at least half way to B and then use another binary digit to cover the upper part of the range as in the previous case.

$$b_i = b^i \quad \text{for} \quad i < n - 1$$

$$b_{n-1} = \left\lceil \frac{B - 1}{2(b - 1)} \right\rceil - \frac{b^{n-1} - 1}{b - 1} \qquad b_n = B - (b - 1)b_{n-1} - b^{n-1}$$

5.3 Inline Multiplicities

I will consider the more complex case first where the multiplicities are inline with the digits. Suppose $\mathbf{b}$ is defined as above for the base b and the range $A \leq v < B$. Then let the vector $\mathbf{d}$ be a vector of base b digits, potentially with a final binary digit if it is necessary for the range, such that $\langle \mathbf{b}, \mathbf{d} \rangle = v$. Let the vector $\mathbf{m}$ be defined in such that m_i for $i = 0..b-2$ is the number of occurrences of the symbol $i+1$ among the base b digits in the value v. If there is a binary digit, let m_{b-1} be equal to the binary digit.

Notice that it is not necessary to store a multiplicity for the zero symbol for any base because the total number of digits is a public constant, so the number of zero digits will always be the total number of digits minus the number of nonzero digits. Consider for simplicity the case where there is no bit first. The reciprocal check without a zero multiplicity is

$$\sum_{i=0}^{\text{len}(\mathbf{d})-1} \frac{1}{e+d_i} - \sum_{j=0}^{\text{len}(\mathbf{m})-1} m_j \left(\frac{1}{e+(j+1)} - \frac{1}{e} \right) = \frac{\text{len}\,(\mathbf{d})}{e}$$

When there is a bit, the prover will perform the same check on the binary digit separately. The core of the reciprocal argument remains the same as before: the prover will commit to the digits and the multiplicities, the verifier will choose the challenge e, and the prover will then commit to the reciprocals. To actually show this the prover will organize the commitments into a polynomial, show that the central coefficient encodes the constraints, and use the blinding protocol to prove knowledge of the error terms. In particular, let $\mathbf{u}(x) = x^2\mathbf{b}$ and the vectors $\mathbf{v}(x)$, and $\mathbf{c}(x)$ be defined so that

$$v_i(x) = \begin{cases} x^3 & \text{if } d_i \text{ is base 2} \\ x^5 & \text{if } d_i \text{ id base } b \\ 0 & \text{if fewer than } i+1 \text{ digits} \end{cases} \qquad c_i(x) = \begin{cases} \frac{x^3}{e} - \frac{x^3}{e+(i+1)} & \text{if } 0 \leq i < b-1 \\ \frac{x^5}{e} - \frac{x^5}{e+1} & \text{if } i = b-1 \text{ and range has bit} \\ 0 & \text{if } \mathbf{m} \text{ undefined} \end{cases}$$

Let the vector $\mathbf{a}$ be 1 on entries where digits are defined, i.e. whenever $\mathbf{b}$ is defined, and 0 elsewhere. This yields the vector valued polynomial

$$\mathbf{p}(y) = \mathbf{s} + y\mathbf{m} + y^2((e\mathbf{1}+\mathbf{d}) + Q^{-1}\mathbf{v}(x)) + y^3(\mathbf{r} + Q^{-1}\mathbf{u}(x)) + y^4Q^{-1}\mathbf{c}(x)$$

Where the y^5 term of the norm of $\mathbf{p}(y)$ is, given the public value $C(q) = \langle \mathbf{u}(x), \mathbf{v}(x) \rangle_{1/q}$

$$\left[y^5\right] |\mathbf{p}(y)|_q^2 = 2\left(x^2 \langle e\mathbf{1}+\mathbf{d}, \mathbf{b} \rangle + (\langle \mathbf{r}, \mathbf{v}(x) \rangle + \langle \mathbf{m}, \mathbf{c}(x) \rangle) + \langle \mathbf{r}, (e\mathbf{1}+\mathbf{d}) \rangle_q\right) + C(q)$$

Breaking this polynomial down further in terms of x powers, the x^2 term encodes the value, the x^3 term encodes the base 2 reciprocal sum, and the x^5 term encodes the base b reciprocal sum. The constant x term encodes multiplicative constraints as a polynomial in q. So, if the following holds, all the digits must be of the correct base and sum with the bases to product the correct value

$$\left[y^5\right] |\mathbf{p}(y)|_q^2 = 2vx^2 + \frac{2\langle \mathbf{v}(x), \mathbf{1} \rangle}{e} + 2|\mathbf{a}|_q^2 + C(q)$$

The prover can easily extend this to check to multiple values with different bases by concatenating the vectors of each value, scaling the base values of each value by distinct even powers of x, and scaling distinct base digits by distinct odd powers of x. Also note that the $\mathbf{d}$, $\mathbf{m}$, and $\mathbf{r}$ must be padded to be the same length for the purposes of concatenation.

5.3.1 Protocol

There is a certain degree of freedom in how one defines the error terms and the public terms. For the purposes of this protocol, all publicly computable products are included in the public terms, although verification might be faster if they are moved into the error terms.

$$\mathbf{n}_{pub}(x,q,y) = y^2(e\mathbf{1} + Q^{-1}\mathbf{v}(x)) + y^3Q^{-1}\mathbf{u}(x) + y^4Q^{-1}\mathbf{c}(x)$$

$$s_{pub}(x,q,y) = |\mathbf{n}_{pub}(x,q,y)|_q^2 + 2y^5\left(|\mathbf{a}|_q^2 + \sum_{j=0}^{k} \frac{\langle \mathbf{v}(x), \mathbf{1} \rangle}{e} - A_j x^{2j+2}\right)$$

$$P(x,q,y) = s_{pub}(x,q,y)G + \langle \mathbf{n}_{pub}(x,q,y), \mathbf{G} \rangle$$

With this definition of the public terms, the error terms are

$$\epsilon_0 = |\mathbf{s}|_q^2$$
$$\epsilon_1 = 2\langle \mathbf{s}, \mathbf{m} \rangle_q$$
$$\epsilon_2 = |\mathbf{m}|_q^2 + 2\left\langle \mathbf{s}, (e\mathbf{1} + \mathbf{d}) + Q^{-1}\mathbf{v}(x) \right\rangle_q$$
$$\epsilon_3 = 2\left(\left\langle \mathbf{s}, \mathbf{r} + Q^{-1}\mathbf{u}(x) \right\rangle_q + \left\langle \mathbf{m}, (e\mathbf{1} + \mathbf{d}) + Q^{-1}\mathbf{v}(x) \right\rangle_q\right)$$
$$\epsilon_4 = |\mathbf{d}|_q^2 + 2\left\langle e\mathbf{1} + Q^{-1}\mathbf{v}(x), \mathbf{d} \right\rangle_q + 2\langle \mathbf{s}, \mathbf{c}(x) \rangle + 2\left\langle \mathbf{m}, \mathbf{r} + Q^{-1}\mathbf{u}(x) \right\rangle_q$$
$$\epsilon_6 = 2\langle \mathbf{d}, \mathbf{c}(x) \rangle + |\mathbf{r}|_q^2 + 2\langle \mathbf{r}, \mathbf{u}(x) \rangle$$
$$\epsilon_7 = 2\langle \mathbf{r}, \mathbf{c}(x) \rangle$$

The reciprocal range proof uses the blinding protocol with three commitments, one for the multiplicities, one for the digits, and one for the reciprocals. The protocol is organized to multiply the digits plus e by the reciprocals. This yields a total of 11 blinding values and places the degree 7 error term in the blinding vector for R. The scalar component of the Bulletproof commitments will be used to store the degree 0 error terms.

Algorithm 3 Inline Reciprocal Range Proof

Require: $A_i \leq v_i < B_i$
  function InlineProver($(b_i, A_i, B_i, C_i = \mathrm{Com}(v_i)) : i$)
    commit $M = \mathrm{BlindIPTerm}(3, 1, \mathrm{Com}_{BP}(\mathbf{m}, \mathbf{G}))$
    commit $D = \mathrm{BlindIPTerm}(3, 2, \mathrm{Com}_{BP}(\mathbf{d}, \mathbf{G}))$
    query $e, x, r_0 \leftarrow \mathbb{F}_p$
    commit $R = \mathrm{BlindFinalIPTerm}(3, \epsilon_7, \mathrm{Com}_{BP}(\mathbf{r}, \mathbf{G}))$
    query $q = q_s^2, r_1 \leftarrow \mathbb{F}_p$
    commit $B = \mathrm{IPBlinding}(3, \epsilon, M, D, R, \mathbf{C})$
    query $y \leftarrow \mathbb{F}_p$
    $C = P(x, q, y) + B + yM + y^2D + y^3R + 2y^5 \sum_{j=0}^{k} x^{2j+2}C_j$
    $\mathbf{l}_c = r_0 r_1 (y, y^2, y^3, y^4/r_1, y^6)$
    NormArgument$(1, 1, 1, q, q_s, \mathbf{l}_c, C)$
  end function

5.4 Shared Digits

There are two versions of the shared digit protocol. In the first, all the multiplicities are shared and the multiplicity commitment M can be completely eliminated. The shared multiplicities are committed to in the D commitment instead. This reduces the number of error terms from 7 to 4 and the number of commitments by 1. The other version combines the inline and shared multiplicity range proofs. The structure of the error terms is the same as that of the inline proof, and the D commitment still commits to the shared multiplicities. The blinding protocol is generic in the number of commitments, which makes especially clear the commonalities between the two range proofs.

For ranges with shared digits, the $\mathbf{m}$ will be zeroed. The shared multiplicity vector $\mathbf{m}^{(s)}$ will contain the sum of the multiplicities for each base. In a multiparty proof, this sum can be computed without revealing the private multiplicities of each prover by simply adding their commitments. The $\mathbf{c}$ coefficients will similarly be zeroed for shared multiplicity digits and the shared coefficients $\mathbf{c}^{(s)}$ will contain the coefficient, as defined before, for each shared multiplicity.

5.4.1 Shared Multiplicity Protocol

The exclusively shared base protocol places the digits on the t term and the reciprocals on the t^2 term. This alters the error terms and the public constants in a predictable manner. Note that the blinding for the linear terms must be accounted for as well in ϵ_2

$$\epsilon_0 = |\mathbf{s}_n|_q^2$$
$$\epsilon_1 = 2\left\langle \mathbf{s}_n, e\mathbf{1} + \mathbf{d} + Q^{-1}\mathbf{v}(x)\right\rangle_q$$
$$\epsilon_2 = |\mathbf{d}|_q^2 + 2\left\langle \mathbf{s}, \mathbf{r} + Q^{-1}\mathbf{u}(x)\right\rangle_q + \left\langle \mathbf{s}_l, \mathbf{c}^{(s)}\right\rangle$$
$$\epsilon_4 = |\mathbf{r}|_q^2 + 2\left\langle \mathbf{r}, \mathbf{v}(x)\right\rangle$$

With public constants

$$\mathbf{n}_{pub}(x,q,y) = y(e\mathbf{1} + Q^{-1}\mathbf{v}(x)) + y^2 Q^{-1}\mathbf{u}(x)$$
$$s_{pub}(x,q,y) = |\mathbf{n}_{pub}(x,q,y)|_q^2 + 2y^3\left(|\mathbf{a}|_q^2 + \sum_{j=0}^{k}\frac{\langle \mathbf{v}(x), \mathbf{1}\rangle}{e} - A_j x^{2j+2}\right)$$
$$P(x,q,y) = s_{pub}(x,q,y)G + \langle \mathbf{n}_{pub}(x,q,y), \mathbf{G}\rangle$$

Algorithm 4 Shared Reciprocal Range Proof

Require: $A_i \leq v_i < B_i$

function SharedProver($(b_i, A_i, B_i, C_i = \mathrm{Com}\,(v_i)) : i$)
- **commit** $D = \text{BlindIPTerm}(2, 2, \mathrm{Com}_{BP}\left(\mathbf{m}^{(s)}, \mathbf{H}, \mathbf{d}, \mathbf{G}\right))$
- **query** $e, x \leftarrow \mathbb{F}_p$
- **commit** $R = \text{BlindFinalIPTerm}(2, \mathrm{Com}_{BP}\,(\mathbf{r}, \mathbf{G}))$
- **query** $q = q_s^2, r_0 \leftarrow \mathbb{F}_p$
- **commit** $B = \text{IPBlinding}(2, \epsilon, D, R, \mathbf{C})$
- **quert** $t \leftarrow \mathbb{F}_p$
- $C = P(x,q,t) + B + tD + t^2R + 2t^3\sum_{j=0}^{k} x^{2j+2}C_j$
- $\mathbf{l}_c = r_0(t, t^2, t^4) \oplus t^2\mathbf{c}^{(s)}$
- NormArgument$(1, 1, 1, q, q_s, \mathbf{l}_c, C)$

end function

5.4.2 Inline and Shared Multiplicity Protocol

The combined inline and shared multiplicity protocol will use the same public values as the inline protocol and the same D commitment with the shared multiplicities as the shared protocol. The error terms will be the same as the inline protocol with the exception of the degree 3 error term which must compensate for the linear blinding values as the degree 2 error term did in the shared protocol.

$$\epsilon_3 = 2\left(\left\langle \mathbf{s}, \mathbf{r} + Q^{-1}\mathbf{u}(x)\right\rangle_q + \left\langle \mathbf{m}, (e\mathbf{1} + \mathbf{d}) + Q^{-1}\mathbf{v}(x)\right\rangle_q\right) - \left\langle \mathbf{s}_l, \mathbf{c}^{(s)}\right\rangle$$

Algorithm 5 Combined Inline Shared Reciprocal Range Proof

Require: $A_i \leq v_i < B_i$

function InlineSharedProver($(b_i, A_i, B_i, C_i = \mathrm{Com}\,(v_i)) : i$)
- **commit** $M = \text{BlindIPTerm}(3, 1, \mathrm{Com}_{BP}\,(\mathbf{m}, \mathbf{G}))$
- **commit** $D = \text{BlindIPTerm}(3, 2, \mathrm{Com}_{BP}\left(\mathbf{m}^{(s)}, \mathbf{H}, \mathbf{d}, \mathbf{G}\right))$
- **query** $e, x, r_0 \leftarrow \mathbb{F}_p$
- **commit** $R = \text{BlindFinalIPTerm}(3, \epsilon_7, \mathrm{Com}_{BP}\,(\mathbf{r}, \mathbf{G}))$
- **query** $q = q_s^2, r_1 \leftarrow \mathbb{F}_p$
- **commit** $B = \text{IPBlinding}(3, \epsilon, M, D, R, \mathbf{C})$
- **quert** $t \leftarrow \mathbb{F}_p$
- $C = P(x,q,t) + B + tM + t^2D + t^3R + 2t^5\sum_{j=0}^{k} x^{2j+2}C_j$
- $\mathbf{l}_c = r_0 r_1(t, t^2, t^3, t^4/r_1, t^6) \oplus t^3\mathbf{c}^{(s)}$
- NormArgument$(1, 1, 1, q, q_s, \mathbf{l}_c, C)$

end function

6 Confidential Transactions

Range proofs are typically used as part of a larger confidential transaction protocol. A confidential transaction is one in which the amounts, types, and other transaction data are kept hidden and transaction validity is established with a zero knowledge proof. This is in contrast to ordinary transactions, like Bitcoin and Ethereum, which do not hide the transaction data and anonymity oriented transaction protocols like ZCash [11] and Monero [6], which also attempt to hide the inter-transaction metadata. Anonymity oriented protocols usually also hide the transaction amounts, and in some cases include a self contained confidential transaction protocol within the larger transaction protocol like Monero.

Confidential transaction protocols typically work by proving that the sum of the inputs equals the sum of the outputs, called conservation of money, and that the outputs commit to positive integers much smaller than the field characteristic. It is not necessary to prove range proofs for the inputs, as they usually have already been proven to satisify the range proof constraint by the transaction which created them.

6.1 Existing Conservation of Money

Conservation of money proofs can surprisingly lightweight, as in the case of Mimblewimble. In that case, the Pedersen commitments to the inputs are subtracted from Pedersen commitments to the outputs, and the result is opened to a commitment to zero. This net commitment is then added to the total net commitment, which allows verification of the chain using only the unspent transaction outputs and the openings from each transaction. Intermediate output commitments can be discarded.

Multitype conservation of money proofs are more complicated. Confidential assets [12] use a distinct, uniformly randomly selected curve point to represent different types of currency. Netting these commitments will produce a commitment to zero only if all the types net to zero, up to the hardness of finding a discrete log relation between the random curve points. However, each output must also be shown to use a curve point corresponding to a type of currency, which is done using ring signatures. This has similar tradeoffs to Monero transactions between type privacy, i.e the number of types an output could be, and transaction size which increases linearly in the number of potential types and the number of inputs.

Bulletproof based implementations of confidential assets like Cloak [4] use a combination of gadgets, essentially small proof fragments, composed into an arithmetic circuit. These gadgets first shuffle the inputs in zero knowledge, followed by a series of conditional mixing operations, followed by another shuffle, followed by another series of splitting operations back out to the outputs of the transaction. The shuffle is an implementation of the same polynomial based technique of Groth [3].

This avoids many of the tradeoffs of confidential assets, but produces a fairly large circuit. Each additional element in the shuffle uses a two multiplications and each mix uses one multiplication. Each factor in the multiplication adds two additional scalars to the witness and each mix gate adds four additional scalars to the witness. In total, the circuit seems to require $8(m-1)$ multiplications and $20n-14$ scalars not counting the inputs or outputs, where n is the maximum of either the number of inputs or the number of outputs. Encoding this circuit using the Bulletproof arithmetic circuit protocol directly will approximately duplicate the additional witness size as all of the multiplications are sequential, so each value will appear twice, except for the factors in the shuffle polynomial which could be linearly combined first.

6.2 Argument

Given two sets I and O of pairs of amounts and types encoded as field elements, the inputs and outputs respectively, the prover wants to show that the amounts of all the inputs of each type add up to the same value as the amounts of all the outputs of the same type for all types. They can encode this as a reciprocal relation by encoding the type as the symbol in the denominator and the amount as the multiplicity in the numerator for a reciprocal term

$$\sum_{(v,t)\in I} \frac{v}{e+t} - \sum_{(v,t)\in O} \frac{v}{e+t} = 0$$

While the reciprocal argument in the range proof is more efficient than the equivalent argument using the preimages of the logarithmic derivative, i.e. polynomials, the multiplicities are small enough that it could plausibly work without being prohibitively inefficient. In this case, the multiplicities are arbitrary

integers, typically around 64 bits in length, which would make the equivalent polynomial argument prohibitively expensive.

Translating this into a system of constraints, the prover will follow the same recipe as the range proofs, with a few modifications. The prover will commit to $\mathbf{t}$ and $\mathbf{v}$, the verifier will choose e, and the prover will commit to a vector of "reciprocals" such that

$$r_i = \frac{v_i}{e + t_i}$$

Because of the similarity in structure to the range proofs, the prover can actually prepend the $\mathbf{t}$ vector to the vector of digits and the type reciprocals to the vector of digit reciprocals. These r_i are importantly different since their numerator is not one, so instead of checking that the product of the digit vector and the reciprocal vector is one for these components

$$(e + t_i)r_i = v_i$$

To put the values in the proper place to balance this equation, the prover will scale the input commitments by $x^{2(i+1)} + q^{2(i+1)}$ and let $\mathbf{a}$ be zero on the type reciprocals. These coefficients will also scale the types in the input commitments, which the prover must show equal the types in the digit vector. To do this, the prover will choose some value x' at the same time as q and set the type component in the linear vector to $-x'$.

Correspondingly, the prover will modify the $\mathbf{u}(x)$ to multiply the types in the norm vector by $qx^{2i+2} + x'q^{2i+2}$ to verify that these value are equal to the types in the commitments. They must also modify $\mathbf{v}(x)$ so that input reciprocals and output reciprocals are scaled by opposite sign.

$$u_i(x, x', q) = x'x^{2i+2} + x'q^{2i+2} \text{ and } c_i(x) = 0 \text{ for } i = 0..k-1$$

$$v_i(x) = \begin{cases} x & \text{if commitment } i \text{ is input} \\ -x & \text{if commitment } i \text{ is output} \end{cases}$$

6.2.1 Multiparty Conservation of Money

In the same way as the reciprocal argument in the range proof, the reciprocal argument for typed values is linear in the reciprocals. Since it is assumed that each type value pair is known to at least one of the provers, as they are committed together, there are no multiplications of secret values from different provers necessary to perform the reciprocal argument for typed inputs. This means the exact same multiparty protocol as for reciprocal range proofs can be used for multiparty confidential transactions.

6.3 Protocol

For confidential transactions, one typically does not want to prove range proofs for the inputs. For this purpose, each transaction input commitment will have a binary flag o_i that is 1 for outputs and 0 for inputs. Technically, there is no reason why a transaction could not prove the inputs lie in ranges, or remove the range proofs for outputs. However, most of the configurability of the protocol is not important for correctness and this is the most common case.

For inputs, that is values with $o_i = 0$, the prover will remove all the corresponding digit, multiplicity, and reciprocal components for the witness. This results in several changes to the other public constants. In particular, the protocol will not subtract A_i if $o_i = 0$ in the public constants and will modify the u_i for the type components. To support public inputs or outputs, for example to support fees, the public constants will also be modified to subtract the associated reciprocals.

$$u_i = o_i x' x^{2i+2} + x' q^{2i}$$

$$v_{pub}(e, x, q) = \sum_{(v,t,o) \in Pub} \frac{(-1)^o v}{e + t} \text{ where } Pub \text{ is the set of public amounts}$$

$$s_{pub}(x, q, y) = |\mathbf{n}_{pub}(x, q, y)|_q^2 + 2y^5 \left(|\mathbf{a}|_q^2 - v_{pub}(e, x, q) + \sum_{j=0}^{k} \frac{\langle \mathbf{v}(x), \mathbf{1} \rangle}{e} - o_j A_j x^{2j+2} \right)$$

With these modifications, the protocol is essentially a super set of the inline and shared multiplicity range proofs for the outputs.

Algorithm 6 Confidential Transactions

Require: $A_i \leq v_i < B_i, \sum_{i:t_i=t}(-1)^{o_i} v_i = 0$ for all t

function InlineSharedProver($(b_i, o_i A_i, B_i, C_i = \text{Com}(v_i, t_i)) : i$)

commit $M = \text{BlindIPTerm}(3, 1, \text{Com}_{BP}(\mathbf{m}, \mathbf{G}))$

commit $D = \text{BlindIPTerm}(3, 2, \text{Com}_{BP}(\mathbf{m}^{(s)}, \mathbf{H}, \mathbf{d}, \mathbf{G}))$

query $e, x, r_0 \leftarrow \mathbb{F}_p$

commit $R = \text{BlindFinalIPTerm}(3, \epsilon_7, \text{Com}_{BP}(\mathbf{r}, \mathbf{G}))$

query $q = q_s^2, r_1 \leftarrow \mathbb{F}_p$

commit $B = \text{IPBlinding}(3, M, D, R)$

quert $t \leftarrow \mathbb{F}_p$

$C = P(x, q, t) + B + tM + t^2 D + t^3 R + 2t^5 \sum_{j=0}^{k}(o_j x^{2j+2} + q^{2j+2}) C_j$

$\mathbf{l}_c = r_0 r_1 (t, t^2, t^3, t^4/r_1, t^6) \oplus t^3 \mathbf{c}^{(s)}$

$\text{NormArgument}(1, 1, 1, q, q_s, \mathbf{l}_c, C)$

end function

7 Arithmetic Circuits

Since the norm argument is equivalently powerful to the inner product argument, it is possible to adapt the arithmetic circuit protocols of Bulletproofs and Bulletproofs+ to the norm linear arguments of Bulletproofs++. The differences are relatively minor, although the witnesses are smaller than both and the proofs can also be smaller. Given public constraints matrices $A, B, C \in \mathbb{F}_p^{n \times m}$, public $\mathbf{d} \in \mathbb{F}_p^m$, and matrix $D \in \mathbb{F}^{k \times m}$ with empty right null space, i.e. $D\mathbf{x} = \mathbf{0}$ only if $\mathbf{x} = \mathbf{0}$, the prover wants to show

$$A\mathbf{a} + B\mathbf{b} + C\mathbf{c} = D\mathbf{v} + \mathbf{d} \qquad a_i b_i = c_i$$

For input commitments $C_i = \text{Com}(v_i)$, the proof will use the same version of the blinding protocol used by the reciprocal range proof, with a slight modification for the inputs and with fewer rounds since there are no reciprocals to compute. After committing to the witness

$$C_a = \text{Com}_{BP}(\mathbf{a}) \qquad C_b = \text{Com}_{BP}(\mathbf{b}) \qquad C_c = \text{Com}_{BP}(\mathbf{c})$$

The verifier will then choose some r uniformly at random to construct the vector $\mathbf{r}$ given by $r_i = r^{i+1}$. The prover will left multiply all the constraint matrices by this vector to collapse the constraints into a single vector constraint. By Schwartz-Zippel, this constraint is satisfied only if all the original constraints are satisfied. The polynomial for the blinding protocol is then

$$\mathbf{p}(y) = \mathbf{bl} + y\mathbf{c} + y^2(\mathbf{a} + Q^{-1}B^\top \mathbf{r}) + y^3(\mathbf{b} + Q^{-1}A^\top \mathbf{r}) + y^4(Q^{-1}C^\top \mathbf{r} - \mathbf{1})$$

This places the constraints on the y^5 coefficient once again with public value $C(q, r) = \left\langle B^\top \mathbf{r}, A^\top \mathbf{r} \right\rangle_{1/q}$

$$\left[y^5\right] |\mathbf{p}(y)|_q^2 = \mathbf{r}^\top (A\mathbf{a} + B\mathbf{b} + C\mathbf{c}) + \langle \mathbf{a}, \mathbf{b} \rangle_q - \langle \mathbf{1}, \mathbf{c} \rangle_q + C(q, r) = \langle \mathbf{r}, \mathbf{d} \rangle$$

Given $(k-1)2^n < \text{len}(\mathbf{a}) \leq k2^n$ where $k = 2, 3$ this will use n rounds, and have a witness of size $(4 + 2n)g + (1 + k)s$. In particular, this proof will be either one or two commitments smaller than a comparable Bulletproof+ arithmetic circuit proof. Due to the shorter witness size, proving and verification should also be faster.

7.1 Products in Linear Portion

Alternatively, if the structure of the protocol to generate the constraints permits, it is possible to commit to portions of the $\mathbf{c}$ vector in the linear portion of the commitments to $\mathbf{a}$ and $\mathbf{b}$ and move the constraints such that they appropriate scale the halves of $\mathbf{c}$ in each commitment

$$C_a = \text{Com}_{BP}(\mathbf{c}_0, \mathbf{H}; \mathbf{a}, \mathbf{G}) \qquad C_b = \text{Com}_{BP}(\mathbf{c}_1, \mathbf{H}; \mathbf{b}, \mathbf{G})$$

$$\mathbf{l}_c = y^1 \left(Q^{-1}C^\top \mathbf{r} - \mathbf{1}\right)_1 + y^2 \left(Q^{-1}C^\top \mathbf{r} - \mathbf{1}\right)_0$$

This will save the commitment to $\mathbf{c}$ and will in the worst case add an additional scalar to the reduced linear vector so the proof size will remain the same. Typically, if the bit length of $5 + \text{len}(\mathbf{c})$ is the same as the bit length of $\text{len}(\mathbf{c})$ this will save one scalar as compared to the original variant. This does increase the witness size by half as compared to the other protocol, but it is still smaller than the Bulletproof+ witness.

8 Performance

There are three aspects of performance to compare: proof size, proving time, and verification time. Proof size is easy to calculate for a particular configuration. Unfortunately, I have not yet completed a competitive implementation of the range proofs which makes it hard to compare the proving and verification times empirically. However, it is possible to estimate the amount of time proving and verification would take by counting the number of elliptic curve scalar multiplications necessary to prove or verify, as they are by far the most expensive operations.

8.1 Proving and Verification Time

To estimate the differences in proving and verification time without a concrete implementation, it is possible to count the number of elliptic curve scalar multiplications. Empirical comparisons will likely be fairly similar, as the elliptic curve scalar multiplications make up the significant majority of the computation. The function $r(n) = 2\lceil \log_2 n \rceil$ will be used to count the number of rounds

8.1.1 Arguments

For a fixed vector length of n the norm argument will use approximately $3n$ scalar multiplications to prove while the inner product argument will use $4n$. This is because the inner product argument has two vectors and each (L, R) commits to an amount of scalars equal to the length of the vectors, and committing to X in the norm argument requires twice as many scalars as R. Taking the geometric sum over all the rounds yields the total number of scalar multiplications.

8.1.2 Range Proofs

The blinding commitment will require one scalar multiplication per element of the witness as well as a quadratic number in the number of terms. It is likely possible to reduce this, but the number of terms is fixed per protocol. Reciprocals also require a scalar multiplication. The witness itself is negligible for the purposes of this estimation, as the digits and multiplicities are small.

8.1.3 Proving

To estimate the cost of proving, I will count three quantities: the number of inner products P, the number of scalar multiplications S, and the number of basis updates B. The logic here is that there is a fixed cost per inner product and a fixed cost per scalar multiplcation. In reality, the cost can depend on the number of scalar multiplications in an inner product. Basis updates are counted separately since they can be performed more efficiently.

Since Bulletproofs+ blind after the inner product argument, they cannot support multiparty proving without some modifications to blind before the witnesses from different provers are combined in the inner product argument. For this reason, it is more appropriate to compare Bulletproofs++ to Bulletproofs as they support an equivalent set of features. To convert from Bulletproofs to Bulletproofs+ it is sufficient to subtract $n + 2$ scalar multiplications from the Bulletproof operation count.

I will consider three cases: n inputs of b bits each in a binary range proof, n inputs of 64 bits each in an inline base 16 range proof, and n inputs of 64 bits each in a base 256 range proof. In all three proofs, the number of basis updates and scalar multiplications performed as part of the arguments is proportional to the witness size.

Table 1: Range Proofs Verification EC Scalar Multiplications

Type	**Param**	**BP++**	**BP**
Binary	$k = nb$	$(4k + 5)S + kB + (r(k) + 2)P$	$(6k + 3)S + (2k)B + (r(k) + 3)P$
Base 16 Inline	$k = 16n$	$(5k + 24)S + kB + (r(k) + 4)P$	$(24k + 3)S + (8k)B + (r(4k) + 3)P$
Base 256 Shared	$k = 8n$	$(5k + 520)S + kB + (r(k) + 3)P$	$(48k + 3)S + (16k)B + (r(8k) + 3)P$

8.1.4 Verification

In the case of verification, the verifier performs one inner product, so it is only necessary to count the number of scalar multiplications. The verifier will perform one scalar multiplication per commitment, one per input, one per element of the witness, and two per round. This is true for all three Bulletproof protocols, which vary only in how many commitments are part of the transcript and how large the witnesses are. For the purposes of comparison, Bulletproof and Bulletproof+ verification are essentially the same complexity with Bulletproofs+ requiring one fewer scalar multiplication. For n inputs with either b bits each, or 64 bits in the case of the latter two rows, the number of elliptic curve scalar multiplications to verify is

Table 2: Range Proofs Verification EC Scalar Multiplications

Type	**BP++**	**BP+**	**Ratio** $n \to \infty$
Binary	$2 + n + nb + r(nb)$	$5 + n + 2nb + r(nb)$	(1 + b) / (1 + 2 b)
Base 16 Inline	$7 + 17n + r(16n)$	$5 + 129n + r(128n)$	$17/129 \approx 0.132$
Base 16 Shared	$260 + 9n + r(8n)$	Same	$9/129 \approx 0.0698$

8.2 Proof Size

I will use the same three bases, binary, base 16 inline, and base 256 share, to compare proof size. There is a much larger configuration space for Bulletproofs++, so if the range is not fixed externally it may be possible to choose ranges that are slightly smaller or larger to save space by choosing a range divisible by one minus the base.

To make comparisons independent of the elliptic curve, I will write them as a linear combination of g and s for group element size and scalar size respectively. Common curve instantiations will be $g = 32.125$ and $s = 32$ bytes for $p \sim 2^{256}$ and $g = s = 32$ for $p \sim 2^{255}$.

Table 3: Binary Range Proofs

Range	**BP++**	**BP+**	**Δ**	**BP**	**Δ**
1x8	4g+5s	9g + 3s	5g - 2s	10g + 5s	6g
2x8	6g+5s	11g + 3s	5g - 2s	12g + 5s	6g
3x8	8g+4s	13g + 3s	5g - s	12g + 5s	6g + s
1x16	6g+5s	11g + 3s	5g - 2s	12g + 5s	6g
2x16	8g+5s	13g + 3s	5g - 2s	14g + 5s	6g
3x16	10g+4s	15g + 3s	5g - s	14g + 5s	6g + s
1x32	8g+5s	13g + 3s	5g - 2s	14g + 5s	6g
2x32	10g+5s	15g + 3s	5g - 2s	16g + 5s	6g
3x32	12g+4s	17g + 3s	5g - s	16g + 5s	6g + s

Table 4: Inline Base 16 Proof Size

Range	**BP++**	**BP+**	**Δ**	**BP**	**Δ**
1x64	10g+3s	15g + 3s	5g	16g + 5s	6g + 2s
2x64	10g+5s	17g + 3s	7g - 2s	18g + 5s	8g
3x64	12g+4s	19g + 3s	7g - s	20g + 5s	8g + s
4x64	12g+5s	19g + 3s	7g - 2s	20g + 5s	8g
5x64	14g+4s	21g + 3s	7g - s	22g + 5s	8g + s
6x64	14g+4s	21g + 3s	7g - s	22g + 5s	8g + s
7x64	14g+5s	21g + 3s	7g - 2s	22g + 5s	8g
8x64	14g+5s	21g + 3s	7g - 2s	22g + 5s	8g

Table 5: Shared Base Proofs Size

Range	**Base**	**BP++**	**BP+**	**Δ**	**BP**	**Δ**
32x64	64	17g+4s	25g + 3s	9g - s	26g + 5s	9g + s
64x64	256	19g+4s	27g + 3s	9g - s	28g + 5s	9g + s
96x64	256	19g+5s	29g + 3s	10g - 2s	30g + 5s	11g
128x64	256	19g+5s	29g + 3s	10g - 2s	30g + 5s	11g
192x64	256	19g+5s	29g + 3s	10g - 2s	30g + 5s	11g
256x64	256	21g+5s	31g + 3s	10g - 2s	32g + 5s	11g
384x64	256	21g+5s	31g + 3s	10g - 2s	32g + 5s	11g

References

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

[2] Heewon Chung et al. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. Cryptology ePrint Archive, Report 2020/735. https://ia.cr/2020/735. 2020.

[3] Stephanie Bayer and Jens Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 263–280. ISBN: 978-3-642-29011-4.

[4] Oleg Andreev and Andrei Ivasko. *cloak*. https://github.com/stellar/slingshot/blob/main/spacesuit/spec.md.

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

[6] Koe, Kurt M. Alonzo, and Sarang Noether. *Zero to Monero: Second Edition*. https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf. 2020.

[7] Tom Elvis Jedusor. *MIMBLEWIMBLE*. https://docs.beam.mw/Mimblewimble.pdf. 2016.

[8] *Grin*. https://grin.mw/.

[9] *Beam*. https://beam.mw/.

[10] Jonathan Bootle et al. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 327–357. ISBN: 978-3-662-49896-5.

[11] Nathan Wilcox Sean Bowe Taylor Hornby. *Zcash Protocol Specification*. https://github.com/zcash/zips/blob/main/protocol/protocol.pdf. 2022.

[12] Andrew Poelstra et al. *Confidential Assets*. https://blockstream.com/bitcoin17-final41.pdf. 2017.

[13] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006, pp. 126–127.

[14] 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.

[15] Ivan Bjerre Damgård and Gudmund Skovbjerg Frandsen. "Efficient algorithms for the gcd and cubic residuosity in the ring of Eisenstein integers". In: *Journal of Symbolic Computation* 39.6 (2005), pp. 643–652. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2004.02.006. URL: https://www.sciencedirect.com/science/article/pii/S0747717105000362.

[16] Daniel Lichtblau. "Half-GCD and fast rational recovery". In: *Proceedings of the 2005 international symposium on Symbolic and algebraic computation*. 2005, pp. 231–236.

[17] damiano and Pete L. Clark. *Quadratic forms over finite fields*. https://mathoverflow.net/questions/17103/quadratic-forms-over-finite-fields. 2010.

9 Proofs

There are three portions to this section. The first deals with the new norm argument and shows its soundness in a manner basically identically to the inner product argument. The responses in the norm argument are not blinded and are not zero knowledge. The essential property that guarantees this section is the linear independence as e polynomials of the C, X, and R coefficients.

The second section shows the SHVZK, CWEE, and soundness of the blinding procedure. In this section, all commitments are shown to be blinded by the leftover degrees of freedom from the error term blinding, or an additional linear blinding value if these are insufficient.

Given these two sections, the correctness of each protocol follows from merely showing that arguments are sound, in particular the reciprocal argument, and that it is possible to extract openings for the inputs to the proofs. The soundness arguments basically all follow from multiple applications of the Schwartz-Zippel lemma and extraction of the inputs from the linear independence, as polynomials in x, of the input coefficients.

9.1 Norm Arguments

The norm arguments are not zero knowledge, so it is sufficient to show that they are sound and that there exists an extractor for the witnesses. In the case of reduction to the inner product argument, it is sufficient to show that the change of basis is invertible. This implies that no known discrete log relations can be known about the new basis, as they would imply discrete log relations in the old basis. The soundness follows from simplification of the algebra.

9.1.1 Norm Argument

To extract the witness for the new norm argument, it is sufficient to show an extractor for a single round of the argument and apply this extractor repeatedly. The existence of the extractor follows directly from the invertibility of the matrix of challenge functions evaluated at three points. These points are distinct with overwhelming probability, and the matrix of challenge functions is

$$\begin{bmatrix} 1 & e_0 & e_0^2 - 1 \\ 1 & e_1 & e_1^2 - 1 \\ 1 & e_2 & e_2^2 - 1 \end{bmatrix} = V(\mathbf{e}) \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where $V(\mathbf{e})$ is the Vandermonde matrix, which is invertible if all the challenge points are distinct. Inverting this matrix allows extracting the witness from three evaluations of one round of the argument. Soundness follows from the Schwartz-Zippel lemma applied to the polynomial relation.

9.2 Blinding

The generic blinding protocol does much of the heavy lifting for the range proofs. In particular, apart from the inputs, it is sufficient to show that there exists an extractor for the blinding procedure to extract the witnesses for all the particular proofs. Also, it is sufficient to show that the blinding procedure is SHVZK to show that all the other protocols are as well.

9.2.1 Soundness

The soundness of the blinding protocol follows directly from the computations describing the blinding protocol. That is, expanding the coefficient $2n - 1$ or $2n$ depending which version is used yields an expression that is, with overwhelming probability, zero only if the system of constraints is satisfied.

9.2.2 Witness Extraction

To extract the witnesses of all the commitments C_i and B, note that each component of C' is a t polynomial in the witnesses of C_i and B, as well as "the" input. For the purposes of the blinding argument, the protocol assumes that there is only one input and leaves it up to the specific protocols to handle multiple inputs.

Given a total of $n + 2$ commitments, the emulator $n + 2$ will be run times on t, construct the Vandermonde matrix, and with overwhelming probability invert this matrix. Applying this to the matrix of openings yields the witnesses for all the commitments.

9.2.3 Zero Knowledge

To show that the protocol is SHVZK, it is sufficient to show that given a transcript and witness there exists blinding values which open the commitments of the transcript to the given witness.

The total number of blinding values in commitment 1 is $2n-2$ and in commitments $i = 2..n$ it is $2n-i$. This makes the total number of blinding values

$$\dim(\mathbf{b}) = -1 + \sum_{i=1}^{n} 2n - i = 2n^2 - \binom{n+1}{2} - 1$$

Each commitment constrains the blinding values, as does each opening of the blinding values. There are $n+1$ commitments and $2n$ openings for a total number of constraints

$$\dim(\mathbf{c}) = 3n + 1$$

The proof transcript corresponds to a matrix of constraints M dependent on the discrete logs of the basis points and the t coefficients. This matrix satisfies

$$M\mathbf{b} = \mathbf{c}$$

For the proof to be zero knowledge, there must exist a $\mathbf{b}$ for every $\mathbf{c}$. The space of possible $\mathbf{c}$ values is constrained by one dimension, as in every valid proof the blinding values cancel in the evaluation of the error polynomial. So, in order for $\mathbf{b}$ to exist for every $\mathbf{c}$ coming from a valid proof, M must have rank $3n$.

It is assumed that the constraints coming from the discrete log problem are linearly independent from the t constraints, since otherwise the prover would have randomly sampled a discrete log relation among the basis points which occurs only with negligible probability. These constraints are also linearly independent of each other, so they can be safely removed from consideration. It is sufficient to show that the remaining constraints have rank $2n-1$.

To show this, it is sufficient to demonstrate the existence of $2n-1$ linearly independent vectors in the image of M. There are $2n-2$ vectors that come from zeroing all the blinding except for each component in the first commitment. That is

$$\mathbf{v}_i \text{ such that } v_{ii} = t,\ v_{i,i+1} = -1 \text{ where } i \neq 2n-2, 2n$$

All lie in the image of the matrix M by setting the i blinding value in C_1 to 1. The final basis element can be constructed from C_2

$$\mathbf{w} \text{ such that } w_{n-2} = t^2,\ w_n = -1$$

Since all of the v_i are zero on the n component, $\mathbf{w}$ is linearly independent of them. Since it is also in the image, the image must have dimension at least $2n-1$. Since the image was already shown to have at most dimension $2n-1$, the dimension of the image must be exactly $2n-1$ so long as there are at least two commitments. In the case of 1 non-blinding commitment, it is sufficient to simply add another blinding component with coefficient 0.

In the case of a square, the basic argument remains the same with some slight adjustments to the structure of the image and number of blinding values. The number of blinding values in commitment C_i is $2n-i$ for all the commitments, and the image can be constructed purely from the first commitment. However, the number of blinding values is still insufficient in the case of $n = 1$.

9.3 Specific Protocols

I will show witness extraction and soundness for three protocols: the binary range proof, confidential transaction proof, and arithmetic circuit proof. The other reciprocal range proof variants follow trivially from the confidential transaction proof proof by simply not using various features of the larger typed protocol and potentially using a different version of the blinding protocol.

These proofs will deal only with extraction of the input commitments and soundness of the protocols. Zero knowledge follows from the zero knowledge property of the blinding portion of the protocol, detailed in the previous section, and the fact that each input is also blinded. The witness extraction for all three protocols will use the following Vandermonde argument.

Given a set of witness n vectors $\mathbf{w}_i$ from each input commitment, and an opening of

$$\mathbf{w}'(x) = \sum_{i=1}^{n-1} x^i \mathbf{w}_i$$

The emulator will be run n times to obtain n evaluations of the above vector valued polynomial. Let the matrix $W'(\mathbf{x})$ have the vectors $\mathbf{w}'(x_j)$ as its rows, where x_j is the j challenge. The modified Vandermonde matrix

$$V_{ij}(\mathbf{x}) = x_j^{i+1}$$

Takes a vector of coefficients and evaluates them as a polynomial at the point x_i, in this case with 0 constant term, For distinct values of x_i not equal to zero this matrix is invertible. Letting W be the matrix $\mathbf{w}_i$ as its rows

$$W'(\mathbf{x}) = V(\mathbf{x})W \qquad W = V(\mathbf{x})^{-1}W'(\mathbf{x})$$

So, after n calls to the random oracle, the emulator can invert the Vandermonde matrix with overwhelming probability and solve for the openings of the inputs. Note that this is true for witnesses of any structure, including witnesses with only a value and blinding or witnesses also including a type.

9.3.1 Binary Range Proof

Extraction of the inputs proceeds according to the Vandermonde extractor applied to the challenge value x in the protocol. It is applied to witness vectors containing the blinding term on the first component of the linear vector of the bulletproof and the value term on the scalar component of the bulletproof.

Soundness of the protocol, following the application of the blinding argument, follows from the expansion of the t^2 term in the blinding polynomial. This system of constraints encodes the quadratic constraints

$$(x - 1/2)^2 - 1/4 = x(x - 1)$$

Which are zero only on binary digits. The computation of the base values ensures that all the representable values lie in the desired range. Therefore the proof is sound.

9.3.2 Typed Reciprocal Range Proof

Extraction in the typed reciprocal range proof invokes the Vandermonde extractor using the challenge value x from step 7 of the protocol. The witness vectors have the type values on the first linear component of the bulletproof, blinding values on the second linear component of the bulletproof, and amounts on the scalar component.

Soundness of the protocol follows from the expansion of the central coefficient properly encoding the constraints and then from the soundness of the reciprocal system of constraints. The soundness of the reciprocal system of constraints follows from the soundness of the reciprocal argument, shown below, and the fact that all the bases represent every value in the range, which also shown.

9.3.3 Reciprocal Argument

Here I show the soundness of the following general version of the reciprocal argument: the prover commits to $\mathbf{v}$ of numerators and $\mathbf{t}$ of roots of the denominators, the verifier chooses e, and the prover commits to the vector $\mathbf{r}$ of rational functions. Assuming that the prover shows

$$(e + t_i)r_i = v_i \qquad \sum_i r_i = 0$$

It follows that all of the sums of v_i with the same denominator equal zero

$$\sum_{t=t_i} v_i = 0$$

Since all of the multiplicative constraints hold, it must be the case that either

$$r_i = \frac{v_i}{e + t_i} \qquad e + t_i = 0, v_i = 0$$

The latter case is equivalent to the challenge e being a root of the polynomial

$$\prod_i (e + t_i) = g(e) = 0$$

Which occurs with negligible probability by the Schwartz-Zippel lemma. Therefore, with overwhelming probability the r_i are equal to the correct rational functions. Given this, the sum of r_i is

$$\sum_i \frac{v_i}{e + r_i} = \frac{f'(e)}{f(e)}$$

Since $g(e) \mid f(e)$ and $g(e) = 0$ with negligible probability, this expression is zero only if $f'(e) = 0$. Once again, this polynomial is zero at a uniformly random e with negligible probability by the Schwartz Zippel lemma. Therefore, if this sum is zero the rational function must be identically zero. This can only all the values with common denominators cancel, as each distinct reciprocal monomial is linearly independent over base field.

9.3.4 Arithmetic Circuit

Given that D is of full column rank, multiplication by D is invertible in the sense that there exists a D' such that

$$D'(D\mathbf{x}) = \mathbf{x}$$

When invoking the Vandermonde extractor in this case, the matrix will also be multiplied by the matrix D' to extract the witness vectors. The witnesses in the arithmetic circuit protocol consist of a blinding value on the first linear term and a value in the scalar component of the bulletproof vectors. The challenge will be the value r such that $r_i = r^{i+1}$. The extracted witnesses will satisfy

$$W'(\mathbf{x}) = V(\mathbf{x})DW \qquad W = D'V(\mathbf{x})^{-1}W'(\mathbf{x})$$

The soundness of the protocol follows from algebraic simplification of the central coefficient. Whether the system of constraints given by A, B, C, and D actually encode the problem they are intended to is outside the scope of this protocol, although there do exist automated conversion tools from certain classes of programs to arithmetic circuits which are provably correct [1].

10 Appendix

10.1 Half GCD for Euclidean Domains

Both of these common cases of complex multiplication, by $i^4 = 1$ and $\omega^3 = 1$, for Euclidean domains, it is also possible to use the extended Euclidean algorithm to find the reduced representation of e. This will also work for other CM fields that are Euclidean domains, like $\alpha = \sqrt{-7}$, but not in general. To find the reduced representative, first one must find a factorization of the field characteristic p as

$$p = u + v\alpha^* \qquad \alpha^* \equiv -\frac{u}{v} \mod p$$

This computation can be carried out once, ahead of time, for the field. Then, to reduce a general $e \in \mathbb{F}_p$ one can compute the reduced representation by "dividing" by the representation of p, rounding the resulting components to integers, scaling by the quotient by the p representation, and finally subtracting it from the original value. Because the scaled quotient is zero in field, the values are equal.

$$\frac{e}{u + v\alpha^*} = \frac{(eu)}{p} + \frac{ev}{p}\alpha = q_0 + q_1\alpha$$

$$e - (\lceil q_0 \rfloor + \alpha \lceil q_0 \rfloor)(u + v\alpha^*) = e_0 + e_1\alpha \equiv e \mod p$$

This optimization is well known [13] and related to GLV [14] techniques for scalar multiplication. Up to sign, e_0 and e_1 are half the field length. Then, the protocol can use a half gcd algorithm for the CM field to find a small rational reconstruction for the reduced representation of e.

There are several techniques for fast gcd, and related euclidean type computations like residue symbols, for the euclidean fields given by small roots of unity [15], but apparently no specialized algorithms for the extended euclidean algorithm or half gcd. I think it is possible to use generic half gcd algorithm like [16] with quotient computations substituted for the above quotient operation. The only additional complication is splitting the numbers in half in a manner guaranteed to produce the minimal solution.

10.2 General Sum of Squares to Inner Product

It is possible to transform a sum of squares to an inner product in a finite field where -1 is not a quadratic residue, although it is not possible via a simple scaling of the inputs as it is in the case when -1 is a quadratic residue. Recall that every quadratic form $Q(\mathbf{x})$ has an associated symmetric bilinear form $B(\mathbf{x}, \mathbf{y})$ such that

$$Q(\mathbf{x}) = B(\mathbf{x}, \mathbf{x}) \qquad 2B(\mathbf{x}, \mathbf{y}) = Q(\mathbf{x} + \mathbf{y}) - Q(\mathbf{x}) - Q(\mathbf{y})$$

This form is sufficient expand $Q(\mathbf{x})$ evaluated over linear combinations of vectors as

$$Q\left(\sum_i a_i \mathbf{u}_i\right) = \sum_i a_i^2 Q(\mathbf{u}_i) + \sum_{i<j} 2a_i a_j B(\mathbf{u}_i, \mathbf{u}_j)$$

Given vectors where $Q(\mathbf{u}_i) = 0$, the expanded expression will have only with cross terms. If it is also possible make all but adjacent cross terms cancel, the transformation will have taken the sum of squares to an inner product. This procedure follows that discussed here [17]. To begin, note that every quadratic form in at least three variables is ansiotropic, i.e. has a nontrivial zero.

Rather than considering the entire quadratic form together, it can instead break it down into three dimensional quadratic subforms. Since the sum of squares is symmetic, given a solution for the generic ternary sum of squares

$$Q_3(x, y, z) = x^2 + y^2 + z^2$$

The same solution will work for all the subforms. To solve the three dimensional case, one will first find linearly independent vectors $Q_3(\mathbf{u}) = Q_3(\mathbf{v}) = 0$ where $B(\mathbf{u}, \mathbf{v}) = 1$. Let $\mathbf{w}$ span the complement of $\mathbf{u}$ and $\mathbf{v}$. By elementary linear algebra, the kernel of $\mathbf{x} \mapsto B_3(\mathbf{u}, \mathbf{x})$ is at least two dimension, contains $\mathbf{u}$, and does not contain the span of $\mathbf{v}$. Therefore, it must contain $\mathbf{w}$. Likewise, for $\mathbf{v}$ and $\mathbf{u}$. Therefore

$$B_3(\mathbf{u}, \mathbf{w}) = B_3(\mathbf{v}, \mathbf{w}) = 0$$

The change of variables by these vectors yields

$$Q_3(a\mathbf{u} + b\mathbf{v} + c\mathbf{w}) = ab + c^2 Q_3(\mathbf{w})$$

For a sum of squares, these vectors can concretely instantiate given a solution $x^2 + y^2 + 1^2 = 0$, which is given by normalizing any solution so $z = 1$. Then

$$\mathbf{u} = (x, y, 1) \qquad \mathbf{v} = (-y, x, 1) \qquad \mathbf{w} = (x - y, x + y, 1)$$

Which yields $Q_3(\mathbf{w}) = -1$. Returning to the original quadratic form $Q(\mathbf{x})$, the variables can be partitioned into groups of three, plus at most two remaining squares. Letting T apply the above linear transformation to each pair, given the number of squares $3n + r$

$$Q(T\mathbf{x}) = \sum_{j=0}^{r} x_{3n+j}^2 + \sum_{i=0}^{n-1} x_{3i}x_{3i+1} - x_{3i+2}^2$$

Pairing up the remainder terms with two of the negated squares, these differences can be factored into products and the procedure repeated on the the remaining $n - r$ values. In the worst case, the final round of the protocol will either include one or two squares, which can be written as the value times itself for one square or a sum of squares

$$(a^2 + b^2) = (1/2)(a + b)(a + b) + (1/2)(b - a)(a - b)$$

Each round has a corresponding linear transformation of the basis vectors by the inverse transpose of the witness transformation. Because each transformation is the same, except for the remained terms, and is only applied locally, the transformation can use a similar technique to the fast scalar multiplications in the norm arguments. The total length of the inner product vectors is given by the geometric sum of thirds times the length n, or

$$\frac{n/3}{1 - 1/3} = \frac{n}{2}$$

Up to the remainder terms. Then same sort of transformation can be applied to arbitrary quadratic forms, but it is not possible in general to handle all triples of variables simultaneously, and the change of basis operation will generally require much more computation.