





Bulletproofs++: Next Generation Confidential Transactions via Reciprocal Set Membership Arguments

Liam Eagen, Sanket Kanjalkar, Tim Ruffing , and Jonas Nick 

Blockstream Research, Victoria, Canada
jonas@n-ck.net

Abstract. Zero-knowledge proofs are a cryptographic cornerstone of privacy-preserving technologies such as “Confidential Transactions” (CT), which aims at hiding monetary amounts in cryptocurrency transactions. Due to its asymptotically logarithmic proof size and transparent setup, most state-of-the-art CT protocols use the Bulletproofs (BP) [8] zero-knowledge proof system for set membership proofs such as range proofs. However, even taking into account recent efficiency improvements, BP comes with a serious overhead in terms of concrete proof size as well as verifier running time and thus puts a large burden on practical deployments of CT and its extensions.

In this work, we introduce Bulletproofs++ (BP++), a drop-in replacement for BP that improves its concrete efficiency and compactness significantly. As for BP, the security of BP++ relies only on the hardness of the discrete logarithm problem in the random oracle model, and BP++ retains all features of Bulletproofs including transparent setup and support for proof aggregation, multi-party proving and batch verification. Asymptotically, BP++ range proofs require only $O(n/\log n)$ group scalar multiplications compared to $O(n)$ for BP and BP+.

At the heart of our construction are novel techniques for permutation and set membership, enabling highly efficient proofs of statements encoded as arithmetic circuits. Concretely, a single BP++ range proof to establish that a committed value is in a 64-bit range (as commonly required by CT) is just 416 bytes over a 256-bit elliptic curve, 38% smaller than an equivalent BP and 27% smaller than BP+. When instantiated on the secp256k1 curve as used in Bitcoin, our benchmarks show that proving is about 5 times faster than BP and verification is about 3 times faster than BP+. When aggregating 32 range proofs, proving and verification are about 9.5 times and 5.5 times faster, respectively.

1 Introduction

Cryptocurrencies like Bitcoin [40] enable decentralized, peer-to-peer payments by maintaining a distributed public ledger called the blockchain. While this innovation has permitted an unprecedented degree of financial autonomy on the Internet, the fact that every transaction leaves a permanent record in the

blockchain poses a substantial threat to the financial privacy of users. Even though cryptocurrency transactions are not typically associated with real-world identities, a surprisingly large amount of information can be extracted from the information in the blockchain [24, 38, 48].

Among the most glaring pieces of data that an observer can extract are the amounts of funds that transactions move from sender to recipient. These monetary amounts are stored as plain integers in many popular cryptocurrencies, including Bitcoin, which makes it easy for blockchain nodes to verify that a transaction is balanced, i.e., that the sum of all its input amounts equals the sum of all its output amounts (except for a small fee given to the miners).

Confidential Transactions. A common countermeasure to this leak of information, e.g., as suggested first in the “Confidential Transactions” proposal [26, 37] (CT), is to hide the monetary amounts in homomorphic commitments such as Pedersen commitments. The additive homomorphism ensures that blockchain nodes can verify the amounts in a confidential transaction without learning the plain amounts, by performing the necessary additions for checking the balance equation on the homomorphic commitments instead of the plain amounts. However, this approach is only sound if the amounts do not overflow during the homomorphic addition, because this would allow an attacker to violate balance and thus create money out of thin air. To exclude overflow, transactions are required to carry a non-interactive zero-knowledge (NIZK) *range proof* that demonstrates that committed amounts are in a range $[0, 2^b)$ of non-negative integers much smaller than the message space of the commitment space.

Bulletproofs. Motivated by this application, the seminal Bulletproofs (BP) by Bünz *et al.* [8] was the first to achieve range proofs with an asymptotic size logarithmic in the number of bits in the range as well as concrete sizes less than 1 kB. Moreover, BP supports aggregate proving, i.e., a single range proof can cover multiple commitments at once, and this proof is significantly more compact than proving each commitment separately. This efficiency makes it feasible to use BP in cryptocurrencies, and BP range proofs have been successfully deployed in Grin [27] and Monero [39] in conjunction with other privacy-preserving features.

However, even though Monero has subsequently upgraded [47] to Chung *et al.* [15]’s recent improvement Bulletproofs+ (BP+), which reduces the size of a single 64-bit range proof to 576 bytes, range proofs still account for 29% to 42% of the size of a typical Monero transaction.¹ These concrete storage costs as well as the concrete verification efficiency still leave much to be desired, considering that all nodes in a cryptocurrency are required to download and verify the entirety of all range proofs created within the system.

¹ A transaction with one input and two outputs has a size of about 1530 bytes, and a transaction with two inputs and two outputs has a size of about 2220 bytes after the v15 hardfork [47]. In either case, the aggregated range proof covering the two output amounts has a size of 640 bytes on a 256-bit elliptic curve (see also Table 1).

Multi-asset Confidential Transactions (MACT). While the initial CT proposal [37] supports only a single asset (e.g., only Bitcoin), the protocol by Poelstra *et al.* [43] (as deployed for instance in the Liquid sidechain [41]) extends the idea to *multi-asset confidential transactions* (MACT), i.e., a single transaction can transfer multiple assets simultaneously, and no observer can learn the transacted amounts or the involved assets. Moreover, the range proof construction used in this protocol supports multi-party proving for transactions created by multiple senders. This is a prerequisite to using coin mixing protocols [36] on top of MACT, which further enhance privacy.

However, it is thus far unclear how to fully leverage the potential of BP in MACT protocols. While it is possible to implement the range proofs in MACT using BP, the protocol by Poelstra *et al.* [43] requires additional zero-knowledge *surjection proofs* to show that the assets on the output side of the transaction are a permutation of the assets on the input side of the transactions. These additional proofs are large and since they are constructed using techniques different from BP, it is not possible to aggregate them together with BP range proofs. The approach taken by the Cloak [50] MACT protocol overcomes this problem by using BP to encode a permutation argument as an arithmetic circuit. This avoids surjection proofs, but the way the circuit is constructed makes it incompatible with known multi-party proving techniques for BP. In summary, there is currently no solution to MACT that is practical and compatible with BP.

1.1 Contributions

The main contribution of this work is Bulletproofs++ (BP++), a zero-knowledge argument of knowledge for arithmetic circuits in the discrete logarithm setting.

Reciprocal Argument. At the core of BP++ is the *reciprocal argument*, a novel interactive argument protocol that generalizes permutation arguments and set membership arguments. This approach builds on the work by Bayer, Groth [3], who encode a multiset as the roots of a polynomial, and whose basic technique has been extended to show richer permutation arguments in plookup [21] and plays a critical role in protocols based on Plonk [22]. These protocols use a “grand product”, i.e., the product of numerous committed values, to show that a particular permutation, which encodes the structure of an arithmetic circuit, was applied correctly. The reciprocal argument of BP++ is essentially the logarithmic derivative of the polynomials used by Bayer-Groth permutation arguments. The logarithmic derivative transforms a product of linear factors into a sum, thereby linearizing the representation of the multiset.

Since the initial publication of a preprint of our work, the reciprocal argument has already been used in several other works: Haböck [31] modifies the “grand product” of Hyperplonk [13] to use a variant of the reciprocal argument, which he rederives via the logarithmic derivative. Eagen, Fiore, Gabizon [18] develop a more asymptotically and concretely performant lookup argument, improving

Table 1. Range proof sizes compared to previous work. The range column of the table ($m \times n$) indicates the number of aggregated proofs (m) and bits of range proven (n) by each proof. We express the resulting proof size in terms of the number of group elements g and scalars s . Values denoted by dash (—) are not provided in the Flashproofs paper [51].

Range	BP++	BP+	BP	SwiftRange	Flashproofs
1×64	$10g + 3s$	$15g + 3s$	$16g + 5s$	$16g + 9s$	$17g + 10s$
2×64	$10g + 5s$	$17g + 3s$	$18g + 5s$	$20g + 9s$	$27g + 17s$
8×64	$14g + 5s$	$21g + 3s$	$22g + 5s$	$28g + 9s$	$65g + 28s$
16×64	$15g + 4s$	$23g + 3s$	$24g + 5s$	$32g + 9s$	$103g + 48s$
64×64	$19g + 4s$	$27g + 3s$	$28g + 5s$	$40g + 9s$	—

upon the sequence of works beginning with Caulk [45, 54].² As evident from these works, the reciprocal argument is clearly of independent interest.

Compactness and Efficiency. BP++’s novel techniques improve the compactness and efficiency of BP(+) significantly. Table 1 compares the size of BP++ range proofs with SwiftRange [52], Flashproofs [51], BP [8] and BP+ [15] range proofs. As demonstrated by the table, BP++ has a clear advantage in terms of proof size compared to the alternatives.

The time needed for proving and verification is dominated in practice by multiplications of group elements with scalars. In BP and BP+ range proofs, the count of these multiplications scales linearly with n . However, BP++ offers an asymptotic improvement, reducing the count to $O(n/\log n)$. The benchmarks in Sect. 7 demonstrate that BP++’s improvements do in fact translate to actual implementations. A 64-bit range proof takes roughly 4 ms for proving and 0.9 ms for verifying, making it $5\times$ quicker than BP in proving and $3\times$ quicker in verification.

Modularity Without Sacrificing Performance. Since BP++ is capable of proving arbitrary statements encoded in arithmetic circuits, it is possible to construct range proofs and MACT simply by an arithmetic circuit that encodes the relation. As opposed to BP(+), we adopted this methodology because our techniques allow for the creation of a range proof that is nearly as efficient as a direct construction of such a proof. Our approach simplifies the security analysis of range proofs and MACT, as they inherit the security properties of the arithmetic circuit protocol, providing the circuit accurately encodes the relation.

² After a per-table setup procedure, these arguments allow the prover to construct an argument for correctness of table look ups in time independent of the table size. This case is particularly interesting, as it is not currently known how to construct an analogous product check that depends only on the number of non-identity values being multiplied.

This demonstrates the potential of reusing the BP++ arithmetic circuit protocol with the reciprocal argument in other applications.

MACT. On the MACT side, we introduce a BP++ MACT protocol (again by specifying an arithmetic circuit) that relies on the same asset representation as Cloak but uses an instance of the reciprocal argument, substantially simplifying the permutation argument. The marginal cost of a BP++ MACT over an aggregated range proof is negligible in prover and verifier time, and proof size.

Compatibility with BP. Since BP++ maintains the same interface and security assumptions established by BP(+), BP++ is a drop-in replacement for existing uses of BP(+). For example, BP range proofs in existing protocols like Grin [27], Monero [39], and Liquid [41] can be replaced without any change in security assumptions and with only minimal modification to existing protocols. This is also true for statements encoded as general arithmetic circuits. Moreover, the MACT protocol uses the same asset representation as Cloak, and so can be directly substituted for Cloak for smaller proof sizes and faster prover and verifier. These replacements retain all benefits of BP:

Aggregate proving. A prover who would like to prove multiple statements simultaneously can create a single aggregated proof, which is more compact than simply giving multiple independent proofs. For example, in the common case that a cryptocurrency transaction creates $m \geq 1$ commitments, an aggregate range proof can prove that m committed values are in range in just $O(\log n + \log m)$ bits, instead of $m \cdot O(\log n)$ bits in the case of m separate range proofs.

Multi-party aggregate proving. For the case that multiple provers want to create a single aggregated proof, BP++ offers a natural MPC protocol. Multi-party proving yields large space savings when CT is combined with coin mixing protocols [46].

Batch verification. Multiple (possibly aggregated) proofs can be verified in a batch computation, improving efficiency further.

Conservative cryptographic assumptions. BP++ is provably secure assuming only the hardness of the discrete logarithm problem and can be made non-interactive in the random oracle model, thus ensuring compatibility with assumptions widely accepted by engineers and users in the cryptocurrency ecosystem. Concretely, BP++ neither requires pairings nor cycles of curves and can be instantiated on the secp256k1 elliptic curve which is used in Bitcoin, for which a wide range of implementations exist.

Transparent setup. Since the public setup parameters only consist of random group elements, the setup is trustless assuming a common random string or the random oracle model.

1.2 Related Work

Range Proofs. An alternative to digit decomposition range proofs are those based on Lagrange’s four square theorem. This theorem states that any positive integer

can be written as a sum of four squares, as originally proposed by Lipmaa [35]. In practice, this is often transformed to an instance of the three square theorem as was originally observed by Groth [28]. To show that a value $v < B$ one can find a four, or three, square decomposition of the value $B - v$, which is positive only if the initial condition is met. These protocols require integer commitments, which require either RSA groups, and hence a trusted setup, or ideal class groups.

More recently, Couteau *et al.* [16] developed a bounded integer commitment protocol that requires only the discrete logarithm assumption in a group of known order. This allows them to construct three-square range proofs using elliptic curves, which are highly performant and smaller than BP and BP+ range proofs. However, BP++ range proofs remain smaller as compared to their approach. Moreover, since their bounded integer commitment scheme requires the committed values to remain in a bounded interval, their approach requires a curve with order somewhat larger than 256 bits at the 128-bit security level. This lower bound on the group size or, equivalently, on the security of their approach is inherent and applies even if one ignores the non-tightness of the security analysis when setting parameters, as often done in practice. This limits their applicability to existing blockchains.

MACT. As explained above, the original Confidential Assets protocol [43] uses surjection proofs to hide the asset type of each output from a set of possible assets. In general, the size of this set is equal to the number m of inputs to the transaction. Thus, for n outputs, the prover will do $O(n \cdot m)$ work as compared to only $O(n + m)$ for BP++. Since it is not known how to aggregate surjection proofs, the proof size is in $O(n \cdot m)$.

Cloak [50] uses a more complex construction to encode a permutation over the assets into a BP circuit. This approach is a large constant factor more expensive than BP++ in terms of prover work.

Generalizations of BP. There are a number of other works building on BP, including BP+ [15] which uses a weighted inner product argument to reduce proving time and uses several other improvements to reduce proof size, and Flashproofs [51] which combine the BP inner product argument with Groth polynomial commitments [29] to reduce verifier complexity and attempt to minimize Ethereum gas costs. There has also been work to unify BP with the large, existing body of work on Sigma protocols [1], and to further generalize this to other related contexts like groups of unknown order [10] to support homomorphic commitments of arbitrary order. BP have also been generalized to inner product arguments in other contexts, including by Lee [33], who propose a general purpose SNARK protocol over a pairing friendly curve that uses an inner product to avoid trusted setup requirements. BP are also core to the structure of Halo [6] and Halo2 [49], which are now implemented in Zcash [7] and have inspired the development of accumulation schemes [9]. These allow a prover to efficiently aggregate multiple proofs in such a way that verification time depends only on the time to verify a single proof.

2 Preliminaries

Notation. Hereafter, we denote the set of *polynomially-bounded* functions in the security parameter λ by $poly = \{f : \exists a \in \mathbb{N}, f(\lambda) \in O(\lambda^a)\}$, the set of *negligible* functions in the security parameter λ by $negl = \{f : f(\lambda)^{-1} \notin poly\}$. A function f is *overwhelming* if $1 - f$ is negligible.

A probabilistic interactive Turing machine \mathcal{A} is *probabilistic polynomial-time (PPT)* if its runtime is in $poly$; it is *probabilistic expected polynomial-time (expected-PPT)* if its expected runtime is in $poly$; it is *deterministic polynomial-time (DPT)* if it is PPT and does not read from its randomness tape.

We denote by \mathbb{G} a cyclic group of prime order p written additively, which is in practice typically a subgroup of an elliptic curve. We write group elements in \mathbb{G} with capital letters and scalars in $\mathbb{F} := \mathbb{F}_p$ with lower case letters. We write $\mathbb{F}[X]$ for the ring of polynomials over \mathbb{F} in indeterminate X ; when we treat it a vector space, then as vector space over the field \mathbb{F} .

Vectors. Vectors are written with bold letters, and matrices with capital letters. These can be distinguished from \mathbb{G} elements from context. We write the diagonal matrix of powers of μ starting with μ^0 as $\text{diag}(\mu)$. Vectors are zero indexed and implicitly padded with zeros on the right as necessary for various operations to be well-defined, i.e. addition and inner products. We denote the vector of all zeros by $\mathbf{0}$ and the vector of all ones by $\mathbf{1}$. We use $|\mathbf{v}|$ to denote the length of \mathbf{v} . We use “slice” notation $\mathbf{v}_{i:j}$ to denote the subvector of \mathbf{v} consisting of components i to $j - 1$; we may omit i if $i = 0$, and j if $j = |\mathbf{v}| - 1$. To access a component of a slice, we write $(\mathbf{v}_{i:})_k = \mathbf{v}_{i+k}$.

We write the inner product of two vectors using angle brackets and an optional subscript to denote weighting by powers of the subscript. If the subscript is not present, it is implicitly 1. Inner products are defined for any vectors of quantities that can be multiplied, i.e. scalars and scalars or scalars and group elements. The norm of a vector refers to its self inner product and uses the same subscripting convention for weights. For example, the weighted inner product of \mathbf{x} and \mathbf{G} and the weighted norm of \mathbf{x} are written

$$\langle \mathbf{x}, \mathbf{G} \rangle_\mu = \sum_{i=0} x_i G_i \mu^{i+1} \quad \text{and} \quad |\mathbf{x}|_\mu^2 = \langle \mathbf{x}, \mathbf{x} \rangle_\mu.$$

We write concatenation of vectors using $\|$, the component-wise product of vectors (Hadamard product) using \circ and tensor product of vectors using \otimes . An iterated tensor product is evaluated from left to right and obeys the convention

$$\bigotimes_{i=0}^n (1, x_i) = \left(1, x_0, x_1, x_0 x_1, x_2, \dots, \prod_{i=0}^n x_i \right).$$

This is convenient for describing, e.g., the vector of challenges used by the verifier for the norm linear argument.

We denote the vector of powers from μ^0 to μ^{n-1} by $\mathbf{e}_n(\mu)$. It obeys the tensor product equation

$$\mathbf{e}_{ab}(\mu) = \mathbf{e}_a(\mu) \otimes \mathbf{e}_b(\mu^a) = (1, \mu, \dots, \mu^{ab-1}).$$

We decompose vectors into subvectors of even (indices 0, 2, ...) and odd (indices 1, 3, ...) components, instead of left and right halves as in BP, written as written as $[\mathbf{a}]_0$ and $[\mathbf{a}]_1$ respectively. This transformation simplifies certain parts of the protocol, and may help with locality in implementations. BP and BP+ can easily be modified to use even and odd halves, as can BP++ to use left and right halves.

Discrete Logarithm Relation Problem. BP++ is provably secure assuming the expected-PPT hardness of the *discrete logarithm relation (DLR) problem*, which is well-known to be tightly equivalent to the standard discrete logarithm problem [32, Lemma 3].

Definition 1 (Discrete Logarithm Relation (DLR) Problem). *The discrete logarithm relation (DLR) problem in \mathbb{G} is hard if for all $n \geq 1$ and for all expected-PPT adversaries \mathcal{A} ,*

$$\Pr[\langle \mathbf{a}, \mathbf{G} \rangle = 0_{\mathbb{G}} \wedge \mathbf{a} \neq \mathbf{0} \mid \mathbb{G} \leftarrow \text{Setup}(1^\lambda); \mathbf{G} \leftarrow_s \mathbb{G}^n; \mathbf{a} \leftarrow \mathcal{A}(\mathbf{G})] \leq \text{negl}(\lambda).$$

2.1 Zero-Knowledge Arguments of Knowledge

A zero-knowledge argument of knowledge consists of a non-interactive PPT Turing machine \mathcal{K} which outputs a *common random string* σ , and two interactive PPT Turing machines \mathcal{P} (prover) and \mathcal{V} (verifier). Critically, the randomness used by \mathcal{K} is public and σ can be reproduced transparently (no trusted setup). The prover and verifier interacting will produce a transcript π and output a bit b indicating whether the verifier accepts, which we write $\pi \leftarrow \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) \rangle = b$. Here, for any σ , a value w is a *witness* for a *statement* u if it satisfies the polynomial time relation $(\sigma, u, w) \in \mathcal{R}$.

A zero-knowledge argument of knowledge must satisfy completeness, soundness, and zero-knowledge.

Definition 2 (Completeness). *The protocol $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ satisfies perfect completeness if for all PPT \mathcal{A} ,*

$$\Pr \left[\begin{array}{l} \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) \rangle = 1 \\ \vee (\sigma, u, w) \notin \mathcal{R} \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda); \\ (u, w) \leftarrow \mathcal{A}(\sigma) \end{array} \right] = 1.$$

The soundness notion we consider in this work is computational witness-extended emulation [30, 34].

Definition 3 (Computational Witness-Extended Emulation). *The protocol $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation (WEE) if for all DPT provers \mathcal{P}^* , there exists an expected-PPT emulator $\mathcal{E}^{\mathcal{O}}$ with access to rewinding oracle $\mathcal{O} = \langle \mathcal{P}^*(\sigma, u, s), \mathcal{V}(\sigma, u) \rangle$ such that for all pairs of adversaries $(\mathcal{A}_1, \mathcal{A}_2)$,*

$$\left| \Pr [\mathcal{A}_2(\sigma, \pi) = 1 \mid \sigma \leftarrow \mathcal{K}(1^\lambda); (u, s) \leftarrow \mathcal{A}_1(\sigma); \pi \leftarrow \mathcal{O}] - \Pr \left[\begin{array}{l} (\pi \text{ is accepting}) \Rightarrow \\ (\sigma, u, w) \in \mathcal{R} \\ \wedge \mathcal{A}_2(\sigma, \pi) = 1 \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda); \\ (u, s) \leftarrow \mathcal{A}_1(\sigma); \\ (\pi, w) \leftarrow \mathcal{E}^{\mathcal{O}}(\sigma, u) \end{array} \right] \right| \leq \text{negl}(\lambda).$$

The protocol has computational witness-extended emulation (CWEE) when adversaries \mathcal{A}_1 and \mathcal{A}_2 are restricted to non-uniform polynomial time.

In the zero-knowledge notion used in this work, the simulator has access to randomness used by the verifier; this is commonly called “special” zero-knowledge in the literature and requires the protocol to be public coin.

Definition 4 (Public Coin). The protocol $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is public coin if the i -th message sent by $\mathcal{V}(\sigma, u; \rho)$ is the i -th component of its randomness argument ρ .

Definition 5 (Perfect Special Honest Verifier Zero-Knowledge). The protocol $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ has perfect Special Honest Verifier Zero-Knowledge (SHVZK) if there exists a PPT simulator \mathcal{S} such that for all pairs of adversaries $(\mathcal{A}_1, \mathcal{A}_2)$,

$$\begin{aligned} & \Pr \left[\begin{array}{l} (\sigma, u, w) \in \mathcal{R} \\ \wedge \mathcal{A}_2(\sigma, \pi) = 1 \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda); (u, w, \rho) \leftarrow \mathcal{A}_1(\sigma); \\ \pi \leftarrow \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u; \rho) \rangle \end{array} \right] \\ &= \Pr \left[\begin{array}{l} (\sigma, u, w) \in \mathcal{R} \\ \wedge \mathcal{A}_2(\sigma, \pi) = 1 \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda); (u, w, \rho) \leftarrow \mathcal{A}_1(\sigma); \\ \pi \leftarrow \mathcal{S}(u, \rho) \end{array} \right]. \end{aligned}$$

General Forking Lemma. To show CWEE, we will use the generalized forking lemma by Bootle *et al.* [5]. It allows handling extractors for multi-round zero-knowledge argument of knowledge generically.

Trustless Common Setup. As a convention, all zero-knowledge arguments in this work use the same setup algorithm \mathcal{K} , which outputs $\sigma = (G, \mathbf{H}, \mathbf{G})$, where G and the components of the two vectors \mathbf{H}, \mathbf{G} (of sufficient size, which will be clear from the context) are random generators in \mathbb{G} . Since \mathcal{K} is transparent, it is possible to use make the setup trustless in the random oracle model.

Non-interactive Proofs from Fiat-Shamir. All zero-knowledge arguments presented in this paper are public coin, interactive protocols between a prover and honest verifier. This means that they can be made non-interactive via the Fiat-Shamir transform [4], and honest-verifier zero-knowledge of the interactive protocols immediately implies that the Fiat-Shamir transformed variants are non-interactive zero-knowledge in the random oracle model. Recent work has shown that also soundness is retained, even for multi-round protocols [2, 25, 53]. Concretely, we establish that our protocols achieve special soundness, which implies that their Fiat-Shamir version achieves knowledge soundness as shown by Attema, Fehr, Kloofß [2, Theorem 4] and further elaborated on by Ganesh *et al.* [23, Section 2.8].

Commitments as Inputs. Our zero-knowledge arguments accept witness inputs in Pedersen vector commitments. For convenience later, given generators $\sigma = (G, \mathbf{H}, \dots)$ from the zero-knowledge setup, we define a commitment to message \mathbf{v} with randomness s to be $\text{Com}(\mathbf{v}; s) = v_0 G + s H_0 + \langle \mathbf{v}_{1:}, \mathbf{H}_{8:} \rangle$. Generators $\mathbf{H}_{1:8} = (H_1, \dots, H_7)$ are intentionally not used for commitments; this will simplify the notation in later sections.

Pedersen commitments are homomorphic, perfectly hiding, and computationally binding up to the hardness of the discrete logarithm relation problem. We omit a formal treatment of these properties because the security analysis of our protocols uses the underlying group directly and does not invoke these abstract properties.

3 Technical Overview

BP++ consists of four primary improvements over earlier, transparent discrete logarithm-based range proof protocols. First, we substitute the BP+ inner product argument by a norm argument, which reduces verifier time by approximately half in many common cases. Second, we introduce a novel set membership and permutation argument called the reciprocal argument, which has already found significant applications beyond BP++. Third, we modify the BP arithmetic circuit protocol to accomplish “blinding” in one round of communication of a single group element, which can be easily adapted to other similarly constructed protocols. These modified circuits are extended to support first order use of the reciprocal argument, similarly to integration of plookup [21] into Halo2 [49]. Finally, we use these techniques to construct the shortest, and most verifier performant transparent range proof and MACT protocols.

3.1 Recap: Bulletproofs and Bulletproofs+

BP, at its core, uses a recursive argument to show the inner product relation

$$\mathcal{R}_{ip} = \left\{ \left(\begin{array}{l} \mathbf{G}, \mathbf{H} \in \mathbb{G}^n, G \in \mathbb{G}; \\ C \in \mathbb{G}; \mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^n \end{array} \right) : C = \langle \mathbf{x}, \mathbf{y} \rangle G + \langle \mathbf{x}, \mathbf{G} \rangle + \langle \mathbf{y}, \mathbf{H} \rangle \right\}. \quad (1)$$

The recursive structure of the argument is itself derived from the recursive structure in Bootle *et al.* [5]. In each round, a commitment to a scalar v and vectors \mathbf{x} and \mathbf{y} of length n is reduced to a commitment to vectors \mathbf{x}' and \mathbf{y}' of length $n/2$. If this commitment satisfies the relation, then the original commitment satisfies the relation with overwhelming probability.

In our notation, given a commitment C , the prover sends the verifier commitments (L, R) , and the verifier chooses a challenge γ . The reduced commitment is defined as

$$C' = C + \gamma^{-2}L + \gamma^2R = v'G + \langle \mathbf{x}', \mathbf{G}' \rangle + \langle \mathbf{y}', \mathbf{H}' \rangle. \quad (2)$$

Each round of the protocol forms essentially a vector valued polynomial commitment. The key to ensuring that the reduced vectors are of length $n/2$ comes from the folding relation. The reduced vectors are defined, in terms of the challenge

$$\mathbf{x}' = \gamma[\mathbf{x}]_0 + \gamma^{-1}[\mathbf{x}]_1 \quad \mathbf{y}' = \gamma^{-1}[\mathbf{y}]_0 + \gamma[\mathbf{y}]_1. \quad (3)$$

Computing the inner product of these vectors as polynomials in γ , we find that the original inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ from the inner product relation occurs as the γ^0 term

$$\langle \mathbf{x}', \mathbf{y}' \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \gamma^2 \langle [\mathbf{x}]_0, [\mathbf{y}]_1 \rangle + \gamma^{-2} \langle [\mathbf{x}]_1, [\mathbf{y}]_0 \rangle. \quad (4)$$

BP applies this same relation to the inner products between the basis points \mathbf{G} and \mathbf{H} and the witness vectors. That is, the reduced basis points are defined in terms of γ to be

$$\mathbf{G}' = \gamma^{-1}[\mathbf{G}]_0 + \gamma[\mathbf{G}]_1 \quad \mathbf{H}' = \gamma[\mathbf{H}]_0 + \gamma^{-1}[\mathbf{H}]_1. \tag{5}$$

This means when the inner products $\langle \mathbf{x}', \mathbf{G}' \rangle$ and $\langle \mathbf{y}', \mathbf{H}' \rangle$ are evaluated, the original inner products will appear on the γ^0 term. The γ^{-2} coefficients from all three reduced inner products are then collected into L and likewise the γ^2 coefficients into R . This reduction is applied until the reduced vectors are of length 2, at which point the reduced vectors are sent to the verifier.

BP+ uses a very similar recursive structure that also incorporates weights to show a weighted inner product relation, with the inner product replaced by a weighted inner product.

3.2 Reciprocal Argument

The primary technique that makes BP++ range proofs and MACT possible is a simple interactive protocol called the *reciprocal argument*. It operates on *collections* that are finite sets A of pairs (m, s) consisting of symbols $s \in \mathbb{F}$ with associated multiplicities $m \in \mathbb{F}$. In more details, the reciprocal argument lets a prover convince a verifier that the total multiplicity $\hat{m}_s = \sum_{(m', s') \in A: s'=s} m'$ of each symbol $s \in \mathbb{F}$ *vanishes* (i.e., equals zero). In that case, we also say that A itself vanishes. (For example, $A = \{(-3, 42), (5, 17), (7, 42), (-4, 42), (-5, 17), (0, 1)\}$ vanishes.) In the protocols we will construct, some or all of the m and s may be private to the prover and thus appear only in committed form.

Vanishing is powerful enough to express many relations commonly used to construct zero-knowledge arguments: For example, assuming that no wrap-around occurs when summing up multiplicities, which is guaranteed if $|A| \ll |\mathbb{F}|$, some (committed) sequence U is a permutation of another (committed) sequence T if and only if $A = \{(-1, u) : u \in U\} \cup \{(1, t) : t \in T\}$ vanishes. As a second example, consider a “lookup argument”: the components of U form a subset of a some public set T (called “table”) if, for each $t \in T$, there exists a multiplicity m_t (only known to the prover) such that $A = \{(-1, u) : u \in U\} \cup \{(m_t, t) : t \in T\}$ vanishes.

The underlying idea of the protocol is that we can associate to A a rational function $f_A(X)$ defined as a sum of reciprocals such that for all $(m, s) \in A$, $f_A(X)$ has a pole $-s$ of multiplicity m :

$$f_A(X) = \sum_{(m,s) \in A} \frac{m}{X + s}. \tag{6}$$

Function f_A vanishes (i.e., is zero everywhere) if and only if the total multiplicity \hat{m}_s for each symbol s vanishes. To show that this function vanishes, it suffices to evaluate it at a uniformly random input X . In the reciprocal argument protocol, this input is a challenge chosen by verifier after the prover has committed to A . We note that the function f_A has the structure of a logarithmic derivative, see the full version [20] for more background.

Application to Range Proofs. Consider the problem of constructing a range proof. We want to prove that some (committed) integer value v is in a range $[0, b^k)$. A natural solution is to consider the k base- b digits d_i of v and use a lookup argument (as described above) that shows that all digits d_i occur in the “table” $T = \{0, \dots, b - 1\}$. In that case, the rational function $f_A(X)$ is

$$f_A(X) = \sum_i \frac{-1}{X + d_i} + \sum_{j=0}^{b-1} \frac{m_j}{X + j}. \quad (7)$$

In contrast, both BP and BP+ construct a range proof by proving the validity of each digit *individually*, then showing that the linear combination of these digits equals the committed value. Binary digits (i.e., $b = 2$) are used since their validity can be checked with just one multiplication per digit: $d \in \{0, 1\}$ if and only if $d(d - 1) = 0$.

However, Camenisch, Chaabouni, shelat [11] suggest to select b such that $b^b \approx B - A$. This base uses only $O(n/\log n)$ digits, where $n = \lceil \log_2(B - A) \rceil$, which is optimal in the sense that the witness length is a function of the base b and the number n of digits and is minimized when they are equal. Unfortunately, the natural generalization of the binary digit check $d_i(d_i - 1)$ to bases $b > 2$ does not result in a more efficient proof in BP. In the binary case, each digit requires a single multiplication, but the number of multiplications increases linearly in the size of the base.

In BP++, we sidestep this performance trade-off via the reciprocal argument, which we use as an efficient lookup argument. Rather than checking each digit is the root of some polynomial separately as in BP, we can use Eq. (7) to check membership of each digit in the set of valid digits. This enables us to construct range proofs with “optimal” bases $b > 2$ while retaining efficiency.

Application to MACT. For MACT, we face a related problem when proving multi-asset conservation of money. In this case, we have two collections of amounts and types of tokens I and O corresponding to the inputs and outputs of a transaction. We want to show that the total amount of each token in I is equal to the total amount of each token in O and that each amount in I and O is a positive integer. The latter claim can be shown using a range proof and the former using a new invocation of the reciprocal argument. Let $A = \{(v, t) : (v, t) \in I\} \cup \{(-v, t) : (v, t) \in O\}$. If A vanishes then the sum of all the amounts in I equals the sum of all the amounts in O for each token t . If the amounts are all positive integers much smaller than p , it follows that no tokens were created or destroyed in the transaction. In this case $f_A(X)$ is

$$f_A(X) = \sum_{(v,t) \in I} \frac{v}{X + t} - \sum_{(v,t) \in O} \frac{v}{X + t}. \quad (8)$$

3.3 Norm Linear Argument

As described in Sect. 3.1, BP and BP+ show a (weighted) inner product relation involving two vectors \mathbf{x} and \mathbf{y} by letting the prover send commitments to both

\mathbf{x} and \mathbf{y} . This introduces undesirable redundancy in some cases. Consider the example of a binary range proof: A prover wants to show $d_i(d_i - 1) = 0$ for each digit d_i in the vector \mathbf{d} that encodes the binary representation of some value v . In a BP range proof, this requires committing to both $\mathbf{x} = \mathbf{d}$ and $\mathbf{y} = -(\mathbf{1} - \mathbf{d})$, even though \mathbf{y} is entirely determined by \mathbf{x} up to the addition of a constant.

To avoid this redundancy in BP++, we can rewrite $d_i(d_i - 1) = 0$ into the equivalent constraint $(2d_i - 1)^2 = 1$. This allows us to substitute the inner product relation by a BP++ *norm relation*, which is a relation involving the inner product of a *single* vector with itself, and thus requires only a commitment to that single vector. As a result, we not only save data to be committed and hence communication, but also roughly half the prover and verifier cost.

However, while this motivating example provides an intuition for why a norm relation can be preferable over an inner product relation, it turns out that in practice, it is almost always more efficient to use a BP++ reciprocal range proof instead of a BP++ binary range proof. As a consequence, we defer the details of BP++ binary range proofs to the full version [20], and now turn our attention towards arithmetic circuits instead.

In the case of arithmetic circuits, similarly as for binary range proofs, using a norm argument allows reducing the verifier time by half, provided we can commit to only a single vector per commitment instead of two. Unfortunately, the inner product relation of BP and the weighted inner product relation of BP+ cannot work for this purpose, since even if the initial $\mathbf{x} = \mathbf{y}$ the reduction is asymmetric so $\mathbf{x}' \neq \mathbf{y}'$. To show a norm relation, we need a new reduction technique that is symmetric in the way it reduces \mathbf{x} and \mathbf{y} . Unlike BP, the reduced vectors are now defined to be

$$\mathbf{x}' = [\mathbf{x}]_0 + \gamma[\mathbf{x}]_1 \quad \mathbf{y}' = [\mathbf{y}]_0 + \gamma[\mathbf{y}]_1. \tag{9}$$

The reduction can be derived by computing the coefficients of the three polynomials $1, \gamma, \gamma^2 - 1 \in \mathbb{F}[\gamma]$ where in BP we computed the coefficients of the polynomials $\gamma^{-2}, 1, \gamma^2 \in \mathbb{F}[\gamma]$. Since these polynomials are linearly independent in $\mathbb{F}[\gamma]$, the reduction is sound. Setting $\mathbf{x} = \mathbf{y} = \mathbf{n}$ we can show a norm relation, and with some modifications can show a weighted norm relation.

A norm by itself is not sufficient; we want to be able to show that the witness satisfies linear constraints without introducing extraneous terms. We can apply this reduction relation to an inner product of an additional vector \mathbf{l} and a public constraint vector \mathbf{c} . This will be especially relevant when handling the blinding procedure for arithmetic circuits and also helps in the MPC proving setting. Thus, BP++ will show the weighted norm linear relation for a witness $(v, \mathbf{l}, \mathbf{n})$ and public (μ, \mathbf{c}) satisfy $v = \langle \mathbf{c}, \mathbf{l} \rangle + |\mathbf{n}|_\mu^2$.

3.4 Arithmetic Circuits

In BP and BP+, arithmetic circuits are given as a separate protocol from range proofs. The circuit is encoded as four matrices and a vector $(W_L, W_R, W_O, W_V, \mathbf{c})$. A witness $(\mathbf{w}_L, \mathbf{w}_R, \mathbf{w}_O, v)$ satisfies the circuit if

$$W_L \mathbf{w}_L + W_R \mathbf{w}_R + W_O \mathbf{w}_O = W_V \mathbf{v} + \mathbf{c} \quad \mathbf{w}_L \circ \mathbf{w}_R = \mathbf{w}_O. \quad (10)$$

While one could use an arithmetic circuit to prove a range proof in BP, it would be less efficient than the specialized range proof protocol. In the BP protocol for circuits, the prover constructs a vector valued polynomial commitment to some $(v(X), \mathbf{x}(X), \mathbf{y}(X))$ and wants to show that when we apply the inner product equation to this witness, the X^2 term of the polynomial $t(X) = v(X) - \langle \mathbf{x}(X), \mathbf{y}(X) \rangle$ vanishes. To show this, the prover commits to all the other “error” terms of $t(X)$ in Pedersen scalar commitments in T_1, T_3, T_4, T_5, T_6 .

BP++ arithmetic circuits avoids these extra commitments, as well as the two final commitments necessary to blind in both BP and BP+. Rather than committing to these other terms in scalar commitments, we commit to them as a vector in the final blinding commitment. This comes at no cost, and conveniently generalizes to larger polynomials without increasing proof size. The norm linear argument naturally allows us to evaluate the committed $t(X)$ at a random X by placing the coefficients in \mathbf{l} and changing the \mathbf{c} vector to be powers of X . We are then able to use the other commitments in the proof to blind these error terms at no additional cost in terms of proof size. This procedure is responsible for the much of the reduction in proof size.

BP++ also modifies the circuit protocol so that instead of the constraint $\mathbf{w}_L \circ \mathbf{w}_R = \mathbf{w}_O$, the arithmetic circuit checks that $\mathbf{w}_L \circ \mathbf{w}_R$ equals a linear combination of the entire witness.

This makes it efficient to formulate reciprocal constraints, where the denominators occur in \mathbf{w}_L , the reciprocals in \mathbf{w}_R , and the numerators can be any linear combination on the right hand side. This new arithmetic circuit protocol allows encoding reciprocal range proofs and MACT more efficiently than existing protocols without the use of specialized protocols.

4 Norm Linear Argument

Unlike BP and BP+ which show inner product relations, BP++ is an argument of knowledge for the weighted norm linear relation

$$\mathcal{R}_{nl} = \left\{ \left(\begin{array}{l} \mathbf{H} \in \mathbb{G}^l, \mathbf{G} \in \mathbb{G}^n, G \in \mathbb{G}; \\ C \in \mathbb{G}, \mathbf{c} \in \mathbb{F}^l, \mu \in \mathbb{F}; \\ \mathbf{l} \in \mathbb{F}^l, \mathbf{n} \in \mathbb{F}^n \end{array} \right) : C = vG + \langle \mathbf{l}, \mathbf{H} \rangle + \langle \mathbf{n}, \mathbf{G} \rangle \right\}. \quad (11)$$

4.1 Reducing the Vectors

We note that the norm linear relation \mathcal{R}_{nl} is equivalent in expressiveness to the weighted inner product relation \mathcal{R}_{ip} , in the sense that both are capable of proving arithmetic circuit satisfiability and more narrowly in the sense that one could, in principle, write the norm linear relation as an inner product and thus construct a norm linear argument by reducing directly to an inner product argument. However, the latter approach requires committing to the vector \mathbf{n} twice, in as both \mathbf{x} and \mathbf{y} from the inner product relation (see Sect. 3.1). While

it is possible to simplify the initial commitment by computing $\langle \mathbf{n}, \mathbf{G} + \mathbf{H} \rangle$ in the inner product commitment, the vectors \mathbf{x} and \mathbf{y} will be reduced asymmetrically following such an approach. This means that even if $\mathbf{x} = \mathbf{y}$, it will not be the case that $\mathbf{x}' = \mathbf{y}'$.

This makes clear what we want from a norm linear argument: given a commitment C to vectors as defined in the relation, we want to reduce this commitment to a new commitment to vectors \mathbf{l}' and \mathbf{n}' of half the length of the original vectors. To this end, we need a folding relation for a pair of vectors that treats both vectors symmetrically. That is, instead of scaling the halves of \mathbf{x} and \mathbf{y} by complementary γ and γ^{-1} , we would like to use reduced vectors that are folded in the same way, such as

$$\mathbf{x}' = \rho^{-1}[\mathbf{x}]_0 + \gamma[\mathbf{x}]_1 \quad \mathbf{y}' = \rho^{-1}[\mathbf{y}]_0 + \gamma[\mathbf{y}]_1. \tag{12}$$

Now if $\mathbf{x} = \mathbf{y}$ then $\mathbf{x}' = \mathbf{y}'$. Here the value is defined as $\rho^2 = \mu$ for weight μ . Taking the weighted inner product of these vectors by μ we can work out a relation that includes the original weighted inner product $\langle \mathbf{x}, \mathbf{y} \rangle_\mu$ as one coefficient of a polynomial in γ

$$\begin{aligned} v_x = \rho^{-1}(\langle [\mathbf{x}]_0, [\mathbf{y}]_1 \rangle_{\mu^2} + \langle [\mathbf{x}]_1, [\mathbf{y}]_0 \rangle_{\mu^2}) \quad v_r = \langle [\mathbf{x}]_1, [\mathbf{y}]_1 \rangle_{\mu^2} \\ \langle \mathbf{x}', \mathbf{y}' \rangle_{\mu^2} = \langle \mathbf{x}, \mathbf{y} \rangle_\mu + v_x \gamma + v_r (\gamma^2 - 1). \end{aligned} \tag{13}$$

Note that this reduction is sound because the polynomials $1, \gamma, \gamma^2 - 1 \in \mathbb{F}[\gamma]$ are linearly independent. As in BP(+), the protocol follows straightforwardly from this relation by applying it to all the inner products in the commitment and grouping alike terms. The prover can commit to the γ and $\gamma^2 - 1$ coefficients (X, R) and then the verifier can select a random γ to evaluate the relation. Because this relation is symmetric, the prover can apply it to the $\mathbf{x} = \mathbf{y} = \mathbf{n}$ case and reduce \mathbf{n} to a single \mathbf{n}' .

4.2 Norm Linear Argument

In the norm linear relation, there are 4 inner products that the prover needs to reduce: $|\mathbf{n}'|_\mu^2, \langle \mathbf{n}, \mathbf{G} \rangle, \langle \mathbf{c}, \mathbf{l} \rangle$, and $\langle \mathbf{l}, \mathbf{H} \rangle$. Since \mathbf{n} participates in a weighted inner product (norm), we need to modify the relation for \mathbf{G} slightly, and since \mathbf{l}, \mathbf{c} , and \mathbf{H} only participate in unweighted relations, there are no weights present. The reduced vectors are thus

$$\begin{aligned} v' = |\mathbf{n}'|_{\mu^2}^2 + \langle \mathbf{c}', \mathbf{l}' \rangle \quad \mathbf{c}' = [\mathbf{c}]_0 + \gamma[\mathbf{c}]_1 \\ \mathbf{l}' = [\mathbf{l}]_0 + \gamma[\mathbf{l}]_1 \quad \mathbf{n}' = \rho^{-1}[\mathbf{n}]_0 + \gamma[\mathbf{n}]_1 \\ \mathbf{G}' = \rho[\mathbf{G}]_0 + \gamma[\mathbf{G}]_1 \quad \mathbf{H}' = [\mathbf{H}]_0 + \gamma[\mathbf{H}]_1. \end{aligned} \tag{14}$$

The commitments X and R follow directly from expanding all the reduced inner products and gathering γ and $\gamma^2 - 1$ coefficients. Explicitly

$$v_x = 2\rho^{-1} \langle [\mathbf{n}]_0, [\mathbf{n}]_1 \rangle_{\mu^2} + \langle \mathbf{c}, ([\mathbf{l}]_1, [\mathbf{l}]_0) \rangle \tag{15}$$

$$v_r = |[\mathbf{n}]_1|_{\mu^2}^2 + \langle [\mathbf{c}]_1, [\mathbf{l}]_1 \rangle \tag{16}$$

$$X = v_x G + \langle ([\mathbf{l}]_1, [\mathbf{l}]_0), \mathbf{H} \rangle + \langle (\rho[\mathbf{n}]_1, \rho^{-1}[\mathbf{n}]_0), \mathbf{G} \rangle \tag{17}$$

$$R = v_r G + \langle [\mathbf{l}]_1, [\mathbf{H}]_1 \rangle + \langle [\mathbf{n}]_1, [\mathbf{G}]_1 \rangle. \tag{18}$$

Evaluating the polynomial commitment at γ yields a commitment on the reduced basis to the reduced witness, i.e., we have

$$C + \gamma X + (\gamma^2 - 1)R = v'G + \langle \mathbf{l}', \mathbf{H}' \rangle + \langle \mathbf{n}', \mathbf{G}' \rangle. \tag{19}$$

The full protocol applies this reduction recursively until doing so does not reduce the overall proof size. This occurs when $|\mathbf{l}| + |\mathbf{n}| \leq 6$, at which point the prover sends the reduced \mathbf{l} and \mathbf{n} to the verifier. If these vectors satisfy the norm linear relation for the reduced \mathbf{c} and μ , then it follows by induction that the original commitment satisfies the relation.

Completeness follows directly from this equation holding and soundness from the linear independence of the polynomials $1, \gamma, \gamma^2 - 1 \in \mathbb{F}[\gamma]$. Linear independence can be used to construct a round extractor, which as in BP can be used to construct an extractor for the entire protocol.

Theorem 1. *The weighted norm linear argument has perfect completeness. Assuming the expected-PPT hardness of the discrete logarithm relation problem, the argument has CWeE and is therefore an argument of knowledge for the weighted norm linear relation.*

See the full version [20] for the proof.

4.3 Full Protocol Description

The setup protocol for the norm linear argument \mathcal{K} simply chooses all group elements $G, \mathbf{H}, \mathbf{G}$ uniformly at random.

Weighted Norm Linear Argument $\langle \mathcal{P}_{nl}, \mathcal{V}_{nl} \rangle$

Common input: $G, \mathbf{G}, \mathbf{H}, \mathbf{c}, C, \rho$ and $\mu = \rho^2$

\mathcal{P} 's input: (\mathbf{l}, \mathbf{n}) and $v = \langle \mathbf{c}, \mathbf{l} \rangle + |\mathbf{n}|_{\mu}^2$ such that $C = vG + \langle \mathbf{l}, \mathbf{H} \rangle + \langle \mathbf{n}, \mathbf{G} \rangle$

1. If $|\mathbf{l}| + |\mathbf{n}| < 6$:
 - 1.1 $\mathcal{P} \rightarrow \mathcal{V} : \mathbf{l}, \mathbf{n}$
 - 1.2 \mathcal{V} computes $v := \langle \mathbf{c}, \mathbf{l} \rangle + |\mathbf{n}|_{\mu}^2$
 - 1.3 \mathcal{V} accepts if $C \stackrel{?}{=} vG + \langle \mathbf{l}, \mathbf{H} \rangle + \langle \mathbf{n}, \mathbf{G} \rangle$, otherwise rejects
2. Else:
 - 2.1 $\mathcal{P} \rightarrow \mathcal{V} : X, R$
 - 2.2 $\mathcal{V} \rightarrow \mathcal{P} : \gamma \leftarrow_{\mathfrak{s}} \mathbb{F}$
 - 2.3 \mathcal{P} computes \mathbf{l}', \mathbf{n}'
 - 2.4 \mathcal{P}, \mathcal{V} compute $\mathbf{G}', \mathbf{H}', \mathbf{c}'$ and $\rho' := \mu, \mu' := \mu^2, C' := C + \gamma X + (\gamma^2 - 1)R$
 - 2.5 Run $\langle \mathcal{P}_{nl}, \mathcal{V}_{nl} \rangle$ with $(G, \mathbf{G}', \mathbf{H}', \mathbf{c}', C', \rho', \mu'; \mathbf{l}', \mathbf{n}')$.

As in BP, it is not necessary for the verifier to actually compute the intermediate $(\mathbf{G}, \mathbf{H}, \mathbf{c}, C)$ values and the final verification check can be replaced with a single linear combination of public curve points. Letting k be the number of rounds before stopping and the vectors γ_l and γ_n be defined as

$$\gamma_l = \bigotimes_{i=0}^{k-1} (1, \gamma_i) \quad \gamma_n = \bigotimes_{i=0}^{k-1} (\rho^{2^i}, \gamma_i), \tag{20}$$

the $(\mathbf{G}, \mathbf{H}, \mathbf{c}, C)$ in the final verification equation can be rewritten in terms of the original $(\mathbf{G}, \mathbf{H}, \mathbf{c}, C)$ as

$$v = \langle \mathbf{c}, \gamma_l \otimes \mathbf{l} \rangle + |\mathbf{n}|_\mu^2 \tag{21}$$

$$vG + \langle \gamma_l \otimes \mathbf{l}, \mathbf{H} \rangle + \langle \gamma_n \otimes \mathbf{n}, \mathbf{G} \rangle \stackrel{?}{=} C + \sum_{i=0}^{k-1} \gamma_i X_i + (\gamma_i^2 - 1)R_i. \tag{22}$$

Also as in BP, when verifying multiple proofs simultaneously, the verifier can take a random linear combination of the equations and combine the $\gamma_l \otimes \mathbf{l}$ and $\gamma_n \otimes \mathbf{n}$ from different proofs if the \mathbf{G} and \mathbf{H} are the same. Thus the marginal cost of verifying an additional proof is only $O(\log n)$ additional scalar multiplications and $O(n)$ field operations. There are additional optimizations that help reduce prover work, as we discuss in the full version [20].

5 Arithmetic Circuits

In BP, arithmetic circuits are represented using four public matrices and one public vector $(W_L, W_R, W_O, W_V, \mathbf{c})$ and four witness vectors $(\mathbf{w}_L, \mathbf{w}_R, \mathbf{w}_O, \mathbf{v})$, which must satisfy Eq. (10). For each multiplication in a BP arithmetic circuit, the prover commits to the left input in $\mathbf{w}_{L,i}$, the right input in $\mathbf{w}_{R,i}$ and the output in $\mathbf{w}_{O,i}$. In some cases, this leads to the prover committing to redundant information. Specifically, if an output of a multiplication is immediately subject to a linear constraint, the prover could avoid committing to it by instead showing

$$\mathbf{w}_L \circ \mathbf{w}_R = W_{m,L} \mathbf{w}_L + W_{m,R} \mathbf{w}_R + W_{m,O} \mathbf{w}_O. \tag{23}$$

This motivates the BP++ circuit encoding, where we make exactly this change. It turns out that effectively every multiplication gate in reciprocal range proofs (Sect. 6.2) and MACTs is of this form. This change makes it more efficient to represent these protocols as arithmetic circuits, rather than using bespoke range proof protocols like other Bulletproof based constructions. We also modify the circuits to accept input vectors from Pedersen vector commitments, rather than just scalars, which removes the matrix W_V .

Concretely, an arithmetic circuit \mathcal{C} will be encoded into two matrices (W_L, W_m) and two vectors $(\mathbf{a}_l, \mathbf{a}_m)$ which constrain a witness $\mathbf{w} = (\mathbf{w}_L, \mathbf{w}_R, \mathbf{w}_O)$. The vectors \mathbf{w}_L and \mathbf{w}_R are the left and right inputs to each multiplication, as in BP. The input vector is the concatenation of vectors

$\mathbf{w}_V = (\mathbf{v}_i)_{i=0}^k$, each of which comes from a Pedersen vector commitment V_i . The circuit is satisfied if both

$$\mathbf{0} = W_l \mathbf{w} + \mathbf{w}_V + \mathbf{a}_l \quad \mathbf{w}_L \circ \mathbf{w}_R = W_m \mathbf{w} + \mathbf{a}_m. \tag{24}$$

The arithmetic circuit protocol is therefore a proof of knowledge for the relation

$$\mathcal{C} = (W_l \in \mathbb{F}^{N_l \times N_w}, \mathbf{a}_l \in \mathbb{F}^{N_l}, W_m \in \mathbb{F}^{N_m \times N_w}, \mathbf{a}_m \in \mathbb{F}^{N_m}) \tag{25}$$

$$\mathcal{R}_{ac} = \left\{ \left(\begin{array}{l} G \in \mathbb{G}, \mathbf{H} \in \mathbb{G}^{N_v+7}, \mathbf{G} \in \mathbb{G}^{N_m}; \\ \mathcal{C}, \mathbf{V} \in \mathbb{G}^k; \mathbf{v}_i \in \mathbb{F}^{N_v} : i = [0, k), \\ \mathbf{s}_V \in \mathbb{F}^k, \mathbf{w}_O \in \mathbb{F}^{N_o}, \mathbf{w}_L, \mathbf{w}_R \in \mathbb{F}^{N_m} \end{array} \right) : \begin{array}{l} V_i = \text{Com}(\mathbf{v}_i; s_{V,i}) \\ \text{Eq. (24)} \end{array} \right\} \tag{26}$$

This new arithmetic circuit format can encode satisfiability of BP circuits and is therefore capable of representing any arithmetic circuit, see the full version [20] for details.

5.1 Protocol Overview

We defer the explicit details of how the arithmetic circuit protocol encodes the statement into the norm linear argument to the full version [20] and limit ourselves to a high-level description here. First, the prover will commit to $(\mathbf{w}_L, \mathbf{w}_R, \mathbf{w}_O)$ in (C_L, C_R, C_O) and send these to the verifier. There is some freedom in how the prover can organize the witness into these three norm linear commitments. Specifically, in some cases it may be more efficient to commit to some of \mathbf{w}_O in the linear portion of C_L and C_R .

Then, the verifier will choose two challenges λ and μ to combine the linear and multiplicative constraints respectively using the vectors of powers $\mathbf{e}_{N_l}(\lambda)$ and $\mathbf{e}_{N_m}(\mu)$. The verifier will also choose challenges β and δ , which will be necessary for blinding. These allow us to transform equations Eq. (24) into a single scalar equation

$$0 = \mathbf{e}_{N_l}(\lambda)^\top (W_l \mathbf{w} + \mathbf{w}_V + \mathbf{a}_l) + \langle \mathbf{w}_L, \mathbf{w}_R \rangle_\mu - \mathbf{e}_{N_m}(\mu)^\top (W_m \mathbf{w} + \mathbf{a}_m). \tag{27}$$

We want to construct a triple of polynomials $(\hat{v}(T), \hat{\mathbf{l}}(T), \hat{\mathbf{n}}(T))$ so that when we apply the norm linear relation and get $\hat{f}(T) = \hat{v}(T) - \langle \hat{\mathbf{c}}(T), \hat{\mathbf{l}}(T) \rangle - |\hat{\mathbf{n}}(T)|_\mu^2$ exactly one term of $\hat{f}(T)$ encodes these randomized constraints. We call this term the value term and the other terms the error terms. To show that the constraints are satisfied, it suffices to show that the value term vanishes, and prove knowledge of the error terms.

To construct this polynomial, we first assign a unique T term to each commitment. The product of the T terms for C_L and C_R will be the value term. Each constraint will be placed on the unique T term so that when multiplied with the T term of the commitment to the witness it acts on, the result will be the value term. So, if T multiplies C_L and $T^2 C_R$, then the value term is T^3 and $\mathbf{e}_{N_l}(\lambda)^\top W_L$ should be multiplied by T^2 . The challenge δ will be used to prevent the norm portion of the \mathbf{w}_V commitments from interfering with $f(X)$.

Then, the prover sends C_S to blind. For the portions of the commitments that commit to w , C_S will consist of uniformly random values. We need to choose a T term for C_S so that none of these random values can interfere with the value term. Our goal now is to introduce some additional elements to the linear portion of C_S to subtract off the non-value terms from $\hat{f}(T)$ and the additional terms that arise from the blinding. If the result of this is zero, then the value term must be zero.

This can have two problems: it might allow interference with the value term and it will not be zero knowledge as it may reveal information about the error terms. The second problem is fixed by allowing the commitments C_L, C_R, C_O to blind the error terms in C_S . The first is fixed by using the challenge δ to prevent interference. Showing that the result is zero knowledge is somewhat more involved than other protocols, and is ultimately reducible to showing that by manipulating the error term blinding in C_L, C_R, C_O can produce any valid opening. Equivalently that a certain matrix is full rank.

Finally, the verifier will send a challenge $T = \tau$. Because of how the blinding was constructed, the prover and verifier can take a linear combination of C_L, C_R, C_S, C_O and public information to produce a valid norm linear instance. Without the blinding protocol, the protocol would need an additional round of interaction. At this point, since the witness is blinded the prover and verifier can run the norm linear argument and complete the protocol.

Arithmetic Circuit Protocol $\langle \mathcal{P}_{ac}, \mathcal{V}_{ac} \rangle$

1. $\mathcal{P} \rightarrow \mathcal{V} : C_L, C_R, C_O$ // Choose blinding r_L, r_R, r_O and commit to w
2. $\mathcal{V} \rightarrow \mathcal{P} : \rho, \lambda, \beta, \delta \leftarrow_{\mathbb{S}} \mathbb{F}$
3. $\mathcal{P} \rightarrow \mathcal{V} : C_S$ // C_S is chosen s.t. all error terms cancel and w is blinded
4. $\mathcal{V} \rightarrow \mathcal{P} : \tau \leftarrow_{\mathbb{S}} \mathbb{F}$
5. \mathcal{P} computes $v(\tau), \mathbf{l}(\tau), \mathbf{n}(\tau)$ // Compute opening of $C(\tau)$.
(Since all error terms cancel, norm linear relation is satisfied.)
6. \mathcal{P}, \mathcal{V} run the weighted norm linear argument $\langle \mathcal{P}_{nl}, \mathcal{V}_{nl} \rangle = b$ with common input $(\mathbf{c}(\tau), C(\tau), \mu = \rho^2)$ and prover input $(\mathbf{l}(\tau), \mathbf{n}(\tau), v(\tau))$.

Theorem 2 (Arithmetic Circuits). *The arithmetic circuit protocol (whose pseudocode can be found in the full version [20]) has perfect completeness and perfect honest verifier zero-knowledge. Assuming the expected-PPT hardness of the discrete logarithm relation problem, the protocol has computational witness-extended emulation.*

See the full version [20] for the proof.

6 Reciprocal Argument

Initially, zero knowledge proof arithmetizations, including that of the original Bulletproof AC protocol, supported only additions and multiplications. This was

sufficient to encode all arithmetic circuits, but more modern proof systems like Halo2 [49] incorporate so called “custom gates” directly into the arithmetization. These custom gates allow circuit designers to “factor out” certain features into the arithmetization, which has a number of benefits for circuit designers. For example, a custom gate to compute x^5 avoids adding the values x^2 and x^4 to the witness.

Another more powerful type of custom gate is the so called “lookup gate”, which is implemented using a variant of plookup [21] in Halo2 [49]. This allows circuit designer to incorporate lookup arguments into their circuits. Unlike raising to the fifth power, this gate cannot be conveniently implemented as a low degree expression since it requires an additional round of prover, verifier interaction. In particular, this means it is not possible to efficiently perform plookup, or the reciprocal argument, inside simpler arithmetizations like BP++ AC. This motivates adding the reciprocal argument directly to BP++ AC, which we call reciprocal form circuits. By formalizing this modification of the protocol, we are also able to provide a single knowledge soundness proof.

To demonstrate the power of this approach, we use the new reciprocal form circuit protocol to define a range proof and a MACT protocol. Since these protocol are simply reciprocal from arithmetic circuits, zero-knowledge and knowledge soundness will follow without the need for additional security proofs.

6.1 Warmup: Reciprocal Argument Protocol

Recall from Sect. 3.2 that the reciprocal argument is an interactive protocol by which the prover can convince a verifier that a *collection* A vanishes.

Definition 6. Let A be a set of pairs (m_i, s_i) of multiplicities $m_i \in \mathbb{F}$ and symbols $s_i \in \mathbb{F}$. Let the total multiplicity of a symbol $s \in \mathbb{F}$ in A be

$$\hat{m}_s = \sum_{i: s_i=s} m_i.$$

We call A a collection, and we say that A vanishes if $\forall s \in \mathbb{F} : \hat{m}_s = 0$.

Let $S = \{s_i : \exists m : (m, s_i) \in A\}$ be the set of symbols in A , and further recall that the reciprocal argument encodes A as a rational function f_A defined as

$$f_A(X) := \sum_{i=0}^{|A|-1} \frac{m_i}{X + s_i} = \sum_{s \in S} \frac{\hat{m}_s}{X + s}. \tag{28}$$

To demonstrate the core idea of the reciprocal argument, we present an informal protocol in which the prover sends the verifier the witness explicitly. This protocol is not used in a blackbox manner by BP++; we will instead embed it into an arithmetic circuit and make additional modifications, e.g., some of \mathbf{s} or \mathbf{m} may be known to the verifier.

The (informal) protocol works as follows: First, the prover sends multiplicities \mathbf{m} and symbols \mathbf{s} in A . Next the verifier selects a random challenge α , and the prover responds by sending the “reciprocals” $r_i = m_i/(\alpha + s_i)$. Finally, the verifier checks that each reciprocal is properly formed and that the sum of all the reciprocals vanish, i.e., if $(\alpha + s_i)r_i = m_i$ and $\sum_i r_i = 0$.

Reciprocal Argument Protocol $\langle \mathcal{P}_{ra}, \mathcal{V}_{ra} \rangle$

1. $\mathcal{P} \rightarrow \mathcal{V} : \mathbf{m}, \mathbf{s}$
2. $\mathcal{V} \rightarrow \mathcal{P} : \alpha \leftarrow_{\mathbb{S}} \mathbb{F}$
3. $\mathcal{P} \rightarrow \mathcal{V} : \mathbf{r}$ s.t. $r_i = m_i / (\alpha + s_i)$
4. \mathcal{V} accepts if $(\alpha + s_i)r_i = m_i$ for $i = 0 \dots |\mathbf{r}| - 1$ and $\sum_i r_i = 0$

This protocol lacks perfect completeness because if $\alpha = -s_i$ for any s_i then r_i is not well-defined. However, this only occurs with negligible probability since $\alpha \leftarrow_{\mathbb{S}} \mathbb{F}$. Informally, soundness follows from the structure of the sum of the reciprocals. If $(\alpha + s_i)r_i = m_i$, then either $\alpha = -s_i$ and $m_i = 0$, or $r_i = m_i / (\alpha + s_i)$. So, with overwhelming probability, if $\sum_i r_i = 0$ we have that $f_A(\alpha) = 0$.

We can show that if $f_A(\alpha_j) = 0$ for $2|S|$ distinct challenges α_j then \hat{m}_s must be zero for all $s \in S$.

Lemma 1 (Reciprocal Argument Vanishing). *Let A be a collection of pairs of multiplicities and symbols. If there exist $2|A|$ accepting transcripts of the reciprocal argument protocol for A with pairwise distinct challenges α_j , then A vanishes (Definition 6).*

Proof. There are at most $|S|$ values α such that there exists $s_i \in S$ for which $\alpha = -s_i$. Let $\boldsymbol{\alpha}'$ be a vector of $|S|$ challenges α_j from the transcripts such that $\alpha'_j \neq -s_i$ for any i, j . Let \mathbf{s} be the vector of elements in S , and note that the components of vector $-\mathbf{s}$ are pairwise distinct and the components in $\boldsymbol{\alpha}'$ are pairwise distinct. This means the $|S| \times |S|$ Cauchy matrix C formed from $-\mathbf{s}$ and $\boldsymbol{\alpha}'$ is well-defined and therefore invertible. Let $f_j = f_A(\alpha'_j)$ for f_A as defined in Eq. (28) and note that $\mathbf{f} = C\hat{\mathbf{m}}_s = \mathbf{0}$. Since C is invertible, $\hat{\mathbf{m}}_s = \mathbf{0}$ and therefore A vanishes.

6.2 Reciprocal Form Circuits

Reciprocal form circuits extend the BP++ AC protocol to support the reciprocal argument. As in the protocol outlined above, this requires an additional round of interaction, where the verifier chooses α , and will require the prover to commit to the witness in several stages. Once we have α and the entire witness, we can use a BP++ AC to verify step 4 of the reciprocal argument protocol.

Suppose we have an arithmetic circuit \mathcal{C} and the arithmetic circuit witness $(\mathbf{w}_L, \mathbf{w}_R, \mathbf{w}_O, \mathbf{w}_V)$. To integrate the reciprocal argument, we want to show that this circuit is satisfied and that some set of rational functions $\mathbf{f}(X)$ vanishes, where each rational function encodes a reciprocal argument instance. In general, we want the symbols and multiplicities to be able to depend on the arithmetic circuit witness \mathbf{w} and ultimately would like to be able to compile the $\mathbf{f}(X)$ vanishing check into an arithmetic circuit for a particular $X = \alpha$.

Let \mathbf{w}_D be the vector of private denominators for all the reciprocal argument instances, and let $\mathbf{w}_P(X)$ be the vector of all the reciprocals associated to each $w_{D,i}$. Define the “initial witness” $\mathbf{w}_I = \mathbf{w}_O \parallel \mathbf{w}_L \parallel \mathbf{w}_D$ and the “entire witness”

to be $\mathbf{w}(X) = \mathbf{w}_D \parallel \mathbf{w}_L \parallel \mathbf{w}_P(X) \parallel \mathbf{w}_R \parallel \mathbf{w}_O$. We can specify all the reciprocal argument instances using three matrices $(W_n, W_d, W_p(X))$

$$w_{P,i}(X) = \frac{(W_n \mathbf{w}_I + \mathbf{a}_n)_i}{X + w_{D,i}} \tag{29}$$

$$W_d \mathbf{w}_I + \mathbf{w}_V + \mathbf{a}_d = \mathbf{0} \tag{30}$$

$$\mathbf{f}(X) = W_p(X) \mathbf{w}(X) + \mathbf{a}_p(X). \tag{31}$$

The intuition here is as follows. First, we take all the reciprocals that occur in all the $\mathbf{f}(X)$ instances and partition them into two groups. The first are the reciprocals with public denominators, and the second are those with denominators that depend on the witness. Those with public denominators do not require multiplicative constraints and can be encoded via $W_p(X)$. The second set are organized into a vector, and \mathbf{w}_D is their denominators. We allow the prover to constrain these values via Eq. (30). The numerators of these reciprocals are encoded via $W_n \mathbf{w}_I + \mathbf{a}_n$, and the reciprocals themselves will be committed via $\mathbf{w}_P(\alpha)$. Finally, we map each reciprocal to its reciprocal argument instance via $W_p(X)$ and add any that consist of entirely public information via $\mathbf{a}_p(X)$.

Following commitment to \mathbf{w}_I , the verifier chooses α , and the prover commits to \mathbf{w}_R and $\mathbf{w}_P(\alpha)$. We can now define the new arithmetic circuit \mathcal{C}' for α . First, prepend the vector \mathbf{w}_D onto \mathbf{w}_L and the vector $\mathbf{w}_P(\alpha)$ onto \mathbf{w}_R to produce \mathbf{w}'_L and \mathbf{w}'_R for \mathcal{C}' . We keep $\mathbf{w}'_O = \mathbf{w}_O$, and can let $\mathbf{w}' = \mathbf{w}'_L \parallel \mathbf{w}'_R \parallel \mathbf{w}'_O$. To verify that the committed vector \mathbf{w}'_P is correctly constructed as $\mathbf{w}_P(\alpha)$, we can clear the denominator of Eq. (29) and check

$$w_{D,i} w'_{P,i} = (W_n \mathbf{w}_I + \mathbf{a}_n)_i - \alpha w'_{P,i}. \tag{32}$$

This is satisfied if $w'_{P,i} = w_{P,i}(\alpha)$ or if $w_{D,i} = -\alpha$ and the numerator is zero. The latter occurs with negligible probability, so this is sufficient to check \mathbf{w}'_P is correctly constructed. The rest of the constraints can be appended onto the W_l and W_m matrices to construct the W'_l and W'_m matrices for \mathcal{C}' as

$$W'_l \mathbf{w}' = (W_d \mathbf{w}_I) \parallel (W_p(\alpha) \mathbf{w}(X)) \parallel (W_l \mathbf{w}) \tag{33}$$

$$W'_m \mathbf{w}' = (W_n \mathbf{w}_I - \alpha \mathbf{w}_P(\alpha)) \parallel (W_m \mathbf{w}). \tag{34}$$

Formally, the reciprocal form arithmetic circuit protocol shows that the reciprocal form arithmetic circuit relation is satisfied for the circuit \mathcal{RC} . In the relation, A_i refers to the collection for the i th instance of the reciprocal argument. That is, the collection A_i is encoded as in Eq. (28) by $f_i(X)$ in Eq. (31).

$$\mathcal{RC} = \left(\begin{array}{l} \mathcal{C}, W_n \in \mathbb{F}^{N_p \times N_I}, W_d \in \mathbb{F}^{N_d \times N_I}, \\ W_p(X) \in \mathbb{F}(X)^{N_p \times N'_w} \\ \mathbf{a}_n \in \mathbb{F}^{N_p}, \mathbf{a}_d \in \mathbb{F}^{N_d}, \mathbf{a}_p(X) \in \mathbb{F}(X)^{N_p} \end{array} \right) \tag{35}$$

$$\mathcal{R}_{rf} = \{(\sigma; x, \mathcal{RC}; w, \mathbf{w}_D \in \mathbb{F}^{N_p}) : A_i \text{ vanishes, Eq. (30), } (\sigma; x; w) \in \mathcal{R}_{ac}\} \tag{36}$$

Given that we can compile reciprocal form circuits to arithmetic circuits for a particular α , the security proofs are able to inherit most of the structure of

those of arithmetic circuits. Zero-knowledge follows immediately, and soundness requires one additional level in the transcript tree for α to extract the vanishing of $f(\alpha)$.

Theorem 3 (Reciprocal Form Arithmetic Circuits). *The arithmetic circuit protocol for circuits in reciprocal form (whose pseudocode can be found in the full version [20]) has completeness and perfect honest verifier zero-knowledge. Assuming the expected-PPT hardness of the discrete logarithm relation problem, the protocol has computational witness-extended emulation.*

See the full version [20] for the proof.

6.3 Reciprocal Range Proofs

Given the reciprocal argument and reciprocal form arithmetic circuits, we can now construct a range proof as an argument of knowledge for

$$\mathcal{R}_{rp} = \left\{ \left(\begin{array}{l} G, H \in \mathbb{G}; \\ \mathbf{V} \in \mathbb{G}^k, \mathbf{A}, \mathbf{B} \in \mathbb{Z}^k, B_i - A_i \in [1, p); \\ \mathbf{v}, \mathbf{s} \in \mathbb{F}^k \end{array} \right) : \begin{array}{l} \forall i : v_i \in [A_i, B_i), \\ V_i = \text{Com}(v_i; s_i) \end{array} \right\}. \quad (37)$$

For simplicity, assume each range $[A_i, B_i)$ uses the same base b . To show that each value lies in the range the prover can break down v_i into digits \mathbf{d}_i , show that each digit is a valid base b digit, and show that for some vector of public constants \mathbf{b}_i , the following linear relation is satisfied $\langle \mathbf{b}_i, \mathbf{d}_i \rangle = v_i - A_i$. To show each \mathbf{d}_i is a valid digit we can use the reciprocal argument and let \mathbf{w}_D consist of all the digits for all the range proofs.

We will also assume for simplicity of presentation that the size of the range is a power of b . That is $B - A = b^k$ for some integer k . This simplifies the range proof description, and is typically sufficient in practice. Especially in cryptocurrencies range proofs are typically used to enforce that a value is not “negative” rather than that it lies in a specific range. It is straightforward to adapt the protocol to support arbitrary ranges using the work of Chaabouni, Lipmaa, shelat [12] and we defer a detailed description to the full version [20].

BP++ arithmetic circuits, as mentioned before, allow placing the vector \mathbf{w}_O of witness elements that participate only linear constraints either in the \mathbf{l}_X portion of the witness, or in the \mathbf{n}_O portion of the witness. For reciprocal range proofs, it makes sense to either place them in \mathbf{n}_O , which we will call “inline” multiplicity range proofs, or in \mathbf{l}_L , which we will call “shared” multiplicity range proofs. The terminology refers to the fact that in the multiparty setting when multiplicities are placed in the linear portion of the witness multiple provers can reuse the same basis points in their separate proofs. Inline range proofs are so called because in the multiparty setting, multiplicities must be represented over the basis elements used by each prover to commit to their digits, so the multiplicities are inline with the digits.

Arithmetic Circuit. In both the inline and reciprocal cases, the vector \mathbf{w}_D consists of the concatenation of the digit vectors for all the ranges. The numerator for each digit reciprocal is always 1, so the numerator matrix is simply zero and $\mathbf{a}_n = \mathbf{1}$. The vector of reciprocals $\mathbf{w}_P(X)$ is the concatenation of the values $r_{i,j} = 1/(\alpha + d_{i,j})$ so they align with \mathbf{w}_D per value that verifies the range using

$$\langle W_{d,i}, \mathbf{w}_D \rangle = \langle \mathbf{b}_i, \mathbf{d}_i \rangle \quad a_{d,i} = A_i \tag{38}$$

Now all that remains is to describe the matrix $W_p(X)$ in terms of the multiplicities. In both the inline and shared cases, the prover shows that the set membership check is satisfied for all the digits of each base. Let the vector \mathbf{m}_i be the number of times each value in $[1, b_i)$ occurs in \mathbf{d}_i . Note this does not include a multiplicity for zero, as this multiplicity is equal to the number of digits minus the sum of the other multiplicities. Let the total multiplicity be $\hat{\mathbf{m}} = \sum_i \mathbf{m}_i$ and the total number of digits be $\hat{n} = \sum_i |\mathbf{d}_i|$. In both the inline and the shared cases, the prover uses the vectors of reciprocals to prove each digit is a valid base b digit

$$\sum_i \langle \mathbf{1}, \mathbf{r}_i \rangle = \frac{\hat{n} - \langle \mathbf{1}, \hat{\mathbf{m}} \rangle}{X} + \sum_{j=0}^{b-2} \frac{\hat{m}_j}{X + j + 1}. \tag{39}$$

The difference arises in how the prover commits to the multiplicities in the inline case, the prover commits to the vector \mathbf{m} in \mathbf{w}_O padded so that they align with \mathbf{d}_i . The partition function \mathcal{F} in the inline case maps all of \mathbf{w}_O to \mathbf{n}_O . Since the $\hat{\mathbf{m}}$ are a linear function of the \mathbf{m}_i , the matrix $W_p(X)$ is defined to compute this function and then the right hand side of Eq. (39).

In the shared case, the prover commits to the all the $\hat{\mathbf{m}}$ directly in \mathbf{w}_O and the partition function maps these values to \mathbf{l}_L . In this case, since neither \mathbf{l}_O or \mathbf{n}_O are used, the commitment can be safely dropped from the protocol. The matrix $W_p(X)$ once again encodes Eq. (39) but now uses the committed total multiplicities.

Theorem 4 (Reciprocal Range Proofs). *Both the inline and shared multiplicity reciprocal range proofs and zero knowledge arguments of knowledge for the reciprocal range proof relation \mathcal{R}_{rp} Eq. (37) assuming the expected-PPT hardness of the discrete logarithm relation problem.*

Proof. The reciprocal range proof protocols are both instances of the reciprocal form arithmetic circuit protocol, so they have SHVZK, CWEE, and completeness. To show they are arguments for Eq. (37), we must establish that the circuit is satisfiable only if the inputs \mathbf{v} satisfy the relation. The protocol applies the reciprocal form circuit protocol to $A = \{(-1, d_i) : i\} \cup \{(m_j, t_j) : j\}$. By the soundness of the reciprocal form circuit protocol, A vanishes. So long as the number of digits is less than \mathbb{F} , which is the case by assumption, this implies all d_i are valid base b digits. Therefore $v_i = \langle \mathbf{b}_i, \mathbf{d}_i \rangle + A_i$ implies that $v_i \in [A_i, B_i)$. Thus, the reciprocal range proof protocol is a zero knowledge argument of knowledge for Eq. (37).

6.4 Multi-asset Confidential Transactions

In a MACT, the prover wants to prove a closely related relation to that of an aggregated range proof. Given a transaction with a set of inputs I ($o_i = 0$) and outputs O ($o_i = 1$), each with a type and amount, the prover wants to show that the amount of input tokens of each type equals the amount of tokens output of each type and that all the output token amounts are “positive.” This is because if one of the outputs were negative it would be possible to secretly create new tokens, by adding more tokens to one of the other outputs to be larger. It is typically not necessary to check that the inputs are positive since they are the outputs of some other transaction.

In a finite field, the positivity condition is checked by bounding each output ($o_i = 1$) by a range much smaller than the field characteristic. More precisely, it must be the case that any negligible amount of inputs and outputs cannot wrap around in the field to create a “negative” value. For simplicity, we can assume that all transaction outputs use the same range in the range proof $[0, B)$, and in practice we can assume that $B = 2^{64}$. The MACT relation is thus

$$\mathcal{R}_{ct} = \left\{ \left(\begin{array}{l} G, H_0, H_1 \in \mathbb{G}; \mathbf{o} \in \{0, 1\}^k, \\ \mathbf{V} \in \mathbb{G}^k, B \in \mathbb{Z}, kB < p, \\ \forall i : o_i = 0 \Rightarrow v_i \in [0, B); \\ \mathbf{v}, \mathbf{t}, \mathbf{s} \in \mathbb{F}^k \end{array} \right) : \begin{array}{l} \forall i : V_i = v_i G + t_i H_0 + s_i H_1 \\ \forall i : o_i = 1 \Rightarrow v_i \in [0, B) \\ \forall t : \sum_{i:t_i=t} (-1)^{o_i} v_i = 0 \end{array} \right\}. \tag{40}$$

To check the range proof part of the relation, we can use any reciprocal range proof over all the transaction outputs, i.e. $o_i = 1$, for the optimal base b and range $[0, B)$. Checking that all the amounts of each type net to zero in \mathbb{F} is essentially a multiset permutation check with large multiplicities, and can be stated in the form of the reciprocal argument as

$$f(X) = \sum_{i=1}^k \frac{(-1)^{o_i} v_i}{X + t_i} = 0. \tag{41}$$

From Lemma 1 it follows that if $f(\alpha) = 0$ for a uniformly random α then with overwhelming probability the total multiplicity associated to each t_i must be zero in \mathbb{F} . From the structure of the function, this total multiplicity is the sum of all the inputs of that type minus the sum of all the outputs of that type, and so the total multiplicity is zero in \mathbb{F} if and only if the amounts net to zero in \mathbb{F} .

Taking these together, we can show that the total amount (i.e., multiplicity) of each type of asset nets to zero in \mathbb{Z} . We know by assumption that each transaction input amount lies in $[0, B)$, and we know from the range proof that each transaction output amount lies in $[0, B)$. Therefore, the total multiplicity \hat{v}_t of any type of asset lies in $(-kB, kB)$, which occurs in a transaction with k inputs or k outputs all of the same type and maximum amount. Since $kB < p$, this value cannot wrap around the field, so if $\hat{v}_t = 0$ in \mathbb{F} and $\hat{v}_t \in (-kB, kB)$, then the amounts net to zero in \mathbb{Z} .

Arithmetic Circuit. Each input and each output commit to two values, so $N_v = 2$. As in the reciprocal range proofs, all multiplicative constraints are reciprocal constraints and the matrices W_L, W_R have zero rows. The protocol can use any reciprocal range proof, and for the purposes of this protocol assume one is fixed by a reciprocal form circuit \mathcal{RC} for either a shared or inline digit range proof for all v_i with $o_i = 1$ for the range $[0, B)$.

We will append the vector \mathbf{t} of types to \mathbf{w}_D from the range proof, and we will add copy constraints to check that these are the same values from the input commitments. Note these copy constraints should be interleaved with the range proof linear constraints to line up with \mathbf{t} in \mathbf{w}_V . Each reciprocal in Eq. (41) has $v_i(-1)^{o_i}$ as its numerator and $X + t_i$ as its denominator. We will define $w_{P,i}(X)$ to be the unsigned reciprocals $w_{P,i}(X) = v_i/(X + t_i)$. Since multiplicative constraints cannot directly access inputs, we also need to add constraints to copy v_i into \mathbf{w}_I and modify W_n such that This lets us simplify $(W_n \mathbf{w}_I + \mathbf{a}_n)_i = v_i$. We can insert dummy constraints that check $t_i = t_i$ in the linear constraints so that the inputs align with the constraint matrix. To check that Eq. (41) holds, we can then append a row $W_p(X)$ so that

$$(W_p(X)_0, \mathbf{w}) = \sum_{i=1}^k (-1)^{o_i} w_{P,i}(X).$$

This completes the MACT arithmetic circuit. In total, each input adds only one element to \mathbf{w}_D and $\mathbf{w}_P(X)$, one copy constraint to W_d and one, trivial, row to W_n . There is also one constraint in $W_p(X)$ to check Eq. (41).

The marginal cost of a MACT over an aggregated range proof is negligible in prover time, verifier time, and proof size. This is in stark contrast to existing protocols which either require large proofs, complex circuits, and require trading off multi-party proving for the full relation.

Theorem 5 (Multi-Asset Confidential Transactions). *The confidential transaction protocol, instantiated with any of the reciprocal range proofs is a zero-knowledge argument of knowledge for the MACT relation Eq. (40) assuming the expected-PPT hardness of the discrete logarithm relation problem.*

Proof. Since the MACT protocol is an instantiation of the reciprocal form arithmetic circuit protocol, it has completeness and perfect SHVZK and CWEE. Therefore, it is sufficient to show that this circuit is satisfied if and only if the protocol inputs \mathbf{v} and \mathbf{t} satisfy the relation. By Theorem 4, we know that all the transaction output commitments commit to values in $[0, B)$ if they satisfy the circuit, and we know by assumption that all inputs lie in this range. Since $kB < p$, the magnitude of the total multiplicity of any type of asset cannot exceed p . The circuit invokes the reciprocal argument on the collection A formed as $\{(v, t) : (v, t) \in I\} \cup \{(-v, t) : (v, t) \in O\}$. By the soundness of the reciprocal form circuits, A vanishes, so total multiplicity of each token type must be zero in \mathbb{F} . Therefore, the total multiplicity of each type of asset must be 0 as an integer.

7 Implementation and Benchmarks

To demonstrate the real-world performance of BP++, we provide a reference implementation in C [19] as well as benchmarks. Our implementation builds on top of the libsecp256k1-zkp library [44] and thus uses secp256k1, the elliptic curve used in Bitcoin and many other cryptocurrencies. All operations on secret data performed by the prover implementation are constant-time. The experiments were performed on an Intel i7-10510U system at 1.80 GHz using a single thread. The implementation uses a single multi-exponentiation algorithm and scalar

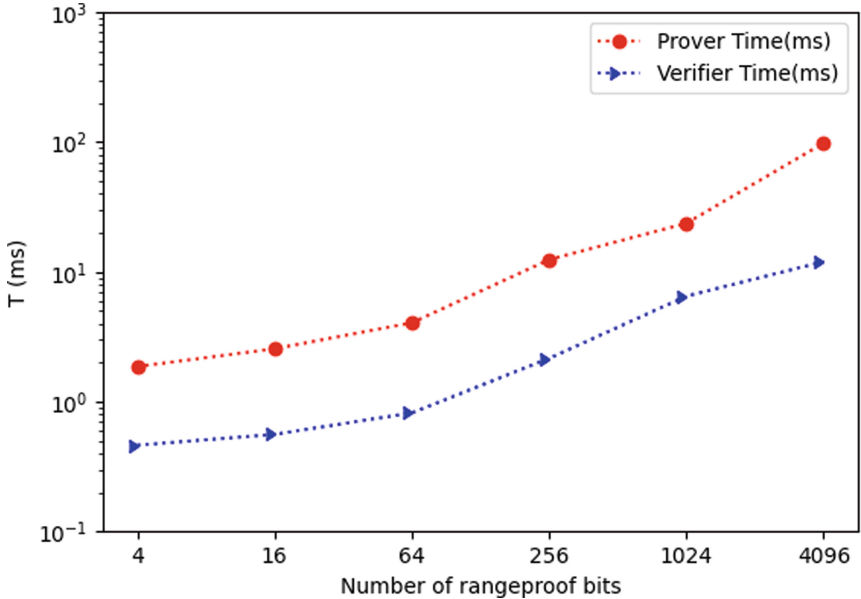


Fig. 1. Proving and verification time for BP++ range proofs. X-axis shows the total number of bits in the range proof. For $x > 64$ bits, we consider an aggregation of 64-bit range proofs. Y-axis shows the time in milliseconds.

Table 2. Proving and verification time compared to prior work.

	BP++	BP+ ([14])	BP ([17])	BP ([42])
Curve	secp256k1	Ristretto255	Ristretto255	secp256k1
Range	Prover time (ms)			
1×64	4.041	11.851	12.136	19.241
32×64	52.108	307.26	384.20	499.060
Range	Verifier time (ms)			
1×64	0.840	1.815	1.907	2.223
32×64	6.424	28.920	29.490	33.548

precomputation optimizations. In summary, verifying a 64-bit range proof took about 0.9 ms and proving about 4 ms. Figure 1 shows the proving and verification time as a function of the total number of range proof bits.

In order to compare the performance of BP++ with existing implementations of BP and BP+, we ran a BP implementation on secp256k1 [42], a BP implementation on Ristretto255 [17] and a BP+ implementation on Ristretto255 [14]. The results are summarized in Table 2. Despite secp256k1 having slower group operations than Ristretto255, for a 64-bit range proof, the BP++ prover is about 3 times and the verifier about 2.2 times faster than the BP+ implementation. The performance improvement in BP++ is amplified when aggregating multiple range proofs, e.g., when aggregating 32 64-bit range proofs, the BP++ prover and verifier are about 5–6 times faster than BP+. Moreover, based on SwiftRange’s [52] comparison to BP, we anticipate BP++ to outperform SwiftRange significantly, with roughly 3 times faster proving speed and 1.3 times faster verification speed.³

References

1. Attema, T., Cramer, R.: Compressed Σ -protocol theory and practical application to plug & play secure algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 513–543. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_18
2. Attema, T., Fehr, S., Kloof, M.: Fiat-Shamir transformation of multi-round interactive proofs. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 113–142. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22318-1_5
3. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_17
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM CCS 93 (1993). <https://doi.org/10.1145/168588.168596>
5. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_12
6. Bowe, S., Grigg, J., Hopwood, D.: Halo: recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019). <https://eprint.iacr.org/2019/1021>
7. Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification (2023). Version 2023.4.0 <https://github.com/zcash/zips/blob/main/protocol/protocol.pdf>
8. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (2018). <https://doi.org/10.1109/SP.2018.00020>

³ A direct comparison has only limited meaning due to the use of different programming languages, but our BP++ C implementation is about 20x times faster than the SwiftRange Java implementation for 64-bit range proof proving and verification.

9. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 1–18. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_1
10. Bünz, B., Fisch, B., Szeponiec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_24
11. Camenisch, J., Chaabouni, R., shelat, A.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_15
12. Chaabouni, R., Lipmaa, H., Shelat, A.: Additive combinatorics and discrete logarithm based range protocols. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 336–351. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14081-5_21
13. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 499–530. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30617-4_17
14. Chung, H., Han, K., Ju, C., Kim, M., Seo, J.H.: Bulletproofs+ implementation. <https://github.com/KyoohyungHan/BulletProofsPlus/commit/2c9dd40>
15. Chung, H., Han, K., Ju, C., Kim, M., Seo, J.H.: Bulletproofs+: shorter proofs for a privacy-enhanced distributed ledger. *IEEE Access* **10** (2022)
16. Couteau, G., Klooß, M., Lin, H., Reichle, M.: Efficient range proofs with transparent setup from bounded integer commitments. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 247–277. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77883-5_9
17. Dalek Cryptography Bulletproofs. <https://github.com/dalek-cryptography/bulletproofs/commit/be67b6d5f5ad1c1f54d5511b52e6d645a1313d07>
18. Eagen, L., Fiore, D., Gabizon, A.: CQ: cached quotients for fast lookups. *Cryptology ePrint Archive*, Report 2022/1763 (2022). <https://eprint.iacr.org/2022/1763>
19. Eagen, L., Kanjalkar, S., Ruffing, T., Nick, J.: Bulletproofs++ C implementation used for benchmarks. <https://github.com/sanket1729/secp256k1-zkp/commit/785f9d728086dd5b9c697ca4d452c517b8243a85>
20. Eagen, L., Kanjalkar, S., Ruffing, T., Nick, J.: Bulletproofs++: next generation confidential transactions via reciprocal set membership arguments. *Cryptology ePrint Archive*, Paper 2022/510 (2022). <https://eprint.iacr.org/2022/510>
21. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. *Cryptology ePrint Archive*, Report 2020/315 (2020). <https://eprint.iacr.org/2020/315>
22. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, Report 2019/953 (2019). <https://eprint.iacr.org/2019/953>
23. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-Shamir bulletproofs are non-malleable (in the random oracle model). *Cryptology ePrint Archive*, Report 2023/147 (2023). <https://eprint.iacr.org/2023/147>
24. Ghesmati, S., Fdhila, W., Weippl, E.R.: SoK: how private is bitcoin? Classification and evaluation of bitcoin privacy techniques. In: ARES 2022 (2022)
25. Ghoshal, A., Tessaro, S.: Tight state-restoration soundness in the algebraic group model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 64–93. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_3

26. Gibson, A.: An Investigation into Confidential Transactions (2016). <https://github.com/AdamISZ/ConfidentialTransactionsDoc/raw/master/essayonCT.pdf>
27. Grin. <https://www.grin-tech.org/>
28. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_32
29. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
30. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_22
31. Haböck, U.: Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530 (2022). <https://eprint.iacr.org/2022/1530>
32. Jaeger, J., Tessaro, S.: Expected-time cryptography: generic techniques and applications to concrete soundness. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 414–443. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64381-2_15
33. Lee, J.: Dory: efficient, transparent arguments for generalised inner products and polynomial commitments. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 1–34. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90453-1_1
34. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptol. **16**(3) (2003). <https://doi.org/10.1007/s00145-002-0143-7>
35. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40061-5_26
36. Maxwell, G.: CoinJoin: Bitcoin privacy for the real world (2013). BitcoinTalk post, <https://bitcointalk.org/index.php?topic=279249.0>
37. Maxwell, G.: Confidential Transactions (2015). https://web.archive.org/web/20190502140939/https://people.xiph.org/~greg/confidential_values.txt
38. Meiklejohn, S., et al.: A fistful of bitcoins: characterizing payments among men with no names. In: Internet Measurement Conference (IMC). <https://doi.org/10.1145/2504730.2504747>
39. Monero. <https://monero.org/>
40. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). <https://bitcoin.org/bitcoin.pdf>
41. Nick, J., Poelstra, A., Sanders, G.: Liquid: a bitcoin sidechain. Technical report (2020). <https://blockstream.com/assets/downloads/pdf/liquid-whitepaper.pdf>
42. Poelstra, A.: Bulletproofs implementation in libsecp256k1-zkp. <https://github.com/BlockstreamResearch/secp256k1-zkp/pull/23/commits/6fb7e05>
43. Poelstra, A., Back, A., Friedenbach, M., Maxwell, G., Wuille, P.: Confidential assets. In: Zohar, A., et al. (eds.) FC 2018. LNCS, vol. 10958, pp. 43–63. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_4
44. Poelstra, A. et al.: libsecp256-zkp. See <https://github.com/ElementsProject/secp256k1-zkp>
45. Posen, J., Kattis, A.A.: Caulk+: table-independent lookup arguments. Cryptology ePrint Archive, Report 2022/957 (2022). <https://eprint.iacr.org/2022/957>
46. Ruffing, T., Moreno-Sanchez, P.: ValueShuffle: mixing confidential transactions for comprehensive transaction privacy in bitcoin. In: Brenner, M., et al. (eds.) FC

2017. LNCS, vol. 10323, pp. 133–154. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_8
47. sethforprivacy: Monero will undergo a network upgrade on 13th August, 2022 (2022). <https://web.getmonero.org/2022/04/20/network-upgrade-july-2022.html>
48. Spagnuolo, M., Maggi, F., Zanero, S.: BitIodine: extracting intelligence from the bitcoin network. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 457–468. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_29
49. The Halo 2 Developers: Halo2 (2020). <https://zcash.github.io/halo2/>
50. Valence, H. de, Yun, C., Andreev, O.: Cloak (2019). <https://github.com/stellar/slingshot/blob/main/spacesuit/spec.md>
51. Wang, N., Chau, S.C.-K.: Flashproofs: efficient zero-knowledge arguments of range and polynomial evaluation with transparent setup. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 219–248. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22966-4_8
52. Wang, N., Chau, S.C.-K., Liu, D.: SwiftRange: a short and efficient zero-knowledge range argument for confidential transactions and more. Cryptology ePrint Archive, Paper 2023/1185 (2023). <https://eprint.iacr.org/2023/1185>
53. Wikström, D.: Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265 (2021). <https://eprint.iacr.org/2021/1265>
54. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., Simkin, M.: Caulk: lookup arguments in sublinear time. In: ACM CCS 2022 (2022). <https://doi.org/10.1145/3548606.3560646>