Bulletproofs++. Construction and Examples

Oleg Fomenko, Mike Sokolov
Distributed Lab

February, 2024

Contents

Abstract

This document introduces some clarification and corrections to Bulletproofs++ [Eag+23] paper. It briefly describes the fundamental protocol "weight norm linear argument", the improved circuits protocol, and the reciprocal range proof protocol. The main contribution of this work is to introduce a fixed version of arithmetic circuit protocol based on the discrete logarithm setting. Reading the original paper before is recommended to understand better the principles described in this work. Also, the original paper contains completeness and soundness proofs that can be easily transformed to be applied to our protocol.

Special thanks to Alex Kurbatov for his kind review.

1 Introduction

Bulletproofs++ protocol presented in [Eag+23] as an improvement to the BP [Bun+17] and BP+ [Chu+20]. Previous versions of the protocol have been developed primarily for zero-knowledge range proofs based on discrete logarithm problems. Bulletproofs++ extends their opportunities and allows proving arithmetic circuits without transferring extra commitments, as in BP and BP+. It consists of three main parts: weight norm linear argument protocol, arithmetic circuit protocol, and high-level protocols that can be built on top of them (for example - range proofs).

Weight norm linear argument protocol is a recursive algorithm for zero-knowledge proving that the user knows two vectors that satisfy some relation on public parameters. That protocol uses the vector reduction approach, allowing us to move from proving knowledge of N-sized vectors to N/2-sized vectors. The main difference from the similar protocols in BP and BP+ is that the weight norm argument keeps the equality relation on vectors: if the original vectors were equal, the reduced vectors would be equal as well.

The high-level idea of the circuit protocol is to combine arithmetic circuit relation into two vectors using challenges from the verifier and, after — prove the knowledge of them using weight norm linear argument protocol. This protocol is constructed in such a way that it will not be possible to make a structure of weight norm linear argument if the circuit conditions are not satisfied. Also, the Bulletproofs++ protocol defines the custom representation of arithmetic circuits. The original paper [Eag+23] describes an approach to format any BP-compatible circuit [Bun+17] to the BP++ style. So, because we can represent any arithmetic circuit in the BP circuit representation and we also can represent any BP circuit in BP++ — this gives us the availability to represent any arithmetic circuit in BP++.

2 Preliminaries

The section is partially taken from [Eag+23] because of the similar definitions and same scope.

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. Vectors are written with bold letters and matrices with capital letters. These can be distinguished from $\mathbb{G}$ elements from context.

Namely, we define a group element $G \in \mathbb{G}$ and vectors of group elements $\boldsymbol{G} \in \mathbb{G}^{N_m}$ and $\boldsymbol{H} \in \mathbb{G}^{9+N_v}$. All of them are independently generated and public. We denote the vector of all zeros by $\mathbf{0}$ and the vector of all ones by $\mathbf{1}$.

Slices are defined using subscript as $\boldsymbol{v}_{i:}$ with meaning that we take elements from i till the end of the vector and $\boldsymbol{v}_{:i}$ with meaning that we take elements from the beginning to $i-1$ inclusive. To access a slice within a vector, we write $(\boldsymbol{v}_{k:})_i = \boldsymbol{v}_{k+i}$.

We write the inner product of two vectors using angle brackets:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n} \boldsymbol{x}_i \cdot \boldsymbol{y}_i \tag{1}$$

Sometimes we will write this inner product as a multiplication of vectors with $*$ or without it to simplify the equations. Also, we define a **weight inner product** as an inner product with a subscript to denote weighting by powers of the subscript:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\mu = \sum_{i=0}^{n} \boldsymbol{x}_i \cdot \boldsymbol{y}_i \cdot \mu^{i+1} \tag{2}$$

Inner products are defined for any vectors of quantities that can be multiplied, i.e. scalars and scalars or scalars and group elements. If the vectors in multiplications have different sizes, the smaller vector will be appended with zeros from the corresponding cyclic group. The norm of a vector refers to its self-inner product and uses the same subscripting convention for weights:

$$|\boldsymbol{x}|_\mu^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle_\mu \tag{3}$$

We write concatenation of vectors using $||$ and tensor product of vectors using $\otimes$. Also, we use the $diag(\mu)^{-1}$ notation for diagonal matrix with $\mu^{-i}, i \in \{0, 1, 2...\}$ on the main diagonal. The size of this matrix can be defined from context. We denote the vector of powers from μ^0 to μ^{n-1} by $e_n(\mu)$.

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 $[\boldsymbol{a}]_0$ and $[\boldsymbol{a}]_1$ 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.

3 Weight norm linear argument

Like BP and BP+, BP++ in [Eag+23] introduces the new version of recursive proving of knowledge of committed vectors that satisfy some conditions. This fundamental basis is called **weight norm linear argument** (WNLA). The following section briefly describes how WNLA works according to the original document without any changes.

3.1 Definition

The WNLA defines the following relation and the proof scheme for it:

- For the public vector $\boldsymbol{c} \in \mathbb{F}^l$ and value $\mu \in \mathbb{F}$
- Public $G \in \mathbb{G}, \boldsymbol{G} \in \mathbb{G}^n, \boldsymbol{H} \in \mathbb{G}^l$
- And private vectors $\boldsymbol{l} \in \mathbb{F}^l, \boldsymbol{n} \in \mathbb{F}^n$
- Let's define the commitment $vG + \langle \boldsymbol{l}, \boldsymbol{H} \rangle + \langle \boldsymbol{n}, \boldsymbol{G} \rangle$, where $v = \langle \boldsymbol{c}, \boldsymbol{l} \rangle + |\boldsymbol{n}|_\mu^2$
- Then, using the weight norm linear argument we can prove the knowledge of private vectors for the given commitment.

3.2 Vector reduction

In the BP's [Bun+17] linear argument proof during vectors reduction we are moving from $\boldsymbol{x}, \boldsymbol{y}$ to a half-length $\boldsymbol{x}', \boldsymbol{y}'$, but unfortunately it does not keep the relation of initial vectors equivalence: if $\boldsymbol{x} = \boldsymbol{y}$ then $\boldsymbol{x}' \neq \boldsymbol{y}'$. The WNLA fixes that, by defining the new rules of vector reduction for the given challenges γ and ρ, such that $\rho^2 = \mu$:

$$\boldsymbol{x}' = \rho^{-1}[\boldsymbol{x}]_0 + \gamma[\boldsymbol{x}]_1 \tag{4}$$

$$\boldsymbol{y}' = \rho^{-1}[\boldsymbol{y}]_0 + \gamma[\boldsymbol{y}]_1 \tag{5}$$

It is easy to check that the following equations allow us to represent the reduced vectors' weight multiplication through the original vectors' weight multiplication:

$$v_x = \rho^{-1}(\langle [\boldsymbol{x}]_0, [\boldsymbol{y}]_1 \rangle_{\mu^2} + \langle [\boldsymbol{x}]_1, [\boldsymbol{y}]_0 \rangle_{\mu^2}) \tag{6}$$

$$v_r = \langle [\boldsymbol{x}]_1, [\boldsymbol{y}]_1 \rangle_{\mu^2} \tag{7}$$

$$\langle \boldsymbol{x}', \boldsymbol{y}' \rangle_{\mu^2} = \langle \boldsymbol{x}, \boldsymbol{y} \rangle_\mu + v_x\gamma + v_r(\gamma^2 - 1). \tag{8}$$

3.3 Protocol description

Now, there are 4 products that we have to reduce: $|\boldsymbol{n}|^2_\mu, \langle \boldsymbol{n}, \boldsymbol{G}\rangle, \langle \boldsymbol{c}, \boldsymbol{l}\rangle$ and $\langle \boldsymbol{l}, \boldsymbol{H}\rangle$. Because $\boldsymbol{n}$ takes part in weight multiplication (weight norm) we also have to modify the relation for $\boldsymbol{G}$ to match our specific reduced vector, and for the $\boldsymbol{c}$, $\boldsymbol{l}$, $\boldsymbol{H}$ vectors we have to use the unweighted reduction (without ρ^{-1}) because they are used only in default multiplications.

$$\boldsymbol{v'} = |\boldsymbol{n}'|^2_{\mu^2} + \langle \boldsymbol{c}', \boldsymbol{l}'\rangle \tag{9}$$

$$\boldsymbol{l}' = [\boldsymbol{l}]_0 + \gamma[\boldsymbol{l}]_1 \tag{10}$$

$$\boldsymbol{c}' = [\boldsymbol{c}]_0 + \gamma[\boldsymbol{c}]_1 \tag{11}$$

$$\boldsymbol{H'} = [\boldsymbol{H}]_0 + \gamma[\boldsymbol{H}]_1 \tag{12}$$

$$\boldsymbol{n}' = \rho^{-1}[\boldsymbol{n}]_0 + \gamma[\boldsymbol{n}]_1 \tag{13}$$

$$\boldsymbol{G'} = \rho[\boldsymbol{G}]_0 + \gamma[\boldsymbol{G}]_1 \tag{14}$$

Using that definition we can define the following equations and commitments:

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

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

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

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

Then, it becomes able to construct a commitment in reduced vectors for both verifier and prover:

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

Finally, we have discussed all protocol peculiarities and now are able to define the algorithm: prover calculates commitments X, R and sends them to the verifier, after that the verifier shares the challenge and both can move to the next iteration with $\rho' = \mu, \mu' = \mu^2$. The original paper [Eag+23] notes that recursive iterations with vector reduction can optimize protocol until $|\boldsymbol{l}| + |\boldsymbol{n}| < 6$. So, once that boundary has been reached, the prover can just send current $\boldsymbol{l'}$ and $\boldsymbol{n'}$ vectors to the verifier for the final check (that they satisfy the current commitment). Protocol completeness follows from the protocol definition and protocol soundness follows from the linear independence of the $1, \gamma, \gamma^2 - 1$ polynomials in the weight multiplication reduction. Check out the original paper for the full algorithm description.

4 Arithmetic circuit

The circuit definition lies on the following parameters and values:

- Public matrices $W_l \in \mathbb{F}^{N_l \times N_w}, W_m \in \mathbb{F}^{N_m \times N_w}$,
- Public vectors $\boldsymbol{a}_l \in \mathbb{F}^{N_l}, \boldsymbol{a}_m \in \mathbb{F}^{N_m}$,
- Two public boolean flags f_l, f_m
- Public point $G \in \mathbb{G}$, vectors of points $\boldsymbol{G} \in \mathbb{G}^{N_m}, \boldsymbol{H} \in \mathbb{G}^{9+N_v}$,
- Commitments to the circuit input witness $\boldsymbol{V} \in \mathbb{G}^k$, where k - number of witness vectors,
- Circuit input witness $\boldsymbol{v}_i \in \mathbb{F}^{N_v}, i \in [0, k-1]$,
- Private commitment blinding $\boldsymbol{s}_{v,i} \in \mathbb{F}$,

- And more private witness $\boldsymbol{w}_L, \boldsymbol{w}_R \in \mathbb{F}^{N_m}, \boldsymbol{w}_O \in \mathbb{F}^{N_O}$.

Additionally, we define witnesses vectors $\boldsymbol{w}$ and $\boldsymbol{w}_v$ as:

$$\boldsymbol{w} = \boldsymbol{w}_L||\boldsymbol{w}_R||\boldsymbol{w}_O \tag{20}$$

$$\boldsymbol{w}_v = v_0||v_1||...||v_{k-1} \tag{21}$$

The arithmetic circuit representation from [Eag+23] should satisfy the following relation:

$$W_l\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l = \boldsymbol{0} \tag{22}$$

$$W_m\boldsymbol{w} + f_m\boldsymbol{w}_v + \boldsymbol{a}_m = \boldsymbol{w}_L \circ \boldsymbol{w}_R \tag{23}$$

where the input commitment to witness is:

$$V_i = Com(v_i, s_{v,i}) = v_{i,0}G + s_{v_i}H_0 + \langle \boldsymbol{H}_{9:}, \boldsymbol{v}_{i,1:}\rangle \tag{24}$$

It is necessary to highlight that W_l should be perceived as consisting of three components related to $\boldsymbol{r}, \boldsymbol{l}, \boldsymbol{o}$ witness (such as $W_l\boldsymbol{w} = W_{l,R}\boldsymbol{w}_R + W_{l,L}\boldsymbol{w}_R + W_{l,O}\boldsymbol{w}_O$). It's also become obvious that $N_w = N_m + N_m + N_O$ and $N_l = N_v * k$.

4.1 General approach

To combine the circuit definition into one equation we can use a general principle in cryptography (for example in ZK-SNARKs) - combine multiple constraints into one by using challenges, so let's represent the arithmetic circuit definition using challenges λ and μ as:

$$\begin{aligned}\boldsymbol{0} = e_{N_l}(\lambda)^T * (W_l\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l) + \langle \boldsymbol{w}_L, \boldsymbol{w}_R\rangle_\mu \\ -\mu * e_{N_m}(\mu)^T * (W_m\boldsymbol{w} + f_m\boldsymbol{w}_v + \boldsymbol{a}_m)\end{aligned} \tag{25}$$

It's easy to observe that the first part of the equation $e_{N_l}(\lambda)^T * (W_l\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l)$ is the same as multiplied by $(1, \lambda, \lambda^2, ...)$ vector $W_l\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l$ and the second part $\langle \boldsymbol{w}_L, \boldsymbol{w}_R\rangle_\mu - \mu * e_{N_m}(\mu)^T * (W_m\boldsymbol{w} + f_m\boldsymbol{w}_v + \boldsymbol{a}_m)$ is derived from $\boldsymbol{w}_L \circ \boldsymbol{w}_R = W_m\boldsymbol{w} + f_m\boldsymbol{w}_v + \boldsymbol{a}_m$ by multiplying on $(1, \mu, \mu^2, ...)$ vector.

According to the described solution in the original document [Eag+23] we will try to represent our circuit in the following function from given challenge T:

$$f(T) = v(T) - \langle \boldsymbol{c(T)}, \boldsymbol{l(T)}\rangle - |\boldsymbol{n(T)}|^2_\mu = 0 \tag{26}$$

It will allow us to use weight norm linear argument proof in the final stage of arithmetic circuit protocol to prove that we know such $\boldsymbol{l(T)}$ and $\boldsymbol{n(T)}$ that satisfies norm linear argument commitment.

The goal that we have to achieve is to construct such commitment $C = vG + \langle \boldsymbol{l}, \boldsymbol{H}\rangle + \langle \boldsymbol{n}, \boldsymbol{G}\rangle$ where the $v = v(T) = \langle \boldsymbol{c(T)}, \boldsymbol{l(T)}\rangle + |\boldsymbol{n(T)}|^2_\mu$ derived from $f(T)$ and $\boldsymbol{l} = \boldsymbol{l(T)}, \boldsymbol{n} = \boldsymbol{n(T)}$. Note that $v = v(T) = \langle \boldsymbol{c(T)}, \boldsymbol{l(T)}\rangle + |\boldsymbol{n(T)}|^2_\mu$ equation will be satisfied only in case when witness satisfies the arithmetic circuit definition.

4.2 Witness commitments

Firstly, let's consider to group all **value terms** (terms that depends on circuit witness) in the same T^i coefficient in $f'(T)$ function. According to the original document it will be the T^3 term. So, the sub-goal for now is to construct function $f'(T) = v(T) - \langle \boldsymbol{\hat{c}_l(T)}, \boldsymbol{\hat{l}(T)}\rangle - |\boldsymbol{n(T)}|^2_\mu$ in which the coefficient near T^3 will be zero in case when the circuit equation satisfied. All the other terms will be nullified by moving from $\boldsymbol{\hat{c}_l(T)}, \boldsymbol{\hat{l}(T)}$ to $\boldsymbol{c(T)}, \boldsymbol{l(T)}$.

Let's start with the definition of the witness (R, L, O) and blinding commitments (S). The C_L, C_R, C_O and C_S commitments that will be defined as following:

$$C_X = \langle \boldsymbol{r}_X || \boldsymbol{l}_X, \boldsymbol{H} \rangle + \langle \boldsymbol{n}_X, \boldsymbol{G} \rangle, X \in \{L, R, O, S\} \tag{27}$$

The $\boldsymbol{r}_X \in \mathbb{F}^9$ vectors will be defined later when we will consider to nullify the entire $f(T)$ function. Also $\boldsymbol{n}_R = \boldsymbol{w}_R$ and $\boldsymbol{n}_L = \boldsymbol{w}_L$, and the other $\boldsymbol{l}_X \in \mathbb{F}^{N_v}$ and $\boldsymbol{n}_X \in \mathbb{F}^{N_m}$ vectors will be a mapping of $\boldsymbol{w}_O$ witness vector by the special injective mapping function $\Psi : \Psi(x, j) = i$, where $x \in \{\boldsymbol{l}_O, \boldsymbol{l}_L, \boldsymbol{l}_R, \boldsymbol{n}_O\}$, j is a index of element in vector $\boldsymbol{x}$ and i is an corresponding index of $\boldsymbol{w}_O$.

Using such mapping we also have to map the circuit public W_m, W_l matrix into the $M_{a,t,X}, a \in \{l, m\}, t \in \{l, n\}, X \in \{L, R, O\}$. The goal is to achieve the following equation:

$$W_a w = \sum_{X \in \{L,R,O\}} M_{a,l,X} l_X + M_{a,n,X} n_X \tag{28}$$

Because we have $\boldsymbol{n}_L = \boldsymbol{w}_L$ and $\boldsymbol{n}_R = \boldsymbol{w}_R$ then $M_{a,n,L} = W_{a,L}$ and $M_{a,n,R} = W_{a,R}$. Other columns will be mapped using also the Ψ function applied to the columns of $W_{a,O}$. Column j of matrix $M_{a,t,X}$ equals to the column i of $W_{a,O}$ if $i = \Psi(t_X, j), t \in \{l, n\}$ and to the zero column otherwise. For example: $M_{l,l,L}$ j column will be equal to column i of $W_{l,O}$ if $\Psi(l_L, j) = i$ and in $M_{m,n,R}$ j column will be equal to column i of $W_{m,O}$ if $\Psi(\boldsymbol{n}_R, j) = i$.

For the simple circuits the Ψ can be defined as a mapping of all elements of $\boldsymbol{w}_O$ into the $\boldsymbol{n}_O$ or $\boldsymbol{l}_L$ but also depends on your conditions.

4.3 Circuit equations

Let's turn back to the initial circuit equation, to describe clearly some following definitions:

$$\begin{gathered}\boldsymbol{0} = e_{N_l}(\lambda)^T * (W_l \boldsymbol{w} + f_l \boldsymbol{w}_v + \boldsymbol{a}_l) + \langle \boldsymbol{w}_L, \boldsymbol{w}_R \rangle_\mu \\ -\mu * e_{N_m}(\mu)^T * (W_m \boldsymbol{w} + f_m \boldsymbol{w}_v + \boldsymbol{a}_m)\end{gathered}$$

Dealing with it we have to define the linear combinations of witness as well as linear combination of circuit parameters based on input challenges μ and λ. Let's define the $\boldsymbol{\lambda}'$ and $\boldsymbol{\mu}'$ vectors:

$$\boldsymbol{\lambda}' = e_{N_l}(\lambda)^T \tag{29}$$

$$\boldsymbol{\mu}' = \mu * e_{N_m}(\mu)^T \tag{30}$$

So, now we have:

$$\begin{gathered}\boldsymbol{0} = \boldsymbol{\lambda}' * W_l \boldsymbol{w} + \boldsymbol{\lambda}' * f_l \boldsymbol{w}_v + \boldsymbol{\lambda}' * \boldsymbol{a}_l + \langle \boldsymbol{w}_L, \boldsymbol{w}_R \rangle_\mu \\ -\boldsymbol{\mu}' * W_m \boldsymbol{w} - \boldsymbol{\mu}' * f_m \boldsymbol{w}_v - \boldsymbol{\mu}' * \boldsymbol{a}_m\end{gathered}$$

Let's substitute the $W_l * \boldsymbol{w}$ and $W_m * \boldsymbol{w}$ multiplications:

$$\begin{aligned}\boldsymbol{0} = \boldsymbol{\lambda}' &\sum_{X \in \{L,R,O\}} (M_{l,l,X} \boldsymbol{l}_X + M_{l,n,X} \boldsymbol{n}_X) + \boldsymbol{\lambda}' f_l \boldsymbol{w}_v + \boldsymbol{\lambda}' * \boldsymbol{a}_l + \langle \boldsymbol{w}_L, \boldsymbol{w}_R \rangle_\mu \\ -\boldsymbol{\mu}' &\sum_{X \in \{L,R,O\}} (M_{m,l,X} \boldsymbol{l}_X + M_{m,n,X} \boldsymbol{n}_X) - \boldsymbol{\mu}' f_m \boldsymbol{w}_v - \boldsymbol{\mu}' \boldsymbol{a}_m\end{aligned}$$

If we group the $\boldsymbol{l}_X$ and $\boldsymbol{n}_X$ coefficients we will have:

$$\begin{aligned}\boldsymbol{0} = &\sum_{X \in \{L,R,O\}} (\boldsymbol{\lambda}' M_{l,l,X} - \boldsymbol{\mu}' M_{m,l,X}) \boldsymbol{l}_X + \boldsymbol{\lambda}' f_l \boldsymbol{w}_v + \boldsymbol{\lambda}' \boldsymbol{a}_l + \langle \boldsymbol{w}_L, \boldsymbol{w}_R \rangle_\mu \\ + &\sum_{X \in \{L,R,O\}} (\boldsymbol{\lambda}' M_{l,n,X} - \boldsymbol{\mu}' M_{m,n,X}) \boldsymbol{n}_X - \boldsymbol{\mu}' f_m \boldsymbol{w}_v - \boldsymbol{\mu}' \boldsymbol{a}_m\end{aligned}$$

Let's define the following vectors to simplify the equation:

$$\boldsymbol{c}_{n,X} = (\boldsymbol{\lambda}' M_{l,n,X} - \boldsymbol{\mu}' M_{m,n,X})(\mu^{-1} diag(\mu)^{-1}) \tag{31}$$

$$\boldsymbol{c}_{l,X} = \boldsymbol{\lambda}' M_{l,l,X} - \boldsymbol{\mu}' M_{m,l,X} \tag{32}$$

The resulting equation that we have is:

$$\begin{aligned}\mathbf{0} = f_l \boldsymbol{\lambda}' \boldsymbol{w}_v - f_m \boldsymbol{\mu}' \boldsymbol{w}_v + \boldsymbol{\lambda}' \boldsymbol{a}_l - \boldsymbol{\mu}' \boldsymbol{a}_m + \langle \boldsymbol{w}_L, \boldsymbol{w}_R \rangle_\mu \\ + \sum_{X \in \{L,R,O\}} (\langle \boldsymbol{c}_{l,X}, \boldsymbol{l}_X \rangle + \langle \boldsymbol{c}_{n,X}, \boldsymbol{n}_X \rangle_\mu)\end{aligned} \tag{33}$$

4.4 Linear combinations

As we've already defined the linear combination of the circuit public parameters we have to define the corresponding linear combination of witness vector $\boldsymbol{w}_v = concat(\boldsymbol{v})$ and its commitment V_i. The linear combination function will be defined as:

$$lcomb(i) = 2(f_l * \lambda^{N_v * i} + f_m * \mu^{N_v * i + 1}) \tag{34}$$

The linear combination of witness will be the vector:

$$\hat{\boldsymbol{v}} = \sum_{i=0}^{i<k} lcomb(i) * \boldsymbol{v}_i \tag{35}$$

The linear combination of private commitment blinding values will be the value:

$$\hat{s} = \sum_{i=0}^{i<k} lcomb(i) * s_{v_i} \tag{36}$$

And the corresponding commitment combination will be the point:

$$\hat{V} = \sum_{i=0}^{i<k} lcomb(i) * V_i \tag{37}$$

4.5 Circuit commitment

Now, using additional challenge δ from verifier let's define the commitment:

$$C = T^{-1} * C_S - \delta * C_O + T * C_L - T^2 * C_R + T^3 * \hat{V} \tag{38}$$

Let's expand the commitments computations:

$$\begin{gathered} C = T^{-1} \langle (\boldsymbol{r}_S || \boldsymbol{l}_S), \boldsymbol{H} \rangle + T^{-1} \langle \boldsymbol{n}_S, \boldsymbol{G} \rangle - \delta \langle (\boldsymbol{r}_O || \boldsymbol{l}_O), \boldsymbol{H} \rangle - \delta \langle \boldsymbol{n}_O, \boldsymbol{G} \rangle \\ + T \langle (\boldsymbol{r}_L || \boldsymbol{l}_L), \boldsymbol{H} \rangle + T \langle \boldsymbol{n}_L, \boldsymbol{G} \rangle - T^2 \langle (\boldsymbol{r}_R || \boldsymbol{l}_R), \boldsymbol{H} \rangle - T^2 \langle \boldsymbol{n}_R, \boldsymbol{G} \rangle \\ + T^3 \sum_{i=0}^{i<k} lcomb(i) * V_i \end{gathered}$$

And group the coefficients G, H:

$$\begin{gathered} C = \langle T^{-1} (\boldsymbol{r}_S || \boldsymbol{l}_S) - \delta (\boldsymbol{r}_O || \boldsymbol{l}_O) + T (\boldsymbol{r}_L || \boldsymbol{l}_L) - T^2 (\boldsymbol{r}_R || \boldsymbol{l}_R), \boldsymbol{H} \rangle + \\ + \langle T^{-1} \boldsymbol{n}_S - \delta \boldsymbol{n}_O + T \boldsymbol{n}_L - T^2 \boldsymbol{n}_R, \boldsymbol{G} \rangle + T^3 \sum_{i=0}^{i<k} lcomb(i) * V_i \end{gathered}$$

For clarity define the vector $\hat{\boldsymbol{sv}} = (\hat{s}, 0, ..., 0)||\hat{\boldsymbol{v}}_{1:}$ with dimension $9 + N_v - 1$. So the final equation is:

$$C = \langle T^{-1}(\boldsymbol{r}_S||\boldsymbol{l}_S) - \delta(\boldsymbol{r}_O||\boldsymbol{l}_O) + T(\boldsymbol{r}_L||\boldsymbol{l}_L) - T^2(\boldsymbol{r}_R||\boldsymbol{l}_R) + T^3\hat{\boldsymbol{sv}}, \boldsymbol{H}\rangle$$
$$+\langle T^{-1}\boldsymbol{n}_S - \delta\boldsymbol{n}_O + T\boldsymbol{n}_L - T^2\boldsymbol{n}_R, \boldsymbol{G}\rangle + T^3\hat{v}_0 G$$

It's obvious that there is something the same to the structure of weight norm linear argument (WNLA), where:

$$\boldsymbol{l(T)} = T^{-1}(\boldsymbol{r}_S||\boldsymbol{l}_S) - \delta(\boldsymbol{r}_O||\boldsymbol{l}_O) + T(\boldsymbol{r}_L||\boldsymbol{l}_L) - T^2(\boldsymbol{r}_R||l_R) + T^3\hat{\boldsymbol{sv}} \tag{39}$$
$$\hat{\boldsymbol{n}}\boldsymbol{(T)} = T^{-1}\boldsymbol{n}_S - \delta\boldsymbol{n}_O + T\boldsymbol{n}_L - T^2\boldsymbol{n}_R \tag{40}$$
$$\hat{v}(T) = T^3\hat{v}_0 \tag{41}$$

4.6 Polynomials

It is easy to see, that the defined commitment requires some changes to be used in weight norm linear argument. Let's define some equations that will be used in the $f'(T)$ definition. For the defined $\hat{V}$ there will be some additional terms that will appear during inner multiplication in case we have both $f_l = 1$ and $f_m = 1$. To deal with it we have to define additional vector that will compensate them. This vector is described in paragraph 5.2.2 of the original document [Eag+23]:

$$\boldsymbol{\lambda_{lm}} = f_l f_m(\mu e_{N_v}(\lambda) \otimes e_k(\mu^{N_v}) + e_{N_v}(\mu) \otimes e_k(\lambda^{N_v})) \tag{42}$$

The final $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ will be:

$$\boldsymbol{\lambda} = (\boldsymbol{\lambda'})^T - \boldsymbol{\lambda_{lm}} \tag{43}$$
$$\boldsymbol{\mu} = (\boldsymbol{\mu'})^T \tag{44}$$

And some polynomials that will be used in the final $f'(T)$ function (the correctness of defined function can be proved by T^3 term calculation — it should be fully derived from circuit equation):

$$\boldsymbol{p_n(T)} = \delta^{-1}T^3\boldsymbol{c}_{n,O} - T^2\boldsymbol{c}_{n,L} + T\boldsymbol{c}_{n,R} \tag{45}$$
$$p_s(T) = |\boldsymbol{p_n(T)}|^2_\mu + 2T^3(\langle\boldsymbol{\lambda}, \boldsymbol{a}_l\rangle - \langle\boldsymbol{\mu}, \boldsymbol{a}_m\rangle) \tag{46}$$
$$\hat{\boldsymbol{c}}\boldsymbol{_l(T)} = 2(\delta^{-1}T^3\boldsymbol{c}_{l,O} - T^2\boldsymbol{c}_{l,L} + T\boldsymbol{c}_{l,R}) + f_m\mu e_{N_v}(\mu)_{1:} - f_l e_{N_v}(\lambda)_{1:} \tag{47}$$
$$\boldsymbol{n(T)} = \boldsymbol{p_n(T)} + \hat{\boldsymbol{n}}\boldsymbol{(T)} \tag{48}$$

Finally, we can define the $f'(T)$ function according to the vectors in raw WNLA commitment definition from the previous section:

$$f'(T) = p_s(T) + \hat{v}(T) - \langle\hat{\boldsymbol{c}}\boldsymbol{_l(T)}, \hat{\boldsymbol{l}}\boldsymbol{(T)}\rangle - |\boldsymbol{n(T)}|^2_\mu \tag{49}$$

where the $\hat{\boldsymbol{l}}\boldsymbol{(T)} = T^{-1}\boldsymbol{l}_S - \delta\boldsymbol{l}_O + T\boldsymbol{l}_L - T^2\boldsymbol{l}_R + T^3\hat{\boldsymbol{v}}_{1:}$. It is simple to check that in the $f'(T)$ function coefficient near T^3 will be equal to:

$$2 * (f_l * \boldsymbol{\lambda'}\boldsymbol{w}_v - f_m * \boldsymbol{\mu'}\boldsymbol{w}_v + \boldsymbol{\lambda'}\boldsymbol{a}_l - \boldsymbol{\mu'}\boldsymbol{a}_m + \langle\boldsymbol{w}_L, \boldsymbol{w}_R\rangle_\mu$$
$$+ \sum_{X\in\{L,R,O\}}(\boldsymbol{c}_{l,X} * \boldsymbol{l}_X + \boldsymbol{c}_{n,X} * \boldsymbol{n}_X))$$

that is equal to 0 in case when the circuit conditions satisfied.

The last thing that we have to deal with is to nullify the whole $f(T)$ function, and then we will be able to express $v(t)$ from it and use the WNLA proof. As we've said earlier the $f(T) = f'(T) - g(T)$ and now we are able do define the $g(T)$:

$$g(T) = \langle\hat{\boldsymbol{c}}\boldsymbol{_r(T)}, \boldsymbol{r(T)}\rangle \tag{50}$$

where $\boldsymbol{\hat{c}_r(T)} = (1, \beta T^{-1}, \beta T, \beta T^2, \beta T^3, \beta T^4, \beta T^5, \beta T^6, \beta T^7)$ for the challenge β and $\boldsymbol{r(T)} = T^{-1}\boldsymbol{r}_S - \delta \boldsymbol{r}_O + T\boldsymbol{r}_L - T^2\boldsymbol{r}_R + T^3(\hat{s}, 0, ..., 0)$.

The $\boldsymbol{r}_S$ vector, as a part of commitment C_S should be defined before receiving the T challenge from verifier, in such way that $g(T)$ will nullify all T^i coefficients in function $f'(T)$. Let's name $f'[X]$ the coefficient in $f'(T)$ near T^X. Then, $\boldsymbol{r}_S$ vector can be obtained by solving the equation $f'[x] = \langle \boldsymbol{\hat{c}_r}, \boldsymbol{r(t)} \rangle [x]$ for unknown $\boldsymbol{r}_S$. For the T^3 coefficient we have special equation because $f'[x]$ is already zero, but additional $T^3\hat{s}$ appears from $\boldsymbol{r}(T)$ function $T^3\hat{s}$ term (and $T^3\hat{V}$ in the commitment). That is why we have to define $r_{S,5} = -\beta^{-1}\hat{s}$.

$$
\begin{aligned}
r_{S,0} &= f'[-1] + \beta\delta r_{O,1} \\
r_{S,1} &= \beta^{-1} f'[-2] \\
r_{S,2} &= \beta^{-1}(f'[0] + \delta r_{O,0}) - r_{L,1} \\
r_{S,3} &= \beta^{-1}(f'[1] - r_{L,0}) + r_{R,1} + \delta r_{O,2} \\
r_{S,4} &= \beta^{-1}(f'[2] + r_{R,0}) - r_{L,2} + \delta r_{O,3} \\
r_{S,5} &= -\beta^{-1}\hat{s} \\
r_{S,6} &= \beta^{-1} f'[4] + r_{R,3} - r_{L,4} + \delta r_{O,5} \\
r_{S,7} &= \beta^{-1} f'[5] + r_{R,4} - r_{L,5} + \delta r_{O,6} \\
r_{S,8} &= \beta^{-1} f'[6] + r_{R,5} - r_{L,6} + \delta r_{O,7}
\end{aligned}
$$

That definition also imposes restrictions on the $\boldsymbol{r}_X$ vectors: $r_{O,4} = r_{L,3} = r_{R,2} = 0$ and $r_{R,6:} = 0$, $r_{O,8} = 0$, $r_{L,7:} = 0$. Other $r_{X,i}, X \in \{L, R, O\}$ shold be selected randomly.

So, for the final function f we have the following equations and commitment that can be used in WNLA proof:

$$
\begin{aligned}
v(T) &= p_s(T) + \hat{v}(T) && (51) \\
\boldsymbol{l(T)} &= T^{-1}(\boldsymbol{r}_S || \boldsymbol{l}_S) - \delta(\boldsymbol{r}_O || \boldsymbol{l}_O) + T(\boldsymbol{r}_L || \boldsymbol{l}_L) - T^2(\boldsymbol{r}_R || \boldsymbol{l}_R) + T^3 \boldsymbol{\hat{s}\hat{v}} && (52) \\
\boldsymbol{n(T)} &= \boldsymbol{p_n(T)} + \boldsymbol{\hat{n}(T)} && (53) \\
\boldsymbol{c(T)} &= \boldsymbol{\hat{c}_r(T)} || \boldsymbol{\hat{c}_l(T)} && (54) \\
f(T) &= v(T) - \langle \boldsymbol{c(T)}, \boldsymbol{l(T)} \rangle - |\boldsymbol{n(T)}|^2_\mu && (55) \\
C(T) &= p_s(T)G + \langle \boldsymbol{p_n(T)}, \boldsymbol{G} \rangle + T^{-1}C_S - \delta C_O + TC_L - T^2 C_R + T^3 \hat{V} && (56) \\
&= v(T)G + \langle \boldsymbol{l(T)}, \boldsymbol{H} \rangle + \langle \boldsymbol{n(T)}, \boldsymbol{G} \rangle
\end{aligned}
$$

If the circuit conditions satisfied then $f(T) = 0$, so:

$$
\begin{aligned}
v(T) - \langle \boldsymbol{c(T)}, \boldsymbol{l(T)} \rangle - |\boldsymbol{n(T)}|^2_\mu &= 0 && (57) \\
v(T) &= \langle \boldsymbol{c(T)}, \boldsymbol{l(T)} \rangle + |\boldsymbol{n(T)}|^2_\mu && (58)
\end{aligned}
$$

That satisfies the commitment $C(T)$ to be used in WNLA.

Algorithm 1 Arithmetic circuit protocol

```
Input: $G, \boldsymbol{G}, \boldsymbol{H}, W_l, W_m, \boldsymbol{a}_l, \boldsymbol{a}_m, f_l, f_m, V, \boldsymbol{s}_v, \boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{w}_O, \boldsymbol{v}$
Require: Circuit equations satisfied
    • $\mathbb{P}$ calculates: $C_X, l_X, r_X, n_X$ for $X \in \{L, R, O\}$
    • $\mathbb{V} \to \mathbb{P} : \rho, \beta, \lambda, \delta$
    • Both calculates: $\mu = \rho^2, \boldsymbol{\lambda}, \boldsymbol{\mu}$, $M_{a,t,X}$ and $\boldsymbol{c}_{t,X}$ where $a \in \{l, m\}, t \in \{l, n\}, X \in \{L, R, O\}$
    • $\mathbb{P}$ selects random $l_S, n_S$ and calculates $r_S, C_S$ using polynomials and $f'(T)$ function coefficients to
      nullify target $f(T)$ function.
    • $\mathbb{P} \to \mathbb{V} : C_S$
    • $\mathbb{V} \to \mathbb{P} : \tau$
    • $\mathbb{P}$ calculates $\boldsymbol{l}(\tau), \boldsymbol{n}(\tau)$
    • Both calculates polynomials $p_s(\tau), \boldsymbol{p}_n(\tau), \hat{\boldsymbol{c}}_r(\tau), \hat{\boldsymbol{c}}_l(\tau)$ and $C(\tau), \boldsymbol{c}(\tau)$
    • Run Weight Norm Linear Argument protocol with input: $\boldsymbol{c} = \boldsymbol{c}(\tau), C = C(\tau), \boldsymbol{l} = \boldsymbol{l}(\tau), \boldsymbol{n} = \boldsymbol{n}(\tau)$
```

5 Reciprocal range proofs

After describing the arithmetic circuit protocol it becomes possible to construct range proofs on top of it. The obvious protocol for the binary system is to prove that every committed digit is a bit by checking that $digit_i * (digit_i - 1) = 0$. You can find description of this circuit in the Appendix. In general, for "b"-base system it corresponds to the check that every digit is a root of polynomial $\prod_{i=0}^{b-1}(x - i)$. It's possible to construct such proof using arithmetic circuits or just a weight norm linear argument protocol, but anyway, such proofs will not be more efficient then their analogs in BP[Bun+17] and BP+[Chu+20]. The BP++ [Eag+23] introduces reciprocal range proofs - the another way to check that every committed digit belongs to the set of valid base values. This section, in turn, describes the straightforward range proof protocol based on reciprocal argument unlike more complicated protocol from the the original paper.

According to the original paper, let's construct a set A of pairs (m_i, v_i) of values and their multiplicities (it can contain repeated pairs). We will say that A set **vanishes** if for the every v, sum of its multiplicities accross all pairs equals to 0. Using the function

$$f(X) = \sum_{(m,v) \in A} \frac{m}{X - v} \tag{59}$$

we can check that A vanishes by calculating $f(\alpha)$ with for a random selected α with overwhelming probability. Then, for the digits vector $\boldsymbol{d}$ and base b we can construct A as:

$$A = (-1, d_i)_{i=0}^{i<n} \cup (m_j, j)_{j=0}^{j<b} \tag{60}$$

where m_j is a total multiplicity for the base digit j — how many times j appears in $\boldsymbol{d}$. Now, if the A set vanishes then all digits satisfies range $[0, b)$. And, we can suspect a zero-knowledge protocol to check that A vanishes: commit to all digits d and multiplicities m, choose a challenge α and commit to the values:

$$r_i = \frac{1}{\alpha + d_i} \tag{61}$$

Then, to show that d_i is a valid base digit prove that:

$$r_i * (d_i + \alpha) = 1 \tag{62}$$

$$\sum_{i=0}^{i<n} r_i = \sum_{j=0}^{j<b} \frac{m_j}{\alpha + j} \tag{63}$$

5.1 Reciprocal circuit

Let's consider to describe the arithmetic circuit for the equations above. Using challenge received from verifier we will consider to generate witness and corresponding circuit matrices. For the given base b and number length n we can describe the following commitments to the value t with s_t blinding:

$$C_V = t * G + s_t * H_0 \tag{64}$$

where $t = \langle e_n(b), \boldsymbol{d} \rangle$ and $d_i \in [0, b)$. Then, after receiving the challenge α from verifier we can commit to the $r_i = \dfrac{1}{\alpha + d_i}$ with s_r blinding:

$$C_R = s_r * H_0 + \langle \boldsymbol{r}, \boldsymbol{H}_{9:} \rangle \tag{65}$$

The total commitment for the circuit will be:

$$C_V + C_R = t * G + (s_t + s_r) * H_0 + \langle \boldsymbol{r}, \boldsymbol{H}_{9:} \rangle \tag{66}$$

where the circuit witness is $v = t||\boldsymbol{r}$ with corresponding blinding $s_t + s_r$.

According to the described proving scheme, we have to prove that $r_i * (d_i + \alpha) = 1$ for our committed witness. It's obvious to use circuit equation that contains witness multiplication $w_L \circ w_R$ to perform $r_i * d_i$ check. Then, using $\boldsymbol{a}_m = \boldsymbol{1}, f_m = 0, \boldsymbol{w}_L = \boldsymbol{d}, \boldsymbol{w}_R = \boldsymbol{r}, \boldsymbol{w}_O = \boldsymbol{m}$ we can construct circuit matrix $W_m(\alpha)$ for the given challenge X in such way that $W_m(\alpha) * \boldsymbol{w} = -\alpha * \boldsymbol{r}$ (by setting $-\alpha$ on on the main diagonal of corresponding part of matrix that will be multiplied on $\boldsymbol{w}_R$ part of $\boldsymbol{w}$ vector). So, in the $W_m(\alpha)\boldsymbol{w} + \boldsymbol{a}_m = \boldsymbol{w}_L \circ \boldsymbol{w}_R$ circuit equation every row will encode check for $-\alpha * r_i + 1 = d_i * r_i$.

To encode check for the reciprocal sum we will use $W_l(\alpha)\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l = \boldsymbol{0}$ circuit equation with $f_l = 1$ and $\boldsymbol{a}_l = \boldsymbol{0}$ parameters. The $W_l(\alpha)$ matrix in this equation have to nullify the witness vector $\boldsymbol{w}_v$, so $W_l(\alpha) * w = -(t||\boldsymbol{r})$. To nullify t in the first row we can encode check that committed value corresponds to the committed digits by encoding of $e_n(b)$ in the first n elements of $W_l(\alpha)_0$ row (that will be multiplied on the $\boldsymbol{w}_L = \boldsymbol{d}$ in the $W_l(\alpha) * \boldsymbol{w}$). For the other r_i values in the $\boldsymbol{w}_v$ we can perform check of the reciprocal sum $\sum_{i=0}^{i<n} r_i - \sum_{j=0}^{j<b} m_j(\alpha + j)^{-1} = 0$ in a way that all of the last n elements of $W_l(\alpha)\boldsymbol{w} + \boldsymbol{w}_v$ will contain this equation. To do that we can set 1 in the all cells except of main diagonal of corresponding sub-matrix (let's call it E_0) that will be multiplied on $\boldsymbol{w}_R$ and $-(\alpha + j)^{-1}$ on main diagonal of the $\boldsymbol{w}_O$ multiplication sub-matrix:

$$W_l(\alpha) = \begin{vmatrix} e_n(b) & \boldsymbol{0} & \boldsymbol{0} \\ O & E_0 & -\dfrac{E}{\alpha + j} \end{vmatrix} \tag{67}$$

Finally, we've defined the circuit and witness vectors so it becomes possible to run the arithmetic circuit protocol. Note, that reciprocal argument gives different proof size for different bases. For the most popular case $uint64$ base $b = 16$ is the most optimal to generate proofs.

Note, that in the described implementation multiplicities vector $\boldsymbol{m}$ committed directly in the $\boldsymbol{w}_O$ witness vector that refers to the "shared" reciprocal argument from the original paper [Eag+23]. In that case all $\boldsymbol{w}_O$ elements will be mapped into l_L by the Ψ partition function, so l_L vector should be able to accommodate all this elements. It imposes additional restriction $N_v >= N_O$ which means that $n + 1 >= b$.

Algorithm 2 Reciprocal range proof protocol

```
Input: Same as in Arithmetic Circuit protocol and $t, n, b, s_t, s_r$
Require: $t \in [0, b^n)$
    • $\mathbb{P}$ calculates: $\boldsymbol{d}, \boldsymbol{m}, C_V$
    • $\mathbb{P} \to \mathbb{V} : C_V$
    • $\mathbb{V} \to \mathbb{P} : \alpha$
    • $\mathbb{P}$ calculates: $\boldsymbol{r}, C_R$
    • Both calculates: $W_m(\alpha), W_l(\alpha)$
    • Run Arithmetic Circuit protocol with witness $\boldsymbol{w}_L = \boldsymbol{d}, \boldsymbol{w}_R = \boldsymbol{r}, \boldsymbol{w}_O = \boldsymbol{m}, \boldsymbol{v} = (t||\boldsymbol{r}), s_v = s_t + s_r$ and circuit $W_l = W_l(\alpha), W_m = W_m(\alpha), f_m = 0, f_l = 1, \boldsymbol{a}_m = \boldsymbol{1}, \boldsymbol{a}_l = \boldsymbol{0}, N_m = n, N_O = b, N_v = 1 + n, k = 1$
```

6 Implementation

To demonstrate the performance of the described arithmetic circuit protocol we've implemented it using Go language over BN256 Curve and using Rust language over Bitcoin secp256k1 Curve. Implementation contains the weight norm linear argument protocol, arithmetic circuit protocol and reciprocal range-proof protocol that were described in this paper. Repositories can be found here:

- Go: https://github.com/distributed-lab/bulletproofs
- Rust: https://github.com/distributed-lab/bp-pp

Besides the described protocols it also contains several test cases on simple circuits: $x + y = r, x * y = z$ and binary range proof. Check the Appendix for more information about the test circuits.

In conclusion, described and implemented approach has $2\mathbb{G}$ points advantage over existing BP[Bun+17] and BP+[Chu+20] protocols in proving of one 64-bit value and this advantage will increase for more values per proof.

Protocol	Elements in $\mathbb{G}$	Elements in $\mathbb{F}$
BP	16	5
BP+	15	3
BP++ (Ours)	13	3

Table 1: Range proof for 64-bit value

References

[Bun+17] Benedikt Bunz et al. *Bulletproofs: Short Proofs for Confidential Transactions and More.* 2017.

[Chu+20] Heewon Chung et al. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger.* 2020. URL: https://eprint.iacr.org/2020/735.pdf.

[Eag+23] Liam Eagen et al. *Bulletproofs++: Next Generation Confidential Transactions via Reciprocal Set Membership Arguments.* 2023. URL: https://eprint.iacr.org/2022/510.pdf.

A Sample circuit

Now, let's examine how to prove simple set of equations in the BP++ arithmetic circuit protocol. For the public constants r, z we want to prove that we know such x, y:

$$x + y = r \tag{68}$$

$$x * y = z \tag{69}$$

It's obvious that we will have $\boldsymbol{w}_L = [x], \boldsymbol{w}_R = [y], \boldsymbol{w}_v = [x, y]$ and $\boldsymbol{w}_O = [x * y, x + y]$. Then $\boldsymbol{w} = [x, y, x * y, x + y]$.

Firstly, lets encode a multiplication constraint into the $W_m\boldsymbol{w} + f_m\boldsymbol{w}_v + \boldsymbol{a}_m = \boldsymbol{w}_L \circ \boldsymbol{w}_R$ circuit relation. The W_m matrix in our case will have dimension $N_m \times N_w = 1 \times 4$ and be perceived as vector where: 0 position corresponds to the w_0 value (x), 1 position to the w_1 value (y), etc. So, to encode our $x * y$ multiplication we can just set $f_m = 0, \boldsymbol{a}_m = [0]$ and $W_m = [0, 0, 1, 0]$. Now, we have:

$$W_m\boldsymbol{w} + f_m\boldsymbol{w}_v + \boldsymbol{a}_m = \boldsymbol{w}_L \circ \boldsymbol{w}_R \tag{70}$$

$$\begin{vmatrix} 0 & 0 & 1 & 0 \end{vmatrix} * \begin{vmatrix} x \\ y \\ x * y \\ x + y \end{vmatrix} + 0 * \boldsymbol{w}_v + \boldsymbol{0} = \begin{vmatrix} x \end{vmatrix} \circ \begin{vmatrix} y \end{vmatrix} \tag{71}$$

Next, we have to encode an addition constraint into the $W_l\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l = \boldsymbol{0}$ circuit equation. Here we will check the following conditions: w_O contains r and z public values (z value is partially checked in previous equation) and $x + y$ equals to public r. In our equation we already have the $f_l\boldsymbol{w}_v = [x, y]$ vector. Let's consider to separate our checks into the corresponding vector positions. In the first position we will check that $x + y - r = 0$ and then, in the second that $x * y - y + y + z = 0$. So, we will have:

$$W_l = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \end{vmatrix} \tag{72}$$

That corresponds to taking y in the first position of resulting $W_l\boldsymbol{w}$ vector and $x * y - y$ in the second position. Then, if we take $\boldsymbol{a}_l = [-r, -z]$, we will have:

$$W_l\boldsymbol{w} + f_l\boldsymbol{w}_v + \boldsymbol{a}_l = \boldsymbol{0} \tag{73}$$

$$\begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \end{vmatrix} * \begin{vmatrix} x \\ y \\ x * y \\ x + y \end{vmatrix} + \begin{vmatrix} x \\ y \end{vmatrix} + \begin{vmatrix} -r \\ -z \end{vmatrix} = \boldsymbol{0} \tag{74}$$

$$\begin{vmatrix} y + x - r \\ x * y - y + y - z \end{vmatrix} = \boldsymbol{0} \tag{75}$$

B Binary range proof

The binary range proof protocol as simple as previous proof of addition and multiplication. Imagine we have the $x = \langle(2^0, 2^1, 2^2, 2^3), (x_0, x_1, x_2, x_3)\rangle$ in 4-bits setting and the goal is to prove that value committed in x vector of bits lies in $[0, 2^4)$ range. To achieve that we will prove that every committed value in vector is a bit (equals to 0 or 1) by checking that $x_i * (x_i - 1) = 0$.

Let's split this equation into two more simple equations that we will encode into circuit:

$$x_i * x_i = a_i \tag{76}$$

$$a_i - x_i = 0 \tag{77}$$

We will encode the multiplication check ($x_i * x_i = a_i$) into the Hadamard product circuit equation.

$$\boldsymbol{w}_L = (x_0, x_1, x_2, x_3) \tag{78}$$

$$\boldsymbol{w}_R = (x_0, x_1, x_2, x_3) \tag{79}$$

$$\boldsymbol{w}_O = (a_0, a_1, a_2, a_3) \tag{80}$$

$$\boldsymbol{w} = \boldsymbol{w}_L || \boldsymbol{w}_R || \boldsymbol{w}_O = (x_0, x_1, x_2, x_3, x_0, x_1, x_2, x_3, a_0, a_1, a_2, a_3) \tag{81}$$

$$v = \begin{vmatrix} x_0 & a_0 \\ x_1 & a_1 \\ x_2 & a_2 \\ x_3 & a_3 \end{vmatrix} \tag{82}$$

$$\boldsymbol{w}_v = (x_0, a_0, x_1, a_1, x_2, a_2, x_3, a_3) \tag{83}$$

For the W_m matrix we should note that it contains three parts related to the $x_{0..3}, x_{4..7}, a_{8..11}$ parts of $\boldsymbol{w}$ vector. Then, to satisfy the circuit relations we should select the following W_m form:

$$W_{m,x} = [\mathbf{0}] \tag{84}$$

$$W_{m,a} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{85}$$

$$W_m = \begin{vmatrix} W_{m,x} & W_{m,x} & W_{m,a} \end{vmatrix} \tag{86}$$

You can easily check that this definition satisfies our circuit relation (with $f_m = 0, \boldsymbol{a}_l = \mathbf{0}$): $W_m \boldsymbol{w} + f_m \boldsymbol{w}_v + \boldsymbol{a}_m = \boldsymbol{w}_L \circ \boldsymbol{w}_R$.

For the W_l matrix we should select values to check the $x_i - a_i = 0$. With $f_l = 1$ we will also have the x_i term during addition that we have to nullify. Note, that W_l as W_m matrix also contains three parts related to the x, x, a parts of $\boldsymbol{w}$ vector.

$$W_{l,x_1} = \begin{vmatrix} -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \end{vmatrix} \tag{87}$$

$$W_{l,x_2} = [\mathbf{0}] \tag{88}$$

$$W_{l,a} = [\mathbf{0}] \tag{89}$$

$$W_l = \begin{vmatrix} W_{l,x_1} & W_{l,x_2} & W_{l,a} \end{vmatrix} \tag{90}$$

This definition allows us to nullify the x_i term and also provide the check that $-x_i + a_i = 0$ and also, satisfy the $W_l \boldsymbol{w} + f_l \boldsymbol{w}_v + \boldsymbol{a}_l = \mathbf{0}$ circuit equation.