A concrete instantiation of Bulletproof zero-knowledge proof

Andrey Jivsov
crypto@brainhub.org

Abstract. This work provides an instantiation of the Bulletproof zero-knowledge algorithm in modulo prime number fields. The primary motivation for this work is to help readers understand the steps of the Bulletproof protocol.

1 Introduction

This work provides specific steps suitable for an implementation of the work by Bünz et al. [1]. We work around the following difficulties:

- Lack of concise protocol steps. Multiple alternative steps are provided in [1].
- It is difficult to follow the entire algorithm due to its complexity. A cookbook-like steps are desired.
- Some quantities are left unspecified, e.g. they require solving equations.
- Only an interactive version is defined.
- Arithmetic in the composite order of a group $\mathbb{G}$ is undefined, yet the algorithm is defined via exponentiation modulo prime number, a group of composite order.
- Random quantities should be derived via KDF for the benefit of low-entropy environments and easier testing.

This work condenses 45 pages of [1] into an algorithm that should be easier to understand to an implementer and easier to maintain in the future.

2 Notations

We follow notations in [1] with following additional notations.

$||$ denotes concatenation. $a \leftarrow a \cdot b$ means that after this line the value of a equals to the previous value of a times b. This is a local operation limited to the relevant function, e.g. we don't change the global a.

$\mathbb{G}^+$ is used to denote "positive" half of elements in $\mathbb{G}$, as defined in sec. 3. We use $\mathbb{G}^+$ for comparison or for public elements in $\mathbb{G}$.

3 Group operations in $\mathbb{G}$ modulo safe prime

In this section we clarify details for the operations in $\mathbb{G}$.

We instantiate $\mathbb{G}$ as operations modulo safe prime q. The p used with [1] is $p = (q-1)/2$, and is a prime as well.

Many intermediate steps in Bulletproof algorithm are exponentiations of elements in $\mathbb{G}$. For example, for a $g \in \mathbb{G}$, we might need to calculate $g^{a \cdot b}$. How is the operation $a \cdot b$ performed in this example, given that the group order of $\mathbb{G}$ is $2p$, a composite number? In general, some operations, such as a multiplicative inverse, are undefined in the group mod $2p$. Some software libraries, such `bn.js` [2], and methods, such as Motgomery multiplication [3], are unsuitable for an even modulo arithmetic.

We adopts the following approach, similiar to [4].

All operations on the exponenet are performed modulo p. This reduction of an exponenet, v.s. $2p$, affects the resulting elelement in G in such a way that it loses the sign of the element in $\mathbb{G}$, in other words, we lose track of whether the result should have been x or $-x = q - x \in \mathbb{G}$.

To see why, consider that $\forall x, y : x > y \pmod{2p}$ we must have $x = p + y$ as the only choice. Observe that $g^p = \{1, -1\} \pmod{q} \in \mathbb{G}$, which explains the above reference to the sign.

We next define the subgroup $\mathbb{G}^+$ of $\mathbb{G}$ that we will use shortly:

$$\mathbb{G}^+ = \{\forall x \in \mathbb{G} : x \leq p\} \tag{1a}$$

We next define the mapping $\mathbb{G} \mapsto \mathbb{G}^+$ via $\mathsf{canonical}(\cdot)$ operation.

A *canonical* representation of any $x \in \mathbb{G}$, via a mapping $\mathbb{G} \mapsto \mathbb{G}^+$, is defined as follows:

$$\forall x \in \mathbb{G}$$
$$\mathsf{canonical(x)} = \begin{cases} x & \text{if } x \leq (q-1)/2 = p \\ q - x & \text{otherwise} \end{cases} \tag{2a}$$

The $\mathsf{canonical}()$ operation returns the smallest element of two, which can be naturally encoded in a fewest number of bits. The following properties of $\mathsf{canonical}()$ follow from the above definitions. For any $x, a, b, c \in \mathbb{G}$:

$$\mathsf{canonical(x)} \in \mathbb{G}^+ \tag{3a}$$
$$\mathsf{canonical(x)} = \mathsf{canonical(-x)} \tag{3b}$$
$$\mathsf{canonical(x)} \leq x \leq p < q \tag{3c}$$
$$\forall x < p : \mathsf{canonical(x)} = x \tag{3d}$$
$$\mathsf{canonical}(\mathsf{canonical}(a) \cdot \mathsf{canonical}(b)) = \mathsf{canonical}(a \cdot b) \tag{3e}$$
$$\mathsf{canonical}(\mathsf{canonical}(a) \cdot \mathsf{canonical}(b) \cdot \mathsf{canonical}(c)) = \mathsf{canonical}(a \cdot b \cdot c) = \\ = \mathsf{canonical}(\mathsf{canonical}(a \cdot b) \cdot c)) \tag{3f}$$

4 The algorithm

In the following algorithm that is an adaptation of [1] the Prover convinces the Verifier that it knows a public commitment V to a secret value v, and provides a proof that $0 \leq v < 2^n$. n must be a power of 2.

We are achieving two main properties:

- **Homomorthic property**. For two pairs of (commitment, secret value), (V_a, v_a) and (V_b, v_b), we can generate the commitment to the sum of secret values $v_a + v_b$ simply as $V_a \cdot V_b$.
- **Protection from negative secret values**. The main contribution of [1] and this work is to provide a publicly verifiable statement that a secret value v is non-negative and less than a specified maximum.

4.1 KDFs

We generate multiple pseudo-random values in this algorithm. There are two sets of these values: private and public.

i is the public identifier of the secret, an integer, as shown in the table 1. The size of the field that encodes i, j, and k is 1 byte.

H256 is a cryptographic hash function with 256-bit output, such as SHA2-256 or Keccak256.

The size of the field that encodes any element in $\mathbb{Z}_p$ is $\lceil log_2(p)/8 \rceil$ bytes. The element is stored in the big-endian format.

We use the following helper function to build KDFs.

$\mathsf{KDFInternal}(s, i, j):$

$\quad \mathsf{if}(log_2(s) > 256) : s = \mathsf{H256}(s)$

$\quad K = s \oplus (i \cdot 2^8) \oplus j$

$\quad m = \lceil log_2(p)/256 \rceil$

$\quad \forall k \in [1, m] : r = \mathsf{H256}(K||1)||...\mathsf{H256}(K||k)...||\mathsf{H256}(K||m) \bmod p$

$\quad$ **return** r

The lowest bit of $\mathsf{H256}(K||m)$ is the lowest bit of the value before reduction mod p.

The private values are generated from the 256-bit seed $\mathsf{SeedPriv}$ with two KDF functions $\mathsf{KDFPriv1}$ and $\mathsf{KDFPrivN}$, as shown next. These functions return values in the range $r : 0 \leq r < p$.

$\mathsf{SeedPriv} \xleftarrow{\$} 1^{256}$ Prover ganerates this seed

$\mathsf{KDFPriv1}(i) = \mathsf{KDFInternal}(\mathsf{SeedPriv}, i, 0)$ return an integer $\in \mathbb{Z}_p$

$\mathsf{KDFPrivN}(i, n)$:
  $\forall j \in [1, n] : r_i = \mathsf{KDFInternal}(\mathsf{SeedPriv}, i, j)$
  return $\mathbf{r} = (r_1, r_2, ...r_n)$ — return a vector $\in \mathbb{Z}_p^n$

Public values are generated with functions KDFPub1 and KDFPubN as follows. These functions return values r in the range $0 < r < p$.

$\mathsf{KDFPub1}(s, i)$:
  $r = \mathsf{KDFInternal}(s, i, 0)$
  $r \leftarrow \lfloor r/2 \rfloor \cdot 2 + 1$ — eliminate a 0
  return r — $\in \mathbb{Z}_p^{*n}$

$\mathsf{KDFPubN}(s, i, n)$:
  $\forall j \in [1, n] : r_j = \mathsf{KDFInternal}(s, i, j)$
  $\forall j \in [1, n] : r_j \leftarrow \lfloor r_j/2 \rfloor \cdot 2 + 1$ — eliminate a 0
  return $\mathbf{r} = (r_1, r_2, ...r_n)$ — return a vector $\in \mathbb{Z}_p^{*n}$

Table 1. Identifier values for pseudo-random values.

Private ID	Value	Use
BULLETPROOF_ID_H	1	Generator h
BULLETPROOF_ID_U	2	Generator u
BULLETPROOF_ID_VG	3	Generator $\mathbf{g}$
BULLETPROOF_ID_VH	4	Generator $\mathbf{h}$
BULLETPROOF_ID_RCPT_GAMMA	5	Pedersen blinding value
BULLETPROOF_ID_RCPT_MASK	6	Secret mask to hide v
BULLETPROOF_ID_ALPHA	7	Blinding value α
BULLETPROOF_ID_SL	8	Exponent $\mathbf{s}_L$
BULLETPROOF_ID_SR	9	Exponent $\mathbf{s}_R$
BULLETPROOF_ID_RHO	10	Exponent ρ
BULLETPROOF_ID_Y	11	Base for vector $\mathbf{y}^n$
BULLETPROOF_ID_Z	12	z to construct $r(X)$
BULLETPROOF_ID_TAU1	13	Blinding for t_1
BULLETPROOF_ID_TAU2	14	Blinding for t_2
BULLETPROOF_ID_X	15	Sample value x for $l(X), r(X)$
BULLETPROOF_ID_INNER_ARG_XU	16	Exponent challenge for u in InnerProductArgumentProver
BULLETPROOF_ID_INNER_ARG_VX	17	Vector $\mathbf{x}$ used as challenges in InnerProductArgumentProver

4.2 Public parameters

We first define public parameters.

Parameters:

$$
\begin{aligned}
& p - \text{prime, such that } q = p \cdot 2 + 1 \text{ is also prime} \\
g, h, u; \mathbf{g}, \mathbf{h} \quad & 5 \text{ generators of unknown relationship to each other} \in \mathbb{G}; \mathbb{G}^n \qquad (7a) \\
& g = 3 \\
& h = \mathsf{KDFPub1}(q, \texttt{BULLETPROOF\_ID\_H}) \qquad \in \mathbb{G} \\
& u = \mathsf{KDFPub1}(q, \texttt{BULLETPROOF\_ID\_U}) \qquad \in \mathbb{G} \\
& \mathbf{g} = \mathsf{KDFPubN}(q, \texttt{BULLETPROOF\_ID\_VG}) \qquad \in \mathbb{G}^n \\
& \mathbf{h} = \mathsf{KDFPubN}(q, \texttt{BULLETPROOF\_ID\_VH}) \qquad \in \mathbb{G}^n
\end{aligned}
$$

p is large subgroup size. q is the prime used for modulo reduction of elements in $\mathbb{G}$. By construction, 5 generators above or their scalars, as appropriate, are less than p.

4.3 Prover steps

v is low-entropy private value. Prover performs the following steps to produce V, a hiding commitment to it, and a proof that $0 \leq v < 2^n$.

$$
\begin{aligned}
\gamma &= \mathsf{KDFPriv1}(\texttt{BULLETPROOF\_ID\_GAMMA}) && \text{a secret} \in \mathbb{Z}_p && (8a) \\
V &= h^\gamma g^v && \text{comm. to } v, \in \mathbb{G} && (8b) \\
M &= \mathsf{H256}(p||g||h||\mathbf{g}||\mathbf{h}||V) && \text{comm. to pub. params and } V && (8c) \\
\mathbf{a}_L &:\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v && \text{Compose } \mathbf{a}_L, \in \mathbb{Z}_p^n && \\
\mathbf{a}_R &= \mathbf{a}_L - \mathbf{1}^n && \in \mathbb{Z}_p^n && \\
\alpha &= \mathsf{KDFPriv1}(\texttt{BULLETPROOF\_ID\_ALPHA}) && \in \mathbb{Z}_p && \\
A &= h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} && \text{comm. to } \mathbf{a}_L \text{ and } \mathbf{a}_R, \in \mathbb{G} && (8d) \\
\mathbf{s}_L &= \mathsf{KDFPrivN}(\texttt{BULLETPROOF\_ID\_SL}, n) && \in \mathbb{Z}_p^n && \\
\mathbf{s}_R &= \mathsf{KDFPrivN}(\texttt{BULLETPROOF\_ID\_SR}, n) && \in \mathbb{Z}_p^n && \\
\rho &= \mathsf{KDFPriv1}(\texttt{BULLETPROOF\_ID\_RHO}) && \in \mathbb{Z}_p && \\
S &= h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} && \text{comm. to } \mathbf{s}_L, \mathbf{s}_R, \in \mathbb{G} && (8e) \\
t &\leftarrow \mathsf{H256}(M||A||S) && \text{transcript} && \\
y &= \mathsf{KDFPub1}(t, \texttt{BULLETPROOF\_ID\_Y}) && \in \mathbb{Z}_p^* && (8f) \\
z &= \mathsf{KDFPub1}(t, \texttt{BULLETPROOF\_ID\_Z}) && \in \mathbb{Z}_p^* && (8g) \\
l(X) &= (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X && \in \mathbb{Z}_p^n[X] && (8h) \\
r(X) &= \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n && \in \mathbb{Z}_p^n[X] && (8i) \\
t(X) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 && \in \mathbb{Z}_p[X] && \\
\mathbf{l}_0 &= \mathbf{a}_L - z \cdot \mathbf{1}^n && \text{free term, see (8h)}, \in \mathbb{Z}_p^n && \\
\mathbf{l}_1 &= \mathbf{s}_L && \text{term at } X\text{, see (8h)}, \in \mathbb{Z}_p^n &&
\end{aligned}
$$

$$\mathbf{r}_0 = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \qquad \text{free term, see (8i)}, \in \mathbb{Z}_p^n$$
$$\mathbf{r}_1 = \mathbf{y}^n \circ \mathbf{s}_R \qquad \text{term at } X\text{, see (8i)}, \in \mathbb{Z}_p^n$$
$$t_0 = \langle \mathbf{l}_0, \mathbf{r}_0 \rangle \qquad \in \mathbb{Z}_p$$
$$t_1 = \langle \mathbf{l}_1, \mathbf{r}_0 \rangle + \langle \mathbf{l}_0, \mathbf{r}_1 \rangle \qquad \in \mathbb{Z}_p$$
$$t_2 = \langle \mathbf{l}_1, \mathbf{r}_1 \rangle \qquad \in \mathbb{Z}_p$$
$$\tau_1 = \mathsf{KDFPriv1}(\texttt{BULLETPROOF\_ID\_TAU1}) \qquad \in \mathbb{Z}_p$$
$$\tau_2 = \mathsf{KDFPriv1}(\texttt{BULLETPROOF\_ID\_TAU2}) \qquad \in \mathbb{Z}_p$$
$$T_1 = g^{t_1} h^{\tau_1} \qquad \text{Pedersen comm. to } t_1, \in \mathbb{G} \tag{8j}$$
$$T_2 = g^{t_2} h^{\tau_2} \qquad \text{Pedersen comm. to } t_2, \in \mathbb{G} \tag{8k}$$
$$t \leftarrow \mathsf{H256}(M||A||S||T_1||T_2) \qquad \text{transcript}$$
$$x = \mathsf{KDFPub1}(t, \texttt{BULLETPROOF\_ID\_X}) \qquad \in \mathbb{Z}_p^* \tag{8l}$$
$$\mathbf{l} = l(X = x) = \mathbf{l}_0 + \mathbf{s}_L \cdot x \qquad \text{Evaluate (8h) at } x, \in \mathbb{Z}_p^n$$
$$\mathbf{r} = r(X = x) = \mathbf{r}_0 + \mathbf{r}_1 \cdot x \qquad \text{Evaluate (8i) at } x, \in \mathbb{Z}_p^n$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \qquad \in \mathbb{Z}_p \tag{8m}$$
$$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \gamma \qquad \text{blinding for } \hat{t}\text{; see (8a)}, \in \mathbb{Z}_p$$
$$\mu = \alpha + \rho \cdot x \qquad \alpha, \rho \text{ blind } A, S;\ \text{(8d), (8e)}, \in \mathbb{Z}_p \tag{8n}$$
$$h' = h^{y^{-i+1}}, \forall i \in [1, n], \qquad \in \mathbb{G}$$
$$\mathbf{h}' = (h_1, h_2^{y^{-1}}, h_3^{y^{-2}}, ..., h_n^{y^{-n+1}}) = \mathbf{h}^{(\mathbf{y}^{-n})} \qquad \in \mathbb{G}^n$$
$$P' = \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}} \qquad \in \mathbb{G}$$
$$\mathsf{Seed} = t \leftarrow \mathsf{H256}(M||A||S||T_1||T_2||\hat{t}||\tau_x||\mu) \qquad \text{compete transcript and Seed} \tag{8o}$$

$$a, b, L_1, ..., L_{log_2(n)}, R_1, ..., R_{log_2(n)} = \qquad a, b \in \mathbb{Z}_p, \text{rest} \in \mathbb{G}$$
$$\mathsf{InnerProductArgumentProver}(\mathbf{g}, \mathbf{h}', u, P', \hat{t}, \mathbf{l}, \mathbf{r}, \mathsf{Seed}) \tag{9a}$$

Finally, Prover sends the following quantities to the Verifier:

$$V \text{ see (8b)}, \in \mathbb{G}$$
$$A, S, \text{ see (8d), (8e)}, \in \mathbb{Z}_p$$
$$T_1, T_2, \text{ see (8j), (8k)}, \in \mathbb{G}$$
$$\hat{t}, \tau_x, \mu \text{ see (8m) - (8n) }, \in \mathbb{Z}_p$$
$$a, b, L_1, ..., L_{log_2(n)}, R_1, ..., R_{log_2(n)} \text{ see (9a)}$$

4.4 Verifier steps

Verifier starts with the input received from the Prover, as specified at the end of the sec. 4.3, copied immediately below.

$$
\begin{aligned}
&V \\
&A, S, \\
&T_1, T_2, \\
&\hat{t}, \tau_x, \mu \\
&a, b, L_1, ..., L_{log_2(n)}, R_1, ..., R_{log_2(n)}
\end{aligned}
$$

Verifier calculates the following pseudo-random values from the above public values:

$$
\begin{aligned}
x, &\text{ as (8l)} \\
y, z, &\text{ as (8f), (8g)} \\
\mathsf{Seed}, &\text{ as (8o)} \\
x_u, x_1, ..., x_{log_2(n)} &\text{ as (15a), (15b)}
\end{aligned}
$$

Verifier performs the following steps:

$$\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle \qquad = (z - z^2) \sum_{i=0}^{n-1} y^i - z^3 (2^n - 1)$$

$$g^{-\hat{t}+\delta(y,z)} h^{-\tau_x} V^{z^2} T_1^x T_2^{x^2} \stackrel{?}{=} 1 \qquad \text{equiv. to (14a)} \tag{13a}$$

$$b(i, j) = \begin{cases} 1 & \text{if the } (log_2(n) - j)\text{th bit of } i - 1 \text{ is } 1 \\ -1 & \text{otherwise} \end{cases}$$

$\forall i \in [1, n]$ **do** :

$$s_i = \prod_{j=1}^{log_2(n)} x_j^{b(i,j)} \qquad \in \mathbb{Z}_p$$

$$l_i = s_i \cdot a + z \qquad \in \mathbb{Z}_p$$

$$r_i = y^{1-i} (s_i^{-1} \cdot b - z^2 \cdot 2^{i-1}) - z \qquad \in \mathbb{Z}_p$$

done

$$\mathbf{l} = (l_1, ..., l_n) \qquad \in \mathbb{Z}_p^n$$

$$\mathbf{r} = (r_1, ..., r_n) \qquad \in \mathbb{Z}_p^n$$

$$\mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} u^{x_u \cdot (ab - \hat{t})} h^{\mu} A^{-1} S^{-x} \left(\prod_{j=1}^{log_2(n)} L_j^{x_j^2} R_j^{x_j^{-2}} \right)^{-1} \stackrel{?}{=} 1 \tag{13b}$$

For higher performance (13a) and (13b) should be be combined and then the calculation performed via multi-exponentiation.

4.5 Verifier steps for a given {l, r}. Debug only.

This section exists for implementation testing. It offers an easier method to check that $0 \le v < 2^n$ based on $\mathbf{l}$, $\mathbf{r}$ directly, without $\mathsf{InnerProductArgumentProver}$.

$$
\begin{aligned}
\delta(y,z) &= (z - z^2)\cdot\langle \mathbf{1}^n, \mathbf{y}^n\rangle - z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle && = (z-z^2)\sum_{i=0}^{n-1} y^i - z^3(2^n - 1), \in \mathbb{Z}_p \\
g^{\hat{t}} h^{\tau_x} &\stackrel{?}{=} V^{x^2}\cdot g^{\delta(y,z)}\cdot T_1^x \cdot T_2^{x^2} && \text{check that } \hat{t} = t(x) = t_0 + t_1 x + t_2 x^2 \quad (14a)\\
P &= A\cdot S^x \cdot \mathbf{g}^{-z}\cdot(\mathbf{h}')^{z\cdot\mathbf{y}^n + z^2\cdot\mathbf{2}^n} && \text{compute a commitment to } l(x), r(x), \in \mathbb{G}\\
P &\stackrel{?}{=} h^\mu \cdot \mathbf{g}^{\mathbf{l}}\cdot(\mathbf{h}')^{\mathbf{r}} && \text{check that } l(x), r(x) \text{ are correct}\\
\hat{t} &\stackrel{?}{=} \langle \mathbf{l}, \mathbf{r}\rangle && \text{check that } \hat{t} \text{ is correct}, \in \mathbb{Z}_p
\end{aligned}
$$

5 Inner-Product Argument for the Prover

This section defines a subroutine used in the main algorithm in sec. 4.

The following $\mathsf{InnerProductArgumentProver}$ is an adaptation of Protocol 1 and Protocol 2 from [1], limited to the prover. We removed recursion, merged two protocols, removed steps not used by the prover, made the protocol non-inteactive, and introduced additional quantities to improve readability, such as (15d) - (15e).

$n \ge 2$, which is also a power of 2, is required. The parameters have following membership: $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p, \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$.

$$
\begin{aligned}
&\mathsf{InnerProductArgumentProver}(\mathbf{g}, \mathbf{h}, u, P, c, \mathbf{a}, \mathbf{b}, \mathsf{Seed}):\\
&\quad x_u = \mathsf{KDFPub1}(\mathsf{Seed}, \mathtt{BULLETPROOF\_ID\_INNER\_ARG\_XU}) && \in \mathbb{Z}_p^* \quad (15a)\\
&\quad P \leftarrow P\cdot u^{x_u\cdot c} && \text{reassign}\\
&\quad u \leftarrow u^{x_u} && \text{reassign}\\
&\quad \mathbf{x} = \mathsf{KDFPubN}(\mathsf{Seed}, \mathtt{BULLETPROOF\_ID\_INNER\_ARG\_VX}) && \in \mathbb{Z}_p^{*n} \quad (15b)\\
\\
&\quad \forall i \in [1, log_2(n)] \textbf{ do} && :\\
&\qquad n' = n/2^i && \{n/2, n/4, ...1\} \quad (15c)\\
&\qquad (\mathbf{a}_L, \mathbf{a}_R) = \mathbf{a} = (\mathbf{a}_{[:\mathbf{n}']}, \mathbf{a}_{[\mathbf{n}':]}) && \text{split in half} \quad (15d)\\
&\qquad (\mathbf{b}_L, \mathbf{b}_R) = \mathbf{b} = (\mathbf{b}_{[:\mathbf{n}']}, \mathbf{b}_{[\mathbf{n}':]}) && \text{split in half}\\
&\qquad (\mathbf{g}_L, \mathbf{g}_R) = \mathbf{g} = (\mathbf{g}_{[:\mathbf{n}']}, \mathbf{g}_{[\mathbf{n}':]}) && \text{split in half}\\
&\qquad (\mathbf{h}_L, \mathbf{h}_R) = \mathbf{h} = (\mathbf{h}_{[:\mathbf{n}']}, \mathbf{h}_{[\mathbf{n}':]}) && \text{split in half} \quad (15e)\\
&\qquad c_L = \langle \mathbf{a}_L, \mathbf{b}_R\rangle && \in \mathbb{Z}_p\\
&\qquad c_R = \langle \mathbf{a}_R, \mathbf{b}_L\rangle && \in \mathbb{Z}_p
\end{aligned}
$$

$L_i = \mathbf{g}_R^{\mathbf{a}_L} \mathbf{h}_L^{\mathbf{b}_R} u^{c_L}$ $\in \mathbb{G}$

$R_i = \mathbf{g}_L^{\mathbf{a}_R} \mathbf{h}_R^{\mathbf{b}_L} u^{c_R}$ $\in \mathbb{G}$

$\mathbf{g} \leftarrow \mathbf{g}_L^{x_i^{-1}} \circ \mathbf{g}_R^{x_i}$ reassign; size is halved

$\mathbf{h} \leftarrow \mathbf{h}_L^{x_i} \circ \mathbf{h}_R^{x_i^{-1}}$ reassign; size is halved

$\mathbf{a} \leftarrow \mathbf{a}_L \cdot x_i + \mathbf{a}_R \cdot x_i^{-1} \in \mathbb{Z}_p$ reassign; size is halved

$\mathbf{b} \leftarrow \mathbf{b}_L \cdot x_i^{-1} + \mathbf{b}_R \cdot x_i \in \mathbb{Z}_p$ reassign; size is halved

done

Return :

a, b single element in $\mathbf{a}, \mathbf{b}, \in \mathbb{Z}_p$

$L_1, ..., L_{log_2(n)}$ $\in \mathbb{G}$

$R_1, ..., R_{log_2(n)}$ $\in \mathbb{G}$

Internal consistency check: $g^a h^b u^{ab} = P \prod_{j=1}^{log_2(n)} L_j^{x_j^2} R_j^{x_j^{-2}}$, immediately before the **Return** statement.

6 Remaining work

– Describe the algorithm in the elliptic curve group with prime group order (beneficial for storage efficiency and simpler).
– Expand to aggregation of proofs and verifies (sec 4.3 and 6.2 of [1]).
– Add multi-exponentiation (sec. 3.1 of [1]).
– Add multi-exponentiation to aggregated proofs (sec 6.2 of [1]).

References

1. Bunz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short Proofs for Confidential Transactions and More. Cryptology ePrint Archive, Report 2017/1066 (2017) `https://eprint.iacr.org/2017/1066`.
2. Indutny, F.: BigNum in pure javascript. GitHub source code (2019) `https://github.com/indutny/bn.js/`.
3. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. 44, 519-521 (1985) `https://doi.org/10.1090/S0025-5718-1985-0777282-X`.
4. Jivsov, A.: Compact representation of an elliptic curve point. Internet draft (2014) `https://tools.ietf.org/id/draft-jivsov-ecc-compact-05.html`.