Zarcanum: A Proof-of-Stake Scheme for Confidential Transactions with Hidden Amounts

sowle[1], koe[2]

[1]Zano project, val@zano.org
[2]Independent researcher, ukoe@protonmail.com

March 2022*

Abstract

This article explores a Proof-of-Stake mining algorithm in an environment where amounts are hidden with homomorphic commitments, in particular, using confidential transactions. Our goal was to avoid revealing amounts and other sensitive information (like which output was used to stake a given block) to blockchain observers when doing staking.

Our contribution is a Proof-of-Stake mining scheme that does not reveal amounts and is compatible with ring confidential transactions. We also present an extension to the Bulletproofs+ protocol that allows range proofs on double-blinded commitments, with corresponding security statements.

1 Notation

Let $\mathbb{G}$ denote the main subgroup of the Ed25519 curve ([1]) and $\mathbb{Z}_p$ denote a ring of integers modulo p.

l is the order of $\mathbb{G}$: $l = \#\mathbb{G} = 2^{252} + 27742317777372353535851937790883648493$.

For any set X, $x \xleftarrow{\$} X$ means uniform sampling of x at random from X.

For any integers x, y, $\left\lfloor \frac{x}{y} \right\rfloor$ denotes the integer part of *integer arithmetic* division.

2 Classic PoS scheme (open amounts)

In this section we describe how PoS mining was originally implemented in Zano[1].

Suppose Alice has some unspent outputs and wants to mine a PoS block using one of them as a stake. In such a scenario she acts as follows (Fig. 2.1):

*Version 4.8. Last update: 2022-03-15. Check here for the latest version.
[1]This scheme is based on ideas from the PeerCoin project [5].

1. Gets the hash identifier of the last PoW block in the blockchain, *last_pow_id*.
2. Gets the last PoS block in the blockchain and gets the stake kernel hash identifier from it, *last_pos_kernel_id*. Together with *last_pow_id* they are called the "stake modifier". It changes each time a new block is added to the blockchain.
3. Makes set T of all possible timestamps for the new PoS block:

$$T = \{t : t_{min} \leq t \leq t_{max},\ t \equiv 0 \mod 15\}$$

where t_{min} and t_{max} are bound to the current blockchain conditions. For the sake of simplicity we can assume that

$$t_{min} = \tau - T,\ t_{max} = \tau + T$$

where T is a constant and τ is the current timestamp established in such a way that it is the same across all the network's nodes.

4. Makes set U of all her unspent transaction outputs (UTXO) that are eligible for staking (i.e. not locked, mature enough, etc.). For each output u from U she also precalculates the key image I_u.
5. Each pair $(t, u) \in T \times U$ is checked against the PoS winning condition as follows:
   (a) For output u build *stake kernel* K_u as a concatenation:

$$K_u = last\_pow\_id \parallel last\_pos\_kernel\_id \parallel t \parallel I_u$$

   where I_u is the key image of the stake output u;
   (b) Calculate hash $h_u = cn\_fast\_hash(K_u)$[2];
   (c) Finally, check the main condition:

$$\frac{h_u D}{a_u} \overset{?}{\leq} 2^{256} \tag{1}$$

   where D is the current PoS difficulty, and a_u is the amount of the stake output u.

   If inequality (1) holds then it means the success of PoS mining! A block with timestamp t and a stake input spending output u can be constructed and broadcast to the network.

If for all pairs (t, u) inequality (1) does not hold, Alice needs to wait until one of the following happens:

- a new block is added to the blockchain (this will change either *last_pow_id* or *last_pos_kernel_id*);
- some time passes (this will change t_{min} and t_{max}).

Once this happens, Alice can attempt mining again (items 1-5), as all K_u, and thus h_u, will have different values, giving new opportunities to meet the main condition.

We'd like to note the following important property of (1): as h_u is the result of a cryptographic hash function and can be considered as distributed evenly over the interval $[0, 2^{256})$, the probability of meeting the main condition is proportional to output amount a_u.

[2] *cn_fast_hash* is an alias for the Keccak-256 hash function, which is similar to SHA3-256 but differs in padding bits.

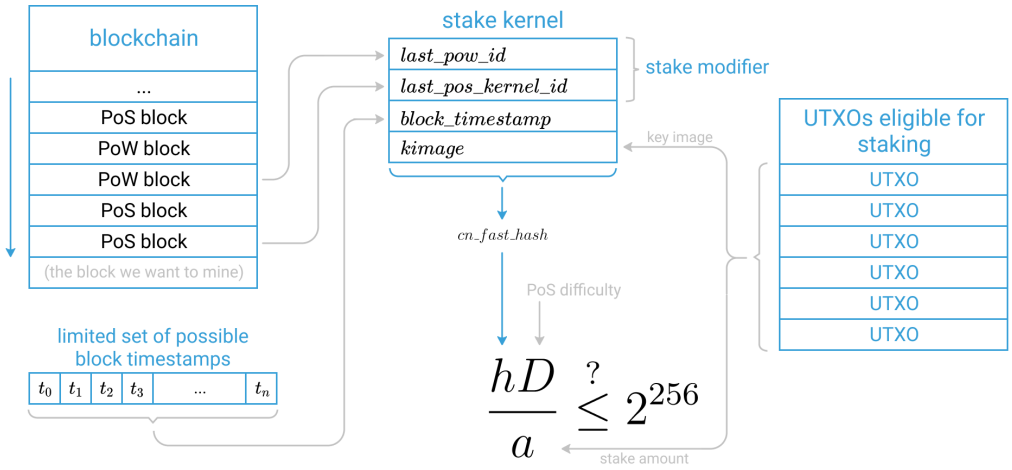

Fig. 2.1. Scheme of the PoS mining process, as originally implemented in Zano

3 Hidden amounts: the problem and the solution

Consider a hidden amount scheme[3], where amount a of an output is hidden using Pedersen[4] commitment A:

$$A = aH + fG \quad (a < 2^{64},\ f \neq 0)$$

where H and G are generators in $\mathbb{G}$ for which DL relation is unknown, and f is a random hiding mask.

It's easy to see that (1) can't be used by verifiers anymore because it requires a to be non-hidden. Let's see how the main inequality could be modified.

Suppose Alice has already prepared sets of timestamps (T) and outputs (U) eligible for staking as mentioned in Section 2. She then considers each pair $(t, u) \in T \times U$ against the PoS winning condition. She calculates (this time using the hash function H_s, which produces scalars in $\mathbb{Z}_l$)

$$h = H_s(last\_pow\_id \parallel last\_pos\_kernel\_id \parallel t \parallel I_u)$$

The use of a hash function means h can be considered as uniform randomness distributed evenly over $\mathbb{Z}_l$. Moreover, since the mask f is $\neq 0$ and fixed for the selected output u (i.e. defined before h can be computed), the multiplication $hf \pmod l$ can also be considered as uniform randomness over $\mathbb{Z}_l$.

Taking this into account, Alice checks the slightly adjusted PoS winning condition:

$$hf \bmod l \ < \ \left\lfloor \frac{l}{D} \right\rfloor a \tag{2}$$

where l is the order of the main subgroup. Here we moved from 2^{256} (used originally in (1)) to l, as all scalar operations in all the following equations hold modulo l except the division $\lfloor \frac{l}{D} \rfloor$.

[3] E.g., Confidential Transactions (CT) by Gregory Maxwell [6].
[4] More information in the original paper by T.P. Pedersen [8].

Note that as soon as $D \geq 2^{64}$ and $a < 2^{64}$, the right side of (2) never exceeds l.

Now we transform the inequality to equality:

$$hf \mod l = da - b_a , \quad 1 \leq d \leq \left\lfloor \frac{l}{D} \right\rfloor , \; b_a < 2^{64} \tag{3}$$

Once (2) holds, Alice needs to calculate d and b_a such that (3) holds. If she can convince verifiers that she knows d and b_a, and that those values are in the correct ranges, she can also convince them that the PoS winning condition (2) holds[5] for a particular h, and thus, for a pair (u, t) (assuming she also convinces them that I_u is the key image of an output u that exists in the ledger).

In the following sections we construct such a proof in a NIZK-manner.

4 Criteria

Let us make a list of criteria for an ideal PoS mining scheme with hidden amounts to help understand the differences between approaches.

1. *(stake proportionality)* The probability of meeting the winning condition is proportional to the stake amount.
2. *(resistantness)* A miner is unable to tamper with the protocol for their benefit.
3. *(amount privacy)* Amounts are kept private; hence observers are not able to calculate amounts from public data.
4. *(sender-recipient anonymity)* The original sender of an output cannot accurately guess when a miner stakes that output and creates a new PoS block.
5. *(untraceability)* It is unreasonably difficult for observers to determine which output was really used as a stake (e.g. when a ring of decoy outputs are used).

It is easy to see that the classic PoS scheme described above in Section 2 satisfies criteria 1, 2 and 5.[6]

5 Direct-spend PoS

In this section, for the sake of clarity, we show how to construct a proof for the winning condition in the simplest case, when there are no decoy outputs, i.e., the stake output is directly referenced in a mining transaction.

Suppose the stake output amount a is committed to in publicly known $A = aH + fG$.

[5] Except with negligible probability in the case when $da < b_a$ (see the discussion in Appendix C.2, setting $z = 1$).

[6] Criterion 5 could be met by using a ring signature to show the key image I_u corresponds to a real output u, and also that a pseudo-output commitment corresponds to that output's amount commitment. Then the prover can open the pseudo-commitment to reveal the amount a_u, without needing to expose the blinding factor of the original amount commitment (which would break Criterion 5).

Rewriting (3) slightly (all scalar equations hold modulo l):

$$(3) \quad \Leftrightarrow \quad hf - da + b_a = 0, \quad 1 \leq d \leq \left\lfloor \frac{l}{D} \right\rfloor, \; b_a < 2^{64} \tag{4}$$

Let $b_f = df - ha$. The following equality holds:

$$ha - df + b_f = 0 \tag{5}$$

Use (4) and (5) as scalar parts for scalar multiplication with H and G correspondingly:

$$(4), (5) \Rightarrow \begin{cases} hf - da + b_a = 0 & | \times H \\ ha - df + b_f = 0 & | \times G \end{cases} \tag{6}$$

Considering commitments $A' = fH + aG$, $A = aH + fG$, and $B = b_a H + b_f G$, we can rewrite (6) in terms of group element operations:

$$hA' - dA + B = \mathbf{0} \tag{7}$$

where $\mathbf{0}$ is the identity element of $\mathbb{G}$.

Now to satisfy range requirements in (3), Alice needs to reveal d and A', prove that A' is the mirror commitment of A, and provide a range proof for B (i.e. show that $b_a < 2^{64}$).

Proving the correctness of A' can be done with a mirror commitment proof as shown in Appendix A.1.

This approach satisfies stake proportionality, resistantness, and amount privacy from (4), but the criteria of sender-recipient anonymity and untraceability are not met.

6 Ring-friendly PoS hidden amount scheme

If Alice would like to improve her mining anonymity and use a ring of decoy outputs to hide her stake, she cannot use the approach in Section 5, because in a RingCT-like mining transaction with a non-empty decoy set [7], the stake input would refer to a *set* of outputs, and thus a *pseudo* output commitment would be used to represent the amount a in the input. Therefore, verifiers on the network would not be able to check (7) as they don't know which A from the set of outputs to use.

This problem can be solved if Alice provides another commitment to the same stake amount that verifiers can use in an equation similar to (7), but that can't be used to link to any amount commitment from the ledger.

6.1 Ring-friendly proof construction

Let $(V, S) = (vG, sG)$ be Alice's public address, where v and s are her view and spend secret keys correspondingly.

Suppose Alice already went through steps 1-4 in Section 2 and found a pair (t, u) for which the PoS winning condition (2) is met. Assume it was Bob who had previously sent output u to Alice.

Following the CryptoNote protocol, Bob calculated a one-time address P for output u:[7]

$$P = H_s(rV)G + S$$

where r is the transaction's secret key. Suppose that for each output Bob also computed group element Q and made it public in addition to P:

$$Q = H_s(rV)V = qG,\ q = vH_s(rV)$$

Note that only Alice and those who get secret view key v from her can calculate secret q, as $q = vH_s(vR)$, where R is the transaction's public key.

Recalling equation (3), suppose Alice has calculated d and b_a such that the PoS winning equality holds.

Also suppose that, following the standard procedure, she randomly selected a set of apparently unspent decoy outputs $\{u_i\}$ from the blockchain and put her output, which met the PoS main condition (2), at random index π of that set.

Let the i-th decoy output's commitment be denoted A_i (and A_π is the commitment to her own output). Note that in general Alice doesn't know amounts a_i and masks f_i for the outputs she selected as decoys.

Let X be a generator in $\mathbb{G}$ for which the DL relations with G and H are unknown.

Consider extended commitment C to the same stake amount a:

$$C = xX + aH + (f + q)G, \quad x \xleftarrow{\$} \mathbb{Z}_l \tag{8}$$

Here x is a secret randomness chosen by Alice.

Extended commitment C can be linked to the stake commitment A_π (without revealing index π) by adding two additional layers[8] to the main ring signature:[9]

1. a proof of knowing the DL x of $C - A_\pi - Q_\pi$ with respect to X;
2. a proof of knowing the DL q of Q_π with respect to G.

We can extend the ring signature by adding two group elements to the calculation of the non-interactive challenge as follows:

$$\begin{aligned}
c_{\pi+1} &= H_s(\ldots,\ \alpha_0 X,\ \alpha_1 G) \\
c_{i+1} &= H_s(\ldots,\ r_i^0 X + c_i(C - A_{i+1} - Q_{i+1}),\ r_i^1 G + c_i Q_{i+1}) \\
r_\pi^0 &= \alpha_0 - c_\pi x \\
r_\pi^1 &= \alpha_1 - c_\pi q
\end{aligned}$$

Note that using randomness x in (8) implies that an external observer would not be able to easily link C with any of A_i, even if a and f are known (which is the case for the sender of A_π).

[7] In CryptoNote, a one-time address is calculated as $P_j = H_s(rV, j)G + S$, where j is the index of an output. Here we skip j in $H_s(rV, j)$ for the sake of clarity.

[8] Here we're using terminology and ideas from Multi-layered Linkable Spontaneous Anonymous Group signatures proposed in [7].

[9] If A_π has a range proof, as it required for confidential transaction amount commitments stored in a ledger, then, taking into account these ring signature layers, observers can be confident that C is composed of the generators X, H, G, and that the amount a in C equals the amount a in A_π.

Consider the mirror extended commitment C':

$$C' = x'X + (f+q)H + aG, \quad x' \xleftarrow{\$} \mathbb{Z}_l,\ x' \neq x \tag{9}$$

Let us introduce randomness $x'' \xleftarrow{\$} \mathbb{Z}_l,\ x'' \neq 0$ which is freely chosen by Alice.

Let $b_x = x'' - hx' + dx$. Then the following equation holds:

$$hx' - dx + b_x = x'' \tag{10}$$

Use (4), (5), and (10) as scalar parts for scalar multiplication with H, G, and X correspondingly. Note that we now use $h(f+q)$ instead of hf for (4) and (5).[10]

$$(4), (5), (10) \Rightarrow \begin{cases} h(f+q) - da + b_a = 0 & | \times H \\ ha - d(f+q) + b_f = 0 & | \times G \\ hx' - dx + b_x = x'' & | \times X \end{cases} \tag{11}$$

Considering $E = b_aH + b_fG + b_xX$ and equations for C and C' above, we can rewrite (11) in terms of group element operations:

$$hC' - dC + E = x''X = F \tag{12}$$

To convince verifiers that (12) holds, Alice discloses C, C', and E, and provides a Schnorr proof for $F = x''X$. Disclosing C, C', and E is safe as each of them are guarded with its own randomness x, x', and x'' respectively. Alice also needs to prove that C' is in fact the mirror commitment of C, and provide a range proof for E to finish proving (12). Proving that (12) holds implies that (4) also holds, and thus the PoS winning condition holds as well.

Let us consider these steps in detail.

1. Assume the correctness of C is proven by modification of the ring signature (see above).
2. For proving $C' = x'X + (f+q)H + aG$ we use an extended mirror commitment proof, as described in Appendix A.2.
3. A range proof for $b_a < 2^{64}$ committed to in $E = b_aH + b_fG + b_xX$ can be done directly by extending an existing range proof protocol like Bulletproofs+[3] to support two blinding factors in commitments. We provide such an extension for Bulletproofs+ in Appendix D.

   However, this can also be achieved using a standard range proof if Alice provides a range proof for commitment $B = b_aH + eG$ on value b_a, where $e \xleftarrow{\$} \mathbb{Z}_l$, and also proves that B and E are commitments to the same value b_a. The latter can be accomplished by proving that

   $$E - B = k_0G + k_1X$$

   for some k_0, k_1 using a linear composition proof (A.3). Indeed: $k_0 = b_f - e$, $k_1 = b_x$.

[10] If we do not include q in $h(f+q)$, and instead just have hf, then the staked output's sender could check inequality (3) for all staking events in the ledger (d and h are public knowledge, and an output's sender knows f and a): $da - hf < 2^{64} \pmod{l}$. The likelihood of that test succeeding yet a different output was staked is negligible, so senders could trivially identify when their outputs are staked by recipients. Including the recipient's secret value q makes that test impossible, preserving sender-recipient anonymity.

Note that, if Alice needs to spend her stake output u in the same transaction, which is the case for the PoS protocol used in Zano, she can construct a pseudo output commitment $W = aH + wG$ to the same value a (with $w \xleftarrow{\$} \mathbb{Z}_l$) and provide a linear composition proof for the fact that $C - W = k_0 G + k_1 X$ for some k_0 and k_1.[11]

We believe this approach satisfies all criteria mentioned in Section 4 above.

6.2 Ring-friendly scheme outline

Let's summarize the whole PoS mining process.

1. Alice prepares a set T of possible block timestamps and set U of outputs eligible for staking.
2. For each pair (t, u) she calculates $h = H_s(\dots)$ and $q = vH_s(vR_u)$, and checks the slightly adjusted winning condition (2), using $h(f + q)$ instead of hf:

$$h(f + q) \bmod l \ < \ \left\lfloor \frac{l}{D} \right\rfloor a \tag{13}$$

3. If (13) holds, she generates random non-zero x, x', x'' and e in $\mathbb{Z}_l$, and calculates:

$$\begin{aligned} d &= \left\lfloor \frac{h(f + q) \bmod l}{a} \right\rfloor + 1 \\ b_a &= da - h(f + q) \\ b_f &= d(f + q) - ha \\ b_x &= x'' - hx' + dx \\ C &= xX + aH + (f + q)G \\ C' &= x'X + (f + q)H + aG \\ E &= b_a H + b_f G + b_x X \\ B &= b_a H + eG \end{aligned}$$

4. To prove correctness of C she adds a proof that $C - A_\pi - Q_\pi = xX$ and proof that $Q_\pi = qG$ as additional layers to the main ring signature.
5. To prove correctness of C' she generates two linear composition proofs (c, y_0, y_1), (c, y_2, y_3) for the fact that $C + C' = k_0 X + k_1(H + G)$ and $C - C' = k_2 X + k_3(H - G)$ (Section A.3).
6. She generates a Schnorr proof (c, y_4) with respect to base X for the fact that $hC' - dC + E = x''X$.
7. She generates a range proof $\mathcal{R}_B$ showing $b_a < 2^{64}$ in commitment B.
8. She generates a linear composition proof (c, y_5, y_6) for the fact that $E - B = k_0 G + k_1 X$.

[11] Such a linear composition proof would only show that $W = aH + n_1 G + n_2 X$, where $n_1, n_2 \geq 0$. It is acceptable for $n_2 > 0$ to be true, because any amount balance proof involving W would have to show that a on generator H is canceled out by any new amounts (in the case of a PoS mining transaction, the block reward plus staked output's amount), regardless of statements about values attached to other generators. In practice, setting $n_2 > 0$ may be either impossible (incompatible with balance proofs) or cause new outputs to be unspendable (incompatible with range proofs).

9. She makes a PoS block with timestamp t, containing a mining transaction with stake output u and extended ring signature, and adds the PoS signature σ to the block's data:

$$\sigma = \{d, C, C', E, B, (c, y_0, y_1, y_2, y_3, y_4, y_5, y_6), \mathcal{R}_B\} \tag{14}$$

Note that all the discrete logarithm proofs can share a Fiat-Shamir challenge c.

Also note that using a range proof protocol with double-blinded commitments (like described in Appendix D) allows us to get rid of B, y_5, y_6, which makes the PoS signature σ more compact:

$$\sigma = \{d, C, C', E, (c, y_0, y_1, y_2, y_3, y_4), \mathcal{R}^e_B\}$$

6.3 Verification of ring-friendly PoS scheme

Verifiers on the network check a PoS block as follows.

1. Check $0 < d \leq \lfloor \frac{l}{D} \rfloor$.
2. Calculate $h = H_s(\dots)$ and $F = hC' - dC + E$.
3. Check the stake input's ring signature, which has additional layers for $C - A_i - Q_i$ and Q_i.
4. Check linear composition proofs (c, y_0, y_1, y_2, y_3) for the fact that $C + C' = k_0 X + k_1(H + G)$ and $C - C' = k_2 X + k_3(H - G)$.
5. Check Schnorr signature (c, y_4) for the fact that $F = x'' X$.
6. Check linear composition proof (c, y_5, y_6) for the fact $E - B = k_0 G + k_1 X$.
7. Check range proof $\mathcal{R}_B$.

6.4 Limitations

The proposed scheme works only under the following conditions:

- Proof-of-stake difficulty: $D > 2^{64}$.
- Output's amount: $a < 2^{64}$.
- Commitment's mask: $f \neq -q$ (in Appendix B we consider this in detail).

6.5 Optional sender-recipient anonymity

If the sender-recipient anonymity criterion mentioned in Section 4 is considered optional in a particular implementation, the protocol can be simplified as follows:

- get rid of Q in outputs' data;
- get rid of the additional layer for $Q_\pi = qG$ in the ring signature;
- let $q = 0$ in all equations with q above.

This data-saving approach can also be used when sender-recipient anonymity is important but it is ensured by other means. For instance, Alice could stake only outputs received from trusted parties (e.g. herself), and make other outputs eligible for staking by sending them to herself using a chain of trusted parties.

6.6 Size of ring-friendly PoS proof

Let's estimate the size of the proof for $n-1$ decoy outputs, where the total size of the ring is n. Assume we're using Bulletproofs+ for range proofing. According to [3] it requires $2 \cdot \lceil \log_2(m) + \log_2(k) \rceil + 3$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_l$, where $m = 64$ for range 2^{64}, and $k = 1$ as we only need it for one element b_a[12].

For the ring signature extension we need to store, presumably, only two extra $\mathbb{Z}_l$ elements per ring member (r_i^0, r_i^1).

Additionally, we need to store 9 elements in $\mathbb{Z}_l$ $(d, c, y_0, \ldots, y_6)$ and 4 elements in $\mathbb{G}$ (C, C', E, B) per PoS signature, and one group element per each output in the blockchain (Q).

In total we have 19 group elements and $2n+12$ field elements. If both field and group elements have a compressed size of 32 bytes, which is the case for Ed25519 used in Zano, then the total size of additional PoS data can be estimated as $2n+31$ elements per PoS signature and 1 element per output, or $64n+992$ bytes per PoS signature and 32 bytes per output.

If the sender-recipient anonymity is ensured by other means and the protocol is simplified as explained in subsection 6.5, the total size of additional data is 19 group elements and $n+12$ field elements per PoS block. Or, in case of Ed25519, the total size is $n+31$ elements or $32n+992$ per PoS block.

Acknowledgment

The authors would like to thank Cypher Stack and Aaron Feickert (Sarang Noether) for reviewing this work ([4]), for valuable comments, and especially for providing rigorous security proofs for the sub-protocols in this paper.

References

[1] Daniel J. Bernstein et al. *Ed25519: high-speed high-security signatures*. https://ed25519.cr.yp.to.

[2] Suyash Bagad, Omer Shlomovits, and Claudio Orlandi. *Monero Bulletproofs+ Security Audit*. https://suyash67.github.io/homepage/assets/pdfs/bulletproofs_plus_audit_report_v1.1.pdf. 2021.

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

[4] Aaron Feickert. *Zarcanum technical notes*. https://github.com/cypherstack/zarcanum-review. 2021.

[12] In assumption that the mining transaction or the block has no other suitable Bulletproofs+ that could be aggregated to reduce the size. If there are range proofs that can be aggregated, it is possible to save up to, per additional aggregated proof, 15 elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_l$.

[5] Sunny King and Scott Nadal. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. https://www.peercoin.net/whitepapers/peercoin-paper.pdf. 2012.

[6] Gregory Maxwell. *Confidential Transactions*. https://web.archive.org/web/20200502151159/https://people.xiph.org/~greg/confidential_values.txt (Archived 2020-05-02). 2015.

[7] Shen Noether, Adam Mackenzie, and Monero Core Team. *Ring Confidential Transactions, MRL-0005*. https://web.getmonero.org/resources/research-lab/pubs/MRL-0005.pdf. 2016.

[8] Torben Pryds Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. https://www.cs.cornell.edu/courses/cs754/2001fa/129.PDF. 1992.

[9] Andrey Sabelnikov. *Zano whitepaper*. https://zano.org/downloads/zano_wp.pdf. 2019.

A Basic proofs

A.1 Mirror commitment proof

Suppose Prover ($\mathcal{P}$) needs to convince a Verifier ($\mathcal{V}$) that $A' = fH + aG$ is the mirror commitment of public $A = aH + fG$ without revealing a or f. We can construct a Schnorr-like proof as follows.

Theorem 1. *Let $\mathbb{G}$ be a finite cyclic group where the discrete logarithm problem is hard, and let $\mathbb{F}$ be its scalar field. Let $0 \neq G, H \in \mathbb{G}$ be group elements with no efficiently-computable discrete logarithm relation. We assume that $\mathbb{G}, G, H$ are implicit public parameters where needed. The protocol presented in Fig. A.1 is complete, special sound, and special honest-verifier zero knowledge as a sigma protocol for the relation $\mathcal{R}_1$.*

$$\mathcal{R}_1 = \{A, A' \in \mathbb{G}; f, a \in \mathbb{F} : A = aH + fG,\ A' = fH + aG\}$$

$\boxed{\mathcal{P}}$: $r_0, r_1 \xleftarrow{\$} \mathbb{F}$ and computes:

$R_0 = r_0(G + H)$

$R_1 = r_1(G - H)$

$\boxed{\mathcal{P} \to \mathcal{V}}$: R_0, R_1

$\boxed{\mathcal{V}}$: $c \xleftarrow{\$} \mathbb{F}$

$\boxed{\mathcal{V} \to \mathcal{P}}$: c

$\boxed{\mathcal{P}}$: computes:

$y_0 = r_0 + c(a + f)$

$y_1 = r_1 + c(a - f)$

$\boxed{\mathcal{P} \to \mathcal{V}}$: y_0, y_1

$\boxed{\mathcal{V}}$: returns *Accept* if and only if the following hold:

$$y_0(H + G) - c(A + A') \stackrel{?}{=} R_0 \tag{15}$$

$$y_1(H - G) - c(A + A') \stackrel{?}{=} R_1 \tag{16}$$

Fig. A.1. Interactive protocol for mirror commitment proof

Proof. Completeness follows trivially by inspection.

We next show that the protocol is 2-special sound by a standard rewinding argument, where we define an extractor that produces valid witness elements on accepting transcripts using distinct verifier challenges. Fix an initial transcript (A, A', R_0, R_1), and let $c \neq c'$ be distinct verifier challenges for this transcript, with corresponding responses (y_0, y_1) and (y_0', y_1'). We apply Equations (15) and (16) to these transcripts to obtain

$$(y_0 - y_0')(H + G) - (c - c')(A + A') = 0$$
$$(y_1 - y_1')(H - G) - (c - c')(A - A') = 0$$

and hence

$$\frac{y_0 - y_0'}{c - c'}(H + G) = A + A' \tag{17}$$

$$\frac{y_1 - y_1'}{c - c'}(H - G) = A - A' \tag{18}$$

Define $\alpha_0 = (y_0 - y_0')/(c - c')$ and $\alpha_1 = (y_1 - y_1')/(c - c')$, and note that both are well defined since $c \neq c'$. Adding and subtracting Equations (17) and (18), we obtain the following expressions for A and A':

$$A = \frac{\alpha_0 + \alpha_1}{2}H + \frac{\alpha_0 - \alpha_1}{2}G$$

$$A' = \frac{\alpha_0 - \alpha_1}{2}H + \frac{\alpha_0 + \alpha_1}{2}G$$

Hence we define

$$a = \frac{\alpha_0 + \alpha_1}{2}$$

and

$$f = \frac{\alpha_0 - \alpha_1}{2}$$

as the extracted witnesses, and the protocol is 2-special sound. Observe that a, f must be unique, or this implies a nontrivial discrete logarithm relationship between G, H.

We finally show that the protocol is special honest-verifier zero knowledge. To do so, we define a simulator that, on a valid statement and uniformly sampled verifier challenge, produces transcripts indistinguishable from those of real proofs. Fix a valid prover statement (A, A') and sample a nonzero challenge $c \in \mathbb{F}$. The simulator samples random $y_0, y_1 \in \mathbb{F}$ and defines R_0, R_1 using Equations (15) and (16), respectively. The resulting simulated proof will be accepted by an honest verifier. Because G, H are independent generators, such a simulated proof is distributed identically to a real proof, and hence the protocol is special honest-verifier zero knowledge.

This completes the proof. □

This protocol may be made non-interactive via the Fiat-Shamir technique, where the verifier challenge is replaced by a suitable transcript hash. This technique further allows for binding an arbitrary proof context into the transcript. Fig. A.2 shows an example non-interactive protocol.

$\mathcal{R}_1 = \{A, A' \in \mathbb{G}; f, a \in \mathbb{F} : A = aH + fG,\ A' = fH + aG\}$
$\boxed{\mathcal{P}}$: $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_l$ and computes: $R_0 = r_0(H + G),\ R_1 = r_1(H - G)$ $c = H_s(R_0, R_1, A', A)$ $y_0 = r_0 + c(a + f),\ y_1 = r_1 + c(a - f)$
$\boxed{\mathcal{P} \to \mathcal{V}}$: (c, y_0, y_1)
$\boxed{\mathcal{V}}$: accepts if and only if the following holds: $c \stackrel{?}{=} H_s(y_0(H + G) - c(A + A'),\ y_1(H - G) - c(A - A'),\ A',\ A)$

Fig. A.2. Non-interactive protocol for mirror commitment proof

A.2 Extended mirror commitment proof

Suppose a prover ($\mathcal{P}$) needs to convince a verifier ($\mathcal{V}$) that $C' = fH + aG + x'X$ is a blinded mirror commitment to $C = aH + fG + xX$ which is publicly known. We assume there are no known nontrivial DL relations between generators $G, H, X \in \mathbb{G}$, and expect that scalars a, f, x, x' and un-blinded commitments $C'_u = fH + aG$ and $C_u = aH + fG$ should not be revealed. For this purpose, we make the following construction.

Theorem 2. *Let $\mathbb{G}$ be a finite cyclic group where the discrete logarithm problem is hard and let $\mathbb{F}$ be its scalar field. Let $0 \neq G, H, X \in \mathbb{G}$ be group elements with no efficiently-computable discrete logarithm relation. We assume that $\mathbb{G}, G, H, X$ are implicit public parameters where needed. The protocol presented in Fig. A.3 is complete, special sound, and special honest-verifier zero knowledge as a sigma protocol for the relation $\mathcal{R}_2$.*

$\mathcal{R}_2 = \{C, C' \in \mathbb{G};\ f, a, x, x' \in \mathbb{F} : C = aH + fG + xX,\ C' = fH + aG + x'X\}$

$\boxed{\mathcal{P}}$: $r_0, r_1, s_0, s_1 \xleftarrow{\$} \mathbb{F}$ and computes:

$R_0 = r_0(H + G) + s_0X$

$R_1 = r_1(H - G) + s_1X$

$\boxed{\mathcal{P} \to \mathcal{V}}$: R_0, R_1

$\boxed{\mathcal{V}}$: $c \xleftarrow{\$} \mathbb{F}$

$\boxed{\mathcal{V} \to \mathcal{P}}$: c

$\boxed{\mathcal{P}}$: computes:

$y_0 = r_0 + c(a + f)$

$y_1 = r_1 + c(a - f)$

$z_0 = s_0 + c(x + x')$

$z_1 = s_1 + c(x - x')$

$\boxed{\mathcal{P} \to \mathcal{V}}$: y_0, y_1, z_0, z_1

$\boxed{\mathcal{V}}$: returns *Accept* if and only if the following hold:

$$y_0(H + G) + z_0X - c(C + C') \stackrel{?}{=} R_0 \tag{19}$$

$$y_1(H - G) + z_1X - c(C - C') \stackrel{?}{=} R_1 \tag{20}$$

Fig. A.3. Interactive protocol for extended mirror commitment proof

Proof. Completeness follows trivially by inspection.

We next show that the protocol is 2-special sound by a standard rewinding argument, where we define an extractor that produces valid witness elements on accepting transcripts using distinct verifier challenges. Fix an initial transcript (C, C', R_0, R_1) and let $c \neq c'$ be distinct verifier challenges for this transcript, with corresponding responses (y_0, y_1, z_0, z_1) and (y'_0, y'_1, z'_0, z'_1). We apply

Equations (19) and (20) to these transcripts to obtain

$$(y_0 - y_0')(H + G) + (z_0 - z_0')X - (c - c')(C + C') = 0 \tag{21}$$
$$(y_1 - y_1')(H - G) + (z_1 - z_1')X - (c - c')(C - C') = 0 \tag{22}$$

and hence

$$\frac{y_0 - y_0'}{c - c'}(H + G) + \frac{z_0 - z_0'}{c - c'}X = C + C' \tag{23}$$
$$\frac{y_1 - y_1'}{c - c'}(H - G) + \frac{z_1 - z_1'}{c - c'}X = C - C'. \tag{24}$$

Define the following:

$$\alpha_0 = \frac{y_0 - y_0'}{c - c'}$$
$$\alpha_1 = \frac{y_1 - y_1'}{c - c'}$$
$$\beta_0 = \frac{z_0 - z_0'}{c - c'}$$
$$\beta_1 = \frac{z_1 - z_1'}{c - c'}$$

Note that all are well defined since $c \neq c'$. Adding and subtracting Equations (23) and (24), we obtain the following expressions for C and C':

$$C = \frac{\alpha_0 + \alpha_1}{2}H + \frac{\alpha_0 - \alpha_1}{2}G + \frac{\beta_0 + \beta_1}{2}X$$
$$C' = \frac{\alpha_0 - \alpha_1}{2}H + \frac{\alpha_0 + \alpha_1}{2}G + \frac{\beta_0 - \beta_1}{2}X$$

Hence we define

$$a = \frac{\alpha_0 + \alpha_1}{2}$$
$$f = \frac{\alpha_0 - \alpha_1}{2}$$
$$x = \frac{\beta_0 + \beta_1}{2}$$
$$x' = \frac{\beta_0 - \beta_1}{2}$$

as the extracted witnesses, and the protocol is 2-special sound. Observe that a, f, x, x' must be unique, or this implies a nontrivial discrete logarithm relationship between G, H, X.

We finally show that the protocol is special honest-verifier zero knowledge. To do so, we define a simulator that, on a valid statement and uniformly-sampled verifier challenge, produces transcripts indistinguishable from those of real proofs. Fix a valid prover statement (C, C'), and sample a nonzero challenge $c \in \mathbb{F}$. The simulator samples random $y_0, y_1, z_0, z_1 \in \mathbb{F}$ and defines R_0, R_1 using Equations (19) and (20), respectively. The resulting simulated proof will be accepted by an honest verifier. Because G, H, X are independent generators, such a simulated proof is distributed identically to a real proof, and hence the protocol is special honest-verifier zero knowledge.

This completes the proof. □

As above, this protocol may be made non-interactive via the Fiat-Shamir technique, where the verifier challenge is replaced by a suitable transcript hash. In Fig. A.4 such a non-interactive protocol is shown as an example.

$\mathcal{R}_2 = \{C, C' \in \mathbb{G};\ f, a, x, x' \in \mathbb{F} : C = aH + fG + xX,\ C' = fH + aG + x'X\}$
$\boxed{\mathcal{P}}$: $r_0, r_1, s_0, s_1 \xleftarrow{\$} \mathbb{Z}_l$ and computes: $R_0 = r_0(H+G) + s_0 X,\ R_1 = r_1(H-G) + s_1 X$ $c = H_s(R_0, R_1, C, C')$ $y_0 = r_0 + c(a+f),\ y_1 = r_1 + c(a-f),\ z_0 = s_0 + c(x+x'),\ z_1 = s_1 + c(x-x')$
$\boxed{\mathcal{P} \to \mathcal{V}}$: (c, y_0, y_1, z_0, z_1)
$\boxed{\mathcal{V}}$: accepts if and only if the following holds: $c \stackrel{?}{=} H_s(y_0(H+G) + z_0 X - c(C+C'),\ y_1(H-G) + z_1 X - c(C-C'),\ C,\ C')$

Fig. A.4. Non-interactive protocol for extended mirror commitment proof

A.3 Linear composition proof

Suppose a prover ($\mathcal{P}$) needs to convince a verifier ($\mathcal{V}$) that $C = aA + bB$, where $A, B \in \mathbb{G}$ are generators with no known nontrivial DL relations, and scalars $a, b \in \mathbb{Z}_l$ should not be revealed. For this purpose, we can construct a Schnorr-like proof as follows.

Theorem 3. *Let $\mathbb{G}$ be a finite cyclic group where the discrete logarithm problem is hard, and let $\mathbb{F}$ be its scalar field. Let $0 \neq A, B \in \mathbb{G}$ be group elements with no efficiently-computable discrete logarithm relation. We assume that $\mathbb{G}, A, B$ are implicit public parameters where needed. The protocol presented in Fig. A.5 is complete, special sound, and special honest-verifier zero knowledge as a sigma protocol for the relation $\mathcal{R}_3$.*

$$\mathcal{R}_3 = \{C \in \mathbb{G}; a, b \in \mathbb{F} : C = aA + bB\}$$

$\boxed{\mathcal{P}}$: $r_0, r_1 \xleftarrow{\$} \mathbb{F}$ and computes:
$R = r_0 A + r_1 B$
$\boxed{\mathcal{P} \to \mathcal{V}}$: R
$\boxed{\mathcal{V}}$: $c \xleftarrow{\$} \mathbb{F}$
$\boxed{\mathcal{V} \to \mathcal{P}}$: c
$\boxed{\mathcal{P}}$: computes:
$y_0 = r_0 + ca$
$y_1 = r_1 + cb$
$\boxed{\mathcal{P} \to \mathcal{V}}$: y_0, y_1
$\boxed{\mathcal{V}}$: returns *Accept* if and only if the following hold:

$$y_0 A + y_1 B - cC \stackrel{?}{=} R \tag{25}$$

Fig. A.5. Interactive protocol for linear composition proof

Proof. Completeness follows trivially by inspection.

We next show that the protocol is 2-special sound by a standard rewinding argument, where we define an extractor that produces valid witness elements on accepting transcript using distinct verifier challenges. Fix an initial transcript (C, R), and let $c \neq c'$ be distinct verifier challenges for this transcript, with corresponding responses (y_0, y_1) and (y'_0, y'_1). We apply Equation (25) to these transcripts to obtain

$$(y_0 - y'_0)A + (y_1 - y'_1)B - (c - c')C = 0$$

and hence

$$\frac{y_0 - y'_0}{c - c'} A + \frac{y_1 - y'_1}{c - c'} B = C.$$

Define

$$a = \frac{y_0 - y'_0}{c - c'}$$

and

$$b = \frac{y_1 - y_1'}{c - c'}$$

as the extracted witnesses, which are well defined since $c \neq c'$, and the protocol is 2-special sound. Observe that a, b must be unique, or this implies a nontrivial discrete logarithm relationship between A, B.

Finally, we show that the protocol is special honest-verifier zero knowledge. To do so, we define a simulator that, on a valid statement and uniformly-sampled verifier challenge, produces transcripts indistinguishable from those of real proofs. Fix a valid prover statement C and sample a nonzero challenge $c \in \mathbb{F}$. The simulator samples random $y_0, y_1 \in \mathbb{F}$ and defines R using Equation (25), The resulting simulated proof will be accepted by an honest verifier. Because A, B are independent generators, such a simulated proof is distributed identically to a real proof, and hence the protocol is special honest-verifier zero knowledge.

This completes the proof. □

This protocol may be made non-interactive via the Fiat-Shamir technique, where the verifier challenge is replaced by a suitable transcript hash. This technique further allows for binding arbitrary proof context into the transcript. In Fig. A.6 such a non-interactive protocol is shown as an example.

$\mathcal{R}_3 = \{C \in \mathbb{G}; a, b \in \mathbb{F} : C = aA + bB\}$
$\boxed{\mathcal{P}}$: $r_0, r_1 \xleftarrow{\$} \mathbb{F}$ and computes: $R = r_0 A + r_1 B$ $c = H_s(R, A, B, C)$ $y_0 = r_0 + ca,\ \ y_1 = r_1 + cb$
$\boxed{\mathcal{P} \to \mathcal{V}}$: (c, y_0, y_1)
$\boxed{\mathcal{V}}$: accepts if and only if the following hold: $c \stackrel{?}{=} H_s(y_0 A + y_1 B - cC,\ \ A,\ \ B,\ \ C)$

Fig. A.6. Non-interactive protocol for linear composition proof

B How to ensure $f \neq -q$

In subsection 6.4 we mentioned that the proposed PoS scheme works only under certain conditions, and one of them is the sum $f + q$ must be non-zero for all staking outputs. The reasoning behind that is simple: suppose Alice prepared a UTXO with amount a committed in A with mask $f = -q$: $A = aH - qG$. Such an output can be staked instantly as the winning condition is met regardless of h:

$$hC' - dC + E = x''X \quad (12) \quad \overset{f=-q}{\Longrightarrow} \quad (b_a - da)H + (b_f + ha)G + (hx' - dx + b_x)X = x''X$$

This implies that for *any* given h, Alice can pick arbitrary $b_a < 2^{64}, d, b_f$ such that $b_a = da$ and $b_f = -ha$, and thus all conditions mentioned in subsection 6.3 will be satisfied.

Below we show two solutions for this problem: the blockchain-independent method by constructing a special proof and a blockchain-specific method.

B.1 Proof for $f + q \neq 0$

Here we construct a proof for $f+q \neq 0$ taking into account criteria from section 4. In particular, sender-recipient anonymity must be preserved. We should not allow the sender, who knows a and f, to identify the prover.

Consider the following:

$$C = xX + aH + (f + q)G$$
$$K = x^{-1}(C - aH) = X + x^{-1}(f + q)G$$

The idea of the proof is to show that $k_0C + k_1H = kG + X$ for non-zero k.

The prover acts as follows:

1. Calculates $K = x^{-1}(C - aH)$.
2. Generates a linear composition proof (c, y_0, y_1) for the fact that $K = k_0C + k_1H$ for some $k_0, k_1 \in \mathbb{Z}_l$.
3. Generates a Schnorr proof (c, y_2) for the fact that $K - X = kG$ for some $k \in \mathbb{Z}_l$. Note that $f + q \neq 0$ implies $k \neq 0$.
4. The proof is $\sigma = \{K, c, y_0, y_1, y_2\}$.

   Note that both discrete logarithm proofs can share a Fiat-Shamir challenge c.

Verifier acts as follows:

1. Checks $K \neq X$ (this check ensures that C has a non-zero component of G).
2. Checks the Schnorr proof $(c, y2)$.
3. Checks the linear composition proof (c, y_0, y_1).

The size of the proof can be estimated as 1 group element and 4 field elements. If both field and group elements have a compressed size of 32 bytes, which is the case for Ed25519 used in Zano, then the total size of additional data is 5 elements, or 160 bytes. Note that the Fiat-Shamir challenge can be shared with other discrete logarithm proofs.

B.2 Blockchain-specific solution

Let's require adding $f'H$ and $f'G$ to commitments C and C' respectively before using them in the PoS protocol, where f' is a non-zero public constant that is unknown to a sender when he generates hiding mask f for the output (so he has no way to define f such that $f + q + f' = 0$).

In the case of the Zano blockchain, which has a hybrid PoW/PoS consensus[13], f' can be calculated as $f' = H_s(last\_pow\_id)$, where $last\_pow\_id$ is the hash identifier of the last PoW block in the blockchain[14]. That way, a malicious miner would have no chance to choose f for his benefit by building an alternative PoS subchain.

Consequently, substituting $f+q+f'$ for $f+q$ in (2) we get this modified PoS winning condition:

$$h(f + q + f') \bmod l \ < \ \left\lfloor \frac{l}{D} \right\rfloor a$$

Doing the same for (11) we get:

$$\begin{cases} h(f + q + f') - da + b_a = 0 & | \times H \\ ha - d(f + q + f') + b_f = 0 & | \times G \\ hx' - dx + b_x = x'' & | \times X \end{cases}$$

Now for (12) we obtain:

$$hC' - dC + E + f'(hH - dG) = x''X$$

With such a modification, neither Alice nor the staked output's sender will be able to gain an advantage when choosing f.

[13] More info in the Zano whitepaper [9].

[14] This would also require that the real output and all the decoy outputs in the staking ring signature to be either older than, or the same age as, the most recent PoW block. Verifiers on the network can easily check such a requirement. Note that $last\_pow\_id$ is also used as part of the $stake\_modifier$ as described in Section 2.

C Brute-force attack, its complexity, and mitigation

The weak point of the scheme proposed in Section 6 is the relation between public scalars d and h, sender-known scalars a and f, and secret scalar q:

$$d = \left\lfloor \frac{h(f+q) \bmod l}{a} \right\rfloor + 1$$

An adversarial sender can reveal secret q by guessing $k \in [0; a-1]$ as follows:

$$q = h^{-1}((d-1)a + k) - f$$

And then reveal Alice's secret view key: $v = (H_s(rV))^{-1}q$, where r is the transaction's secret key which is known to the sender.

C.1 Complexity

An adversarial sender would need to scan the blockchain and for each PoS block that is referencing his output in the miner transaction, guess k and check $Q_j \stackrel{?}{=} qG$ for each attempt. The most expensive operation here is EC scalar multiplication, and each attempt would require one such operation. The average number of attempts when testing an output that was staked is upper-bounded at $\frac{1}{2}a$.

Let us use the Zano blockchain as a reference for estimation. As the vast majority of PoS blocks in Zano now have their stake value in $[100 \cdot 10^{12}; 600 \cdot 10^{15}]$, the expected number of attempts can be roughly estimated as 2^{50} in practice, which arguably isn't secure enough given that the reward is Alice's secret view key v.

Below we suggest two ways of mitigation: by increasing the complexity of the attack and by eliminating all risk to secret v.

C.2 Solution 1: increasing complexity

We start with a modification of the direct-spend scheme (Section 5). Let $z = const$. Consider the following modification to (4) and (7):

$$hf - dza + b_a = 0, \quad 1 \le d \le \left\lfloor \frac{l}{zD} \right\rfloor, \; b_a < z2^{64} \tag{26}$$

$$hA' - dzA + B = \mathbf{0}$$

As the range of b_a is z times bigger, it would require the range proof for B to be extended correspondingly.

For the ring-friendly scheme variant, equation (12) can be modified like this:

$$hC' - dzC + E = x''X = F$$

and similarly $B = b_a H + eG$ would require a wider range proof.

With this modification, the expected number of attempts to guess the correct q or v is z times bigger. At the same time, z cannot be arbitrarily big, because the bigger z is, the more likely (26) will hold without satisfying the main PoS condition (2) due to the rounding error[15].

This approach is certainly a trade off. In the case of the Zano blockchain, values of $2^{64} \leq z \leq 2^{106}$ look reasonable because the complexity of a brute force attack will rise up to $2^{114} \dots 2^{156}$ without increasing rounding error too much[16]. If using Bulletproofs+ for range proofs, the additional data for increasing the range's upper boundary to $2^{65} \dots 2^{128}$ is 2 group elements and for increasing it to $2^{129} \dots 2^{256}$ is only 4 group elements, comparing to the standard 2^{64} range[17].

C.3 Solution 2: eliminating all risk to secret key v

If secret key v shouldn't be put even at negligible risk, the following solution can be used.

- All addresses are 3-key tuples: $(V, S, T) = (vG, sG, tG)$;
- Sender calculates Q for each output along with one-time address P:

  $$Q = H_s(rV)T$$

  Note that only Alice and those who get secret keys v and t from her can calculate secret $q = H_s(vR)t$, where R is the transaction's public key.
- Follow the rest of the protocol using the calculated q.

A brute force attack in this case would only reveal secret key t, which is not used in balance calculation nor for transferring coins. If a sender exposes a recipient's t using a staked output they sent to that person, it would only allow them to identify when other outputs they sent to that person are staked.

[15] From (26) we get:

$$(hf \bmod l + b_a) \bmod l \leq \left\lfloor \frac{l}{zD} \right\rfloor za$$

Note that the right side never exceeds l. As b_a can be arbitrary chosen in $[0; z2^{64})$, this inequality also holds for $hf \bmod l > l - z(2^{64} - a)$ (if $d = 1$), resulting in an overall acceptable interval for $hf \pmod l$ of

$$\left[0; \left\lfloor \frac{l}{zD} \right\rfloor za\right] \cup \big(l - z(2^{64} - a); l\big)$$

Now we can estimate the relative increase in possible values of $hf \pmod l$ that satisfy the PoS winning condition due to this 'rounding error' as the 'length' of the rounding-zone compared to the 'length' of the intended PoS zone (note real arithmetic division):

$$P_{\%} = \frac{z(2^{64} - a) - 1}{\left\lfloor \frac{l}{zD} \right\rfloor za + 1} \cdot 100\%$$

[16] Assuming maximum difficulty $D \approx 2^{70}$ for the Zano blockchain, and $a \ll 2^{64}$, we can approximate $P_{\%}$:

$$P_{\%} \approx \frac{z}{a} 2^{-118} \cdot 100\%$$

For $z = 2^{106}$ using the smallest possible stake amount of 1, we get a $P_{\%} \approx 2^{-12} \cdot 100\% \approx 0.024\%$ increase in the probability of satisfying the PoS winning condition (compared to what we intend). However, meeting the PoS winning condition for $a = 1$ and $D \approx 2^{70}$ is already very unlikely, and for greater values of a the value of $P_{\%}$ will be even smaller. Therefore, if $z \lessapprox 2^{106}$, $a < 2^{64}$, and $D \lessapprox 2^{70}$, then the rounding error will be negligible.

[17] According to [3], Bulletproofs+ requires $2 \cdot \lceil \log_2(m) + \log_2(k) \rceil + 3$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_l$, where 2^m is the upper boundary, and $k = 1$ as we only need it for one element B.

D Bulletproofs+ with double-blinded commitments

D.1 Introduction

The original Bulletproofs+ protocol [3] uses the following group-based range relation[18]:

$$\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n,\ g, h, V \in \mathbb{G},\ v, \gamma \in \mathbb{Z}_p) : V = g^v h^\gamma \ \wedge\ v \in [0, 2^n - 1]\}$$

which is only compatible with single-blinded commitments of the form $V = g^v h^\gamma$. Below we extend the Bulletproofs+ protocol to support double-blinded commitments using a group-based range relation as follows:

$$\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n,\ g, h_1, h_2, V \in \mathbb{G},\ v, \gamma_1, \gamma_2 \in \mathbb{Z}_p) : V = g^v h_1^{\gamma_1} h_2^{\gamma_2} \ \wedge\ v \in [0, 2^n - 1]\}$$

Instead of a single blinding mask generator h we introduce two generators $h_1, h_2 \in \mathbb{G}$ and corresponding blinding masks $\gamma_1, \gamma_2 \in \mathbb{Z}_p$. It is assumed the discrete logarithm relation is unknown for all used generators.

D.2 Zero knowledge argument for weighted inner product (WIP) relation

A WIP agrument protocol with double-blinded commitments support is shown in Fig. D.1. We provide the security statement for the proposed updated zk-WIP protocol in Theorem 4. The proof of Theorem 4 mostly corresponds to the original proof in Bulletproofs+ paper [3]. For the reader's convenience all changes are highlighted.

Theorem 4. *Let y be a constant in $\mathbb{Z}_p^*$. The zero-knowledge argument for WIP presented in Fig.D.1 has perfect completeness, perfect honest verifier zero-knowledge, and computational witness-extended emulation.*

Proof. (perfect completeness) We show that the WIP argument has perfect completeness. First, we assume that $P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^{\boldsymbol{a} \odot_y \boldsymbol{b}} h_1^{\alpha_1} h_2^{\alpha_2}$ and show that the case $n = 1$ satisfies perfect completeness. That is, we show that the verification equation holds. It is sufficient to show that the following five equalities hold (corresponding to bases $\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2$).

$$\begin{aligned}
\boldsymbol{a}e^2 + re = (\boldsymbol{a}e + r)e &= r'e && \in \mathbb{Z}_p \\
\boldsymbol{b}e^2 + se = (\boldsymbol{b}e + s)e &= s'e && \in \mathbb{Z}_p \\
\boldsymbol{a}y\boldsymbol{b}e^2 + (ry\boldsymbol{b} + sy\boldsymbol{a})e + rys = (\boldsymbol{b}e + s)(\boldsymbol{a}ye + ry) &= r' \odot_y s' && \in \mathbb{Z}_p \\
\alpha_1 e^2 + \delta_1 e + \eta_1 &= \delta_1' && \in \mathbb{Z}_p \\
\alpha_2 e^2 + \delta_2 e + \eta_2 &= \delta_2' && \in \mathbb{Z}_p
\end{aligned}$$

From the above five equalities, the perfect completeness for the case $n = 1$ is proven.

Next, we move to the case $n > 1$. At the end of every recursion, if the parameters $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, g, h_1, h_2, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, \widehat{\alpha}_1, \widehat{\alpha}_2)$ that will be used for the next call satisfy the relation $\widehat{P} = \widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}} \widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}} g^{\widehat{\boldsymbol{a}} \odot_y \widehat{\boldsymbol{b}}} h_1^{\widehat{\alpha}_1} h_2^{\widehat{\alpha}_2}$ when $P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^{\boldsymbol{a} \odot_y \boldsymbol{b}} h_1^{\alpha_1} h_2^{\alpha_2}$, then we can be sure that the protocol will end up with a correct input

[18] For clarity, we use original notation from the Bulletproofs+ [3] paper throughout this entire appendix.

Original Bulletproofs+	Extended Bulletproofs+
$\boxed{\text{zk-WIP} \underset{y^n}{\longrightarrow} (\boldsymbol{g}, \boldsymbol{h}, g, h, P; \boldsymbol{a}, \boldsymbol{b}, \alpha)}$	$\boxed{\text{zk-WIP} \underset{y^n}{\longrightarrow} (\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, P; \boldsymbol{a}, \boldsymbol{b}, \alpha_1, \alpha_2)}$
Relation:	
$\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n, \alpha \in \mathbb{Z}_p) : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^{\boldsymbol{a} \odot_y \boldsymbol{b}} h^{\alpha}\}$	$\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h_1, h_2, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n, \alpha_1, \alpha_2 \in \mathbb{Z}_p) : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^{\boldsymbol{a} \odot_y \boldsymbol{b}} h_1^{\alpha_2} h_2^{\alpha_2}\}$
$\mathcal{P}$'s input:	
$(\boldsymbol{g}, \boldsymbol{h}, g, h, P; \boldsymbol{a}, \boldsymbol{b}, \alpha)$	$(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, P; \boldsymbol{a}, \boldsymbol{b}, \alpha_1, \alpha_2)$
$\mathcal{V}$'s input:	
$(\boldsymbol{g}, \boldsymbol{h}, g, h, P)$	$(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, P)$
$\mathcal{P}$'s output: none	
$\mathcal{V}$'s output: Accept or Reject	
$\boxed{\text{If } n = 1:}$	
$\boxed{\mathcal{P}}$: $r, s, \delta, \eta \xleftarrow{\$} \mathbb{Z}_p$ and computes: $A = \boldsymbol{g}^r \boldsymbol{h}^s g^{r \odot_y \boldsymbol{b} + s \odot_y \boldsymbol{a}} h^{\delta} \in \mathbb{G}$ $B = g^{r \odot_y s} h^{\eta} \in \mathbb{G}$	$\boxed{\mathcal{P}}$: $r, s, \delta_1, \delta_2, \eta_1, \eta_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes: $A = \boldsymbol{g}^r \boldsymbol{h}^s g^{r \odot_y \boldsymbol{b} + s \odot_y \boldsymbol{a}} h_1^{\delta_1} h_2^{\delta_2} \in \mathbb{G}$ $B = g^{r \odot_y s} h_1^{\eta_1} h_2^{\eta_2} \in \mathbb{G}$
$\boxed{\mathcal{P} \to \mathcal{V}}$: A, B $\boxed{\mathcal{V}}$: $e \xleftarrow{\$} \mathbb{Z}_p^*$ $\boxed{\mathcal{P} \leftarrow \mathcal{V}}$: e $\boxed{\mathcal{P}}$: computes: $r' = r + \boldsymbol{a} \cdot e \in \mathbb{Z}_p$ $s' = s + \boldsymbol{b} \cdot e \in \mathbb{Z}_p$	
$\delta' = \eta + \delta \cdot e + \alpha \cdot e^2 \in \mathbb{Z}_p$	$\delta_1' = \eta_1 + \delta_1 \cdot e + \alpha_1 \cdot e^2 \in \mathbb{Z}_p$ $\delta_2' = \eta_2 + \delta_2 \cdot e + \alpha_2 \cdot e^2 \in \mathbb{Z}_p$
$\boxed{\mathcal{P} \to \mathcal{V}}$: r', s', δ'	$\boxed{\mathcal{P} \to \mathcal{V}}$: $r', s', \delta_1', \delta_2'$
$\boxed{\mathcal{V}}$: outputs *Accept* iff the following holds: $P^{e^2} A^e B = \boldsymbol{g}^{r' \cdot e} \boldsymbol{h}^{s' \cdot e} g^{r' \odot_y s'} h^{\delta'} \in \mathbb{G}$	$\boxed{\mathcal{V}}$: outputs *Accept* iff the following holds: $P^{e^2} A^e B = \boldsymbol{g}^{r' \cdot e} \boldsymbol{h}^{s' \cdot e} g^{r' \odot_y s'} h_1^{\delta_1'} h_2^{\delta_2'} \in \mathbb{G}$
$\boxed{\text{else } (n > 1):}$ Let $\widehat{n} = \frac{n}{2},\ \boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2),\ \boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2),\ \boldsymbol{g} = (\boldsymbol{g}_1, \boldsymbol{g}_2),\ \boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$ (where $\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{g}_i$ and $\boldsymbol{h}_i$ are of the same length $\widehat{n}$)	
$\boxed{\mathcal{P}}$: $d_L, d_R \xleftarrow{\$} \mathbb{Z}_p$ and computes:	$\boxed{\mathcal{P}}$: $d_L', d_R', d_L'', d_R'' \xleftarrow{\$} \mathbb{Z}_p$ and computes:
$c_L = \boldsymbol{a_1} \odot_y \boldsymbol{b_2} \in \mathbb{Z}_p$ $c_R = (y^{\widehat{n}} \cdot \boldsymbol{a_2}) \odot_y \boldsymbol{b_1} \in \mathbb{Z}_p$	
$L = \boldsymbol{g}_2^{(y^{-\widehat{n}} \cdot \boldsymbol{a}_1)} \boldsymbol{h}_1^{\boldsymbol{b}_2} g^{c_L} h^{d_L} \in \mathbb{G}$ $R = \boldsymbol{g}_1^{(y^{\widehat{n}} \cdot \boldsymbol{a}_2)} \boldsymbol{h}_2^{\boldsymbol{b}_1} g^{c_R} h^{d_R} \in \mathbb{G}$	$L = \boldsymbol{g}_2^{(y^{-\widehat{n}} \cdot \boldsymbol{a}_1)} \boldsymbol{h}_1^{\boldsymbol{b}_2} g^{c_L} h_1^{d_L'} h_2^{d_L''} \in \mathbb{G}$ $R = \boldsymbol{g}_1^{(y^{\widehat{n}} \cdot \boldsymbol{a}_2)} \boldsymbol{h}_2^{\boldsymbol{b}_1} g^{c_R} h_1^{d_R'} h_2^{d_R''} \in \mathbb{G}$
$\boxed{\mathcal{P} \to \mathcal{V}}$: L, R $\boxed{\mathcal{V}}$: $e \xleftarrow{\$} \mathbb{Z}_p^*$ $\boxed{\mathcal{P} \leftarrow \mathcal{V}}$: e $\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: compute: $\widehat{\boldsymbol{g}} = \boldsymbol{g}_1^{e^{-1}} \circ \boldsymbol{g}_2^{e \cdot y^{-\widehat{n}}} \in \mathbb{G}^{\widehat{n}}$ $\widehat{\boldsymbol{h}} = \boldsymbol{h}_1^{e} \circ \boldsymbol{h}_2^{e^{-1}} \in \mathbb{G}^{\widehat{n}}$ $\widehat{P} = L^{e^2} P R^{e^{-2}} \in \mathbb{G}$ $\boxed{\mathcal{P}}$: computes: $\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 \cdot e + (\boldsymbol{a}_2 \cdot y^{\widehat{n}}) \cdot e^{-1} \in \mathbb{Z}_p^{\widehat{n}}$ $\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 \cdot e^{-1} + \boldsymbol{b}_2 \cdot e \in \mathbb{Z}_p^{\widehat{n}}$	
$\widehat{\alpha} = d_L \cdot e^2 + \alpha + d_R \cdot e^{-2} \in \mathbb{Z}_p$	$\widehat{\alpha_1} = d_L' \cdot e^2 + \alpha_1 + d_R' \cdot e^{-2} \in \mathbb{Z}_p$ $\widehat{\alpha_2} = d_L'' \cdot e^2 + \alpha_2 + d_R'' \cdot e^{-2} \in \mathbb{Z}_p$
$\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: run: zk-WIP$_{\overrightarrow{y}^{\widehat{n}}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, g, h, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, \widehat{\alpha})$	$\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: run: zk-WIP$_{\overrightarrow{y}^{\widehat{n}}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, g, h_1, h_2, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, \widehat{\alpha_1}, \widehat{\alpha_2})$

Fig. D.1. Zero Knowledge Argument for WIP relation

for the last step of $n = 1$. Therefore we show that if the input P is of the form $\boldsymbol{g}^{\boldsymbol{a}}\boldsymbol{h}^{\boldsymbol{b}}g^{\boldsymbol{a}\odot_y\boldsymbol{b}}h_1^{\alpha_1}h_2^{\alpha_2}$ and $\widehat{P}, \widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}$ are computed as in the protocol, then $\widehat{P}$ has the desired form $\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}}\widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}}g^{\widehat{\boldsymbol{a}}\odot_y\widehat{\boldsymbol{b}}}h_1^{\widehat{\alpha}_1}h_2^{\widehat{\alpha}_2}$.

Let $P, \widehat{P}, \widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}$ be the form in the protocol description for the case $n > 1$. If L and R are computed as described in the protocol, then $\widehat{P}$ is computed as $\widehat{P} = L^{e^2}PR^{e^{-2}}$ and we can write $\widehat{P}$ according to the corresponding bases.

$$
\begin{aligned}
\boldsymbol{a}_1 + y^{\widehat{n}}\boldsymbol{a}_2e^{-2} &= (\boldsymbol{a}_1e + y^{\widehat{n}}\boldsymbol{a}_2e^{-1})e^{-1} = \widehat{\boldsymbol{a}}e^{-1} && \in \mathbb{Z}_p \\
y^{-\widehat{n}}\boldsymbol{a}_1e^2 + \boldsymbol{a}_2 &= (\boldsymbol{a}_1e + y^{\widehat{n}}\boldsymbol{a}_2e^{-1})ey^{-\widehat{n}} = \widehat{\boldsymbol{a}}ey^{-\widehat{n}} && \in \mathbb{Z}_p \\
\boldsymbol{b}_2e^2 + \boldsymbol{b}_1 &= (\boldsymbol{b}_2e + \boldsymbol{b}_1e^{-1})e = \widehat{\boldsymbol{b}}e && \in \mathbb{Z}_p \\
\boldsymbol{b}_2 + \boldsymbol{b}_1e^{-2} &= (\boldsymbol{b}_2e + \boldsymbol{b}_1e^{-1})e^{-1} = \widehat{\boldsymbol{b}}e^{-1} && \in \mathbb{Z}_p \\
c_Le^2 + \boldsymbol{a}\odot_y\boldsymbol{b} + c_Re^{-2} &= \boldsymbol{a}_1\odot_y\boldsymbol{b}_2e^2 + \boldsymbol{a}\odot_y\boldsymbol{b} + y^{\widehat{n}}\boldsymbol{a}_2\odot_y\boldsymbol{b}_1e^{-2} && \in \mathbb{Z}_p \\
d'_Le^2 + \alpha_1 + d'_Re^{-2} &= \widehat{\alpha}_1 && \in \mathbb{Z}_p \\
d''_Le^2 + \alpha_2 + d''_Re^{-2} &= \widehat{\alpha}_1 && \in \mathbb{Z}_p
\end{aligned}
$$

Furthermore, from the definition of $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$, we see that

$$
\begin{aligned}
&\widehat{\boldsymbol{a}}\odot_y\widehat{\boldsymbol{b}} \\
&= (\boldsymbol{a}_1e + (\boldsymbol{a}_2y^{\widehat{n}})e^{-1})\odot_y(\boldsymbol{b_1}e^{-1} + \boldsymbol{b}_2e) \\
&= \boldsymbol{a}_1\odot_y\boldsymbol{b}_1 + \boldsymbol{a}_1\odot_y\boldsymbol{b}_2e^2 + (y^{\widehat{n}}\boldsymbol{a}_2)\odot_y\boldsymbol{b}_1e^{-2} + (y^{\widehat{n}}\boldsymbol{a}_2)\odot_y\boldsymbol{b}_2 \\
&= \boldsymbol{a}_1\odot_y\boldsymbol{b}_2e^2 + \boldsymbol{a}\odot_y\boldsymbol{b} + (y^{\widehat{n}}\boldsymbol{a}_2)\odot_y\boldsymbol{b}_1e^{-2} \quad \in \mathbb{Z}_p,
\end{aligned}
$$

which is equal to the g-base exponent of $\widehat{P}$. Using the above observation, we can easily check that the following holds.

$$
\begin{aligned}
\widehat{P} &= \boldsymbol{g}_1^{\widehat{\boldsymbol{a}}e^{-1}}\boldsymbol{g}_2^{\widehat{\boldsymbol{a}}ey^{-\widehat{n}}}\boldsymbol{h}_1^{\widehat{\boldsymbol{b}}e}\boldsymbol{h}_2^{\widehat{\boldsymbol{b}}e^{-1}}g^{\widehat{\boldsymbol{a}}\odot_y\widehat{\boldsymbol{b}}}h_1^{\widehat{\alpha}_1}h_2^{\widehat{\alpha}_2} \\
&= (\boldsymbol{g}_1^{e^{-1}}\boldsymbol{g}_2^{ey^{-\widehat{n}}})^{\widehat{\boldsymbol{a}}}(\boldsymbol{h}_1^e\boldsymbol{h}_2^{e^{-1}})^{\widehat{\boldsymbol{b}}}g^{\widehat{\boldsymbol{a}}\odot_y\widehat{\boldsymbol{b}}}h_1^{\widehat{\alpha}_1}h_2^{\widehat{\alpha}_2} \\
&= \widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}}\widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}}g^{\widehat{\boldsymbol{a}}\odot_y\widehat{\boldsymbol{b}}}h_1^{\widehat{\alpha}_1}h_2^{\widehat{\alpha}_2} \quad \in \mathbb{G}
\end{aligned}
$$

This completes the proof of perfect completeness.

(perfect SHVZK) To prove the argument system is perfect special honest verifier zero-knowledge, we construct a simulator which, given only the public input, outputs a simulated transcript that is identical to the valid transcript produced by the prover and verifier in the real interaction.

We first describe our simulator construction, and then analyze it. The simulator begins with taking the statement and the randomness ρ of the verifier as input. Using ρ, the simulator can generate all challenges whose distribution is identical to that of the real argument. We describe how the simulator generates the non-challenge part. For each $n > 1$, the simulator chooses two random group elements to be L_n, R_n. For the case of $n = 1$, the simulator chooses $A_s \xleftarrow{\$} \mathbb{G}$ and $r'_s, s'_s, \delta'_{1,s}, \delta'_{2,s} \xleftarrow{\$} \mathbb{Z}_p$ at random and computes

$$
B_s = (P^{e^2}A^e\boldsymbol{g}^{-r'_s\cdot e}\boldsymbol{h}^{-s'_s\cdot e}g^{-r'_s\odot_y s'_s}h_1^{-\delta'_{1,s}}h_2^{-\delta'_{2,s}})^{-1} \quad \in \mathbb{G}.
$$

Next, we analyze the distribution of the simulated transcript for the non-challenge part $(\{(L_i, R_i)\}_i, A_s, B_s, r'_s, s'_s, \delta'_{1,s}, \delta'_{2,s})$. In the protocol description, $\forall i, (L_i, R_i)$ are distributed uniformly and independently due to blinding factors $d'_{L_i}, d'_{R_i}, d''_{L_i}$, and d''_{R_i}, and all (L_i, R_i) contribute to generate the point P used in the case $n = 1$. The simulator generates all (L_i, R_i) uniformly at random so that their distributions are identical to that of the real argument. From now, we analyze the distribution of $(A_s, B_s, r'_s, s'_s, \delta'_{1,s}, \delta'_{2,s})$ for a given P in the case $n = 1$.

Before analyzing the simulated transcript $(A_s, B_s, r'_s, s'_s, \delta'_{1,s}, \delta'_{2,s})$, we first analyze the real transcript $(A, B, r', s', \delta'_1, \delta'_2)$ and then show the two distributions are identical.

Here, we focus on $(A, r', s', \delta'_1, \delta'_2)$ and claim that it is uniformly distributed in $\mathbb{G} \times \mathbb{Z}_p^4$ when $(r, s, \delta_1, \delta_2, \eta_1, \eta_2)$ is uniformly distributed in $\mathbb{Z}_p^6$. To this end, it is sufficient to prove the following claim.

Claim: *There exists a one-to-one correspondence between* $(r, s, \delta_1, \delta_2, \eta_1, \eta_2)$ *and* $(A, r', s', \delta'_1, \delta'_2)$.

Proof: First, consider the following function mapping from $(r, s, \delta_1, \delta_2, \eta_1, \eta_2)$ to $(A, r', s', \delta'_1, \delta'_2)$.

$$\begin{pmatrix} A \\ r' \\ s' \\ \delta'_1 \\ \delta'_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}^r \boldsymbol{h}^s g^{r \odot_y \boldsymbol{b} + s \odot_y \boldsymbol{a}} h_1^{\delta_1} h_2^{\delta_2} \\ r + \boldsymbol{a} \cdot e \\ s + \boldsymbol{b} \cdot e \\ \eta_1 + \delta_1 \cdot e + \alpha_1 \cdot e^2 \\ \eta_2 + \delta_2 \cdot e + \alpha_2 \cdot e^2 \end{pmatrix} \quad \in \mathbb{G} \times \mathbb{Z}_P^4$$

Assume that there is another tuple $(\tilde{r}, \tilde{s}, \tilde{\delta_1}, \tilde{\delta_2}, \tilde{\eta_1}, \tilde{\eta_2}) \neq (r, s, \delta_1, \delta_2, \eta_1, \eta_2)$ whose image via the above function is also $(A, r', s', \delta'_1, \delta'_2)$. Then, comparing the two function values we obtain

$$\begin{pmatrix} \boldsymbol{g}^r \boldsymbol{h}^s g^{r \odot_y \boldsymbol{b} + s \odot_y \boldsymbol{a}} h_1^{\delta_1} h_2^{\delta_2} \\ r + \boldsymbol{a} \cdot e \\ s + \boldsymbol{b} \cdot e \\ \eta_1 + \delta_1 \cdot e + \alpha_1 \cdot e^2 \\ \eta_2 + \delta_2 \cdot e + \alpha_2 \cdot e^2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{g}^{\tilde{r}} \boldsymbol{h}^{\tilde{s}} g^{\tilde{r} \odot_y \boldsymbol{b} + \tilde{s} \odot_y \boldsymbol{a}} h_1^{\tilde{\delta_1}} h_2^{\tilde{\delta_2}} \\ \tilde{r} + \boldsymbol{a} \cdot e \\ \tilde{s} + \boldsymbol{b} \cdot e \\ \tilde{\eta_1} + \tilde{\delta_1} \cdot e + \alpha_1 \cdot e^2 \\ \tilde{\eta_2} + \tilde{\delta_2} \cdot e + \alpha_2 \cdot e^2 \end{pmatrix}$$

From the second and third rows we get $r = \tilde{r}$ and $s = \tilde{s}$. This lets us simplify the first row:

$$\begin{aligned} \boldsymbol{g}^r \boldsymbol{h}^s g^{r \odot_y \boldsymbol{b} + s \odot_y \boldsymbol{a}} h_1^{\delta_1} h_2^{\delta_2} \cdot (\boldsymbol{g}^{\tilde{r}} \boldsymbol{h}^{\tilde{s}} g^{\tilde{r} \odot_y \boldsymbol{b} + \tilde{s} \odot_y \boldsymbol{a}} h_1^{\tilde{\delta_1}} h_2^{\tilde{\delta_2}})^{-1} = 1_{\mathbb{G}} \Rightarrow \\ h_1^{\delta_1 - \tilde{\delta_1}} h_2^{\delta_2 - \tilde{\delta_2}} = 1_{\mathbb{G}} \end{aligned}$$

Using the discrete logarithm relation assumption for h_1 and h_2, we get

$$\begin{aligned} \delta_1 &= \tilde{\delta_1} \\ \delta_2 &= \tilde{\delta_2} \end{aligned}$$

Using that, from the last two equations for η_1 and η_2 we finally obtain $\eta_1 = \tilde{\eta_1}$ and $\eta_2 = \tilde{\eta_2}$. Thus:

$$(\tilde{r}, \tilde{s}, \tilde{\delta_1}, \tilde{\delta_2}, \tilde{\eta_1}, \tilde{\eta_2}) = (r, s, \delta_1, \delta_2, \eta_1, \eta_2)$$

This contradiction concludes the proof of the claim. □

In the generation of the real transcript $(A, B, r', s', \delta'_1, \delta'_2)$, only six random integers r, s, $\delta_1, \delta_2, \eta_1$ and η_2 are used. Therefore, the above result implies that the distribution of $(A, B, r', s', \delta'_1, \delta'_2)$ is identical to the distribution that $(A, r', s', \delta'_1, \delta'_2)$ is uniformly distributed and B is uniquely defined by the others and the verification equation. In fact, the latter process is exactly the same as the simulated transcript. Therefore, the simulated transcript is identical to that of the real transcript for given P in the case $n = 1$. This concludes our proof of the perfect special honest verifier zero-knowledge.

(witness-extended emulation) For witness extended emulation, we construct an expected polynomial time extractor χ that extracts a witness using a $poly(\lambda)$-bounded tree of accepting transcripts, in

order to meet the requirements of the general forking lemma. Consider the case $n = 1$. At the first move, the prover sends A and B to the verifier. By rewinding the oracle $\langle \mathcal{P}^*, \mathcal{V} \rangle$ four times with five distinct challenges e_1, e_2, e_3, e_4, and e_5 while using the same A and B, the extractor obtains five tuples $(r'_i, s'_i, \delta'_{1,i}, \delta'_{2,i})$ for $i = 1, \ldots, 5$ satisfying the following verification equation.

$$P^{e_i^2} A^{e_i} B = \boldsymbol{g}^{r'_i \cdot e_i} \boldsymbol{h}^{s'_i \cdot e_i} g^{r'_i \odot_y s'_i} h_1^{\delta'_{1,i}} h_2^{\delta'_{2,i}} \quad \text{for } i = 1, \ldots, 5 \tag{27}$$

Using the first three challenges and the corresponding valid responses, we can interpret the exponents as the product of a 3×3 Vandermonde matrix (which is invertible in $\mathbb{Z}_p^{3\times 3}$ since all e_i's are distinct):

$$\begin{pmatrix} e_1^2 & e_1 & 1 \\ e_2^2 & e_2 & 1 \\ e_3^2 & e_3 & 1 \end{pmatrix} \begin{pmatrix} a_P & b_P & c_P & d_P & f_P \\ a_A & b_A & c_A & d_A & f_A \\ a_B & b_B & c_B & d_B & f_B \end{pmatrix} = \begin{pmatrix} r'_1 e_1 & s'_1 e_1 & r'_1 \odot_y s'_1 & \delta'_{1,1} & \delta'_{2,1} \\ r'_2 e_2 & s'_2 e_2 & r'_2 \odot_y s'_2 & \delta'_{1,2} & \delta'_{2,2} \\ r'_3 e_3 & s'_3 e_3 & r'_3 \odot_y s'_3 & \delta'_{1,3} & \delta'_{2,3} \end{pmatrix}$$

The other exponents in the right hand side of Eq. (27) are public as well. Thus, from those three challenges and responses, we can obtain the exponents $a_P, b_P, c_P, d_P, f_P, a_A, b_A, c_A, d_A, f_A, a_B, b_B, c_B, d_B, f_B$ such that

$$\begin{aligned} P &= \boldsymbol{g}^{a_P} \boldsymbol{h}^{b_P} g^{c_P} h_1^{d_P} h_2^{f_P} \\ A &= \boldsymbol{g}^{a_A} \boldsymbol{h}^{b_A} g^{c_A} h_1^{d_A} h_2^{f_A} \\ B &= \boldsymbol{g}^{a_B} \boldsymbol{h}^{b_B} g^{c_B} h_1^{d_B} h_2^{f_B} \end{aligned}$$

Using the above three equations and the verification equation, we obtain for each $e_i \in \{e_1, \ldots, e_5\}$,

$$\begin{aligned} &\boldsymbol{g}^{r'_i e_i - a_P e_i^2 - a_A e_i - a_B} \boldsymbol{h}^{s'_i e_i - b_P e_i^2 - b_A e_i - b_B} \\ \cdot\ &g^{r'_i \odot_y s'_i - c_P e_i^2 - c_A e_i - c_B} h_1^{\delta'_{1,i} - d_P e_i^2 - d_A e_i - d_B} h_2^{\delta'_{2,i} - f_P e_i^2 - f_A e_i - f_B} = 1_{\mathbb{G}}. \end{aligned}$$

Thus, under the discrete logarithm relation assumption, we have five equations of exponents according to the bases $\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2$,

$$\begin{aligned} r'_i e_i - a_P e_i^2 - a_A e_i - a_B &= 0 \\ s'_i e_i - b_P e_i^2 - b_A e_i - b_B &= 0 \\ r'_i \odot_y s'_i - c_P e_i^2 - c_A e_i - c_B &= 0 \\ \delta'_{1,i} - d_P e_i^2 - d_A e_i - d_B &= 0 \\ \delta'_{2,i} - f_P e_i^2 - f_A e_i - f_B &= 0 \end{aligned}$$

and, equivalently,

$$r'_i = a_P e_i + a_A + a_B e_i^{-1} \tag{28}$$
$$s'_i = b_P e_i + b_A + b_B e_i^{-1} \tag{29}$$
$$r'_i \odot_y s'_i = c_P e_i^2 + c_A e_i + c_B \tag{30}$$
$$\begin{aligned} \delta'_{1,i} &= d_P e_i^2 + d_A e_i + d_B \\ \delta'_{2,i} &= f_P e_i^2 + f_A e_i + f_B \end{aligned}$$

By eliminating r'_i and s'_i from Eq. (28), Eq. (29), and Eq. (30), we have for $i \in \{1, \ldots, 5\}$

$$\begin{aligned} & a_P \odot_y b_P \cdot e_i^2 + (a_P \odot_y b_A + b_P \odot_y a_A) \cdot e_i \\ & + (a_P \odot_y b_B + b_P \odot_y a_B + a_A \odot_y b_A) + (a_A \odot_y b_B + b_A \odot_y a_B) \cdot e_i^{-1} \\ & + a_B \odot_y b_B \cdot e_i^{-2} \\ & = c_P e_i^2 + c_A e_i + c_B \quad \in \mathbb{Z}_p \end{aligned} \tag{31}$$

This equation can be considered as an inner-product between $(e_i^2, e_i, 1, e_i^{-1}, e_i^{-2})$ and a constants vector. Since Eq. (31) holds for all five distinct challenges $e_i \in \{e_1, \ldots, e_5\}$ and each $(e_i^2, e_i, 1, e_i^{-1}, e_i^{-2})$ tuple is linearly independent, each coefficient in the left hand side of Eq. (31) must be equal to the corresponding coefficient in the right hand side of Eq. (31). As we intended, the extractor either extracts a witness (a_P, b_P) satisfying $a_P \odot_y b_P = c_P$, or a discrete logarithm relation between the generators.

Next, we move to the case $n > 1$. We prove the case $n > 1$ recursively. That is, we construct an extractor χ_{2k} for the case $n = 2k$ using an extractor χ_k and let χ_1 be the extractor χ we constructed for the case $n = 1$. We start with input $(\boldsymbol{g}, \boldsymbol{h}, g, h_2, h_2, P)$ for the case $n = 2k$. Assume that we have the extractor χ_k for the case $n = k$. The extractor χ_{2k} runs the prover to get L and R. At this point, the extractor χ_{2k} rewinds the oracle three times, using four distinct challenges e_i for $i = 1, \ldots, 4$, and sets

$$\widehat{\boldsymbol{g}}_i = \boldsymbol{g}_1^{e_i^{-1}} \circ \boldsymbol{g}_2^{e_i \cdot y^{-k}}, \quad \widehat{\boldsymbol{h}}_i = \boldsymbol{h}_1^{e_i} \circ \boldsymbol{h}_2^{e^{-1}}, \quad \widehat{P}_i = L^{e_i^2} P R^{e_i^{-2}} \quad \in \mathbb{G} \text{ for } i = 1, \ldots, 4$$

Then, for each i, it feeds $(\widehat{\boldsymbol{g}}_i, \widehat{\boldsymbol{h}}_i, g, h_1, h_2, \widehat{P}_i)$ to χ_k and obtain the corresponding witness $\widehat{\boldsymbol{a}}_i, \widehat{\boldsymbol{b}}_i$, $\widehat{\alpha}_{1,i}$, and $\widehat{\alpha}_{2,i}$ that satisfy

$$L^{e_i^2} P R^{e_i^{-2}} = (\boldsymbol{g}^{e_i^{-1}} \circ \boldsymbol{g}_2^{e_i \cdot y^{-k}})^{\widehat{\boldsymbol{a}}_i} (\boldsymbol{h}_1^{e_i} \circ \boldsymbol{h}_2^{e_i^{-1}})^{\widehat{\boldsymbol{b}}_i} g^{\widehat{\boldsymbol{a}}_i \odot_y \widehat{\boldsymbol{b}}_i} h_1^{\widehat{\alpha}_{1,i}} h_2^{\widehat{\alpha}_{2,i}}, \quad i \in [1, 4] \tag{32}$$

For the first three challenges e_1, e_2, e_3, the tuples $(e_i^2, 1, e_i^{-2})$ are linearly independent and so compose a 3×3 invertible matrix in $\mathbb{Z}_p^{3 \times 3}$. We can see that all exponents are constants known to the extractor. Thus, by applying elementary linear algebra to the public exponent of the first three equations of Eq. (32), we can find the exponents $\boldsymbol{a}_P, \boldsymbol{b}_P, c_P, d_P, f_P, \boldsymbol{a}_L, \boldsymbol{b}_L, c_L, d_L, f_L, \boldsymbol{a}_R, \boldsymbol{b}_R, c_R, d_R, f_R$ satisfying

$$\begin{aligned} P &= \boldsymbol{g}^{\boldsymbol{a}_P} \boldsymbol{h}^{\boldsymbol{b}_P} g^{c_P} h_1^{d_P} h_2^{f_P} \quad \in \mathbb{G}, \\ L &= \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{b}_L} g^{c_L} h_1^{d_L} h_2^{f_L} \quad \in \mathbb{G}, \\ R &= \boldsymbol{g}^{\boldsymbol{a}_R} \boldsymbol{h}^{\boldsymbol{b}_R} g^{c_R} h_1^{d_R} h_2^{f_R} \quad \in \mathbb{G}. \end{aligned}$$

Now we prove that those exponents satisfy the desired relation $c_P = \boldsymbol{a}_P \odot_y \boldsymbol{b}_P$. Putting the above representations of P, L, R into Eq. (32) for each i, we have the following equations with bases $\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2$ under the discrete logarithm relation assumption.

$$\boldsymbol{g}^{\boldsymbol{a}_L e_i^2} \boldsymbol{g}^{\boldsymbol{a}_P} \boldsymbol{g}^{\boldsymbol{a}_R e_i^{-2}} = (\boldsymbol{g}_1^{e_i^{-1}} \circ \boldsymbol{g}_2^{e_i \cdot y^{-k}})^{\widehat{\boldsymbol{a}}_i} \tag{33}$$

$$\boldsymbol{h}^{\boldsymbol{b}_L e_i^2} \boldsymbol{h}^{\boldsymbol{b}_P} \boldsymbol{h}^{\boldsymbol{b}_R e_i^{-2}} = (\boldsymbol{h}_1^{e_i} \circ \boldsymbol{h}_2^{e_i^{-1}})^{\widehat{\boldsymbol{b}}_i} \tag{34}$$

$$g^{c_L e_i^2} g^{c_P} g^{c_R e_i^{-2}} = g^{\widehat{\boldsymbol{a}}_i \odot_y \widehat{\boldsymbol{b}}_i} \tag{35}$$

$$h_1^{d_L e_i^2} h_1^{d_P} h_1^{d_R e_i^{-2}} = h_1^{\widehat{\alpha}_{1,i}}$$

$$h_2^{f_L e_i^2} h_2^{f_P} h_2^{f_R e_i^{-2}} = h_2^{\widehat{\alpha}_{2,i}}$$

That is, Eq. (32) is separated into the above five equations according to the bases $\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2$. If we find exponents satisfying Eq. (32) but not the above five equations, it directly implies a non-trivial relation between the generators which breaks the discrete logarithm assumption. We use the above five equations to prove $\boldsymbol{a}_P \odot_y \boldsymbol{b}_P = c_P$. To this end, we first find a relation between $\widehat{\boldsymbol{a}}_i$ and $\boldsymbol{a}_P$ with Eq. (33), second find another relation between $\widehat{\boldsymbol{b}}_i$ and $\boldsymbol{b}_P$ with Eq. (34), and then finally use Eq. (35) to show the desired relation between $c_P, \boldsymbol{a}_P$, and $\boldsymbol{b}_P$.

First, we show the relation between $\widehat{\boldsymbol{a}}_i$ and $\boldsymbol{a}_P$ with Eq. (33). By the discrete logarithm assumption, it is infeasible to find a relation between $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, so Eq. (33) can be interpreted as two equations on bases $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, as follows.

$$\begin{aligned} \boldsymbol{a}_{L,1} e_i^2 + \boldsymbol{a}_{P,1} + \boldsymbol{a}_{R,1} e_i^{-2} &= e_i^{-1} \widehat{\boldsymbol{a}}_i \\ \boldsymbol{a}_{L,2} e_i^2 + \boldsymbol{a}_{P,2} + \boldsymbol{a}_{R,2} e_i^{-2} &= y^{-k} e_i \widehat{\boldsymbol{a}}_i \ , \end{aligned}$$

where $\boldsymbol{a}_P = (\boldsymbol{a}_{P,1}, \boldsymbol{a}_{P,2})$, $\boldsymbol{a}_L = (\boldsymbol{a}_{L,1}, \boldsymbol{a}_{L,2})$, $\boldsymbol{a}_R = (\boldsymbol{a}_{R,1}, \boldsymbol{a}_{R,2}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p^k$. By eliminating $\widehat{\boldsymbol{a}}_i$ from the above two equations, we obtain

$$\boldsymbol{a}_{L,1} e_i^3 + \boldsymbol{a}_{P,1} e_i + \boldsymbol{a}_{R,1} e_i^{-1} = \boldsymbol{a}_{L,2} y^k e_i + \boldsymbol{a}_{P,2} y^k e_i^{-1} + \boldsymbol{a}_{R,2} y^k e_i^{-3} \tag{36}$$

Eq. (36) holds for all four challenges $e_1, \ldots, e_4$ and there are four variable terms $e_i, e_i^3, e_i^{-1}, e_i^{-3}$. This implies that the following must hold.

$$\begin{aligned} \boldsymbol{a}_{L,1} &= 0 && \in \mathbb{Z}_p^k \\ \boldsymbol{a}_{P,1} &= \boldsymbol{a}_{L,2} y^k && \in \mathbb{Z}_p^k \\ \boldsymbol{a}_{R,1} &= \boldsymbol{a}_{P,2} y^k && \in \mathbb{Z}_p^k \\ \boldsymbol{a}_{R,2} &= 0 && \in \mathbb{Z}_p^k \end{aligned}$$

Using the above result with Eq. (33), we see that the exponent of the base $\boldsymbol{g}_1$ in Eq. (33) is

$$\boldsymbol{a}_{P,1} + \boldsymbol{a}_{P,2} y^k e_i^{-2} = e_i^{-1} \widehat{\boldsymbol{a}}_i \ ,$$

which means we have a relation between $\widehat{\boldsymbol{a}}_i$ and $\boldsymbol{a}_P$,

$$\widehat{\boldsymbol{a}}_i = \boldsymbol{a}_{P,1} e_i + \boldsymbol{a}_{P,2} y^k e_i^{-1} \tag{37}$$

Second, we show the relation between $\widehat{\boldsymbol{b}}_i$ and $\boldsymbol{b}_P$ with Eq. (34). Like with Eq. (33), we use the discrete logarithm assumption to extract the exponents of the bases $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ from Eq. (34).

$$\boldsymbol{b}_{L,1}e_i^2 + \boldsymbol{b}_{P,1} + \boldsymbol{b}_{R,1}e_i^{-2} = e_i\widehat{\boldsymbol{b}}_i$$
$$\boldsymbol{b}_{L,2}e_i^2 + \boldsymbol{b}_{P,2} + \boldsymbol{b}_{R,2}e_i^{-2} = e_i^{-1}\widehat{\boldsymbol{b}}_i$$

where $\boldsymbol{b}_P = (\boldsymbol{b}_{P,1}, \boldsymbol{b}_{P,2}), \boldsymbol{b}_L = (\boldsymbol{b}_{L,1}, \boldsymbol{b}_{L,2}), \boldsymbol{b}_R = (\boldsymbol{b}_{R,1}, \boldsymbol{b}_{R,2}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p^k$. By eliminating $\widehat{\boldsymbol{b}}_i$ from the above two equations, we obtain

$$\boldsymbol{b}_{L,1} \cdot e_i + \boldsymbol{b}_{P,1} \cdot e_i^{-1} + \boldsymbol{b}_{R,1} \cdot e_i^{-3} = \boldsymbol{b}_{L,2}y^k \cdot e_i^3 + \boldsymbol{b}_{P,2} \cdot e_i + \boldsymbol{b}_{R,2} \cdot e_i^{-1} \tag{38}$$

Eq. (38) holds for all four challenges $e_1, \ldots e_4$ and there are four variable terms $e_i, e_i^3, e_i^{-1}, e_i^{-3}$. This implies that the following must hold.

$$\begin{aligned}
\boldsymbol{b}_{L,1} &= \boldsymbol{b}_{P,2} && \in \mathbb{Z}_p^k \\
\boldsymbol{b}_{P,1} &= \boldsymbol{b}_{R,2} && \in \mathbb{Z}_p^k \\
\boldsymbol{b}_{R,1} &= 0 && \in \mathbb{Z}_p^k \\
\boldsymbol{b}_{L,2} &= 0 && \in \mathbb{Z}_p^k
\end{aligned}$$

Using the above result with Eq. (34), we see that the exponent of the base $\boldsymbol{h}_1$ in Eq. (34) is

$$\boldsymbol{b}_{P,2}e_i^2 + \boldsymbol{b}_{P,1} = e_i\widehat{\boldsymbol{b}}_i\,,$$

which means we have a relation between $\widehat{\boldsymbol{b}}_i$ and $\boldsymbol{b}_P$,

$$\widehat{\boldsymbol{b}}_i = \boldsymbol{b}_{P,2}e_i + \boldsymbol{b}_{P,1}e_i^{-1} \tag{39}$$

Finally, we use Eq. (35) to show the relation between $c_P, \boldsymbol{a}_P$, and $\boldsymbol{b}_P$. Taking the WIP $\odot_y$ on Eq. (37) and Eq. (39), we have

$$\begin{aligned}
&\widehat{\boldsymbol{a}}_i \odot_y \widehat{\boldsymbol{b}}_i \\
&= (\boldsymbol{a}_{P,1} \odot_y \boldsymbol{b}_{P,2})e_i^2 + (\boldsymbol{a}_{P,1} \odot_y \boldsymbol{b}_{P,1} + \boldsymbol{a}_{P,2} \odot_y \boldsymbol{b}_{P,2} \cdot y^k) \\
&\quad + (\boldsymbol{a}_{P,2} \odot_y \boldsymbol{b}_{P,1} \cdot y^k)e_i^{-2}
\end{aligned}$$

Combining this result with Eq. (35), we have

$$\left(\boldsymbol{a}_{P,1} \odot_y \boldsymbol{b}_{P,2} - c_L,\ \boldsymbol{a}_{P,1} \odot_y \boldsymbol{b}_{P,1} + \boldsymbol{a}_{P,2} \odot_y \boldsymbol{b}_{P,2}y^k - c_P,\ \boldsymbol{a}_{P,2} \odot_y \boldsymbol{b}_{P,1}y^k - c_R\right) \cdot \begin{pmatrix} e_i^2 \\ 1 \\ e_i^{-2} \end{pmatrix} = 0$$

The above equation holds for the first three distinct challenges $e_1, \ldots, e_3$. Since the vectors $(e_i^2, 1, e_i^{-2})$'s are linearly independent, this implies that the following must hold.

$$\boldsymbol{a}_P \odot_y \boldsymbol{b}_P = \boldsymbol{a}_{P,1} \odot_y \boldsymbol{b}_{P,1} + \boldsymbol{a}_{P,2} \odot_y \boldsymbol{b}_{P,2} \cdot y^k = c_P$$

For each recursive step, the extractor χ_{2k} uses 4 transcripts and χ_1 uses 5 transcripts, so that the final extractor χ_n uses $5 \cdot 4^{\log_2(n)}$ transcripts in total and this runs in expected polynomial time in λ since n is polynomial in λ. Then, by the general forking lemma, we conclude that the proposed WIP argument system has computational witness extended emulation.

□

D.3 Zero Knowledge Arguments for Range Proof and Aggregate Range Proof

A range proof protocol with support for double-blinded commitments is shown on Fig. D.2 and a corresponding aggregated range proof protocol is shown on Fig. D.3. We provide the security statement for the proposed extended protocols in Theorem 6. The proof of Theorem 6 mostly corresponds to the original proof in the Bulletproofs+ paper[3]. The WEE proof for Theorem 6 originally has a mistake, that was found and discussed in [2]. We provide a fixed WEE proof with the exact equations. For the reader's convenience all changes are highlighted.

As soon as Theorem 5 can be shown to be a particular case of Theorem 6, we, like in the original paper, omit the proof of Theorem 5.

Original Bulletproofs+	Extended Bulletproofs+
Relation $\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h, V \in \mathbb{G}; v, \gamma \in \mathbb{Z}_p) :$ $V = g^v h^\gamma \bigwedge v \in [0, 2^n - 1]\}$	Relation $\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h_1, h_2, V \in \mathbb{G}; v, \gamma_1, \gamma_2 \in \mathbb{Z}_p) :$ $V = g^v h_1^{\gamma_1} h_2^{\gamma_2} \bigwedge v \in [0, 2^n - 1]\}$
$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, V; v, \gamma)$ $\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, V)$	$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, V; v, \gamma_1, \gamma_2)$ $\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, V)$
$\mathcal{P}$'s output: none $\mathcal{V}$'s output: *Accept* or *Reject*	
$\boxed{\mathcal{P}}$: $\alpha \xleftarrow{\$} \mathbb{Z}_p$, sets $\boldsymbol{a}_L \in \{0,1\}^n$ such that $\langle \boldsymbol{a}_L, \boldsymbol{2}^n \rangle = v$ and $\boldsymbol{a}_R = \boldsymbol{a}_L - \boldsymbol{1}^n \in \mathbb{Z}_p^n$, and computes $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h^\alpha \in \mathbb{G}$.	$\boxed{\mathcal{P}}$: $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p$, sets $\boldsymbol{a}_L \in \{0,1\}^n$ such that $\langle \boldsymbol{a}_L, \boldsymbol{2}^n \rangle = v$ and $\boldsymbol{a}_R = \boldsymbol{a}_L - \boldsymbol{1}^n \in \mathbb{Z}_p^n$, and computes $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h_1^{\alpha_1} h_2^{\alpha_2} \in \mathbb{G}$.
$\boxed{\mathcal{P} \to \mathcal{V}}$: A $\boxed{\mathcal{V}}$: $y, z \xleftarrow{\$} \mathbb{Z}_p$ $\boxed{\mathcal{V} \to \mathcal{P}}$: y, z $\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: compute $\widehat{A} = A \cdot \boldsymbol{g}^{-\boldsymbol{1}^n \cdot z} \cdot \boldsymbol{h}^{\boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z} \cdot V^{y^{n+1}}$ $\cdot g^{\langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle \cdot z - \langle \boldsymbol{1}^n, \boldsymbol{2}^n \rangle \cdot y^{n+1} z - \langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle \cdot z^2} \in \mathbb{G}$ $\boxed{\mathcal{P}}$: computes $\widehat{\boldsymbol{a}}_L = \boldsymbol{a}_L - \boldsymbol{1}^n \cdot z \in \mathbb{Z}_p^n$ $\widehat{\boldsymbol{a}}_R = \boldsymbol{a}_R + \boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z \in \mathbb{Z}_p^n$	
$\widehat{\alpha} = \alpha + \gamma \cdot y^{n+1} \in \mathbb{Z}_p$	$\widehat{\alpha}_1 = \alpha_1 + \gamma_1 \cdot y^{n+1} \in \mathbb{Z}_p$ $\widehat{\alpha}_2 = \alpha_2 + \gamma_2 \cdot y^{n+1} \in \mathbb{Z}_p$
$\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: run zk-WIP$_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$	$\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: run zk-WIP$_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha}_1, \widehat{\alpha}_2)$

Fig. D.2. Zero Knowledge Argument for Range Proof $v \in [2^n - 1]$

Original Bulletproofs+	Extended Bulletproofs+
Relation $\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^{m \cdot n}, g, h \in \mathbb{G}, \boldsymbol{V} \in \mathbb{G}^m; \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m) :$ $V_j = g^{v_j} h^{\gamma_j} \bigwedge v_j \in [0, 2^n - 1]$ for $j \in [1, m]\}$	Relation $\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^{m \cdot n}, g, h_1, h_2 \in \mathbb{G}, \boldsymbol{V} \in \mathbb{G}^m;$ $\boldsymbol{v}, \boldsymbol{\gamma_1}, \boldsymbol{\gamma_2} \in \mathbb{Z}_p^m) : V_j = g^{v_j} h_1^{\gamma_{1,j}} h_2^{\gamma_{2,j}}$ $\bigwedge v_j \in [0, 2^n - 1]$ for $j \in [1, m]\}$
$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, \boldsymbol{V}; \boldsymbol{v}, \boldsymbol{\gamma})$ $\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, \boldsymbol{V})$	$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, \boldsymbol{V}; \boldsymbol{v}, \boldsymbol{\gamma_1}, \boldsymbol{\gamma_2})$ $\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, \boldsymbol{V})$
$\mathcal{P}$'s output: none $\mathcal{V}$'s output: *Accept* or *Reject*	
$\boxed{\mathcal{P}}$: for $j \in [1, m]$, let $\boldsymbol{d}_j := (\underbrace{0, \ldots, 0}_{(j-1) \cdot n}, \boldsymbol{2}^n, \underbrace{0, \ldots, 0}_{(m-j) \cdot n})$, sets $\boldsymbol{a}_L \in \{0,1\}^{m \cdot n}$ such that $\langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j$ and $\boldsymbol{a}_R = \boldsymbol{a}_L - \boldsymbol{1}^{mn} \in \mathbb{Z}_p^{mn}$	
$\boxed{\mathcal{P}}$: chooses $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and computes $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h^{\alpha} \in \mathbb{G}$	$\boxed{\mathcal{P}}$: chooses $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p$, and computes $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h_1^{\alpha_1} h_2^{\alpha_2} \in \mathbb{G}$
$\boxed{\mathcal{P} \to \mathcal{V}}$: A $\boxed{\mathcal{V}}$: $y, z \xleftarrow{\$} \mathbb{Z}_p^*$ $\boxed{\mathcal{V} \to \mathcal{P}}$: y, z $\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: let $\boldsymbol{d} := \sum_{j=1}^{m} z^{2j} \cdot \boldsymbol{d}_j$, and compute $\widehat{A} = A \cdot \boldsymbol{g}^{-\boldsymbol{1}^{mn} \cdot z} \cdot \boldsymbol{h}^{\boldsymbol{d} \circ \overleftarrow{y}^{mn} + \boldsymbol{1}^{mn} \cdot z} \cdot \left(\prod_{j=1}^{m} V_j^{z^{2j}} \right)^{y^{mn+1}}$ $\cdot g^{\langle \boldsymbol{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z - \langle \boldsymbol{1}^{mn}, \boldsymbol{d} \rangle \cdot y^{mn+1} z - \langle \boldsymbol{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2} \in \mathbb{G}$ $\boxed{\mathcal{P}}$: computes $\widehat{\boldsymbol{a}}_L = \boldsymbol{a}_L - \boldsymbol{1}^{mn} \cdot z \in \mathbb{Z}_p^{mn}$ $\widehat{\boldsymbol{a}}_R = \boldsymbol{a}_R + \boldsymbol{d} \circ \overleftarrow{y}^{mn} + \boldsymbol{1}^{mn} \cdot z \in \mathbb{Z}_p^{mn}$	
$\widehat{\alpha} = \alpha + \sum_{j=1}^{m} z^{2j} \cdot \gamma_j \cdot y^{mn+1} \in \mathbb{Z}_p$	$\widehat{\alpha}_1 = \alpha_1 + \sum_{j=1}^{m} z^{2j} \cdot \gamma_{1,j} \cdot y^{mn+1} \in \mathbb{Z}_p$ $\widehat{\alpha}_2 = \alpha_2 + \sum_{j=1}^{m} z^{2j} \cdot \gamma_{2,j} \cdot y^{mn+1} \in \mathbb{Z}_p$
$\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: run zk-WIP$_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$	$\boxed{\mathcal{P} \text{ and } \mathcal{V}}$: run zk-WIP$_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha}_1, \widehat{\alpha}_2)$

Fig. D.3. Zero Knowledge Argument for Aggregate Range Proof $v_j \in [2^n - 1]$ for $j \in [1; m]$

Theorem 5. *The zero-knowledge argument for range proof presented in Figure D.2 has perfect completeness, perfect honest verifier zero-knowledge, and computational witness extended emulation.*

Theorem 6. *The zero-knowledge argument for range proof presented in Figure D.3 has perfect completeness, perfect honest verifier zero-knowledge, and computational witness extended emulation.*

Proof. (perfect completeness) Since the proposed range proof argument runs the WIP argument as a subprotocol, we prove perfect completeness by showing that if $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h_1^{\alpha_1} h_2^{\alpha_2}$ is correctly

computed, then $\widehat{A}$ satisfies the WIP relation such that

$$\widehat{A} = \boldsymbol{g}^{\widehat{\boldsymbol{a}}_L} \boldsymbol{h}^{\widehat{\boldsymbol{a}}_R} g^{\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_R} h_1^{\widehat{\alpha}_1} h_2^{\widehat{\alpha}_2} \in \mathbb{G} \tag{40}$$

From the computation

$$\widehat{A} = A \cdot \boldsymbol{g}^{-\mathbf{1}^{mn} \cdot z} \boldsymbol{h}^{\boldsymbol{d} \circ \overleftarrow{y}^{mn} + \mathbf{1}^{mn} \cdot z} \left(\prod_{j=1}^{m} V_j^{z^{2j}} \right)^{y^{mn+1}} g^{\langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z - \langle \mathbf{1}^{mn}, \boldsymbol{d} \rangle \cdot y^{mn+1} z - \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2}$$

one can easily check that the exponents of bases $\boldsymbol{g}, \boldsymbol{h}$, h_1, and h_2 are equal to $\widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_L$, $\widehat{\alpha}_1$ and $\widehat{\alpha}_2$ respectively. The only thing that remains is to confirm whether the exponent of the base g is equal to $\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_L$. This can be checked as follows.

$$\begin{aligned}
&\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_R \\
&= (\boldsymbol{a}_L - \mathbf{1}^{mn} \cdot z) \odot_y (\boldsymbol{a}_R + \boldsymbol{d} \circ \overleftarrow{y}^{mn} + \mathbf{1}^{mn} \cdot z) \\
&= \boldsymbol{a}_L \odot_y \boldsymbol{a}_R + \boldsymbol{a}_L \odot_y (\boldsymbol{d} \circ \overleftarrow{y}^{mn}) + \boldsymbol{a}_L \odot_y \mathbf{1}^{mn} \cdot z - \mathbf{1}^{mn} \odot_y \boldsymbol{a}_R \cdot z \\
&\qquad - \mathbf{1}^{mn} \odot_y (\boldsymbol{d} \circ \overleftarrow{y}^{mn}) \cdot z - \mathbf{1}^{mn} \odot_y \mathbf{1}^{mn} \cdot z^2 \\
&= \langle \boldsymbol{a}_L, \boldsymbol{d} \rangle \cdot y^{mn+1} + \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z - \langle \mathbf{1}^{mn}, \boldsymbol{d} \rangle \cdot z \cdot y^{mn+1} - \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2
\end{aligned} \tag{41}$$

The first component in the far right hand side in Eq. (41) is equal to

$$\langle \boldsymbol{a}_L, \sum_{j=1}^{m} z^{2j} \cdot \boldsymbol{d}_j \rangle \cdot y^{mn+1} = \sum_{j=1}^{m} z^{2j} \cdot \langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle \cdot y^{mn+1} = \sum_{j=1}^{m} z^{2j} \cdot v_j \cdot y^{mn+1}$$

which is the g-base exponent of $(\prod_{j=1}^{m} V_j^{z^{2j}})^{y^{mn+1}}$. This means Eq. (41) is exactly equal to the g-base exponent of $\widehat{A}$, so Eq. (40) holds. Since Eq. (40) has the format of a bilinear argument, we can utilize the perfect completeness of the WIP argument, which lets us conclude that the proposed aggregatable range proof argument has perfect completeness.

(perfect SHVZK) For perfect special honest verifier zero-knowledge, we construct a simulator. The simulator samples $A \xleftarrow{\$} \mathbb{G}$ and sets the input of the WIP argument according to the description of the range protocol. Then, the simulator runs the simulator from the perfect SHVZK proof in the WIP argument as a subalgorithm. In the real transcript, A is uniformly distributed due to the blinding factors α_1, α_2, and A, along with y and z, contributes to define the input of the bilinear argument. In the simulated transcript, A is also uniformly generated, and for any fixed input of the WIP argument we have already proved that the simulator for the perfect SHVZK proof in the WIP argument can perfectly simulate the real transcript. Therefore, we conclude that the simulated transcript's distribution is identical to that of the real transcript.

(witness-extended emulation) We prove that the proposed protocol has witness-extended emulation. To this end, we construct the extractor χ_R that extracts a witness of the range proof argument by using $3n + 3$ distinct y challenges and $2m + 2$ distinct z challenges. In the proof, we first explain how the extractor extracts openings v_j of the Pedersen commitments and next prove that it satisfies the desired relation $v_j \in [0, 2^n - 1]$.

In order to extract openings of the Pedersen commitments, the extractor uses another extractor χ_B for the bilinear arguments whose existence is already proved in the proof of Theorem 4. More precisely, after fixing the first message A sent by the prover, χ_R rewinds the prover by using one

y challenge and $m+1$ distinct z challenges, computes the corresponding $\widehat{A}$, calls χ_B by giving $(\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2, \widehat{A})$ as input, and then obtains the corresponding values $\widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R$, $\widehat{\alpha}_1$ and $\widehat{\alpha}_2$ such that

$$\begin{aligned} & \boldsymbol{g}^{\widehat{\boldsymbol{a}}_L} \boldsymbol{h}^{\widehat{\boldsymbol{a}}_R} g^{\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_R} h_1^{\widehat{\alpha}_1} h_2^{\widehat{\alpha}_2} \\ = \; & A \cdot \boldsymbol{g}^{-\mathbf{1}^{mn} \cdot z} \boldsymbol{h}^{\boldsymbol{d} \circ \overleftarrow{y}^{mn} + \mathbf{1}^{mn} \cdot z} \left(\prod_{j=1}^{m} V_j^{z^{2j}} \right)^{y^{mn+1}} g^{\langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z - \langle \mathbf{1}^{mn}, \boldsymbol{d} \rangle \cdot y^{mn+1} z - \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2} \end{aligned} \tag{42}$$

Here, χ_R knows all exponents in Eq. (42), so we can efficiently perform any linear operations among the exponents in Eq. (42). The tuple of exponents corresponding to A and each V_j in Eq. (42) is the $(m+1)$-dimensional vector $(1, y^{mn+1} z^2, \ldots, y^{mn+1} z^{2m})$. We know that a set of vectors $(1, z^2, ..., z^{2m})$ for $m+1$ distinct z's is linearly independent since it composes the Vandermonde matrix with distinct rows. By multiplying the inverse of such a Vandermonde matrix to the exponents in Eq. (42), the extractor can compute the decompositions of A and each $V_j^{y^{mn+1}}$ with respect to bases $\boldsymbol{g}, \boldsymbol{h}, g$, h_1 and h_2, and so do V_j for $j = 1, \ldots, m$ by exponentiation with y^{-mn-1}. Therefore, now the extractor χ_R has values $\boldsymbol{a}_L, \boldsymbol{a}_R, \alpha, \beta, \boldsymbol{v}_{L,j}, \boldsymbol{v}_{R,j}, v_j$, $\gamma_{1,j}$ and $\gamma_{2,j}$ such that $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} g^{\beta} h_1^{\alpha_1} h_2^{\alpha_2}$ and $V_j = \boldsymbol{g}^{\boldsymbol{v}_{L,j}} \boldsymbol{h}^{\boldsymbol{v}_{R,j}} g^{v_j} h_1^{\gamma_{1,j}} h_2^{\gamma_{2,j}}$ for $j = 1, \ldots, m$. In particular, we successfully extracted v_j for $j = 1, \ldots, m$ openings of the Pedersen commitments.

Next, we show that the extracted values $\boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{v}_L, \boldsymbol{v}_R$, and v_j for $j = 1, \ldots, m$ satisfy the desired relations $v_j \in [0, 2^n - 1]$ and $\boldsymbol{v}_{L,j} = \boldsymbol{v}_{R,j} = \mathbf{0}$ for $j = 1, \ldots, m$. To this end, it is sufficient to show that these values satisfy the five equations $\boldsymbol{a}_R = \boldsymbol{a}_L - \mathbf{1}^{mn}$, $\boldsymbol{a}_L \circ \boldsymbol{a}_R = \mathbf{0}$, $\langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j$, $\boldsymbol{v}_{L,j} \circ \boldsymbol{v}_{R,j} = \mathbf{0}$, and $\boldsymbol{v}_{L,j} + \boldsymbol{v}_{R,j} = \mathbf{0}$ for $j \in [1, m]$. First, we consider Eq. (42). Although $\widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha}_1, \widehat{\alpha}_2$ are computed by using only the $m+1$ challenges z_j, these are fixed exponents for the commitments A and public V_j for $j = 1, \ldots, m$, regardless of challenges. That is, if we find a challenge pair that does not satisfy Eq. (42), then we directly obtain a non-trivial discrete logarithm relation between the $2mn+3$ generators $\boldsymbol{g}, \boldsymbol{h}, g, h_1, h_2$ of $\mathbb{G}$. If Eq. (42) holds for all challenge pairs (y, z), again under the discrete logarithm relation assumption, we can change Eq. (42) with the following five equations according to the bases $\boldsymbol{g}, \boldsymbol{h}, g$, h_1 and h_2.

$$\widehat{\boldsymbol{\alpha}}_L = \boldsymbol{a}_L - \mathbf{1}^{mn} \cdot z + y^{mn+1} \cdot \boldsymbol{u}_L \quad \in \mathbb{Z}_p^{mn}, \text{ where } \boldsymbol{u}_L = \sum_{j=1}^{m} z^{2j} \cdot \boldsymbol{v}_{L,j}$$

$$\widehat{\boldsymbol{\alpha}}_R = \boldsymbol{a}_R + \boldsymbol{d} \circ \overleftarrow{y}^{mn} + \mathbf{1}^{mn} \cdot z + y^{mn+1} \cdot \boldsymbol{u}_R \; \in \mathbb{Z}_p^{mn}, \text{ where } \boldsymbol{d} = \sum_{j=1}^{m} z^{2j} \cdot \boldsymbol{d}_j, \; \boldsymbol{u}_R = \sum_{j=1}^{m} z^{2j} \cdot \boldsymbol{v}_{R,j}$$

$$\widehat{\boldsymbol{\alpha}}_L \odot_y \widehat{\boldsymbol{\alpha}}_R = \beta + \sum_{j=1}^{m} v_j \cdot z^{2j} \cdot y^{mn+1} + \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z - \langle \mathbf{1}^{mn}, \boldsymbol{d} \rangle \cdot y^{mn+1} z - \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2 \in \mathbb{Z}_p$$

$$\widehat{\alpha}_1 = \alpha_1 + \sum_{j=1}^{m} \gamma_{1,j} \cdot z^{2j} \cdot y^{mn+1} \quad \in \mathbb{Z}_p$$

$$\widehat{\alpha}_2 = \alpha_2 + \sum_{j=1}^{m} \gamma_{2,j} \cdot z^{2j} \cdot y^{mn+1} \quad \in \mathbb{Z}_p$$

The right hand sides of the above equations can be considered as polynomials in variables y and z. By eliminating $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ in the left hand sides of the first three equations, we obtain the following

equation in y and z.

$$
\begin{aligned}
&\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_R \\
&= \underbrace{\boldsymbol{a}_L \odot_y \boldsymbol{a}_R}_{y,\ldots,y^{mn}\ \text{terms}} + \underbrace{y^{mn+1} \cdot \langle \boldsymbol{a}_L, \boldsymbol{d} \rangle}_{y^{mn+1}z^2,\ldots,y^{mn+1}z^{2m}\ \text{terms}} + \underbrace{z \cdot \boldsymbol{a}_L \odot_y \mathbf{1}^{mn}}_{yz,\ldots,y^{mn}z\ \text{terms}} + \underbrace{y^{mn+1} \cdot \boldsymbol{a}_L \odot_y \boldsymbol{u}_R}_{y^{mn+2}z^2,\ldots,y^{2mn+1}z^{2m}\ \text{terms}} \\
&- \underbrace{z \cdot \mathbf{1}^{mn} \odot_y \boldsymbol{a}_R}_{yz,\ldots,y^{mn}z\ \text{terms}} - \underbrace{y^{mn+1} z \cdot \langle \mathbf{1}^{mn}, \boldsymbol{d} \rangle}_{y^{mn+1}z^{2+1},\ldots,y^{mn+1}z^{2m+1}\ \text{terms}} - \underbrace{z^2 \cdot \mathbf{1}^{mn} \odot_y \mathbf{1}^{mn}}_{yz^2,\ldots,y^{mn}z^2\ \text{terms}} - \underbrace{y^{mn+1} z \cdot \mathbf{1}^{mn} \odot_y \boldsymbol{u}_R}_{y^{mn+2}z^3,\ldots,y^{2mn+1}z^{2m+1}\ \text{terms}} \\
&+ \underbrace{y^{mn+1} \cdot \boldsymbol{u}_L \odot_y \boldsymbol{a}_R}_{y^{mn+2}z^2,\ldots,y^{2mn+1}z^{2m}\ \text{terms}} + \underbrace{y^{2mn+2} \cdot \langle \boldsymbol{u}_L, \boldsymbol{d} \rangle}_{y^{2mn+2}z^4,\ldots,y^{2mn+2}z^{4m}\ \text{terms}} + \underbrace{y^{mn+1} z \cdot \boldsymbol{u}_L \odot_y \mathbf{1}^{mn}}_{y^{mn+2}z^3,\ldots,y^{2mn+1}z^{2m+1}\ \text{terms}} \\
&+ \underbrace{y^{2mn+2} \cdot \boldsymbol{u}_L \odot_y \boldsymbol{u}_R}_{y^{2mn+3}z^4,\ldots,y^{3mn+2}z^{4m}\ \text{terms}} = \underbrace{\beta}_{\text{constant term}} + \underbrace{\sum_{j=1}^{m} v_j \cdot z^{2j} \cdot y^{mn+1}}_{y^{mn+1}z^2,\ldots,y^{mn+1}z^{2m}\ \text{terms}} + \underbrace{\langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z}_{yz,\ldots,y^{mn}z\ \text{terms}} \\
&\qquad - \underbrace{\langle \mathbf{1}^{mn}, \boldsymbol{d} \rangle \cdot y^{mn+1} z}_{y^{mn+1}z^{2+1},\ldots,y^{mn+1}z^{2m+1}\ \text{terms}} - \underbrace{\langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2}_{yz^2,\ldots,y^{mn}z^2\ \text{terms}}
\end{aligned} \tag{43}
$$

The above equation holds for all y, z challenges, which are $3n+3$ distinct y challenges and $2m+2$ distinct z challenges, and thus the following equations must hold. (See Lemma 2 in Appendix B.2 in the original paper [3] for a rigorous proof for this argument.)

Variables in Eq. (43)	Left Hand Side		Right Hand Side	
$y, \ldots, y^{mn}$	$\boldsymbol{a}_L \odot_y \boldsymbol{a}_R$	$=$	0	$\Rightarrow \boldsymbol{a}_L \circ \boldsymbol{a}_R = \mathbf{0}$
$y^{mn+1}z^2, \ldots, y^{mn+1}z^{2m}$	$\langle \boldsymbol{a}_L, \sum_{j=1}^{m} z^{2j} \boldsymbol{d}_j \rangle y^{mn+1}$	$=$	$\sum_{j=1}^{m} v_j z^{2j} y^{mn+1}$	$\Rightarrow \langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j \forall j$
$yz, \ldots, y^{mn}z$	$z \cdot \boldsymbol{a}_L \odot_y \mathbf{1}^{mn} - z \cdot \mathbf{1}^{mn} \odot_y \boldsymbol{a}_R$	$=$	$\langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle z$	$\Rightarrow \boldsymbol{a}_L - \boldsymbol{a}_R = \mathbf{1}^{mn}$
$y^{2mn+3}z^4, \ldots, y^{3mn+2}z^{4m}$	$y^{2mn+2} \sum_{j=1}^{m} z^{2j} \boldsymbol{v}_{L,j} \odot_y \boldsymbol{v}_{R,j}$	$=$	0	$\Rightarrow \boldsymbol{v}_{L,j} \circ \boldsymbol{v}_{R,j} = \mathbf{0}\ \forall j$
$y^{mn+2}z^3, \ldots, y^{2mn+1}z^{2m+1}$	$y^{mn+1} z \sum_{j=1}^{m} z^{2j} (\boldsymbol{v}_{L,j} - \boldsymbol{v}_{R,j}) \odot_y \mathbf{1}^{mn}$	$=$	0	$\Rightarrow \boldsymbol{v}_{L,j} - \boldsymbol{v}_{R,j} = \mathbf{0}\ \forall j$

Therefore, the extracted values v_j and γ_j satisfy the desired relations $v_j \in [0, 2^n - 1]$ and $V_j = g^{v_j} h_1^{\gamma_{1,j}} h_2^{\gamma_{2,j}}$ for all $j = 1, \ldots, m$.

Finally, the extractor χ_R runs the prover with $3n + 3$ distinct y challenges and $2m + 2$ distinct z challenges and also invokes χ_B on each of the transcripts, so that in total χ_R uses $(3n+3) \cdot (2m+2) \cdot (5 \cdot 4^{\log(n)})$ valid transcripts, which is polynomial in λ since both n and m are polynomial in λ. The extractor χ_R rewinds the prover $(3n + 3) \cdot (2m + 2) \cdot (5 \cdot 4^{\log(n)})$ times, so that it runs in expected polynomial time in λ. Combining the result of the general forking lemma, we conclude that the proposed protocol has witness-extended emulation.

□