A Proof-of-Stake scheme for confidential transactions with hidden amounts

sowle

Zano project
val@zano.org
https://zano.org

July 2021

Abstract

This article explores a way of implementing Proof-of-Stake mining algorithm in an environment where amounts are hidden with homomorphic commitments.

We propose an algorithm which is compatible with such transactions, including transactions with mixed-in decoys.

1 Notation

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

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

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

2 PoS scheme using open amounts

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

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

1. Gets hash identifier of the last PoW block in the blockchain, *last_pow_id*.
2. Gets the last PoS block in the blockchain, and gets stake kernel hash identifier from it, *last_pos_kernel_id*. Together with *last_pow_id* they are called "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:

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

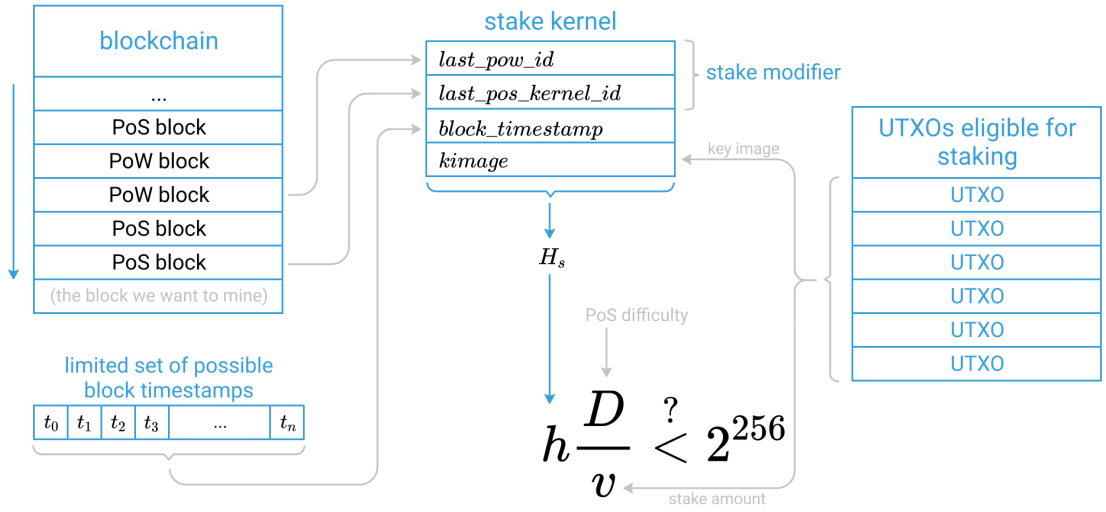

Fig. 2.1. Scheme of the original PoS mining process, as originally implemented in Zano

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 current timestamp established in such a way, that it is the same across all the network 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 precalculate key image I_u.

5. Each pair $(t, u) \in T \times U$ is being checked against PoS winning condition as the following:

   (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 $h_u = H_s(K_u)$;

   (c) Finally, check the main condition:

   $$h_u \frac{D}{v_u} \stackrel{?}{\leq} 2^{256} \tag{1}$$

   where D is the current PoS difficulty, and v_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 stake input, spending output u, could be constructed and broadcasted to the network.

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

- a new block 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 could perform another attempt of mining (items 1-5), as all K_u, and thus h_u, will have different values, giving more chances to win the main condition.

We'd like to note the important property of (1): as h_u is the result of cryptographic hash function H_s and could be considered as distributed evenly on $(0..L)$, the probability of wining the main condition is proportional to output amount v_u.

3 PoS direct spending scheme using hidden amounts

(*Note: this variant of scheme requires stake inputs to directly refer to their outputs without decoys.*)

Consider a hidden amount scheme, where amount v of an output is hidden using Pedersen[1] commitment A:

$$A = vG + fH \quad (v < 2^{64},\ f \neq 0)$$

where G and H are generators for which DL relation is unknown, and f is a random hiding mask.

It's easy to see that (1) can't be used anymore because it's requires using non-hidden v. 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 PoS win condition. She calculates

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

As we mentioned above, h can be considered as uniform randomness distributed evenly over $\mathbb{Z}_l$. Because hiding mask $f \neq 0$ and it is fixed for the selected output u, then the multiplication $hf \pmod l$ can also be considered as uniform randomness over $\mathbb{Z}_l$. TODO: formal proof may be needed

Taken this into account, Alice checks slightly adjusted main inequality:

$$hf < \left\lfloor \frac{l}{D} \right\rfloor v \pmod l \tag{2}$$

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 $\lfloor \frac{d_0}{D} \rfloor$.

Note, that as soon as $D > 2^{64}$ and $v < 2^{64}$, the right side of (2) never overlaps the l.

[1] More information in original paper by T.P. Pedersen: 3.

Now turn inequality into the equality:

$$hf = \left\lfloor \frac{d_0}{D} \right\rfloor v - b_v, \quad D \leq d_0 \leq l,\ b_v < 2^{64} \tag{3}$$

Once (2) holds, Alice needs to calculate d_0 and b_v so that (3) holds. If then she could convince someone of knowing d_0, b_v to be in corresponding ranges, she could also convince, that (2) holds for particular h, and thus, for pair (u, t). Now we construct such a proof in NIZK-manner.

Rewrite (3) slightly:

$$(3) \quad \Leftrightarrow \quad hf - dv + b_v = 0, \quad d = \left\lfloor \frac{d_0}{D} \right\rfloor \tag{4}$$

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

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

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

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

Considering commitments $A' = fG + vH,\ A = vG + fH,\ B = b_vG + b_fH$, we can rewrite (6) in terms of group elements operations:

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

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

3.1 A' proof

As the part of the whole PoS proof, Alice needs to convince a verifier that $A' = fG + vH$ without revealing v and f. As the verifier knows public $A = vG + fH$ we can construct Schnorr-like proof as the following:

1. Alice generates randomnesses $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_l$
2. Calculates $R_0 = r_0(G + H), R_1 = r_1(G - H)$
3. Calculates non-interactive challenge $c = H_s(R_0, R_1, A', A)$
4. Calculates $y_0 = r_0 + c(v + f), y_1 = r_1 + c(v - f)$
5. Sends (c, y_0, y_1) to the verifier.
6. Verifier makes sure that
$$c \stackrel{?}{=} H_s(y_0(G + H) - c(A + A'), y_1(G - H) - c(A - A'), A', A)$$
if the above equation holds, the verifier is convinced that $A + A' = k_0(G + H)$ and $A - A' = k_1(G - H)$ for some k_0 and k_1, and thus due to Lemma 1 he is convinced that $A' = fG + vH$.

3.2 PoS proof

Let's summarize the whole scheme.

1. Alice prepares set of possible timestamps T and staking outputs U.
2. For each pair (t, u) she calculates $h = H_s(\dots)$ and checks win condition (2).
3. If (2) holds she calculates:

$$\begin{aligned} A' &= fG + vH \\ d &= \left\lfloor \frac{hf}{v} \right\rfloor + 1 \\ b_v &= dv - hf \\ b_f &= df - hv \\ B &= b_v G + b_f H \end{aligned}$$

4. Generates proof (c, y) for the fact, that $A' = fG + vH$ (section 3.1).
5. Makes PoS block with stake output u and timestamp t, and adds PoS proof σ to the block's data:

$$\sigma = \{d, A', (c, y_0, y_1), B, \mathcal{R}_B\} \tag{8}$$

where $\mathcal{R}_B$ is a range proof (e.g. Bulletproofs) for the fact, that $b_v < 2^{64}$

Verifiers on the network check PoS block as the following:

1. $d \leq \left\lfloor \frac{l}{D} \right\rfloor$
2. Check Schnorr-like signature (c, y_0, y_1) for A'.
3. Check $hA' - dA + B = \mathbf{0}$
4. Check range proof $\mathcal{R}_B$

3.3 Limitations

The proposed scheme works only under the following conditions:

- Proof-of-stake difficulty: $D > 2^{64}$
- Output's amount: $v < 2^{64}$
- Commitment's mask: $f \neq 0$

3.4 Size of PoS proof

Let's estimate size of the proof (8).

It has two group elements, four field elements plus the size of the range proof $\mathcal{R}_B$.

In case of using Bulletproofs+ [2] the size of a single proof with $n = 64$ is

$$2 \cdot \lceil \log_2(n) \rceil + 3 = 15$$

group elements and 3 field elements.

In total, for PoS proof we have 17 group elements and 7 field elements. If both field and group elements have compressed size of 32 bytes, that is the case for Ed25519 used in Zano, then the total size of proof could be estimated as $17 + 7 = 24$ elements or $24 \cdot 32 = 768$ bytes.

4 Ring-friendly PoS hidden amount scheme

The solution we proposed in section 3 can't be used in case of CT-like mining transaction with a non-empty decoy set: in such a transaction stake input would refer to a *set* of outputs and thus a set of *pseudo* output commitments is used instead of a single commitment A in which input's hidden amount v is committed to. Therefore verifiers on the network would not be able to check (7), as they don't know the correct A.

Here we propose a solution to this problem.

4.1 Ring-friendly PoS scheme

Suppose, Alice wants to mine a PoS block and she already went though steps 1-3 above (subsection 3.2). She has calculated A', d, B so that $hA' - dA + B = \mathbf{0}$ holds.

Then she goes as the following:

4. Randomly selects a set of apparently unspent decoy outputs $\{u_i\}$ from the blockchain, and puts her output, which won the PoS main condition (2) at random index π into that set. Note, that i-th decoy output has its hidden amount committed to in A_i (and A_π is the commitment to her own output). Also note, that Alice doesn't know amounts v_i and masks f_i for the outputs she selected as decoys in general.

5. For each decoy output u_i $(i \neq \pi)$ Alice generates random values $b_{v,i} \xleftarrow{\$} \mathbb{Z}_{2^{64}}$, $b_{f,i} \xleftarrow{\$} \mathbb{Z}_l$ and commitments to them:

$$B_i = b_{v,i}G + b_{f,i}H \tag{9}$$

She also calculates complementary commitment A':

$$A'_i = h^{-1}(dA_i - B_i) \tag{10}$$

As before, B_π and A'_π were calculated earlier and correspond to her own output.

Now, consider system of equations $hA'_i - dA_i + B_i = \mathbf{0}$ or:

$$\begin{cases} hA'_0 & -dA_0 & +B_0 & = \mathbf{0} \\ & \dots & & \\ hA'_\pi & -dA_\pi & +B_\pi & = \mathbf{0} \\ & \dots & & \\ hA'_{n-1} & -dA_{n-1} & +B_{n-1} & = \mathbf{0} \end{cases} \tag{11}$$

Here π-th equation holds, because it equivalents to (7), and others hold because of (9) and (10).

6. Consider pair of group elements $(A_i + A'_i, A_i - A'_i)$. If $i = \pi$ Alice is able to calculate secrets k_0 and k_1 so the following holds:

$$\begin{cases} A_i + A'_i = k_0(G + H) \\ A_i - A'_i = k_1(G - H) \end{cases}$$

For all the others $i \neq \pi$ Alice would not be able to calculate such k_0, k_1, unless she selected her own output as a decoy (and thus, she knows the corresponding hidden amount and mask). But in such a case (11) holds only if she is able to calculate appropriate d and $b_{v,i}$ as well, which is equivalent to satisfying (2) for that "decoy" output.

According to Lemma 1, a proof of knowing k_0, k_1 is equivalent to proof that $A'_i = fG + vH$.

Alice adds proof of knowing secret keys k_0, k_1 into the main ring signature as additional two layers[2], in order to convince verifiers that $A'_i = fG + vH$ and $A_i = vG + fH$ hold together for the same output that is being spent.

To achieve this, main ring signature may be extended by adding two more group elements to the calculation of the non-interactive challenge as the following:

$$\begin{aligned} c_{\pi+1} &= H_s(\dots,\ \alpha_0(G + H),\ \alpha_1(G - H) \\ c_{i+1} &= H_s(\dots,\ r_i^0(G + H) + c_i(A_i + A'_i),\ r_i^1(G - H) + c_i(A_i - A'_i) \\ r_\pi^0 &= \alpha_0 - c_\pi k_0 \\ r_\pi^1 &= \alpha_1 - c_\pi k_1 \end{aligned}$$

7. Finally, Alice adds PoS signature $\sigma = \{d, \{A'_i\}, \{B_i\}, \{\mathcal{R}_{Bi}\}\}$ where $\{\mathcal{R}_{Bi}\}$ are range proofs (e.g. Bulletproofs) for the fact, that $b_{v,i} < 2^{64}$, to the mining transaction.

4.2 Verification of ring-friendly PoS scheme

Verifiers on the network check PoS block as the following:

1. $d \leq \left\lfloor \frac{l}{D} \right\rfloor$
2. Calculate h and check $hA'_i - dA_i + B_i = \mathbf{0}$

[2]Here we're using terminology and ideas from Multi-layered Linkable Spontaneous Anonymous Group signature proposed in 4.

3. Check stake input's ring signature with additional layers for A'_i
4. Check range proofs $\mathcal{R}_{Bi}$

4.3 Size of ring-friendly PoS proof

Let's estimate the size of the proof for $n-1$ decoy outputs, so the total size of the ring is n. Assume, we're using Bulletproofs+ for range proof, which is 15 group elements and 3 field elements per value (without aggregation).

For each ring member we have: a complementary commitment A'_i, group element B_i and 15 groups elements for it's range proof $\mathcal{R}_{Bi}$.

Ring signature extension would require, supposedly, only adding r_i^0 and r_i^1, that gives $2n$ additional field elements.

In total we have $17n$ group elements and $5n+1$ field elements. If both field and group elements have compressed size of 32 bytes, that is the case for Ed25519 used in Zano, then the total size of proof could be estimated as $22n+1$ elements or $704n+32$ bytes.

We'd like to note, that the total size could be greatly reduced by using aggregated Bulletproofs+.

References

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

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

[3] T.P. Pedersen. *Non-interactive andinformation-theoreticsecureveriØablesecretsharing.Advancesin Cryptology-CRYPTO'91*. 1991.

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

A Lemmas

Lemma 1. *Let $v, f \in \mathbb{Z}_l$ and $A = vG + fH$, where G and H – public generators in cycle group $\mathbb{G}$ of prime order l, for which DL relation is unknown. The Proover knows v and f. The Verifier knows only A. If the Proover wants to convince the Verifier that the given $A' = fG + vH$ without revealing secrets v and f, it is enough to provide a proof to the fact that he knows y_0 and y_1 such, that:*

$$A + A' = y_0(G + H)$$

$$A - A' = y_1(G - H)$$

Proof. Suppose the Verifier is convinced that the Proover knows y_0 and y_1 such, that the equations above hold. Also suppose, that contrary to the lemma's statement $A' = aG + bH \neq fG + vH$ where $a, b \in \mathbb{Z}_l$.

Substitute equations for A and A':

$$\begin{cases} vG + fH + aG + bH = y_0(G + H) \\ vG + fH - aG - bH = y_1(G - H) \end{cases}$$

As DL relation between G and H is unknown, we can split the equations:

$$\begin{cases} (v + a)G = y_0G \\ (f + b)H = y_0H \\ (v - a)G = y_1G \\ (f - b)H = -y_1H \end{cases}$$

$$\Rightarrow \begin{cases} v + a = f + b \\ v - a = b - f \end{cases}$$

$$\Leftrightarrow \begin{cases} 2v = 2b \\ 2a = 2f \end{cases} \Rightarrow A' = aG + bH = fG + vH$$

Contradiction.

□