StakeNote - A Proof-of-Stake Protocol for CryptoNote Payments

Bernardo David[1,2] and Dimitris Karakostas[1]

[1] Common Prefix
[2] IT University of Copenhagen (ITU)
{bernardo,dimitris}@commonprefix.com

Abstract. This work proposes a distributed ledger protocol that combines Proof-of-Stake (PoS) with privacy and anonymity preserving payments. We combine Ouroboros Praos, a well-known PoS protocol, with CryptoNote, a privacy-preserving payment system based on ring signatures, which has been widely used in practice. We demonstrate a practical approach for combining a PoS ledger with ring signature-based payments, such that the ledger's security properties and the payment system's privacy guarantees carry over.

Acknowledgements *This paper is a collaboration between the Zano open-source cryptocurrency and ecosystem, and Common Prefix. The project was commissioned and funded by Zano. The academic paper was authored by Prof. Bernardo David and Dr. Dimitris Karakostas from Common Prefix, with helpful discussions, questions, and feedback from Andrey Sabelnikov and Sowle from Zano.*

1 Introduction

Bitcoin [22] introduced a new paradigm for distributed computing and financial applications. On the technical side, the Proof-of-Work (PoW), blockchain-based ledger enabled the creation of an open and permissionless database, where participants join and leave the protocol arbitrarily. On the application side, the Bitcoin cryptocurrency founded the domain of Decentralized Finance (DeFi) [26], which was later enhanced with programmability capabilities via smart contracts. Despite its innovations, Bitcoin left a lot to be desired.

The first concern about Bitcoin was the sustainability of PoW usage. Bitcoin mining is consuming such large amounts of energy [1,8] that has led to calls for regulatory or policy interventions [5]. This concern was identified early and drove research towards more sustainable approaches. The most promising alternative was Proof-of-Stake (PoS) [16,7], which replaces the usage of physical assets, namely energy in the case of PoW, with digital assets, namely the assets ("stake") that are traded on the ledger itself.

A second concern regarding usability stemmed from Bitcoin's public nature. By default all information logged on the Bitcoin ledger is public and readable by everyone, revealing sensitive financial information. Consequently, various cryptocurrencies that guarantee user anonymity and privacy have been proposed. The arguably most popular are Zerocash [3] and CryptoNote [24], which are used by the Zcash and Monero platforms, respectively.

Naturally, these concerns have led to the exploration of privacy-preserving PoS protocols. Such protocols include Ouroboros Crypsinous [15] and privacy-aware blockchains [10], both of which build on the ideas and primitives behind Zerocash. Another line of work explored anonymous committee election, either in a PoS setting [2] or with accountability guarantees [6]. Finally, an elegant work showed the theoretical privacy limitations of anonymous PoS protocols [17], which further led to the development of protocols that offer security and privacy under different assumptions and more relaxed, albeit arguably more realistic, settings [25].

None of them systems using Cryptonote rely on a pure PoS mechanism for consensus. Monero uses PoW, Zano uses an ad-hoc hybrid of PoW and PoS, whereas Sentz uses a custom quorum-based mechanism inspired by Stellar [19]. Interestingly, little research has been done so far on *practical* privacy-preserving PoS protocols which support CryptoNote. The only attempt is Zarcanum [23], which is seems practical but lacks security proofs showing that it offers liveness and safety guarantees assuming honest stake majority.

This lack of research can be attributed to multiple reasons. First, Zerocash is more suitable for academic research, since it is provably secure and offers higher privacy guarantees than CryptoNote, namely a full anonymity set of all coins in the system, as opposed to a subset of coins as in CryptoNote. On the contrary, CryptoNote has not been formally defined and peer reviewed yet, while also a list of works have broken its privacy guarantees by demonstrating how to trace CryptoNote transactions [18,21,29,14,27,28].

Our work aims to fill this gap by proposing a *ring-friendly, secure, and practical* Proof-of-Stake distributed ledger protocol. Specifically, we present a consensus protocol that: (i) is secure, meaning it guarantees safety and liveness; (ii) inherits the privacy guarantees of ring-signature based payment protocols, like CryptoNote; (iii) is practical, i.e., does not rely on heavy and inefficient cryptographic primitives. Our primary goal is to offer a realistic alternative to existing CryptoNote-based ledgers, in order to move away from PoW and towards a more environmentally sustainable PoS alternative.

2 StakeNote Protocol

At a high level, the protocol follows the same steps as Ouroboros Praos [7], with some changes in the public setup and leader eligibility proofs. In our setting, each output consists of a public key and a commitment to the output's stake value. To compute each epoch's snapshot, parties should know when and if an output has been spent. This is not possible in standard RingCT, so we introduce a new

type of output, called "staking outputs", and enforce that they are identifiably spent at a consensus level (cf. Section 2.1).

On each slot, every party takes the following steps to create a block:[3]

1. $\mathcal{P}$ creates a VRF output, using the output's VRF key, in the same manner as Praos, i.e., passing as input the epoch's nonce and slot number.[4]
2. If the VRF output is below some threshold, which is a function of the output's stake value, then $\mathcal{P}$ is the slot leader and proceeds as follows:
   (a) $\mathcal{P}$ collects a subset of the snapshot's UTxOs including the chosen output.
   (b) $\mathcal{P}$ creates a block with the following proofs:
      i. A proof that $\mathcal{P}$ knows the spending key of an output in the chosen subset (without revealing which one).
      ii. A proof that the private VRF key, which created the VRF output from Step 1, is deterministically derived from the private payment key of the same output from Step 2(b)i.
      iii. A proof that the stake value of the same output from Step 2(b)i was used to compute the threshold of Step 2.
   (c) $\mathcal{P}$ signs the block by turning the proofs from Step 2b to a signature of knowledge, proving that the private signing key is the same as the spending key of the output from Step 2(b)i.

Building Blocks Our protocol uses Verifiable Random Functions (VRF) [20] with unpredictability under malicious key generation [7,11] as in Ouroboros Praos. Moreover, we use a modified Concise Linkable Spontaneous Anonymous Group (CLSAG) [13], which we define in Section 2.2. We now briefly recall the syntax of our building blocks.

Verifiable Random Functions (VRF) [7] with unpredictability under malicious key generation consist of three algorithms: $\mathsf{KeyGen}(\lambda) \to (\mathsf{sk}, \mathsf{pk})$: on input a security parameter λ outputs a secret key sk and a public key pk; $\mathsf{Eval}(\mathsf{sk}, x) \to (y, \pi_y)$: on input a secret key sk and a value x, outputs a pseudorandom value $y \in \{0,1\}^{l_{\mathsf{VRF}}}$ and a proof π_r; $\mathsf{Verify}(\mathsf{pk}, x, y, \pi_y) \to 0/1$: on input a public key pk, a value x, a value y, and a proof π_y, outputs 1 if and only if the proof verifies.

2.1 Stake Snapshot

In Ouroboros Praos, the epoch's eligible VRF keys are chosen at a fixed point in time before the epoch starts (e.g., two epochs prior). At that point, a snapshot of the stake distribution is taken and only the VRF keys that belong in the snapshot are eligible for leader election. In CrypoNote this is not possible, since the amounts of each output are hidden and parties cannot identify which historical outputs have been spent at any point in time. However, as in Ouroboros, the epoch's snapshot should only capture unspent outputs.

[3] If $\mathcal{P}$ controls multiple UTxOs, then the steps are executed for every output.

[4] As we describe in Section 2.2, for each output a VRF key pair is deterministically created by evaluating a PRF on the output's payment key.

To bypass this issue, we define protocol-specific "staking" outputs, which are identifiably spent. These differ from standard payment UTxOs only by a designated flag. With the introduction of staking outputs, a snapshot can now be directly computed, by collecting all *unspent* staking outputs.

A user creates a staking output in the same manner as any other UTxO, i.e., via typical transaction which consumes some UTxO. Note that each staking output should be created via a separate transaction. If multiple staking outputs are created within the same transaction, then this leaks the information that these outputs belong to the same party (who created the transaction).

In order to consume a staking output, the user creates a special transaction which identifiably spends the output. Specifically, the transaction uses a ring signature with $n = 1$, i.e., without any decoys. This is *enforced on the consensus level*, meaning that a transaction which consumes a staking output is valid *only if* it does not use any decoys.

Because the adversary corrupts parties at the beginning of each epoch, each staking output should be used only for *a single epoch*. Specifically, let a staking output o which is active during an epoch e. Honest parties must: (i) spend o before the snapshot for epoch $e + 1$ is taken; (ii) delete the private key x_o of o at the end of epoch e and before epoch $e + 1$ starts.[5]

2.2 Construction

At its core, our construction relies on a proof which shows that the block is produced by some eligible UTxO, among a subset of UTxOs in the snapshot, without revealing which specific UTxO is being used.

The eligibility criterion is the same as Praos, that is a VRF output should be below a certain threshold. The party $\mathcal{P}$'s VRF key is derived deterministically from the eligible UTxO's private key by hashing the latter.

Briefly, $\mathcal{P}$ first creates the VRF key, computes the VRF output and checks whether he is elected as slot leader — in the same manner as Ouroboros Praos. If so, $\mathcal{P}$ creates the block, proves that the VRF key is the hash of the UTxO's payment key and that the UTxO's (committed) stake value is above the eligibility threshold, and finally signs the block with the UTxO's payment key. Crucially, all of the above is done in a way that *does not reveal* the exact UTxO that is being used, but only reveals a set of UTxOs which includes the correct one.

Objects and notation

- Global protocol parameters:
  - G_1, G_2, G_{PAY}, G_{VRF}, G_{ST}: group generators of finite cyclic groups over a finite field F_p for some prime p
  - n: ring size of CryptoNote payment transactions
  - random oracle H_{clsag} and a PRF[6] H_{vrf}.

[5] This behavior is not enforced at the consensus level, but rather defines how honest parties should use their staking outputs, akin to key erasures in Ouroboros Praos.

[6] A Pseudorandom Function is used because we later prove knowledge of a preimage of the PRF when showing that the VRF key has been correctly derived.

  - δ: snapshot height delta, i.e., the slot distance between the snapshot's taking and the beginning of the epoch
- For every UTxO (output) o, we have the following:
  - private elements: (i) v_o: stake value; (ii) x_o: private payment key; (iii) r_o: salt of value's Pedersen commitment.
  - public elements: (i) $P_o = x_o \cdot G_{\mathsf{PAY}}$: public payment key; (ii) $C_o = v_o \cdot G_1 + r_o \cdot G_2$: Pedersen commitment to output's stake value; (iii) b_{ST}: "staking" flag bit.
  - public elements that are *not linkable* to o: (i) $Z_o = x_o \cdot G_{\mathsf{ST}}$: virtual staking public key; (ii) $Y_o = H_{\mathsf{vrf}}(x_o) \cdot G_{\mathsf{VRF}}$: VRF public key.
- Output sets: (i) $\mathbb{O}$: all (historical) outputs; (ii) $\hat{\mathbb{O}}_{\mathsf{sl}_r}$: all outputs with $b_{\mathsf{ST}} = 1$, which were *unspent* at slot sl_r
- Epoch e: (i) sl_e: epoch start (slot); (ii) η_e: epoch random nonce; (iii) $\mathbb{O}_e = \{o \in \hat{\mathbb{O}}_{\mathsf{sl}_e - \delta}\}$: epoch's snapshot (set of eligible outputs)

Block creation On a slot sl_r, for every output o that the party $\mathcal{P}$ controls, they check whether $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(v_o)$, where $\mathsf{Eval}(\mathsf{sk}_o, \eta_e || \mathsf{sl}_r) \rightarrow (y_o, \pi_{y_o})$ is verifiable under Y_o and $\phi_f(v_o)$ is defined as in [7]. If the check holds, then o is eligible for creating a block on slot sl_r, so $\mathcal{P}$ creates the following objects:

- T: the minimum value such that $v_o \geq T$ and $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(T)$
- B: block payload (transactions, metadata like timestamps, etc)
- $\mathcal{M} = \langle B, (Y_o, y_o, \pi_{y_o}), Z_o, T \rangle$: the unsigned block message

At this point, $\mathcal{P}$ needs to prove that they know x_o such that:

1. o can be spent using x_o;
2. the VRF output (y_o, π_{y_o}) was created using the private key $H_{\mathsf{vrf}}(x_o)$;
3. the stake value of o is $v_o \geq T$, such that $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(T)$;
4. the block is signed using x_o.

Importantly, $\mathcal{P}$ *does not reveal* o itself. Instead, $\mathcal{P}$ reveals only a subset of outputs $\mathbb{O}_{\mathsf{block}} \subseteq \mathbb{O}_e$ that contains o. The main proof of our construction concerns the statement: "I know the private payment key of an output in $\mathbb{O}_{\mathsf{block}}$, for which all of the following statements hold at the same time: (i) the output's key is equal to the scalar of the point Z_o; (ii) the value $\mathcal{C}$ is a commitment of the output's stake minus the value T."[7]

This proof is constructed via a three-column Concise Linkable Spontaneous Anonymous Group (CLSAG) [13]. Then, the proof is turned into a signature σ_{CLSAG} using the Fiat-Shamir heuristic [9]. For ease of reading, we give the full proof's construction at the end of this subsection.

Finally, $\mathcal{P}$ creates two more proofs. First, they create π_{RANGE}. This is a range proof (e.g., using Bulletproofs [4]) such that, for the correct output o, it holds $(v_o - T) \geq 0$. Second, they create a non-interactive zero-knowledge (NIZK) proof π_{NIZKVRF}, which shows that the private key of Y_o is the output of a PRF evaluated on the private key of Z_o. Specifically, π_{NIZKVRF} proves the statement: "I know a value a, such that $Z_o = a \cdot G_{\mathsf{ST}}$ and $Y_o = H_{\mathsf{vrf}}(a) \cdot G_{\mathsf{VRF}}$".

The final published block is: $\langle \mathcal{M}, \mathbb{O}_{\mathsf{block}}, \sigma_{\mathsf{CLSAG}}, \pi_{\mathsf{RANGE}}, \pi_{\mathsf{NIZKVRF}} \rangle$

[7] Note that $\mathbb{O}_{\mathsf{block}}, Z_o, \mathcal{C}, T$ are all part of the statement, hence public.

Block verification A full node validates a new block $\langle \mathcal{M}, \mathbb{O}_{\mathsf{block}}, \sigma_{\mathsf{CLSAG}}, \pi_{\mathsf{RANGE}}, \pi_{\mathsf{NIZKVRF}} \rangle$, by checking all of the following:

- (y_o, π_{y_o}) is valid w.r.t. Y_o, *i.e.*, $\mathsf{Verify}(Y_o, \eta_e || \mathsf{sl}_r, y_o, \pi_{y_o} \rightarrow 0/1$
- $y_o < 2^{l_{\mathsf{VRF}}} \phi_f(T)$
- $\mathbb{O}_{\mathsf{block}} \subseteq \mathbb{O}_e$
- σ_{CLSAG} is valid w.r.t. $\mathbb{O}_{\mathsf{block}}$ and $\mathcal{M}$
- π_{RANGE} is valid w.r.t. T and σ_{CLSAG} (see below for more details)
- π_{NIZKVRF} is valid w.r.t. Y_o, Z_o

CLSAG signature We will now describe our construction's main signature scheme (Definition 1). Our signature builds on the ideas of CLSAG [13], albeit without the linkability aspect. Intuitively, the proof can be imagined as a matrix of n rows and 3 columns, comprising an OR statement across rows and an AND statement across columns. Specifically, each row corresponds to one of the outputs in $\mathbb{O}_{\mathsf{block}}$. Each column corresponds to a value of interest that the party should prove knowledge: column 1 corresponds to the payment key P_o, column 2 corresponds to the key Z_o, and column 3 corresponds to the commitment $\mathcal{C}$. The proof is transformed to a signature via the Fiat-Shamir transformation [9].

Definition 1. *Our signature scheme consists of the tuple* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- $\mathsf{Setup} \rightarrow$ *par:* Setup *selects: (i) a prime* p*; (ii) group generators* $G_1, G_2, G_{\mathsf{PAY}}, G_{\mathsf{ST}}$ *uniformly at random; (iii) cryptographic hash function* H_{clsag} *with codomain* F_p *(modeled as random oracle). Then,* Setup *outputs* $par = (p, G_1, G_2, G_{\mathsf{PAY}}, G_{\mathsf{ST}}, n, H_{\mathsf{clsag}})$.
- $\mathsf{KeyGen} \rightarrow (\mathsf{sk}, \mathsf{pk})$*: When queried for a new key,* KeyGen *samples a fresh secret key and computes the associated public key:* [8]

$$\mathsf{sk} = (x_o, (v_o, r_o))$$
$$\mathsf{pk} = (x_o \cdot G_{\mathsf{PAY}}, v_o \cdot G_1 + r_o \cdot G_2)$$

- $\mathsf{Sign}(\mathcal{M}, \mathbb{O}_{\mathsf{block}}, \mathsf{sk}) \rightarrow \sigma$*:* Sign *parses the following:*
  - $\langle B, (Y_o, y_o, \pi_{y_o}), Z_o, T \rangle \leftarrow \mathcal{M}$
  - $\langle (P_0, C_0), \ldots, (P_{n-1}, C_{n-1}) \rangle \leftarrow \mathbb{O}_{\mathsf{block}}$
  - $\langle x_o, (v_o, r_o) \rangle \leftarrow \mathsf{sk}$

  Next:
  - Sign *picks a random scalar* r' *and computes:* [9] $\mathcal{C} = (v_o - T) \cdot G_1 + r' \cdot G_2$
  - Sign *finds the index* k*, such that* $\mathsf{pk}_k = (x_o \cdot G_{\mathsf{PAY}}, v_o \cdot G_1 + r_o \cdot G_2)$.
  - *For each* $i \in [0, n-1]$*,* Sign *computes* $D_i = C_i - T \cdot G_1 - \mathcal{C}$.[10]
  - Sign *picks* $(\alpha_x, \alpha_r) \xleftarrow{\$} (\mathsf{F}_p)^2$ *and computes:*

[8] Here, the "public key" is in practice a UTxO. We use the term "key" and "KeyGen"to be consistent with digital signature terminology.

[9] *Note:* $\mathcal{C}$ is used for the block's range proof π_{RANGE}. Specifically, the block producer constructs π_{RANGE} to show that, given $\mathcal{C}$ and a threshold T, it holds that $(v_o - T) \geq 0$ without revealing v_o.

[10] Observe that: $D_k = C_o - T \cdot G_1 - \mathcal{C} = (v_o - T - v_o + T) \cdot G_1 + (r_o - r') \cdot G_2 = (r_o - r') \cdot G_2$.

* $L_k = \alpha_x \cdot G_{\mathsf{PAY}}$
* $M_k = \alpha_x \cdot G_{\mathsf{ST}}$
* $R_k = \alpha_r \cdot G_2$
* $c_{(k+1) \mod n} = H_{\mathsf{clsag}}(\mathcal{M}||k||L_k||M_k||R_k)$

- Sign *sets* $i = (k+1) \mod n$ *and does the following, until* $i = k$*:*
  1. *Picks* $(s_i^{(x)}, s_i^{(r)}) \xleftarrow{\$} (\mathsf{F}_p)^2$.
  2. *Computes:*
     * $L_i = s_i^{(x)} \cdot G_{\mathsf{PAY}} + c_i \cdot P_i$.
     * $M_i = s_i^{(x)} \cdot G_{\mathsf{ST}} + c_i \cdot Z_o$.
     * $R_i = s_i^{(r)} \cdot G_2 + c_i \cdot D_i$.
     * $c_{i+1 \mod n} = H_{\mathsf{clsag}}(\mathcal{M}||i||L_i||M_i||R_i)$.
  3. *Sets:* $i = (i+1) \mod n$.
- Sign *computes:*
  * $s_k^{(x)} = \alpha_x - c_k \cdot x_o$
  * $s_k^{(r)} = \alpha_r - c_k \cdot (r_o - r')$
- *Finally,* Sign *outputs:* $\sigma_{\mathsf{CLSAG}} = \langle \mathcal{C}, c_0, (s_0^{(x)}, s_0^{(r)}), \ldots, (s_{n-1}^{(x)}, s_{n-1}^{(r)}) \rangle$

– $\mathsf{Verify}(\mathcal{M}, \mathbb{O}_{\mathsf{block}}, \sigma) \to \{0,1\}$*:* Verify *takes as input a message* $\mathcal{M}$*, a matrix* $\mathbb{O}_{\mathsf{block}}$*, and a signature* σ *and then:*

- Verify *parses the following:*
  * $\langle \mathcal{C}, c_0, (s_0^{(x)}, s_0^{(r)}), \ldots, (s_{n-1}^{(x)}, s_{n-1}^{(r)}) \rangle \leftarrow \sigma_{\mathsf{CLSAG}}$
  * $((P_0, C_0), \ldots, (P_{n-1}, C_{n-1})) \leftarrow \mathbb{O}_{\mathsf{block}}$
  * $\langle B, (Y_o, y_o, \pi_{y_o}), Z_o, T \rangle \leftarrow \mathcal{M}$
- Verify *sets* $c'_0 = c_0$.
- *For every* $i \in [0, \ldots, n-1]$, Verify *computes:*
  * $L_i = s_i^{(x)} \cdot G_{\mathsf{PAY}} + c_i \cdot P_i$.
  * $M_i = s_i^{(x)} \cdot G_{\mathsf{ST}} + c_i \cdot Z_o$.
  * $D_i = C_i - T \cdot G_1 - \mathcal{C}$.
  * $R_i = s_i^{(r)} \cdot G_2 + c_i \cdot D_i$.
  * $c'_{(i+1)} = H_{\mathsf{clsag}}(\mathcal{M}||i||L_i||M_i||R_i)$
- *If* $c_0 = c'_n$ Verify *outputs* 1*, otherwise it outputs* 0.

2.3 Enhancement: Fixed Staking Output Value

In order to avoid the range proof of the output's value, the protocol can enforce that each staking output has a fixed value. Specifically, When creating a staking output, the user sets its value to (a global parameter) X. This value *is not* hidden, as with standard UTxOs, and is enforced on the consensus level, i.e., honest parties check whether newly-created staking outputs hold exactly X tokens.

This approach has various implications. First, each staking output holds the same value, so the eligibility threshold is now a global parameter: $T' = 2^{l_{\mathsf{VRF}}} \phi_f(X)$. Therefore, when validating a block, it suffices to check whether the VRF output is below this threshold: $y_o \leq T'$. This removes the costly generation and verification of the range proof π_{RANGE}. Second, the CLSAG signature becomes simpler, containing only two columns. Finally, the following information is leaked: (i) for

each transaction which creates a staking UTxO, the information that one of the transaction's inputs holds at least X tokens;[11] (ii) for the transaction which consumes a staking UTxO, the information that the transaction's output holds exactly X tokens.[12]

3 Discussion

CLSAG Our CLSAG proof σ_{CLSAG} satisfies two different purposes. First, as is the original purpose of CLSAG, the block producer proves that they control one of the outputs in the group. Note that, in our case, linkability is not needed since we allow the same private key to produce multiple messages, that is multiple blocks per epoch. Second, it offers a discrete logarithm equality proof, between Z_o and P_o. This is achieved by CLSAG's elegant idea of reusing the same response $s_i^{(x)}$ for the two columns, that correspond to P_o and Z_o, effectively showing that the secret of both columns is the same.

Ledger security Our protocol changes Ouroboros Praos as follows. First, the honest majority assumption now applies on the staking outputs, since the snapshot is taken over them, instead of the entire stake distribution. Second, slot leader eligibility relies on the CLSAG, range, and NIZK proofs $(\sigma_{\mathsf{CLSAG}}, \pi_{\mathsf{RANGE}}, \pi_{\mathsf{NIZKVRF}})$. If the primitives used to build these proofs are secure, then the block producer cannot forge them and produce a block without being the correct slot leader, hence the security analysis of Ouroboros Praos is directly applicable here. Third, our protocol assumes a weaker adversary than Ouroboros Praos in terms of adaptive corruptions, since our adversary can only corrupt parties at the beginning of each epoch. Nonetheless, by requiring that parties delete their keys at the end of each epoch, the security argument of Ouroboros Praos carries over in our case as well, since the adversary cannot retroactively produce blocks for past slots when adaptively corrupting parties.

Staking outputs Staking outputs have the following implications. First, a staking output cannot be used as decoy for standard payment transactions, so the set of decoys for regular payment transactions is reduced, since staking outputs are excluded. Second, they reveal that a specific staking output has been consumed and also the addresses that receive this output's assets. Consequently, if a third party receives a payment directly from a staking output, then it knows that the sender participated in consensus using this output. For example, imagine that Alice has a staking output o. Alice creates a transaction that consumes o and which sends some assets to Bob. In this case, Bob, who knows that Alice created the transaction, can deduce that Alice controlled o. To avoid this hazard,

[11] If multiple inputs are used, then it leaks the information that the UTxOs of at least one of the input vectors hold on aggregate at least X tokens.

[12] If the transaction creates multiple outputs, then it leaks the information that these outputs hold on aggregate X tokens.

the miner should send a staking output's assets to a payment output *before* transferring them to a third party. In our example, Alice should consume o in a transaction that sends the its assets to a regular payment output α (that she controls), before sending the payment to Bob by consuming α via a standard transaction, which does not reveal (to Bob) that the funds originate from α (and consequently o). Finally, stake is not frozen throughout the epoch. The minimum time that staking outputs should remain active is until the snapshot is taken, as they can be spent immediately afterwards. This approach is consistent with Ouroboros Praos, where stake is also not frozen. Still, the protocol's designer *can* optionally enforce a freezing period, if needed.

References

1. for Alternative Finance, C.C.: Cambridge bitcoin electricity consumption index (2025), https://ccaf.io/cbnsi/cbeci
2. Baldimtsi, F., Madathil, V., Scafuro, A., Zhou, L.: Anonymous lottery in the proof-of-stake setting. In: Jia, L., Küsters, R. (eds.) CSF 2020 Computer Security Foundations Symposium. pp. 318–333. IEEE Computer Society Press (2020). https://doi.org/10.1109/CSF49147.2020.00030
3. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press (May 2014). https://doi.org/10.1109/SP.2014.36
4. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00020
5. Chamanara, S., Ghaffarizadeh, S.A., Madani, K.: The environmental footprint of bitcoin mining across the globe: Call for urgent action. Earth's Future **11**(10), e2023EF003871 (2023)
6. Christ, M., Choi, K., McKelvie, W., Bonneau, J., Malkin, T.: Accountable secret leader election (2024)
7. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 66–98. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_3
8. Digiconomist: Bitcoin energy consumption index (2025), https://digiconomist.net/bitcoin-energy-consumption
9. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
10. Ganesh, C., Orlandi, C., Tschudi, D.: Proof-of-stake protocols for privacy-aware blockchains. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 690–719. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17653-2_23
11. Giunta, E., Stewart, A.: Unbiasable verifiable random functions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part IV. LNCS, vol. 14654, pp. 142–167. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58737-5_6

12. Goldberg, I., Moore, T. (eds.): FC 2019, LNCS, vol. 11598. Springer, Cham (Feb 2019)
13. Goodell, B., Noether, S., Blue, A.: Concise linkable ring signatures and forgery against adversarial keys. Cryptology ePrint Archive, Paper 2019/654 (2019), https://eprint.iacr.org/2019/654
14. Hinteregger, A., Haslhofer, B.: Short paper: An empirical analysis of monero cross-chain traceability. In: Goldberg and Moore [12], pp. 150–157. https://doi.org/10.1007/978-3-030-32101-7_10
15. Kerber, T., Kiayias, A., Kohlweiss, M., Zikas, V.: Ouroboros crypsinous: Privacy-preserving proof-of-stake. In: 2019 IEEE Symposium on Security and Privacy. pp. 157–174. IEEE Computer Society Press (May 2019). https://doi.org/10.1109/SP.2019.00063
16. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 357–388. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_12
17. Kohlweiss, M., Madathil, V., Nayak, K., Scafuro, A.: On the anonymity guarantees of anonymous proof-of-stake protocols. In: 2021 IEEE Symposium on Security and Privacy. pp. 1818–1833. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00107
18. Kumar, A., Fischer, C., Tople, S., Saxena, P.: A traceability analysis of monero's blockchain. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017, Part II. LNCS, vol. 10493, pp. 153–173. Springer, Cham (Sep 2017). https://doi.org/10.1007/978-3-319-66399-9_9
19. Mazieres, D.: The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation **32**(4), 1–45 (2015)
20. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th FOCS. pp. 120–130. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814584
21. Möser, M., Soska, K., Heilman, E., Lee, K., Heffan, H., Srivastava, S., Hogan, K., Hennessey, J., Miller, A., Narayanan, A., Christin, N.: An empirical analysis of traceability in the monero blockchain. PoPETs **2018**(3), 143–163 (Jul 2018). https://doi.org/10.1515/popets-2018-0025
22. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
23. sowle, koe: Zarcanum: A proof-of-stake scheme for confidential transactions with hidden amounts. Cryptology ePrint Archive, Report 2021/1478 (2021), https://eprint.iacr.org/2021/1478
24. Van Saberhagen, N.: Cryptonote v 2.0 (2013)
25. Wang, C., Pujol, D., Nayak, K., Machanavajjhala, A.: Private proof-of-stake blockchains using differentially-private stake distortion. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 1577–1594. USENIX Association (Aug 2023), https://www.usenix.org/conference/usenixsecurity23/presentation/wang-chenghong
26. Werner, S.M., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., Knottenbelt, W.J.: Sok: Decentralized finance (defi) (2022), https://arxiv.org/abs/2101.08778
27. Wijaya, D.A., Liu, J.K., Steinfeld, R., Liu, D., Yu, J.: On the unforkability of monero. In: Galbraith, S.D., Russello, G., Susilo, W., Gollmann, D., Kirda, E., Liang, Z. (eds.) ASIACCS 19. pp. 621–632. ACM Press (Jul 2019). https://doi.org/10.1145/3321705.3329823

28. Yu, J., Au, M.H.A., Veríssimo, P.J.E.: Re-thinking untraceability in the CryptoNote-style blockchain. In: Delaune, S., Jia, L. (eds.) CSF 2019 Computer Security Foundations Symposium. pp. 94–107. IEEE Computer Society Press (2019). https://doi.org/10.1109/CSF.2019.00014
29. Yu, Z., Au, M.H., Yu, J., Yang, R., Xu, Q., Lau, W.F.: New empirical traceability analysis of CryptoNote-style blockchains. In: Goldberg and Moore [12], pp. 133–149. https://doi.org/10.1007/978-3-030-32101-7_9