HydRand
Practical Continuous Distributed Randomness

Philipp Schindler*, Nicholas Stifter*†, Aljosha Judmayer*, Edgar Weippl*†
*SBA Research
†Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI), TU Wien
Email: (firstletterfirstname)(lastname)@sba-research.org

Abstract*—A reliable source of randomness is not only an essential building block in various cryptographic, security, and distributed systems protocols, but also plays an integral part in the design of many new blockchain proposals. Consequently, the topic of publicly-verifiable, bias-resistant and unpredictable randomness has recently enjoyed increased attention in a variety of scientific contributions, as well as projects from the industry. In particular *random beacon protocols*, which are aimed at *continuous* operation, can be a vital component for many current Proof-of-Stake based distributed ledger proposals. We improve upon existing random beacon approaches by introducing *HydRand*, a novel distributed protocol based on publicly-verifiable secret sharing (PVSS) to ensure unpredictability, bias-resistance, and public-verifiability of a *continuous* sequence of random beacon values. Furthermore, HydRand is able to provide guaranteed output delivery of randomness at regular and predictable intervals in the presence of adversarial behavior and does not rely on a trusted dealer for the initial setup. In comparison to existing PVSS based approaches, our solution improves scalability by lowering the communication complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$. Furthermore, we are the first to present a comparison of recently described schemes in the area of random beacon protocols.

I. Introduction

The question of how to generate trustworthy random values among a set of mutually distrusting participants over a message passing network was first addressed by Blum in 1983, thereby introducing the notion of *coin tossing protocols* [6]. Lately, coin tossing protocols have received increased attention, in part because randomness is proving to be a vital component of most scalable distributed ledger approaches (e.g. [5], [15], [19]) that do not require a computationally intensive *Proof-of-Work* (PoW) mechanism as found in Bitcoin [21] and similar cryptocurrencies. Specifically, *Proof-of-Stake* (PoS) blockchain proposals, which rely on virtual resources in the form of digital assets, call for manipulation resistant and unpredictable leader election as part of a secure protocol design. In this regard Kiayias et al. identified leader election as a fundamental problem of PoS based protocols, since any introduced entropy is subject to potential manipulation by an adversary [19]. The distributed generation of trustworthy random values can hence be considered a complementary problem to the development of such protocols.

Random beacon protocols aim to generate publicly-verifiable, bias-resistant and unpredictable randomness[1] in distributed environments. The concept of a random *beacon* was first formalized by Rabin, which proposed a service that emits a fresh random number at regular intervals [23]. Random beacons can be particularly useful in the context of randomized consensus protocols, where participants may rely on a shared common coin [13], [22] to effectively break ties and ensure eventual progress. However, establishing this common coin generally relies on a trusted dealer, at least for the initial setup. In addition to the scenario of leader election and establishing consensus in Proof-of-Stake (PoS) based distributed ledgers, random beacons are also useful in a variety of other scenarios: This includes gambling and lottery services, publicly-auditable selections such as soccer World Cup draws and the verifiable assignment of a limited number of resources. Syta et al. [26] list additional use cases for randomness including Tor hidden services, generation of elliptic curve parameters, Byzantine consensus and electronic voting. One prominent example from the domain of cryptocurrencies is the provision of randomness to Smart Contracts, which often rely on insecure sources (such as the hash of block headers which is subject to manipulation by miners) or trusted third parties (e.g. the NIST random beacon service) [2], [10]. For all the mentioned scenarios the following properties, as previously outlined in [3], [9], [26], are desiderata of a random beacon protocol:

1) **Availability/Liveness:** Any single participant or a colluding adversary should not be able to prevent progress.
2) **Unpredictability:** Correct as well as adversarial nodes should not be able to predict (precompute) future random beacon values.
3) **Bias-Resistance:** Any single participant or colluding adversary should not be able to influence future random beacon values to their advantage.
4) **Public-Verifiability:** Third parties, i.e. processes which are not directly partaking in the protocol, should also be able to verify generated values. As soon as a new random beacon value becomes available, all parties can verify the correctness of the new value using public information only.

We give formal definitions and security proofs for these properties in section VI. Although not always explicitly stated, practical solutions should also achieve good **efficiency** in terms of computational resources as well as communication complexity. Furthermore, we suggest that **guaranteed output delivery**, i.e., the inability for an adversary to prevent correct nodes of the protocol from obtaining an output [14], can also be considered a valuable property in practical random beacon protocols. Another particular desirable property for random beacons in the context of (permissionless) distributed ledgers is the **avoidance of an initial trusted setup**, e.g. a trusted dealer, [26].

[1]In the following we will simply refer to this by the term *randomness*.

Current random beacon protocols aim to provide solutions by employing different techniques, reaching from Proof-of-Delay [10], [12] and incentive based solutions [11], [24] over publicly-verifiable secret sharing (PVSS) [3], [14], [19], [26] and unique signatures [15], [17] to utilizing Bitcoin itself as a source of randomness [4], [9]. The diversity of these approaches, as well as the differences in their underlying assumptions and characteristics, make them difficult to compare and not equally suited for all use-cases. Moreover, some recently described protocols in this field tend to be closely coupled with their respective (PoS) blockchain schemes and are therefore not easily comparable or deployable in other settings, e.g. as a stand alone protocol.

A. Contribution

We present *HydRand*, a new PVSS based distributed random beacon protocol geared towards the *continuous* provision of randomness at regular intervals in a Byzantine failure setting. HydRand provides *guaranteed output delivery*, i.e., it guarantees the generation of new, *bias-resistant* randomness in every round of the protocol. *Unpredictability* is furthermore ensured with absolute certainty if a commitment is made to use the random output of the beacon after at least $f + 1$ rounds in the future.[2] The protocol assumes a synchronous system model and $n = 3f + 1$ participants. In respect to previous approaches based on PVSS, the communication complexity is hereby lowered from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ as the initial overhead of $f+1$ rounds is quickly amortized during continuous operation. Our protocol is described in a self contained manner and does not rely on a trusted dealer or distributed key generation (DKG) protocol. Moreover, to the best of our knowledge, we are the first to provide a detailed comparative overview of recent random beacon protocols in this field.

B. Related Work

In recent years a substantial amount of research related to random beacon protocols for distributed ledgers has been published in academia as well as the industry:

Algorand [15] builds a distributed ledger by combining (i) a randomness beacon based on unique signatures and hash functions with (ii) a newly proposed randomized Byzantine agreement protocol [20]. The protocol can be parameterized to achieve good probabilistic liveness guarantees that are sufficient for all intents and purposes. Considering the produced randomness as part of the protocol execution, it is not fully bias-resistant as protocol participants can decide to withold information if they are selected as leader. *Ouroboros* [19] is a provably secure Proof-of-Stake blockchain protocol. It relies on a combination of publicly-verifiable secret sharing (PVSS) and other cryptographic primitives to obtain randomness. The agreed and bias-resistant randomness is then used as the basis for the respective Proof-of-Stake algorithm, which is the main focus of Ouroboros. The newer Ouroboros Praos protocol [16] improves upon the communication complexity of Ouroboros in a similar way to Algorand, but cannot ensure our strong notion of bias-resistance. The protocol family *RandShare*, *RandHound* and *RandHerd* [26] also employs PVSS in combination with a Byzantine fault tolerant (BFT) consensus algorithm (RandShare, RandHound) and additionally with Collective Signing (RandHerd). For the scenario outlined by the authors, the more scalable protocols RandHound and RandHerd however operate with a failure probability of 0.08%.

In an orthogonal work, I. Cascudo and B. David [14] present Scrape, thereby introducing an optimized variant of Schoenmakers' secret sharing protocol [25], which we use as main building block for our protocol, and that can also be used to reduce computation complexity in Ouroboros, RandShare, RandHound and RandHerd. Scrape, Caucus [3] and Proof-of-Delay [10], [12] build upon the assumption that a shared bulletin board, i.e., a distributed ledger, is available for exchanging information between participants, thereby necessitating some form of external blockchain or other consensus protocol if it is to be used as a stand-alone random beacon implementation. HydRand helps to close this gap by presenting a self-contained protocol that is focused towards a permissioned system model.

The *Dfinity* project [18] of the equally named foundation is aiming to build a decentralized verifiable random function as the key ingredient for reaching consensus among network nodes. Compared to the other schemes, they utilize BLS signatures for that purpose [17]. BLS provides signature uniqueness as well as support for signature aggregation [7], [8]. Dfinity combines both of these key properties to obtain a random beacon protocol. However, the security assumptions required for BLS signatures are less analysed when compared to traditional elliptic curve cryptography. HydRand does not rely on assumptions for pairing based cryptography to achieve its security goals.

C. Structure of this paper

The paper is structured as follows: Our system model is described in section II. Section III gives a high level overview of our protocol and outlines the basic properties of Publicly-Verifiable Secret Sharing (PVSS), which is one of the main cryptographic primitives in our design. The details of our protocol are described in section IV and an example execution of the protocol is provided in section V. Proofs showing that the protocol indeed achieves the desired properties are presented in section VI. Section VII compares the protocol to other related schemes and sections VIII and IX discuss and conclude the paper.

II. System and Threat Model

We assume a fixed set of known participants, hereby referred to as nodes, of size $n = 3f+1$, of which at most f nodes may exhibit Byzantine failures and can deviate arbitrarily from the specified protocol. A node is considered to be *correct* if it does not exhibit any incorrect behavior during the entirety of the protocol execution, else it is considered to be *faulty*. The terms *Byzantine* or *malicious* are used synonymous to refer to faulty nodes. The set of nodes is denoted by $\mathcal{P} = \{1, 2, ..., n\}$ and each node $i \in \mathcal{P}$ is assumed to have a private / public key pair $\langle sk_i, pk_i \rangle$. The public keys of these keypairs are known to all participants. We assume a synchronous system model with a fully connected network of authenticated and reliable bidirectional point-to-point messaging channels.

[2]Before the bound of $f + 1$ rounds, the probability of prediction decreases exponentially with each round to predict.

III. Protocol Overview

The aim of the HydRand protocol is to provide a bias-resistant, publicly-verifiable and unpredictable stand-alone random beacon which emits random values at a regular interval. We target HydRand at a permissioned setting with a fixed set of participants and assume a known upper bound Δ on both computation and message transmission times.

During the protocol setup, all participants have to exchange their public keys and prepare an initial commitment. The protocol operation itself is separated into *rounds*, where each round consists of three distinct *phases*. In each round, the previously generated random value is used for uniquely selecting the current *leader* of this round. Generally speaking, the selected leader has two main choices: (i) The leader *reveals* the correct secret value he has committed himself to the last time he was leader (or during protocol setup) and attaches his next commitment. (ii) The leader does not reveal his secret value and therefore cannot attach another commitment. In the later case, this previously committed secret value will be *reconstructed* by $f+1$ other nodes, including at least one correct participant. The properties of the underlying PVSS scheme ensure that the random beacon value obtained by reconstruction is equal to the value that would have been obtained if a leader has revealed his secret – this establishes *bias-resistance*. Once the leader's previous commitment is reconstructed, the current leader is excluded from being eligible as leader in further rounds since he has not provided a new valid commitment.

If the leader is correct he constructs a new *dataset*, which (simply speaking) includes: (i) the revealed secret value he previously committed himself to, (ii) a new commitment to a randomly chosen value and (iii) a reference to the dataset of the previous round. The leader signs this dataset using his private key and broadcasts this message and signature to all other nodes in the network. After receiving and verifying the dataset, each node can compute a new random value.

In case a leader fails or purposely does not broadcast any data, other participants can collaborate to reconstruct the missing secret value, i.e. the value the leader has previously committed himself to in (ii). This reconstructed value can be used by each node to obtain a new random beacon value and thereby advance the protocol to the next round and hence to the next leader. This process is repeated until eventually a leader is selected that creates a new dataset that accounts for all reconstructed datasets in between.

To ensure that a correct node is selected as leader after (at most) $f+1$ rounds, all previously selected leaders of the last f rounds are not allowed to become leader in the current round. Since malicious nodes do not know how a revealed or reconstructed commitment of a correct node influences future random beacon values, they cannot precompute future random values once a correct node has been selected. Moreover, correct participants agree on a single history after a correct node is selected as leader, because correct leaders are assumed to build on top of a single dataset and never sign different datasets in the same round. The correct node hence acts as a barrier for *unpredictability* and anchor for agreement on the protocol state. Unpredictability is ensured with certainty for any round after $f+1$ rounds in the future. By leveraging the properties of the underlying PVSS scheme *public-verifiability* is established.

A. Publicly-Verifiable Secret Sharing

We use publicly-verifiable secret sharing (PVSS) as a primary building block in the HydRand protocol. More specifically, we make use of Scrape's PVSS protocol [14], which is an optimization of Schoenmakers' PVSS scheme [25], and allows a node (dealer) to efficiently share a secret value $s \in \mathbb{Z}_q$ among a set of n recipients, such that any subset of size $\geq t$ of these nodes is able to recover / reconstruct the value $h^s \in \mathbb{G}_q$, , where h is one of two independent generators of this group. The value of the reconstruction threshold t is set in a way that does not enable a colluding adversary to successfully recover a shared secret without requiring the collaboration of at least one correct node, i.e. $t = f+1$. A key property of a *publicly-verifiable* secret sharing protocol is that, upon receiving the secret shares, not only the recipients but any third party with access to the public keys of the participants can verify the correctness of the shares prior to reconstruction of the secret. We use the term *PVSS commitment*, denoted by $Com(s_d)$, to refer to the result of the share distribution process of Scrape's PVSS. To form a PVSS commitment, a dealer d provides:

- The encrypted shares for a secret s_d, i.e. one encrypted share $\hat{s_i}$ for each node i encrypted with the receiver's public key.
- The commitments $v_1, v_2, ..., v_n$ to the individudal shares.
- A non-interactive zero-knowledge (NIZK) proof ensuring the correctness of the encrypted shares

For additional details we refer to the reader to [14].

B. Design Rationale

To bias the resulting sequence of random beacon values, a malicious leader could try to construct and send different commitments and hence different datasets to participating nodes or selectively withhold information. Such a construction necessitates some form of (Byzantine) *consensus protocol* for participants to reach agreement upon either the existence of a single, valid commitment or that the leader was faulty. In this respect HydRand leverages on its intended application as a continuous random beacon by reducing the communication overhead of Byzantine agreement (BA) that would be incurred at each round through bundling messages. Specifically, HydRand implements its own variation of a Byzantine agreement protocol that defers consensus decisions for up to $f+1$ rounds and combines information from multiple instances of consensus that are executed with every consecutive new round in the HydRand protocol. Thereby, the overall communication (bit) complexity of comparable PVSS based random beacon schemes is reduced from $O(n^3)$ to $O(n^2)$ as HydRand only requires a single PVSS share distribution and potentially a single PVSS recovery per round. Still the protocol outputs a new random beacon value once per round, because these values are not dependent on immediate agreement on the protocol state.

IV. Protocol Details

The protocol proceeds in rounds. Each round $r \geq 1$ consists of three phases: *propose*, *acknowledge* and *vote*. Further, each round has an associated (randomly selected) leader $\ell_r \in \mathcal{P}$, denoted by ℓ if r is clear from the context.

In each round, ℓ_r is selected uniformly at random from the set of all nodes, which have not been selected as leader during the last $f+1$ rounds.[3] At the end of each round all nodes learn a new random beacon value R_r. For simplicity, we hereby assume that the correct nodes agree on the first random beacon value R_0 used to select the leader of round 0 as well as the set of initial commitments of all nodes. R_o becomes public knowledge only after the set of initial commitments was defined during setup.[4]

To simplify our notation we assume that a node or leader, which broadcasts a message is also recipient of that message. Similarly, the dealer in the PVSS protocol provides a share for himself. We denote a cryptographic signature on a message m by $\langle m \rangle_i$, where i denotes the node signing the message with its private key sk_i. We further assume, that all correct nodes discard invalidly signed messages and process only messages for the round and phase it is currently working on.

A. Phase: Propose

During this phase the round's leader reveals his previously committed value s_ℓ and provides a new commitment $Com(s_\ell^\star)$. For this purpose, it is the leader's task to propose a new dataset D_r for the current round r. A dataset D_r consists of two parts, a header and a body, where the hash of the header is simply denoted by $H(D_r)$. The header $header(D_r)$ of dataset D_r contains:

- the current round index r
- the rounds random beacon value R_r
- the revealed secret value s_ℓ
- the most recent round index $\hat{r}$ for which ℓ_r is in possession of a valid dataset $D_{\hat{r}}$, as well as a confirmation certificate $CC(D_{\hat{r}})$ for this dataset. In the first round of the protocol, this value is set to $\hat{r} = 0$ since no such dataset can exist
- the hash $H(D_{\hat{r}})$ for the referenced dataset $D_{\hat{r}}$ if $\hat{r} > 0$
- a list of random beacon values $\{R_k, R_{k+1}, ...\}$ for all recovered rounds between $\hat{r}$ and r such that $\hat{r} < k < r$
- coefficient C_0 of the new commitment $Com(s_\ell^\star)$, which allows for later verification of $s_\ell^\star$
- the Merkle tree root hash M_r over all encrypted shares in the new commitment $Com(s_\ell^\star)$ (see the definition of the body below)

The body $body(D_r)$ of dataset D_r contains:

- the commitment $Com(s_\ell^\star)$ to a new randomly chosen secret $s_\ell^\star$
- a confirmation certificate $CC(D_{\hat{r}})$, which confirms that $D_{\hat{r}}$ was previously accepted as valid dataset
- a recovery certificate $RC(k)$ for all rounds $k \in \{\hat{r}+1, \hat{r}+2, ..., r-1\}$, which confirms that there exists a recovery for all rounds between $\hat{r}$ and r. If $\hat{r} = r-1$ then no such intermediate round exists and this value is omitted.

A correct leader ℓ broadcasts a signed *propose* message

$$\big\langle propose, \langle header(D_r) \rangle_\ell, body(D_r) \big\rangle_\ell$$

to all nodes. Each node i, that receives such a message from the leader before the end of the propose phase, checks the validity of the dataset D_r. For this purpose i verifies that D_r is constructed as defined and properly signed. This includes a check that the revealed secret s_ℓ corresponds to the commitment $Com(s_\ell)$ submitted previously by the current leader. Additionally the validity of the confirmation and recovery certificates is checked. A *confirmation certificate* for dataset $D_{\hat{r}}$ is valid iff it consists of $f+1$ signed messages of the form

$$\langle confirm, \hat{r}, H(D_{\hat{r}}) \rangle_i$$

from $f+1$ different senders. Similarly, a *recovery certificate* for some round k is a collection of $f+1$ signed messages of the form $\langle recover, k \rangle_i$ from $f+1$ different senders. The leader selects $\hat{r}$ as the highest possible round index for which he is only in possession of a valid confirmation certificate $CC(D_{\hat{r}})$ but does not know a recovery certificate $RC(\hat{r})$.

B. Phase: Acknowledge

If a node i receives a valid dataset D_r from the round's leader ℓ_r during the propose phase, it constructs and broadcasts a signed acknowledge message

$$\big\langle \langle acknowledge, r, H(D_r) \rangle_i, \langle header(D_r) \rangle_\ell \big\rangle_i$$

thereby also forwarding the revealed secret value s_ℓ. Further, each node i collects and validates acknowledge messages from all nodes.

C. Phase: Vote

Each node i checks the following conditions:

- During the current propose phase a valid dataset D_r was received.
- During the current acknowledge phase $\geq 2f+1$ acknowledge messages from different senders have been received.
- All of those messages acknowledge the received dataset's hash[5] $H(D_r)$.

If all conditions are met, node i broadcasts a signed confirmation message:

$$\langle confirm, r, H(D_r) \rangle_i$$

Otherwise node i, broadcasts a recover message:

$$\big\langle \langle recover, r \rangle_i, s_\ell, Com(s_\ell)[s_i], \hat{s_i}, M_k[\hat{s_i}] \big\rangle_i$$

[3]The detailed leader selection mechanism is described in section IV-D.

[4]In practice this initial random value can be obtained via *Proof-of-Delay* [10] or a *Proof-of-Work* [4].

[5]Valid acknowledge messages for more than one value of $H(D_r)$ form a cryptographic proof of leader equivocation. E.g. in a Proof-of-Stake setting, the protocol might be extended such that this equivocation proof in used to seize the security deposit of the leader.

Here, $Com(s_\ell)[s_i]$ denotes i's decrypted share s_i and its share decryption proof according to Scrape's PVSS, which cryptographically proves that s_i is a valid decryption of $\hat{s_i}$ under i's secret key. Round k denotes the round in which ℓ has provided the commitment $Com(s_\ell)$ and a Merkle tree root hash M_k. The Merkle branch $M_k[\hat{s_i}]$ proofs that the encrypted share $\hat{s_i}$ was previously distributed as part of $Com(s_\ell)$ and therefore also of D_k. The values $\hat{s_i}$ and $M_k[\hat{s_i}]$ are required to enable nodes which are not in possession of $Com(s_\ell)$ to verify the share decryption proof for s_i.

Correct nodes always include values for $s_\ell, Com(s_\ell)[s_i], \hat{s_i}$ and $M_k[\hat{s_i}]$ if they are in possession of the required data. Otherwise the unknown value(s) are omitted. Upon receiving recovery messages from other nodes, correct nodes accept messages with omitted values. This is not a problem since the protocol ensures that there are always at least $f+1$ correct nodes that have received the dataset with a valid confirmation certificate, and hence can provide the shares necessary for reconstructing the secret of the respective dataset. An example is presented in section V.

At the end of this phase each node i can obtain the round's random beacon value R_r. We distinguish two cases: (i) node i already knows the secret value s_ℓ, because it received the dataset D_r or an acknowledge message for D_r, and (ii) node i has received at least $f+1$ valid recover messages which include at least $f+1$ decrypted secret shares for s_ℓ. In this case the reconstruction procedure of Scrape's PVSS can be executed to produce the value h^{s_ℓ}. In both cases R_r is then obtained by computing:

$$R_r \leftarrow H(R_{r-1} \,||\, h^{s_\ell}) \tag{1}$$

D. Leader selection

At the beginning of each round $r \geq 1$, a node i determines the round's leader ℓ_r based on the available local information it gathered so far. For this purpose node i uses the randomness R_{r-1} of the previous round to deterministically select ℓ_r from the set $\mathcal{L}_r$ of potential leaders. We denote the canonical representation of $\mathcal{L}_r$ as $\langle l_0, l_1, ..., l_{|\mathcal{L}_r|-1}\rangle$ and obtain ℓ_r as follows:

$$\ell_r \leftarrow l_{(R_{r-1} \bmod |\mathcal{L}_r|)} \tag{2}$$

Let $D_{\hat{r}}$ denote the most recent valid dataset, for which node i is *not* in possession of a corresponding recovery certificate $RC(\hat{r})$. If no such dataset exists[6] we set $\hat{r} = 0$. Now we introduce a method to determine *recovered nodes* $rn(\cdot)$ as a component needed for the definition of $\mathcal{L}_r$. Intuitively, the set defined by $rn(\cdot)$ contains all nodes, which have not provided valid datasets for some round where the node has been selected as leader. We define this set of all leaders which have been recovered in some round up to a referenced dataset as follows:

$$rn(D_x) = \begin{cases} \emptyset & \text{if } \hat{x} = 0 \\ \{\ell_k \mid RC(k) \in D_x\} \cup rn(D_{\hat{x}}) & \text{otherwise} \end{cases} \tag{3}$$

Here $D_{\hat{x}}$ denotes the previous dataset referenced by D_x. This function is used to construct the set of available nodes $\mathcal{P}_r$ for round r recursively by excluding all nodes which have been selected as leader in a round for which a valid reconstruction certificate exists:

$$\mathcal{P}_r = \mathcal{P} \setminus rn(D_{\hat{r}}) \tag{4}$$

Based on this notion, the definition of the set of potential leaders $\mathcal{L}_r$ for round r follows:

$$\mathcal{L}_r = \mathcal{P}_r \setminus \{\ell_{r-f}, \ell_{r-f+1}, ..., \ell_{r-1}\} \tag{5}$$

Intuitively, the set $\mathcal{L}_r$ only includes nodes, which have not been selected as leader for at least f rounds in the past and have not been reconstructed in any previous round, i.e., distributed valid datasets for all rounds in which they have been selected as leader.

V. Example Protocol Execution

Figure 1 shows four rounds of an example execution of the HydRand protocol in a setting of $f = 2$ Byzantine nodes. We assume that the leaders in this specific execution got selected randomly as described in section IV-D. The sequence of leaders in this example execution includes a worst case scenario, where f succinct leaders come from the set of byzantine nodes (nodes n_4 and n_5), followed by a correct node and then again the first byzantine node (n_4).

Round r_1: In this execution the first node that gets selected as the leader (i.e., node n_4) belongs to the set of byzantine nodes. This leader selectively sends a *propose* message only to a subset of correct nodes. In our case the nodes n_1, n_2 and n_3. Moreover, the Byzantine node n_5 only sends *acknowledge* messages to the very same nodes (n_1, n_2, n_3). After that phase, the Byzantine node n_5 sends a *recover* message to the nodes n_6 and n_7.

This leads to a situation where the correct nodes n_1, n_2 and n_3 receive $f+1$ acknowledge messages. Therefore, those nodes (n_1, n_2 and n_3) broadcast *confirm* messages which together form a valid confirmation certificate known to every node. Further, the nodes n_6 and n_7 as well as the adversary are in possession of a valid recovery certificate $RC(r_1)$, as nodes n_5, n_6 and n_7 sent a *recover* messages.

Round r_2: The next node (n_5) that gets selected as leader is also in the set of byzantine nodes and does not broadcast any message. Therefore, the secret value of the rounds leader gets reconstructed at the end of the *vote* phase and all nodes are only in possession of a *reconstruction certificate* $RC(r_2)$ for this round.

Round r_3: The leader (n_3) of this round belongs to the set of correct nodes and has received $f+1$ *confirm* messages in round r_1. Moreover, node n_3 is not in possession of a valid recovery certificate for r_1 since he has only received f *recover* messages, i.e. from node n_6 and n_7 but not from node n_5. Therefore, the leader broadcasts a new dataset D_3 containing a valid *confirmation certificate* $CC(D_1)$ for round r_1, as well as a *recovery certificate* $RC(r_2)$ for round r_2.

After receiving the *propose* message, all correct nodes, including n_6 and n_7, are safe to assume that at least $f+1$ correct nodes are in possession of dataset D_1. The justification for this assumption comes from the fact that the *propose* message contains a *confirmation certificate* composed of $f+1$ signed messages including a hash the header of D_1. This

[6]In this scenario all rounds since protocol start can be recovered.

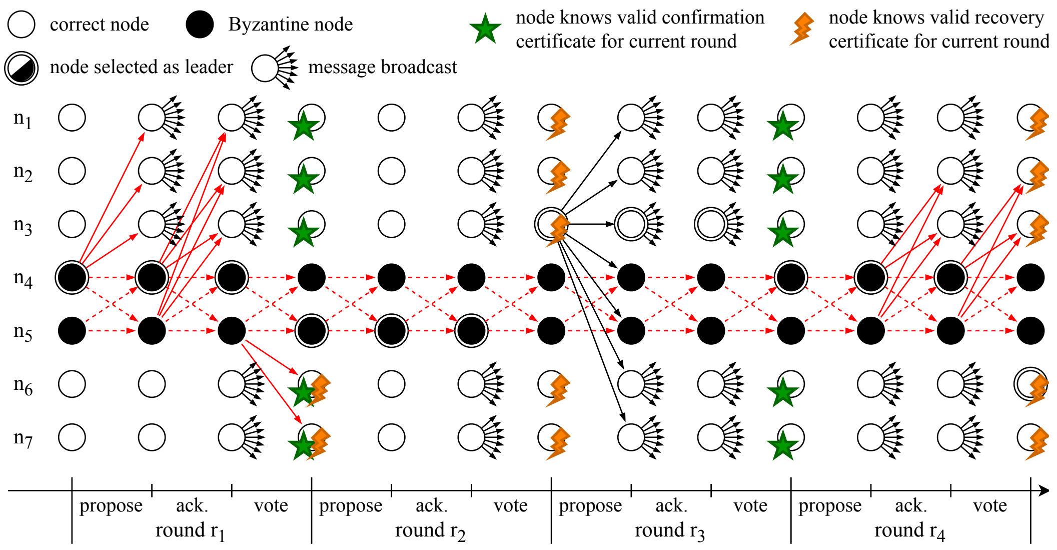

Fig. 1. Example execution of four rounds of the HydRand protocol with $n = 3f + 1 = 7$.

necessarily includes at least one honest node which, per definition, only sends a confirm message if he has received $2f + 1$ valid *acknowledge* messages in advance. Therefore, at least $f+1$ correct nodes have to be in possession of dataset D_1. As a result, all correct nodes accept this rounds new dataset D_3 containing $CC(D_1)$. This holds true, even for nodes n_6 and n_7 although they have not received dataset D_1.

If node n_6 or n_7 would have been selected as leader in round r_3, then this node would have constructed a dataset D_3 that contains a valid *recovery certificate* for round r_1 and r_2 as well. In that case the nodes n_1, n_2 and n_3 would have discarded their dataset D_1.

Round r_4: In this round node n_4 is again selected as leader. This is valid since f rounds have passed since this node has been selected as leader. Therefore, at least one correct node was selected as leader in between – in this case node n_3. Since there is no *recovery certificate* $RC(r_3)$ for round r_3 available, all further leaders have to include the *confirmation certificate* $CC(D_3)$ for round r_3 to extend upon the chain of valid datasets. Otherwise their future datasets would not be valid and rejected by all correct nodes. Therefore, all nodes including node n_4, have to accept the view of node n_3 in this case.

In our example, node n_4 tries to stall the protocol by selectively releasing a new dataset D_4 only to the nodes n_2, n_3. But since those nodes are not able to reach the required number of $2f + 1$ *acknowledge* messages (together with the byzantine nodes n_4 and n_5), no correct node will send a *confirmation* message in the last phase of this round. As a result all correct nodes will send *reconstruct* messages leading to a total of $2f + 1$ reconstruct messages, which is more than $f + 1$ and hence enough to form a *reconstruction certificate* and to reconstruct the leader's secret for round r_4.

Note that, although possible, the PVSS reconstruction of the secret from r_1 would not be necessary here, since in this example the leader of r_4 selectively sent out a new dataset and therefore revealed the secret to at least one correct node. Per definition, correct nodes broadcast the revealed secret in their *acknowledge* messages. Therefore, all other correct nodes receive the revealed secret in round r_4 even if they have not received the dataset D_3 directly.

VI. Protocol Properties

In the following, we show that HydRand achieves the desirable properties of a random beacon protocol as outlined in section I: *liveness*, *guaranteed output delivery*, *unpredictability*, *bias-resistance*, and *public-verifiability*. We furthermore show that our protocol also achieves *uniform agreement*. In our proofs we might refer to the definitions introduced in section IV.

Lemma 1: (Possibility of construction of valid datasets) For each round r a correct leader ℓ_r can construct a valid dataset D_r.

Proof: Since we are in a fully synchronous setting, we assume a correct leader always knows the round number r. Further, a correct leader is in possession of its own secret s_ℓ and thus knows R_r. Furthermore, the leader can always construct a new PVSS commitment for a new secret $Com(s^\star_\ell)$ and is able to provide valid values for M_r and C_0. Therefore, it only remains to show that each correct node is able to provide the required confirmation certificate $CC(\cdot)$ (and its round number) and recovery certificates $RC(\cdot)$. During the vote phase of all previous rounds, all correct nodes either broadcast a recover or confirm message. As there are at least $2f+1$ correct nodes, each node receives at least $f+1$ recover messages or at least $f+1$ confirm messages (or both) for each of these rounds. As $f + 1$ recover messages form recovery certificate and $f + 1$ confirm messages form a confirmation certificate, each node is in possession of a recovery certificate or a confirmation certificate (or both) for each previous round, and is therefore able to provide the required certificates for D_r. ■

Lemma 2: (No recovery of correct leaders) If the leader ℓ_r is correct, there does not exist a node i, which is in possession of a valid recovery certificate $RC(r)$.

Proof: A correct leader ℓ_r sends valid proposal D_r to all nodes during the propose phase. By lemma 1, ℓ_r can always construct such a dataset. As all correct nodes consider D_r as valid, at least $2f+1$ nodes broadcast acknowledge messages for D_r during the acknowledge phase. All $2f+1$ correct nodes therefore receive $2f+1$ valid acknowledge messages for D_r. As there cannot exist a valid acknowledge for a different dataset D'_r (because the leader only provided his signature for D_r) all correct nodes broadcast *confirm* messages during the vote phase. As correct nodes only broadcast either confirm or recover messages, there are at most f recover messages (from Byzantine nodes). A valid recovery certificate $RC(r)$ however requires at least $f+1$ recover messages from different nodes, and therefore cannot exist. ■

Lemma 3: (Availability of leaders) For each round $r \geq 1$, the set of potential leaders $\mathcal{L}_r$ contains at least $f+1$ correct nodes.

Proof: We first show that for each round r, the set of available nodes $\mathcal{P}_r$ contains at least $2f+1$ correct nodes. By definitions 3 and 4, we have that only leaders ℓ_k for some round k, in which a recovery certificate $RC(k)$ exists, are excluded from the set $\mathcal{P}$ to form $\mathcal{P}_r$. As we have shown in lemma 2 there are no recovery certificates for rounds with correct leaders. Therefore correct nodes cannot be excluded from $\mathcal{P}$ to form $\mathcal{P}_r$, and thus $\mathcal{P}_r$ contains at least $2f+1$ correct nodes.

Using the above result and definition 5, which excludes at most $f+1$ nodes from $\mathcal{P}_r$ to form $\mathcal{L}_r$, $\mathcal{L}_r$ contains at least $f+1$ correct nodes. ■

Lemma 4: If a correct node knows the random beacon value R_{r-1}, it can output the random beacon value R_r by the end of round r (independent of the actions of the round's leader ℓ_r).

Proof: Following lemma 3 we guarantee the existence of a leader ℓ_r. Since $\ell_r \in \mathcal{L}_r$ and $\mathcal{L}_r \subset \mathcal{P}_r$, we know that $\ell_r \in \mathcal{P}_r$. By applying definition 4 we get $\ell_r \notin rn(D_{\hat{r}})$. This means that there exists some history of datasets with head $D_{\hat{r}}$ in which there does not exist a recovery certificate $RC(k)$ for any round $k < \hat{r}$ in which ℓ_r was also leader. Such a history for any valid dataset D_k can only exist if at least one correct node confirmed that D_k was correctly distributed and acknowledged by $2f+1$ nodes by providing a confirm message. Hence, at least $f+1$ correct nodes know a common dataset D_k for all rounds k where ℓ_r was previously selected as leader. In addition all nodes know the shares for ℓ_r's first commitment provided (and agreed upon) during the protocol setup. Thus at least $f+1$ correct nodes can (and will) broadcast the decrypted share in case a recovery of the leader ℓ_r in round r is necessary. Hence all nodes learn the value h^{s_ℓ} corresponding to ℓ_r's last commitment $Com(s_\ell)$, and thus obtain R_r using h^{s_ℓ} and R_{r-1} via definition 1. ■

Theorem 1: (Liveness / Guaranteed Output Delivery) For each round r correct nodes output a new random beacon value R_r.

Proof: We use lemmas 3 and 4 and proof the theorem by induction on the round index r. For the base case we have an agreed random beacon value R_0 as given by the protocol setup. For the induction step, we assume that R_{r-1} is known by all correct nodes. Lemma 3 ensures that the set of potential leaders $\mathcal{L}_r$ contains at least $f+1$ correct nodes. Therefore, definition 2 can always be applied to selected a leader ℓ_r using $\mathcal{L}_r$ and R_{r+1}. Hence, we can use lemma 4, to show that by the end of round r each correct node outputs a value R_r. ■

Lemma 5: (Selection of correct leaders) In each interval $\{k, k+1, k+2, ..., k+f\}$ of $f+1$ consecutive rounds there is at least one round $\hat{k} \in \{k, k+1, k+2, ..., k+f\}$ such that the leader $\ell_{\hat{k}}$ of that round is correct.

Proof: We assume that there is no correct leader in $\{\ell_k, \ell_{k+1}, \ell_{k+2}, ..., \ell_{k+f}\}$ and derive a contradiction. We apply the definiton of the set of potential leaders for round $k+f$:

$$\mathcal{L}_{k+f} = \mathcal{P}_{k+f} \setminus \{\ell_k, \ell_{k+1}, ..., \ell_{k+f-1}\}$$

Notice that $\{\ell_k, \ell_{k+1}, ..., \ell_{k+f-1}\}$ denotes a set of f Byzantine nodes. As there are only f Byzantine nodes in total, $\mathcal{L}_{r+f}$ cannot contain any Byzantine nodes. However, the Byzantine node ℓ_{k+f} is leader of round $k+f$ and therefore $\ell_{k+f} \in \mathcal{L}_{k+f}$, which completes the contradiction. ■

Lemma 6: (Agreement on potential leaders) If a node constructs a valid set of potential leaders $\mathcal{L}_r$ in round r then every correct node constructs the same value for $\mathcal{L}_r$.

Proof: Using lemma 5, for the interval $\{r-f-1, r-f, ..., r-1\}$, we know that there is some round $\hat{r}$ with a correct leader $\ell_{\hat{r}}$ in this interval. Using lemma 1, we know that $\ell_{\hat{r}}$ is able to construct a valid dataset $D_{\hat{r}}$ in round $\hat{r}$. As $\ell_{\hat{r}}$ is correct, it has distributed this dataset to all nodes during the propose phase of round $\hat{r}$. All correct nodes therefore acknowledge $D_{\hat{r}}$ in the acknowledge phase of round $\hat{r}$. Since there are at least $2f+1$ correct nodes, all correct nodes receive at least $2f+1$ valid acknowledge messages for $D_{\hat{r}}$ by the end of the acknowledge phase. No node can receive a valid acknowledge for some different dataset $D'_{\hat{r}}$, because the correct leader $\ell_{\hat{r}}$ does not provide a signature for a different value. Therefore, all correct nodes broadcast confirm messages for $D_{\hat{r}}$. As all correct nodes broadcast either one confirm or one recovery message, there are at most f recover messages (by Byzantine nodes). Therefore, there is no valid recovery certificate $RC(\hat{r})$ for round $\hat{r}$. Thus, any valid future dataset needs to (indirectly) reference the common and unique dataset $D_{\hat{r}}$. Consequently, we established agreement on $D_{\hat{r}}$ and its common history provided by the references to the predecessor datasets.

As the set of available nodes $\mathcal{P}_{\hat{r}}$ for round $\hat{r}$ is defined using only the agreed set of all nodes $\mathcal{P}$ and $D_{\hat{r}}$, $\mathcal{P}_{\hat{r}}$ is also agreed upon. Since the definition of $\mathcal{L}_r$ does not depended on whether or not leaders are recovered during the rounds $\{r-f, r-f+1, ..., r-1\}$ and $\hat{r} \geq r-f-1$ agreement on the set $\mathcal{L}_r$ follows. ■

Theorem 2: (Uniform Agreement) If a node outputs a valid random beacon value R_r in round r then every node that outputs a valid beacon value in round r outputs the same R_r.

Proof: We proof the theorem by induction on the round index r. For the base case we have an agreed common random beacon value R_0 as given by the protocol setup.

For the induction step, we assume that every node that outputs a valid beacon value in round $r-1$ output the same R_{r-1}. We have agreement on R_{r-1} by the induction hypothesis and shown agreement on the set of potential leaders $\mathcal{L}_r$ in lemma 6. As the leader selection mechanism given in definition 2 only depends on those two argument, all correct nodes agree on a common unique leader ℓ_r. By applying lemma 4 we obtain that each correct nodes learns the leader's previously commited secret h^{s_ℓ}. By either checking the revealed value of s_ℓ against the leaders commitment or verifiying the validity of the share decryption proof accoring to Scrape's PVSS description [14], uniqueness of a valid h^{s_ℓ} and consequently of R_r is ensured. ■

Theorem 3: (Unpredictability) At the beginning of round r, no node can predict the outcome R_{r+f} of the random beacon protocol in round $r + f$.

Proof: By applying lemma 5 we know that there is at least one correct leader during the interval of the $f + 1$ consecutive rounds $\{r, r+1, r+2, ..., r+f\}$. Let k denote any round during this interval in which the leader ℓ_k is correct. As ℓ_k follows the protocol, it has not distributed the its secret value s_{ℓ_k} to any node at the beginning of round r. Additionally no correct nodes does provide a decrypted secret share, which could be used for the recovery process of the secret value. Therefore only f secret share are available to Byzantine nodes which try to recover the secret in order to compute R_k (and potentially consecutive random beacon values). However, the protocol defines the reconstruction threshold t used by the PVSS scheme to be $f + 1$. Therefore, an adversary cannot obtain the underlying secret before it is revealed or recovered during round k. Consequently, R_k and all consecutive random beacon values (including R_{r+f}) are unpredictable at the begining of round r. ■

Theorem 4: (Bias-Resistance) No node i can, for any round r, influence the value R_r of the random beacon protocol in a meaningful (i.e. predictable) way.

Proof: This property follows from unpredictability and the fact that the protocol is constructed in a way that ensures that any action a (Byzantine) nodes takes in some round r, can only influence the value of the random beacon at round $r+f+1$ or later. In theorem 3 we have shown that the random beacon value at round $r + f$ is unpredictable at the beginning of round r. Therefore, a (Byzantine) node cannot influence the random beacon values for rounds r to $r + f$, and may only influence values at round $r+f+1$ or later in an unpredictable manner. ■

Theorem 5: (Public-Verifiability) For each round r, an external verifier can check the correctness of the random beacon value R_r, at the end of round r.

Proof: The external verifier asks any correct node (i.e. at most $f + 1$ nodes) to provide its history up to and including round r. Then the verifier can, by following the protocol rules, obtain the same random beacon value R_r if and only if the provided history is correct. Additionally, any dataset D_r and a its confirmation certificate $CC(D_r)$ allow an external verifier to obtain and check the random beacon value for round r and all rounds $k \in \{\hat{r} + 1, \hat{r} + 2, ..., r - 1\}$. ■

Lemma 7: (Efficient-Verification) For each round r, an external verifier can check the correctness of the random beacon value R_r, without validation of all previous rounds.

Proof: The external verifier asks any correct node (i.e. at most $f + 1$ nodes) to provide (i) the header of a dataset $D_{r'}$ which includes the value of R_r (either directly or in the list of random beacon values for recovered rounds) and (ii) a confirmation certificate $CC(D_{r'})$ for this dataset. Under our security assumption of $n = 3f + 1$, each valid confirmation certificate includes at least one signature of a honest node, the verification of the confirmation certificate is sufficient to check the validity of R_r. As a confirmation certificate includes $f+1$ signatures this verifcation process in $\mathcal{O}(n)$. ■

VII. Comparison of Random Beacon Protocols

In this section, we provide an overview regarding the characteristics of various random beacon protocols. Thereby, we focus on designs that are suitable as building blocks in (PoS) blockchain protocols. For a broader comparison, we are also include Proof of Work (PoW) and Proof-of-Delay [10], [12] in the given table. Simply speaking, a random beacon value via Proof-of-Delay is computed by repeated hashing of a seed value. Hereby, the number of iterations is set such that a computationally bounded participant cannot bias the random beacon value before the successor blocks are found.

The underlying models, assumptions, notations as well as the context may differ from protocol to protocol. Therefore, comparing existing approaches in this field is a non trivial task. We performed the hereby presented comparison to the best of our knowledge and explicitly state whenever we have not been able to pinpoint certain properties or had to estimate them. Table I presents a first step towards comparing current approaches. A property *prop* is marked as uncertain using the notation ∼*prop* if we have not been able to fully assess the property using the available information. For cells marked with '**?**' we cannot provide an adequate evaluation due to a lack of available information. The symbol ✓ is used to describe that a property is fulfilled, whereas ✗ refers to unfulfilled properties. Additionally, we use (✓) to indicate that a property is achieved with probabilistic guaranties over time. Further information on specific properties is indicated using the notation $prop_{(1-13)}$. For the complexity evaluations, n refers to the number of participants in the network, and c describes the size of the subset used in the specific protocol. Notice that c is different depending on the protocol.

In the following, we provide additional details in regard to the assessment provided in table I:

(1) In [18], the author's claim that their protocol "gracefully handles temporary losses of network synchrony including network splits" but only proof the protocol secure in the synchronous case.

(2) The authors of the RandShare, RandHound and RandHerd protocols explicitly state asynchronous communication only for their RandShare protocol. However, the author's statement "The client chooses a subset of server inputs from each group, omitting servers that *did not respond on time* or with proper values [...]" [26] indicates that the communication model for RandHound is synchronous.

(3) The exact probability is configurable as a protocol parameter. The given value represents a suggestion by the by the respective authors.

(4) Liveness in the asynchronous communication model is only achieved after a barrier point. Whether or not this point is reached depends on the outcome of a Byzantine agreement protocol, which RandShare uses as a subprotocol [26].

(5) The protocols are not presented in a standalone setting and assume an underlying blockchain or bulletin

TABLE I. COMPARISON OF RANDOM BEACON PROTOCOLS

		Communication model	Liveness / Failure probability	Comm. complexity (overall system)	Unpredictability	Bias-Resistance	Comp. complexity (per participant)	Verification complexity (per verifier)	Characteristic cryptographic primitive(s)	Trusted dealer or DKG required
	PoW	syn.	✓	$\mathcal{O}(n)$	(✓)	✗	very high$_{(10)}$	$\mathcal{O}(1)$	hash func.	no
[15]	Algorand	semi-syn.	$10^{-12}{}_{(3)}$	?	(✓)	✗$_{(9)}$	?	$\sim\mathcal{O}(1)$	VRF	no
[13]	Cachin et al.	asyn.	✓	$\mathcal{O}(n^2)$	✓	✓	$\sim\mathcal{O}(n)$	$\sim\mathcal{O}(1)$	uniq. thr. sig.	yes
[3]	Caucus	syn.	✓	$\mathcal{O}(n)_{(5)}$	(✓)	✗$_{(9)}$	$\sim\mathcal{O}(1)$	$\sim\mathcal{O}(1)$	hash func.	no
[18]	Dfinity	syn.$_{(1)}$	10^{-17}	$\sim\mathcal{O}(cn)_{(6)}$	✓	✓	?	$\sim\mathcal{O}(1)$	BLS sig.	yes
[19]	Ouroboros	syn.	✓	$\mathcal{O}(n^3)$	✓	✓	$\mathcal{O}(n^3)_{(11)}$	$\mathcal{O}(n^3)_{(11)}$	PVSS	no
[16]	Ourob. Praos	semi-syn.	✓	?	(✓)	✗$_{(9)}$	?	$\sim\mathcal{O}(1)$	VRF	no
[10]	Proof-of-Delay	syn.	✓	$\mathcal{O}(n)_{(5)}$	✓	✓	high$_{(10)}$	high$_{(13)}$	hash func.	no
[26]	RandShare	asyn.	✗$_{(4)}$	$\mathcal{O}(n^3)$	✓	✓	$\mathcal{O}(n^3)_{(11)}$	$\mathcal{O}(n^3)_{(11)}$	PVSS	no
[26]	RandHound	∼syn.$_{(2)}$	0.08%	$\sim\mathcal{O}(c^2n)$	✓	✓	$\sim\mathcal{O}(c^2n)_{(12)}$	$\sim\mathcal{O}(c^2n)_{(12)}$	PVSS/CoSi	no
[26]	RandHerd	∼syn.$_{(2)}$	0.08%	?$_{(7)}$	✓	✓	$\mathcal{O}(c^2 \log n)$	$\mathcal{O}(c^2 \log n)$	PVSS/CoSi	?
[14]	Scrape	syn.	✓	$\mathcal{O}(n^3)_{(5)}$	✓	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	PVSS	no
	HydRand	syn.	✓	$\mathcal{O}(n^2)$	(✓)$_{(8)}$	✓	$\mathcal{O}(n)$	$\mathcal{O}(n)$	PVSS	no

board as communication channel. The herein presented communication complexity does not account for this additional overhead.

(6) Due to a lack of information, we can only estimate the communication complexity. Assuming that the only communication strictly necessary to produce the random beacon values is the broadcast of partial signatures, which each member of the correct group has to perform, the complexity $\mathcal{O}(cn)$ can be derived. This estimate excludes further potential messages exchange required for Dfinity's group setup.

(7) According to our interpretation, the communication complexity $\mathcal{O}(c^2 \log n)$ for RandHerd is stated per server only. Therefore, this value is not comparable to the other approaches, which consider the communication complexity of the overall system.

(8) HydRand reaches unpredictability with absolute certainty after $f + 1$ rounds in the future. Before that point, the protocol provides unpredictability with increasingly high probability (the likelihood of prediction decreases exponentionally with the number of round to predict).

(9) For Algorand, Ouroboros Praos and the Caucus protocol, our strong notion of bias-resistance is not achieved because malicious leaders can selectively withhold values to bias the produced randomness.

(10) The computation complexity is not dependent on the number of participants and therefore is $\mathcal{O}(1)$. However, as PoW and Proof-of-Delay inherently computation intensive, the notation of $\mathcal{O}(1)$ would be misleading compared to other schemes.

(11) Using the optimization of Schoenmakers' PVSS proposed by the authors of the Scrape protocol, the complexity can be reduced by a factor of n.

(12) Again using Scrape's optimization, the complexity can be reduced. Since the PVSS protocol is executed among a subset of participants, a reduction by a factor of c is possible.

(13) For verification the delay function has to be recalculated. Using checkpoints, the verification time can be parallelized to some extend. The validity / invalidily of values is shown using an interactive process upon disagreement.

VIII. DISCUSSION

For easier comparison and evaluation, HydRand is designed as a stand-alone protocol, but with an application in the area of blockchains in mind. Potential use-cases could be as part of future Proof-of-Stake approaches or permissioned blockchains. A desirable property for random beacons that our protocol can provide is *guaranteed output delivery*, i.e. a new random beacon value is guaranteed to be produced at each round regardless of the adversary's actions. This is important for all application scenarios in which continuous operation is required and an improvement compared to other commitment schemes, such as collateral based approaches which cannot guarantee output delivery at every round.

The main advantage of HydRand, compared to other PVSS based random beacon protocols in this field, is its low communication complexity of $O(n^2)$ compared to $O(n^3)$. The complexity of $O(n^2)$ includes all messages required to establish Byzantine agreement. We estimate the required communication overhead for the overall system for $n = 100$ and $n = 250$. For $n = 100$, a typical round without recovery results in an overall communication amount of $\sim$ 5.4 MB, while a round with recovery leads to $\sim$ 5.6 MB transmitted. In the scenario of $n = 250$, the respective values are $\sim$ 34.0 MB and $\sim$ 31.0 MB. This is an important improvement regarding the practicality of such approaches, and a further step towards widespread deployment. The communication complexity is reduced by shifting the transmission of messages of size n to the leader and employing cryptographically signed *conformation/recovery certificates* to converge on a history of datasets. Messages that need to be broadcasted by all nodes are always of constant size.

Commit/reveal schemes and protocols like Algorand, Ouroboros Praos and Caucus fail to establish the strong notion of bias-resistance we set out to achieve. HydRand ensures that a unique, bias-resistant random number is produced at *every round* of the protocol.

An additional advantage is that Hydrand does not require a trusted dealer or distributed key generation protocol. Furthermore, the protocol setup and execution is publicly verifiable by utilizing the underlying characteristics of the PVSS scheme. Additionally, we do not rely on less established cryptographic primitives such as cryptographic pairings.

However, our protocol assumes $n = 3f + 1$ in a synchronous communication model, while existing efficient BFT protocols (e.g. [1]) achieve a bound of $n = 2f + 1$. Protocols like Algorand and Ouroboros Praos require weaker assumptions in regard to synchrony, while early coin-tossing protocols (e.g. [13]) can operate under full asynchrony.

IX. Conclusion

We propose HydRand, a synchronous Byzantine fault tolerant random beacon protocol that tolerates up to one third Byzantine nodes and show that the protocol achieves *liveness*, *public-verifiability*, *bias-resistance*, as well as *unpredictability* for all output values after (at most) $f + 1$ rounds in the future. The presented protocol is designed for stand-alone usage, but it could also find utility in the context of current and future (PoS) and permissioned blockchain protocols. Furthermore, we provide the first step towards a systematization of novel random beacon proposals, which enables researchers to compare current as well as future designs objectively with each other. Thereby, we show that the HydRand protocol poses an improvement regarding performance and scalability in respect to comparable random beacon solutions.

References

[1] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren. Efficient synchronous byzantine consensus. Cryptology ePrint Archive, Report 2017/307, 2017.

[2] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on Ethereum smart contracts. In *International Conference on Principles of Security and Trust*, pages 164–186. Springer, 2017.

[3] S. Azouvi, P. McCorry, and S. Meiklejohn. Winning the caucus race: Continuous leader election via public randomness. *arXiv preprint arXiv:1801.07965*, 2018.

[4] I. Bentov, A. Gabizon, and D. Zuckerman. Bitcoin beacon, 2016.

[5] I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake, 2016.

[6] M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Eurocrypt*, volume 2656, pages 416–432. Springer, 2003.

[8] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *Advances in Cryptology ASIACRYPT 2001*, pages 514–532, 2001.

[9] J. Bonneau, J. Clark, and S. Goldfeder. On Bitcoin as a public randomness source. *IACR Cryptology ePrint Archive*, 2015:1015, 2015.

[10] B. Bünz, S. Goldfeder, and J. Bonneau. Proofs-of-delay and randomness beacons in Ethereum. In *S&B '17: Proceedings of the 1^{st} IEEE Security & Privacy on the Blockchain Workshop*, April 2017.

[11] V. Buterin. Validator Ordering and Randomness in PoS, 2016. Accessed: 2018-04-24.

[12] V. Buterin. Randao++, 2017. Accessed: 2018-04-24.

[13] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 123–132. ACM, 2000.

[14] I. Cascudo and B. David. Scrape: Scalable randomness attested by public entities, 2017.

[15] J. Chen and S. Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2017.

[16] B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Technical report, Cryptology ePrint Archive, Report 2017/573, 2017., 2017.

[17] Dfinity Stiftung. Threshold Relay: How to Achieve Near-Instant Finality in Public Blockchains using a VRF, 2017. Accessed: 2017-08-20.

[18] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series consensus system, 2018. Rev. 1.

[19] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol, 2016.

[20] S. Micali. Byzantine agreement, made trivial, Apr 2017. Accessed:2018-02-21.

[21] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008.

[22] M. O. Rabin. Randomized byzantine generals. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 403–409. IEEE, 1983.

[23] M. O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256–267, 1983.

[24] randao.org. Randao: Verifiable Random Number Generation, 2017. Accessed: 2018-04-24.

[25] B. Schoenmakers. A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.

[26] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable Bias-Resistant Distributed Randomness. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 444–460. IEEE, 2017.