SoK: Decentralized Randomness Beacon Protocols

Mayank Raikwar[0000−0002−5479−5748] and Danilo Gligoroski[0000−0002−7078−6139]

Norwegian University of Science and Technology (NTNU) Trondheim, Norway
{mayank.raikwar,danilog}@ntnu.no

Abstract. The scientific interest in the area of Decentralized Randomness Beacon (DRB) protocols has been thriving recently. Partially that interest is due to the success of the disruptive technologies introduced by modern cryptography, such as cryptocurrencies, blockchain technologies, and decentralized finances, where there is an enormous need for a public, reliable, trusted, verifiable, and distributed source of randomness. On the other hand, recent advancements in the development of new cryptographic primitives brought a huge interest in constructing a plethora of DRB protocols differing in design and underlying primitives.
To the best of our knowledge, no systematic and comprehensive work systematizes and analyzes the existing DRB protocols. Therefore, we present a Systematization of Knowledge (SoK) intending to structure the multi-faced body of research on DRB protocols. In this SoK, we delineate the DRB protocols along the following axes: their underlying primitive, properties, and security. This SoK tries to fill that gap by providing basic standard definitions and requirements for DRB protocols, such as Unpredictability, Bias-resistance, Availability (or Liveness), and Public Verifiability. We classify DRB protocols according to the nature of interactivity among protocol participants. We also highlight the most significant features of DRB protocols such as scalability, complexity, and performance along with a brief discussion on its improvement. We present future research directions along with a few interesting research problems.

Keywords: Random Beacon · Bias-resistance · Unpredictability · Secret Sharing · Verifiable Delay Function

1 Introduction

Public digital randomness is an essential building component for a large spectrum of applications and protocols. For example, a reliable source of continuous randomness providing high entropy, also known as *random beacon*, is crucial for many security applications. A notion of *coin tossing protocol* [9] was proposed by Blum in 1983 that addressed the question of generating the trustworthy random value in a network of mutually distrustful participants. Further, Rabin [58] formalized the notion of the random beacon. Since then, randomness generation has been advanced significantly due to the underlying modern cryptography.

Lately, coin tossing protocols became more appealing in Proof-of-Work (PoW) or Proof-of-Stake (PoS) [39] consensus. Random beacon has a range of applications that includes cryptographic parameter generation [49], design of byzantine fault tolerant (BFT) protocols [16, 39], privacy-preserving message services [40], e-voting protocols [1], online gaming [13], publicly auditable selections [13], anonymous browsing [41], sharded blockchains [23] and smart contracts [48].

Due to the applicability of shared randomness in a variety of applications, a rich body of literature has emerged that proposes many DRB protocols differing in their designs and underlying cryptography. Nevertheless, the system models and design challenges in these DRB protocols are highly disparate. Therefore, to address these challenges and to provide a general definition of a DRB protocol, we present a Systematization of Knowledge (SoK). The purpose of this SoK is to provide a systematic overview of existing DRB protocols that can help researchers and practitioners to find suitable solutions for randomness generation.

Background. An easy approach to achieve continuous randomness is through a single node or a trusted third party such as NIST [46], Random.org [43] or Oraclize.it [56]. The NIST beacon continuously outputs hardware-generated random values from a quantum-mechanical effect. Since these beacon services are centralized, they can be unreliable. Moreover, in the past, they suffered a significant public trust deterioration after the revealed backdoor in the NSA-designed Dual elliptic curve pseudorandom number generator [68]. Due to these problems, these centralized beacon services are undesirable for secure applications.

As a consequence, *Decentralized Randomness Beacon* (DRB) protocols were proposed and constructed where trust is distributed across multiple nodes that jointly generate random values at a regular interval. More concretely, a consortium of organizations launched a distributed publicly verifiable randomness beacon that periodically provides unbiasable and unpredictable random outputs. The deployment is known as League of Entropy (LoE)) [51] that aims to provide collaborative governance for protection of its random beacon. The consortium believes that their beacon can become a fundamental service on the internet.

DRB protocols can be constructed by employing different cryptographic primitives e.g. Publicly Verifiable Secret Sharing schemes (PVSS) [47, 69, 66, 17, 18, 8, 25], Threshold Crypto-Systems [16, 45, 31, 20, 55], Verifiable Random Functions (VRF) [27, 39, 36, 71, 22, 24], Verifiable Delay Functions (VDF) [49, 30, 34, 65, 44]. Randomness can also be extracted from external data sources such as [7, 21, 13] or from the blockchain schemes having their own random beacon [47, 36, 45]. These DRB protocols are not equally-suited in all applications or use-cases due to the diversity in their designs, characteristics, and underlying assumptions.

DRB protocols differ significantly due to their underlying techniques. A DRB protocol should have a list of desirable beacon properties along with low communication complexity, low computational cost, and low trust requirement (e.g., setup assumptions). Additionally, the DRB protocol should be efficient in practical settings. Therefore, despite having many constructions of DRB protocols, a few problems such as scalability, trust, and network assumptions need to be addressed to construct a desirable DRB protocol for practical applications.

Motivated by the above, the contributions of this SoK are as follows:

1. We provide a formal definition of a Decentralized Randomness Beacon (DRB) with a brief description of its security properties (Sect. 2).
2. We present a classification of DRB protocols in Interactive and Non-interactive DRB protocols and we describe these protocols in detail (Sect. 3).
3. We give a brief discussion on several crucial issues related to DRB protocols, including complexity, scalability, and assumptions. We also identify a few efficient building components to construct efficient DRB protocols (Sect. 4).

2 Decentralized Randomness Beacon (DRB)

A DRB allows a group of participants to collaboratively produce random values without the need of a central party. A DRB consists of n participants [1] $\mathcal{P} = (P_1, P_2, \ldots, P_n)$. These participants are connected in a distributed manner. Without loss of generality, we assume that a DRB protocol works in rounds and maintains a beacon state st for each round. For every round $e \in \{1, 2, \ldots\}$, given the current state st_{e-1}, the DRB protocol collectively produces a random output v_e; the state st_0 is jointly computed and agreed from the protocol participants during the bootstrapping of the DRB protocol. Following, we present a formalization of DRB and we formally define the required security properties of a DRB. Additionally, we define a secure DRB protocol in Appendix A.

Definition 1. *(Decentralized Randomness Beacon (DRB)) A DRB on a set of participants $\mathcal{P} = (P_1, \ldots, P_n)$ is defined as a tuple $\mathcal{B}$ of polynomial algorithms:* $\mathcal{B} = (\mathsf{Setup}, \mathsf{LocalRand}, \mathsf{GlobalRand}, \mathsf{VerifyRand}, \mathsf{UpdateSt})$

- $\mathsf{Setup}(1^\lambda, n)$*: Given input security parameter λ, and n participants, it generates public parameter pp and keypair for each participant (pk_i, sk_i). All participants agree on public parameter pp and $\{pk_i\}$.*
- $\mathsf{LocalRand}(st_{e-1}, pp, sk_i, s_{e,i})$*: Given input state st_{e-1} from round $e-1$, public parameter pp, and input seed $s_{e,i}$, a participant P_i computes a local output value $v_{e,i}$ with a proof $\pi_{e,i}$ using sk_i and $s_{e,i}$ for round e. Output $(i, v_{e,i}, \pi_{e,i})$.*
- $\mathsf{GlobalRand}(st_{e-1}, pp, \mathcal{S} = \{(i, v_{e,i}, \pi_{e,i})\}, m)$*: Given input state st_{e-1}, public parameter pp, a set $\mathcal{S}$ of local output values from $|\mathcal{S}|$ participants, if $|\mathcal{S}| \geq m$, where m is the minimum number of required local output values, the algorithm computes the beacon output v_e for round e by executing a function f on $\{v_{e,i}\}$ from set $\mathcal{S}$. It also computes proof of correctness π_e using $\{\pi_{e,i}\}$ from set $\mathcal{S}$. Output (v_e, π_e) or $\perp$.*
- $\mathsf{VerifyRand}(st_{e-1}, pp, v_e, \pi_e)$*: Given input state st_{e-1}, public parameter pp, a beacon output v_e, and a proof π_e, the algorithm verifies the beacon value v_e and the corresponding proof π_e. Output* $\mathsf{0}$ *or* $\mathsf{1}$*.*
- $\mathsf{UpdateSt}(st_{e-1}, pp, v_e, \pi_e)$*: Given input state st_{e-1}, public parameter pp, a beacon output v_e, and a proof π_e generated at the round e, the algorithm updates the state from st_{e-1} to st_e for round e. Output st_e or $\perp$.*

[1] We use node and participant interchangeably in protocols throughout the paper.

The security properties of a DRB corresponds to: *Unpredictability*: An adversary should not be able to predict (precompute) future beacon outcomes; *Bias-Resistance*: A single participant or a colluding adversary cannot bias the future beacon values; *Availability (or Liveness)*: A single participant or a colluding adversary can not prevent the generation of the new beacon value; *Public Verifiability*: Any third party can verify the correctness of the new beacon value. **Note:** *We use DRB protocols and DRBs interchangeably throughout the paper.*

These formal security guarantees of a DRB protocol are evolved during the time. Initial proposals lack the formal definitions and mathematical proofs of their DRB protocols. Nevertheless, the recent proposals put an emphasis on the security of their protocols. These protocols define and prove the security properties of their DRB using the mathematical properties of the underlying cryptographic primitives. Due to different designs, setting up a formal provability framework for DRBs should define the least common security requirements, therefore, we formulate the desiderata of a DRB protocol as follows where λ is a security parameter and $\mathsf{negl}(\lambda)$ is a negligible function of λ.

Definition 2. *(Unpredictability) Let $\mathcal{A}(v_1, \ldots, v_e, st_e)$ be a probabilistic polynomial time algorithm that receives the values $v_1, \ldots, v_e$ and the current state st_e as the input values. Let $\mathcal{A}$ outputs a value v_{e+f} for any value (future rounds) $f \geq 2$, and for all rounds $e \geq 1$. Then*

$$\Pr[\mathcal{A}(v_1, \ldots, v_e, st_e) = v_{e+f}] \leq \mathsf{negl}(\lambda) \tag{1}$$

Definition 3. *(Bias-Resistance) Let $\mathsf{bit}_i(v_e)$ denotes the i-th bit in the binary representation of v_e, let $b = |v_e|$ is the number of bits of v_e, and let $\mathcal{A}_i(v_1, \ldots, v_{e-1}, st_{e-1})$ for $i = 1, \ldots, b$, be b probabilistic polynomial-time algorithms that receive the values $v_1, \ldots, v_{e-1}$ and the current state st_{e-1} as input and output one bit: 0 or 1. Then for every round $e \geq 1$, every $\mathcal{A}_i(\ )$ and for all $i = 1, \ldots, b$*

$$\Pr[\mathsf{bit}_i(v_e) = \mathcal{A}_i(v_1, \ldots, v_{e-1}, st_{e-1})] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \tag{2}$$

$$\Pr[\mathsf{bit}_i(v_e) = 0] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \tag{3}$$

More concretely, we say that a DRB protocol is Bias-resistant if predicting any single bit of the random beacon output v_e has only a non-negligible advantage over the trivial guessing strategy that has a probability of $1/2$.

Definition 4. *(Availability) Let $\mathcal{A}$ be an adversary controlling a fraction of participants and $\mathcal{P}^h \subseteq \mathcal{P}$ be a set of honest participants in the DRB protocol. Given v_e, π_e, pp and st_{e-1}, for every round $e \geq 1$ and for every participant $P_i \in \mathcal{P}^h$*

$$\Pr[\mathsf{UpdateSt}(st_{e-1}, pp, v_e, \pi_e) \neq st_e] \leq \mathsf{negl}(\lambda) \tag{4}$$

Definition 5. *(Public Verifiability) Given $\mathsf{VerifyRand}(\ )$ as a public probabilistic polynomial-time algorithm run by an external verifier $P_x \notin \mathcal{P}$ that receives v_e, π_e and the state st_{e-1} at the end of round e as input values and outputs a bit $\mathsf{0}$ or $\mathsf{1}$ based on the verification of v_e using π_e. Then for every round $e \geq 1$*

$$\Pr[\mathsf{VerifyRand}(v_e, \pi_e, st_{e-1}) \neq \mathsf{1}] \leq \mathsf{negl}(\lambda) \tag{5}$$

3 DRB Classification

We classify DRB protocols in two ways: *Interactive* and *Non-Interactive*. *Interactive* DRB protocols generate a beacon output in an interactive manner which involves multiple rounds of communication among participants. However, *Non-Interactive* DRB protocols do not involve interactions among participants to produce a random beacon value for each round. Therefore, non-interactive DRBs are preferable for decentralized applications. Nevertheless, the setup for the public parameter generation can be interactive for both types of DRBs.

3.1 Interactive Decentralized Randomness Beacon Protocols

Interactive DRB protocols employ multiple rounds of interaction among participants in order to produce one beacon output. These protocols are constructed using interactive cryptographic primitives such as Publicly Verifiable Secret Sharing (PVSS) or Interactive Threshold Signature Scheme. The existing interactive DRB protocols are based on PVSS involving two logical rounds of coin-tossing wherein the first round, the participants broadcast commitments to their shares, and further, these commitments are revealed in another round. Constructions of DRBs with other interactive cryptographic primitives, we left as open problems.

Research Problem 1 *Construct a DRB protocol based on interactive threshold signature scheme with better complexity compared to existing interactive DRBs.*

The main advantage of PVSS-based DRBs is that the generated randomness is indistinguishable from uniform. Nevertheless, due to the interaction and broadcast, interactive DRBs incur high communication cost. Some of the PVSS-based DRBs improve upon the general PVSS scheme to reduce the communication complexity by utilizing a threshold version of PVSS or electing a committee to perform PVSS or introducing a leader to relay the messages. Hence, these optimized versions of DRB protocols can be used to obtain periodic fresh randomness in real-world applications. A PVSS scheme consists of a tuple of algorithms ($\mathsf{PVSS.Setup}, \mathsf{PVSS.Share}, \mathsf{PVSS.Verify}, \mathsf{PVSS.Recon}$) described in Appendix B.

PVSS-based DRB protocols are mainly of two types: with leader [66, 8, 25] and without leader [47, 17, 18]. In a leader-based protocol, a leader L_e is elected in each round e which is responsible for performing the distribution of the secret shares of the PVSS scheme. A further illustration can be found in Appendix B. Following we present a description of PVSS-based interactive DRB protocols.

– Ouroboros [47]: Ouroboros is a PoS-based blockchain where a set of elected participants run the DRB protocol to fetch the randomness for the leader election. It operates in two phases *commit* and *reveal*. In *commit* phase, participants encrypt the shares for all other participants by running $\mathsf{PVSS.Share}$ and submit the shares on the blockchain. In *reveal* phase, each participant decrypts all the encrypted shares that are encrypted using his public key. Then, each participant computes a local random value using all the decrypted shares and posts it in the blockchain. Finally, a beacon output is computed by performing an XOR operation on all the published local random values.

– RandHound, RandHerd [69]: Syta et al. constructed scalable randomness generation protocols by following client-server architecture and threshold cryptography. RandHound is a one-shot on-demand protocol to generate single randomness. However, RandHerd is a beacon protocol that emits continuous random values. RandHound divides the servers into groups, and each group is responsible for running PVSS among the group members. RandHound employs the commit-reveal technique as defined in Ouroboros for each group. Finally, to produce global randomness in RandHound, a client operates on all the received valid local randomness from each server group. RandHerd improves upon the complexity of RandHound by leveraging communication trees among the server groups and collective signing to produce beacon outputs.
– SCRAPE [17]: Cascudo et al. constructed an honest majority coin-tossing protocol SCRAPE with guaranteed output delivery. It constructs a threshold PVSS scheme where sharing, verification, and reconstruction take only a linear number of exponentiations compared to quadratic in basic PVSS scheme [67]. In SCRAPE, all participants have access to a ledger where messages are posted similar to Ouroboros. Cascudo et al. constructed an efficient share verification procedure with linear complexity by observing the fact that sharing a secret using PVSS is equivalent to encoding the secret with a Reed Solomon error correcting code [61]. The dealer in the PVSS scheme [67] not only encrypts the shares but also commits to the shares. Therefore, to prove that shares in encrypted shares are the same as shares in commitments, the efficient share verification procedure involving error-correcting code is applied. SCRAPE improves the computation and verification cost compared to Ouroboros.
– HydRand [66]: HydRand improves upon the complexity of SCRAPE's PVSS protocol. HydRand works in rounds consisting of three phases: *propose*, *acknowledge* and *vote*. In each round, a leader is selected deterministically from the set of potential leaders and by using the last round randomness. In *propose* phase, the leader reveals his previously committed value which is acknowledged, signed and further broadcast by the other participants in *acknowledge* phase. In *vote* phase, each participant performs some checks, including the checks on the number of received acknowledgments. If the leader does not reveal his secret, the secret is reconstructed using PVSS.Recon. The beacon value is computed using the revealed secret and the last round randomness.
– ALBATROSS [18]: ALBATROSS significantly improves, amortizes the computation complexity of SCRAPE and provides a universal composability (UC)-secure model. It shows efficiency gain through the packed Shamir secret sharing scheme in PVSS or by using a linear t-resilient function to extract randomness as a vector of random group elements. It utilizes Cooley-Tukey fast Fourier transformation to amortize the complexity and for further improvement, it uses $\sum$-protocol to prove that the published sharing is correct. ALBATROSS provides two variants of UC security: 1) First variant uses UC-Non-Interactive Zero-Knowledge (NIZK) proofs for discrete logarithm, 2) Second variant introduces and uses a new primitive named "designated verifier" homomorphic commitments where a sender can open a commitment for one specific receiver. Later, the receiver can prove the same opening to a third party.

– RandPiper [8]: Bhat et al. constructed a reconfiguration-friendly DRB protocol RandPiper with strong security guarantees and quadratic communication complexity. It combines PVSS with State-Machine Replication protocol and presents two protocols: GRandPiper and BRandPiper. GRandPiper is a communication optimal DRB with strong unpredictability in the presence of a static adversary. However, BRandPiper shows the best communication complexity and the best possible unpredictability in case of a dynamic adversary.
– SPURT [25]: SPURT protocol constructs a new PVSS scheme using pairing to produce beacon output and involves a leader. The new PVSS scheme relies on Decisional Bilinear Diffie-Hellman (DBDH) assumption [12]. In addition, SPURT uses State Machine Replication to lower the communication complexity compared to the broadcast channel used by other DRBs e.g., HydRand. SPURT operates in a semi-synchronous network and has no trusted setup.

3.2 Non-Interactive Decentralized Randomness Beacon Protocols

We categorize Non-Interactive DRB (NI-DRB) protocols based on the main constituent cryptographic primitive, further, we illustrate these protocols in Table 1.

VDF-based These DRBs are based on stand-alone Verifiable Delay Function $\mathsf{VDF} = (\mathsf{VDF.Setup}, \mathsf{VDF.Eval}, \mathsf{VDF.Verify})$ described in Appendix C. A VDF is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that takes a prescribed number of sequential steps to compute the output and provides exponentially easy verification of the output. In a VDF-based DRB, the participants evaluate an Iteratively Sequential Function (ISF) to generate their local random values. The verification of these values can be efficiently done using $\mathsf{VDF.Verify}$. Due to the non-parallelizable property of VDF, an adversary cannot bias the output of the random beacon.

Lenstra and Wesolowski [49] constructed a DRB protocol, Unicorn, using a slow-time hash function named *sloth*. This function takes inputs from a set of distrusting participants and outputs a random value. Keeping Unicorn protocol as a successor to VDF, the following VDF-based DRB protocols are constructed.

– Minimal VDF Randomness Beacon [30]: Justin Drake constructed a minimal randomness beacon using RANDAO [59] and VDF. RANDAO is a smart contract based DRB where participants submit their local entropy to the smart contract, and further, the smart contract produces a single global entropy. RANDAO biasable entropy is used as input to the VDF to produce unbiasable randomness. Nevertheless, there is no formal security analysis of this protocol.
– Continuous VDF [34]: Ephraim et al. presented a new notion of Continuous Verifiable Delay Function (cVDF) by adapting Pietrzak scheme [57]. A cVDF f provides the output computation of each intermediate steps (i.e. $f(t)$ for $t < T$) with an efficient proof π^t (used for public verification of the output). A cVDF can be used to construct a DRB protocol where beacon outputs are generated by applying a suitable hash to the intermediate outputs of each step. The drawback with this protocol is that the nodes having the most efficient (fastest) processors can always learn the beacon outputs before the other participating nodes. A similar argument goes for the Unicorn protocol.

– RandRunner [65]: RandRunner leverages trapdoor VDF with strong uniqueness to construct a DRB protocol. Each participant P_i of RandRunner initializes its public parameter pp_i with a corresponding trapdoor sk_i. The participants exchange their public parameters and verify the received ones. RandRunner executes in consecutive rounds where in each round, a leader is elected. Further, the leader tries to solve the VDF using its trapdoor, and other participants attempt to solve the VDF using the common $\mathsf{VDF.Eval}$ algorithm. The drawback with the RandRunner protocol is that once a powerful adversary becomes a leader, it can keep corrupting the round leaders (e.g., via DoS), withhold its output computed via trapdoor, and keep working on for the next outputs for many subsequent rounds hence breaking unpredictability.
– RANDCHAIN [44]: RANDCHAIN is a competitive DRB where in each round, nodes compete to be a leader which solely produces the beacon output. RANDCHAIN constructs a non-parallelizable Sequential Proof-of-Work (SeqPoW) puzzle by employing VDF or Sloth. A node solves the SeqPoW puzzle by incrementing an ISF for a randomized time. RANDCHAIN works as a Nakamoto-based blockchain where nodes synchronize their local blockchains and keep solving the puzzle to mine new blocks to the main blockchain. RANDCHAIN mimics a blockchain structure, so it can suffer from front-running (block withholding) attacks and can also have forks due to problems with blocks' finality.

VRF-based These DRBs compute randomness using Verifiable Random Function $\mathsf{VRF} = (\mathsf{VRF.KeyGen}, \mathsf{VRF.Eval}, \mathsf{VRF.Verify})$ described in Appendix D. A VRF is a pseudorandom function that produces pseudorandom output along with proof about the correctness of the output. Participants in these DRBs apply VRF on an input seed to generate their local entropy which is used to compute the beacon output. VRF-based DRBs are explained as follows:

– Blockchain Protocol Designs: Ouroboros Praos [27], Algorand [39] and Erlond [33] blockchains have their DRB as a byproduct. In these DRBs, each participant P_i runs $\mathsf{VDF.Eval}$ on a seed (e.g., previous output or state) using its secret key sk_i and the DRB output is computed from the participants' VRF outputs. These DRBs do not guarantee generation of uniformly random values and do not have strong bias-resistance as an adversary can include/exclude the corrupted participants' VRF outputs used for DRB output computation.
– Distributed VRF-based DRBs: A distributed VRF (DVRF) [29] based DRB was first introduced by DFINITY [45]. Later DRBs [22, 24] employed DFINITY-DVRF along with BLS cryptography. Nevertheless, these DRBs do not provide formal security analysis. A recent paper [36] provides two new constructions of DVRF: 1) DDH-DVRF based on elliptic curve cryptography; 2) GLOW-DVRF based on cryptographic pairings. These constructions also formalize a security model with proper security analysis. DRBs based on DDH-DVRF, and GLOW-DVRF show strong bias resistance and strong pseudorandomness.
– RandChain [71]: RandChain follows *commit-and-reveal* strategy by building a sub-routine *RandGene* using VRF. RandChain has a two-layer hierarchical blockchain structure where nodes form distinct committees. Each committee

has a local blockchain and generates local entropy through the RandGene protocol, further, global randomness is computed from these local entropy by forming a RandChain block. RandChain security depends on a secure sharding process, followed by a leader election for each shard (committee). However, both processes can be influenced by an adversary to obstruct DRB properties.

HE-based These DRBs utilize homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.Setup}, \mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$. Homomorphic encryption allows performing arithmetic operations on ciphertext directly without decryption (details in Appendix E). Following DRBs employ ElGamal encryption [32] as partial HE.

- Nguyen-Van et al. [55]: Their DRB has three components: a Requester, a Public Distributed Ledger (PDL), and a Core Layer. The protocol works in rounds where, first, the Requester sends a nonce to the PDL that computes a ticket T and publishes it. Further, participants of the core layer run a VRF using the ticket T to check if they are selected as a contributor. Each contributor publishes a ciphertext computed on a random value using the Requester's public key. Later, the Requester performs a homomorphic operation on the published ciphertexts and computes a single ciphertext. Finally, the Requester publishes the decrypted value as DRB output with a proof of correct decryption. There are two drawbacks: 1) A malleable ElGamal encryption, 2) The Requester can collude with contributors or refuse to decrypt the resulting ciphertext.
- HERB [20]: Homomorphic Encryption Random Beacon (HERB) DRB uses threshold ElGamal encryption scheme with a distributed key generation (DKG) protocol. DKG is used to generate a common public key and secret key shares for participants. Each participant publishes a ciphertext share with proof of correct encryption (NIZK Proof) on a public bulletin board. These shares generate an aggregated ciphertext through ElGamal aggregation which is subsequently decrypted by a threshold of participants to produce the DRB output.

External Source-based In these DRBs, participants extract the randomness from an external entropy source, i.e., real-world entropy. These entropy sources can be public blockchains [14, 7], real-time financial data [21] or national lottery results [3]. PoW-based blockchains are promising sources but an adversarial miner can manipulate the generated randomness. Therefore, to achieve most of the beacon properties, the following DRBs apply different defense mechanisms.

- Rand Extractor [21]: Clark et al. [21] created a model to generate randomness by combining the information theory with computational finance. They used the closing prices of the stock market to compute a random output. During the market's closing in the day, one entity publishes this random output in the protocol. This entity can also induce its own local entropy to transparently construct a publicly verifiable final randomness, but liveness is hard to achieve.
- Proofs of Delay [14]: In this DRB, a beacon smart contract (BC) publishes the random beacon values on a public blockchain. The DRB is built on *Proof-of-Delay* which uses an ISF such as sloth [49]. In this DRB, a beacon maintainer executes this ISF and publishes the result to BC with queryable access to the

Scheme	Advantages	Disadvantages
VDF-based	1. These DRBs achieve liveness under the period of full asynchrony. 2. These DRBs avoid byzantine agreement consensus hence have less communication complexity. 3. It shows strong bias-resistance as long as there is an honest node.	1. Front-running attack can hinder some DRB properties. 2. In most of these DRBs, the significant powerful adversary can learn the output of DRB earlier than other nodes. 3. These protocols rely on the new assumptions of VDF.
VRF-based	1. Most of these DRBs do not have any trusted setup and achieve strong notion of pseudo-randomness and bias-resistance [36]. 2. These DRBs incur less computation and communication cost.	1. In some of these DRBs, leader uniqueness is not guaranteed that introduces additional consensus protocol to agree on the beacon output.
HE-based	1. The output of these DRBs for a round e does not depend on the output of the previous round $e-1$. 2. Partial homomorphic encryption schemes used in these DRBs can be replaced by a lattice-based fully homomorphic scheme to ensure the post-quantum security.	1. Scalability issue due to the homomorphic evaluation of multiple ciphertexts. 2. The existing DRBs use public ledger to publish the local and global entropy. But distributing the local entropy in DRBs using a consensus incur a high communication cost.
External Source -based	1. These DRBs do not incur communication cost as the DRB output is published in a public bulletin board. 2. These DRBs work perfectly even in the asynchronous network.	1. Most of these DRBs do not provide public verifiability. 2. Proof-of-Work based beacons are not energy efficient and nodes with better hardware can outperform other nodes in producing the beacon output.
Threshold Signature -based	1. These DRBs provide strong bias resistance and unpredictability. 2. Consortium of organizations can participate to construct such beacon due to threshold property (e.g. Drand [31]).	1. These DRBs require either a trusted setup or DKG, hence do not offer a reconfiguration-friendly setup. 2. Security of the DRBs depend on the security assumptions of elliptic curve pairings due to the use of BLS-signature.

Table 1. Advantages and Disadvantages of different Non-Interactive DRB protocols

beacon output using a *refereed delegation of computation* protocol. To show the honest behavior, the maintainer is incentivized; otherwise punished.

– Bitcoin Beacon [13] [7]: These DRBs extract randomness from the bitcoin blockchain [53] and follows the security of the bitcoin. In [13], an extractor fetches the randomness from the block headers. As each block contains several transactions involving ECDSA signatures [37] that rely on strong randomness for security hence, the extractor gives good public randomness as a beacon output. Bentov et al. [7] constructed a bitcoin beacon protocol that fetches m consecutive blocks $B_1, B_2, \ldots, B_m$ such that the block B_m already have l subsequent blocks. Further, the protocol acquires a bit b_i from each block and runs a majority function on all these bits as input to get the DRB output.

Threshold Signature-based These DRBs are based on a non-interactive threshold signature scheme that requires a single round of communication among participants to produce the unique group signatures from a threshold number of participants' signature shares. Most of the existing threshold signature-based DRBs employ threshold BLS signature. These DRBs require a setup to generate the secret shares for the participants. Additionally, the complexity of unique signature construction comply with DRB protocol for practical use.

– Cachin et al. [16], Drand [31]: Cachin et al. presented a common coin protocol using threshold signature along with a random-access coin-tossing scheme. In this DRB, a trusted dealer distributes the secret key shares to the participants. The DRB output is a unique signature on the hash of a counter (epoch number). Drand [31] follows a similar idea, but it replaces the threshold secret to the threshold BLS key. Drand can be considered as an implementation of the Cachin et al. scheme. Drand utilizes the DKG protocol of Gennaro et al. [38] during the setup phase that yields a high communication complexity.
– DFINITY [45]: It also employs a threshold BLS signatures scheme but the selection of the best initialization vector in the scheme creates a challenge. The protocol works well even in the partial synchronous network model. It employs a non-interactive DKG setup and achieves better communication complexity than Drand. The DRB acts as a VRF that produces unbiasable output.

Note: We present Hybrid DRB protocols in Appendix F.

4 Discussion

4.1 Security Assumptions

The security of all these DRBs depends on well-defined security assumptions. These assumptions can be assumptions about the underlying network, adversary, setup, or cryptographic primitives. If these assumptions are failed in some cases, then the DRB using these assumptions will break its security properties.

- *Cryptographic Assumptions (Primitive)*: As the above described DRBs are based on cryptographic primitives such as PVSS, VDF, VRF, these DRBs inherit the security assumptions from the primitives. These assumptions are well-known hard problems of cryptography such as standard decisional or computational Diffie-Hellman assumptions [10] (or their variants) depending on the underlying cryptographic scheme (e.g., PVSS, DVRF). VDF-based DRBs depend on the new security assumptions on sequential computation (e.g., iterated squaring over groups of unknown order [62]) that are not well studied and understood in the current literature. Modeling of the hash function as a random oracle [6] is also considered in security assumptions in some DRBs.
- *Network Assumptions (Model)*: Most of the PVSS-based and VRF-based DRBs assume a strong *synchronous* network which can be an unrealistic setting in the real world. Hence, these DRBs require a lock-step synchronous network where the messages are delivered before the end of each round. In case of no lock-step synchrony, participants might employ round synchronization protocols [54, 74]. Some of the DRBs work well in *semi-synchronous* network where the messages are delivered within a known finite time-bound. VDF-based and external-source-based DRBs work well in an *asynchronous* network where messages are delivered without a known time-bound. However, the trust of these models depends on the underlying setup assumptions or on the public blockchain, where the local entropy of the participants are posted.
- *Setup Assumptions*: Many DRB protocols [31, 20, 8, 69, 45] require an initial trusted setup assumption where private keys for the participants and uniformly random public parameters are generated by a trusted third party (dealer) or by a distributed key generation (DKG) protocol. The Security of DRBs with a trusted third party crucially depends on the action and ability of the trusted party. Nevertheless, DKG incurs a considerable setup cost (high communication complexity) with its limitation of adding or replacing the participants. Therefore, DKG-based DRBs are preferred when the participants are fixed. Hence, many recent DRB protocols [66, 17, 27, 2, 39] have a transparent setup where the public parameters are trapdoor free.

Following the above security assumption, most of the DRB protocols perform well in permissioned systems. However, permissionless systems have a highly dynamic set of nodes that maintain the system state. Due to the dynamically changing participants, integrating an existing DRB with the system is challenging. Moreover, setting the assumption on a number of adversarial nodes is hard.

Research Problem 2 *Study the hardness of embedding the existing DRB protocols in permissionless systems, based on Proof-of -Work (PoW) or -Stake (PoS).*

4.2 Complexity

DRB protocols following different approaches exhibit different complexity. Finding a good balance between computation and communication complexity in a DRB protocol is a challenging task. Therefore, an extensive amount of work has been devoted to reduce the complexity of DRB protocols.

– *Communication Complexity*: Most of the interactive DRB protocols assume a broadcast channel. Therefore, Ouroboros [47], RandShare [69], and SCRAPE [17] have a communication complexity of $\mathcal{O}(n^3)$ due to the broadcasting of $\mathcal{O}(n)$ size message. HydRand [66] improves upon the communication complexity to $\mathcal{O}(n^2)$ by having a leader-based approach where a leader node performs the PVSS share distribution. Relaying the messages through a single node to reduce the communication complexity is also embraced by ALBATROSS [18], GLOW [36]. RandHound, RandHerd [69], DFINITY [45] employ sharding to sample a committee for output generation that results in lower communication complexity. But such a procedure can be immediately subject to attacks by an adaptive adversary who can corrupt the committee once it is determined. Most of the non-interactive DRBs [27, 39, 14, 55] have less communication complexity as a successful participant (e.g., leader) usually need to perform one broadcast. Therefore, it incurs the communication complexity in $\mathcal{O}(n)$. Moreover, most of the NI-DRBs involving blockchain [59, 44] to publish shared local and global randomness also have lower communication complexity.
DKG setup based DRBs [20, 45, 31, 16] suffer from additional communication cost. Complexity of DRBs can be improved using asynchronous data dissemination (ADD) [26] or using hbACSS [75](for PVSS-based DRBs).

Research Problem 3 *Design a DRB protocol with sub-quadratic communication complexity together with optimal fault-tolerance.*

– *Computation Complexity*: It is defined as the number of operations needed to be performed by a participant during one round of DRB protocol. PVSS-based protocols such as RandShare [69] and Ouroboros [47] requires a computation complexity of $\mathcal{O}(n^3)$. An improved version of PVSS further reduces this cost in SCRAPE [17]. Puzzle-based DRBs [13, 14] have a high computational cost due to the involved puzzle. VDF-based DRBs also have the drawback of high computational complexity due to the repeated squaring. On the contrary, VRF-based DRBs incur a minimum computation cost.

Research Problem 4 *Design a puzzle-based DRB protocol incurring low computation complexity.*

– *Verification Complexity* Verification cost refers to the number of operations performed by an external participant to verify the output of a beacon protocol. Although VDF-based DRBs have high computational costs, they do provide efficient verification hence incur less verification cost. The most efficient DRB protocols with regard to computation and verification complexity are based on VRF [27, 39, 71] or threshold crypto-systems [16, 45, 31, 55].

Research Problem 5 *Design a PVSS-based DRB protocol with a constant verification complexity, linear communication cost and no trusted setup.*

4.3 Scalability

Despite a decade of research on DRB protocols, only quite a few recent DRBs emphasize the scalability of their DRB. Scalability in a DRB protocol refers to the number of participants it can support. Many of the described DRB protocols do not offer good scalability. Especially, DRBs involving DKG setup provide poor scalability as DKG does not support frequent modification in the set of key holders. In addition, the high complexity along with the underlying network model in many of these DRBs significantly affect the scalability of the DRBs.

A general approach for achieving good scalability is "*sharding*" which is considered in recent DRBs, including RandHerd [69], DFINITY [45] and Algorand [39] with the cost of slightly degrading the fault-tolerance. RandHerd shows a direct consequence of the sharding where nodes are split into smaller groups. Each group produces local entropy and each group's entropy is combined to produce the DRB output. Algorand and DFINITY show selection of a committee to generate the DRB output. Therefore, this orthogonal technique of randomly sampling a committee for protocol execution can improve scalability.

Another way for improving scalability is using a *leader-based approach* where a leader relays the messages to the participants. Moreover, having a *public ledger* where participants post their local entropy messages also improves scalability.

Reconfiguration Friendliness directly impact scalability. A protocol is reconfiguration friendly when the parameters and list of participants can be changed dynamically without affecting the current execution. When there is no binding between the setup and the system, the reconfiguration becomes easier. DRBs involving DKG setup are not reconfiguration-friendly, hence poor scalability. On the contrary, non-interactive DRBs (with no DKG) show better scalability.

Research Problem 6 *Study the (im)possibility of designing a reconfiguration-friendly sub-quadratic DRB protocol that do not employ committee sampling.*

4.4 Adversarial Model

Most of these DRBs consider a fixed set of n nodes; out of these nodes, f nodes may exhibit byzantine behavior. An adversary in these DRBs can be defined as:

Active vs. Passive An active adversary actively modifies the messages (e.g., public shares, DRB output) in DRB; a passive adversary observes the transcript (i.e. messages) of an honest run of the DRB and predicts the DRB's next output.

Adaptive vs. Static An adaptive adversary corrupts the nodes during the protocol execution, while a static adversary does corruption before the execution.

An adversary can affect the security guarantees of a DRB in many ways, such as by trying to bias the produced random output, withholding the output, predicting the future output, or tricking an outsider (third party) into accepting invalid beacon output. Leader-based DRBs suffer from targeted attacks, however, blockchain-based DRBs suffer from blockchain-specific attacks. Moreover, unpredictability can be affected by network model.

Research Problem 7 *Choose a static secure DRB protocol and transform it to an adaptively secure DRB that retains the efficiency standard of the static one.*

Table 2. Comparison of Existing Decentralized Randomness Beacon Protocol

Protocol	Network Model	Adaptive Adversary	Liveness	Unpredictability	Bias-Resistance	Fault-tolerance	Communication Complexity	Computation Complexity	Verification Complexity	Cryptographic Primitive	No Trusted Dealer or DKG required
ALBATROSS [18]	syn.	✗	✓	✓	✓	1/2	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	PVSS	✓
Algorand [39]	semi-syn.	✗	✓	✓‡	✗	1/3$^{\diamond}$	$\mathcal{O}(cn)$	$\mathcal{O}(c)$	$\mathcal{O}(1)$	VRF	✓
BRandPiper [8]	syn.	✓	✓	✓	✓	1/2	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	PVSS	✓
Cachin et. al [16]	asyn.	✗	✓	✓	✓	1/3	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	Uniq. thr-sig.	✗
Caucus [2]	syn.	✗	✓	✓‡	✗	1/2	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Hash func.	✓
Continuous VDF [34]	asyn.	✗	✗†	✓	✓	1/2	$\mathcal{O}(1)$	VDF	$\mathcal{O}(1)$	VDF	✓
DFINITY [45]	semi-syn.	✗	✓	✓	✓	1/3	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	BLS thr-sig.	✗
Drand [31]	syn.	✗	✓	✓	✓	1/2	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	Uniq. thr-sig.	✗
GLOW [36]	syn.	✗	✓	✓	✓	1/3	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	DVRF	✗
GRandPiper [8]	syn.	✗	✓	✓‡	✓	1/2	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	PVSS	✓
HERB [20]	syn.	✗	✓	✓	✓	1/3	$\mathcal{O}(n^2)^{*}$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	PHE	✗
HydRand [66]	syn.	✗	✓	✓‡	✓	1/3	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	PVSS	✓
Nguyen-Van et. al [55]	syn	✗	✗	✓	✗	1/2	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	PHE, VRF	✓
Ouroboros [47]	syn.	✗	✓	✓	✓	1/2	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	PVSS	✓
Ouroboros Praos [27]	semi-syn.	✓	✓	✓‡	✗	1/2	$\mathcal{O}(n)^{\star}$	$\mathcal{O}(1)^{\star}$	$\mathcal{O}(1)^{\star}$	VRF	✓
Proof-of-Delay [14]	syn.	✗	✓	✓	✓	1/2	$\mathcal{O}(n)$	very high	$\mathcal{O}(\log \Delta)^{\circ}$	Hash func.	✓
Proof-of-Work [53]	syn.	✗	✓	✓‡	✗	1/2	$\mathcal{O}(n)$	very high	$\mathcal{O}(1)$	Hash func.	✓
RandChain [71]	syn.	✗	✓	✓	✓	1/3	$\mathcal{O}(cn)$	$\mathcal{O}(cn)$	$\mathcal{O}(n)$	VRF	✓
RANDCHAIN [44]	syn.	✓	✓	✓	✓	1/3	$\mathcal{O}(n)$	VDF	$\mathcal{O}(1)$	VDF	✓
RANDAO [59]	asyn.	✗	✓	✗	✗	1/2	$\mathcal{O}(n)$	VDF	$\mathcal{O}(1)$	VDF	✓
RandHerd [69]	syn.	✗	✓	✓	✓	1/3	$\mathcal{O}(c^2 \log n)$	$\mathcal{O}(c^2 \log n)$	$\mathcal{O}(1)$	PVSS, CoSi	✗
RandHound [69]	syn.	✗	✓	✓	✓	1/3	$\mathcal{O}(c^2 n)$	$\mathcal{O}(c^2 n)$	$\mathcal{O}(c^2 n)$	PVSS	✓
RandRunner [65]	syn.	✓	✓	✓‡	✓	1/2	$\mathcal{O}(n^2)$	VDF	$\mathcal{O}(1)$	VDF	✓
RandShare [69]	asyn.	✓	✗$^{\odot}$	✓	✓	1/3	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	VSS	✓
Rand Extractor [21, 13]	asyn.$^{\pm}$	✓	✓$^{\text{II}}$	✓	✓	1/2	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Hash func.	✓
SCRAPE [17]	syn.	✗	✓	✓	✓	1/2	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	PVSS	✓
SPURT [25]	semi-syn	✗	✓	✓	✓	1/3	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	PVSS, Pairing	✓
Unicorn [49]	asyn.	✗	✓†	✓	✓	1/2	$\mathcal{O}(1)$	high	$\mathcal{O}(1)$	Sloth	✓

Fault-tolerance refers to number of byzantine faults a DRB can tolerate and c is average committee size.
‡ refers to probabilistic guarantees for unpredictability, and has a bound on the number of future rounds an adaptive rushing adversary can predict the beacon output.
$\diamond$ Due to the randomly sampling a committee of size c in Algorand, the fault-tolerance reduces slightly.
† The node with more computational power learns the beacon output earlier than others.
$*$ HERB achieves communication complexity of $\mathcal{O}(n^2)$ when nodes use Avalanche algorithm or public blockchain to share their ciphertexts.
$\star$ Ouroboros Praos is not a stand-alone DRB and does not describe randomness generation approach, so the presented complexity does not account the additional complexity for communication or verification.
$\circ$ The verification in beacon smart contract has complexity $\mathcal{O}(\log \Delta)$ in the security parameter Δ.
$\odot$ An additional synchrony assumption is needed to provide liveness in RandShare.
$\pm$ DRB protocols built on public blockchain also follow the network structure of the respective blockchain. Therefore, [13] uses a synchronous network as it relies on the bitcoin blockchain.
II Liveness can be hindered due to the limited availability of financial data caused by closed exchanges in [21] or due to the fork situation in the blockchain in [13].

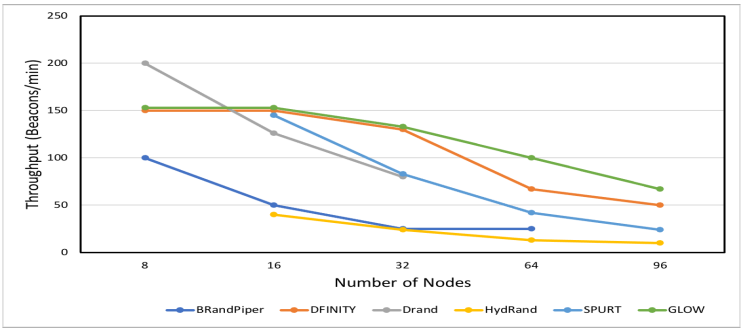

Fig. 1. Overview of throughput for various DRB protocols

4.5 Throughput Evaluation

We report throughput of state-of-the-art public implementations of various DRBs in Figure 1. DFINITY and SPURT operate on a semi-synchronous network but BRandPiper, Drand, GLOW, and HydRand assume synchronous networks. The network delay parameter in these DRBs directly affects the throughput of DRBs.

Drand is a practically deployed DRB protocol. However, when the number of nodes increases to more than 64, nodes in the DRand abort the DKG step of the protocol, and yet it suffers from significant network delay. For HydRand, we chose the public implementation [64] available on Github. For BRandPiper, we depict throughput for the Merkle-tree-based implementation, which is quantitatively practical for real-world scenarios. For SPURT, we used the throughput values directly from their paper. For DFINITY and GLOW, we followed the public implementation [35] of DVRFs to get the throughput while assuming no failure.

4.6 Others

- *Incentive* Some DRBs [14, 59] involve incentivizing or punishing the participants to enforce fairness against rational adversaries. In particular, these (dis)incentivizing approaches (e.g., [4]) are considered mostly in smart contract-based DRBs. Putting an economic incentivization scheme to reward the participants of beacon enforces the honest behavior from the participants. An incentivization scheme can also reward the right computation or verification. Interesting research would be to create an *incentive structure* for a DRB.
- *Output Uniqueness* It states that the DRB produces a unique output even in the presence of an adversary having the trapdoor information of honest participants. It implies strong bias-resistance in DRBs. Therefore, DRBs such as RandRunner [65], DFINITY [45] and GLOW [36] provide strong bias-resistance due to their output uniqueness. Having this property also prevents an adversary from manipulating the beacon output for any financial gain.
- *Universal Composability* (UC) It is arguably one of the strongest security guarantees. A UC-secure protocol ensures that the protocol can be employed as a building block in more complex systems while preserving its security. The earlier UC-secure DRBs [47] do not provide bias-resistance. The first UC-secure DRB ALBATROSS [18] leverages UC-secure NIZK proofs. UC-secure time-lock puzzles (TLP) [5] can be scrutinized to construct a UC-secure DRB.

5 New Components for Construction of DRB Protocols

There have been many new efficient constructions of cryptographic primitives in recent years. These primitives can be embedded as new building blocks or replace old ones in the DRB protocols to improve the performance of DRBs.

- *Using New Verifiable Functions*: Gurkan et al. [42] constructed a new aggregatable DKG scheme that leverages gossip instead of broadcast communication to reduce the communication complexity. Further, they introduced an efficient Verifiable Unpredictable Function (VUF) and combined it with DKG. This threshold VUF can be utilized to construct a DRB protocol.
  VDF-based DRBs can benefit from the recent work [63] about batch verification of VDF in which the verification of beacon outputs during the last several rounds can be batched and verified efficiently. Work [52] on VDF can be investigated and applied to construct a practical VDF-based DRB protocol.
  Current DRBs are not Post-Quantum (PQ) secure (except DRB [50] that does not depend on any third party to construct a quantum-safe beacon). Recent constructions of Post-Quantum VRF [15] and Post-Quantum VDF [19] can be carefully studied and applied to construct practical PQ-secure DRB protocols.
- *Using New Threshold Signatures*: Tomescu et al. [70] designed a fast BLS-based threshold signature scheme (BLS-TSS). Their scheme has fast signature aggregation and verification. There are some DRBs that use the BLS signature scheme. The new BLS-TSS scheme can be directly applied to these DRBs to improve their performance. Otherwise, a new large-scale, simple DRB protocol can be designed and implemented using this new BLS-TSS scheme.
- *Using New Erasure Codes*: All known Verifiable Secret Sharing (VSS) schemes published so far in the open literature (without exception) use the well-known Reed-Solomon codes [61]. Reed-Solomon codes are Maximum Distance Separable (MDS) erasure codes of type (t, n), where the original message is equally split in t parts and is encoded to n (where $n > t$) parts. In the recent decade, the coding theory community constructed new MDS erasure codes. The most significant line of work was done by Dimakis et al., in [28] where they constructed Minimum Bandwidth Regenerating (MBR) codes (optimal in terms of the repair bandwidth) and Minimum Storage Regenerating (MSR) codes (optimal in terms of the storage). Soon after that, those MDS codes were practically employed in Facebook data centers [60], and new variants of MDS codes e.g. [73] were proposed. Therefore, it would be interesting to research the potential replacement of the Reed-Solomon code in VSS schemes with another MDS (MSR) code to improve the performance of VSS-based DRBs.

6 Conclusion

Within recent years, there has been a dramatic surge in the construction of new Decentralized Randomness Beacon (DRB) protocols due to its emergence in cryptographic protocols. We present the first systematization of knowledge (SoK) for the existing efforts on DRB protocols. This SoK provides a comprehensive

review of the design paradigms and approaches of DRB protocols. This SoK can serve as a starting point to explore DRB protocols and can help researchers or practitioners to pick a DRB protocol well-suited for their application.

In this SoK, we presented basic standard definitions of a DRB protocol and its required properties. We discussed the key components and the most significant features of DRB protocols and summarized the existing DRB protocols in Table 2. We identified several research challenges related to the complexity, scalability, and security of DRB protocols. We highlighted respective solutions to encounter some of the challenges. Finally, we proposed promising research directions for the future design of DRB protocols by employing the new cryptographic components that can help to advance the state-of-the-art of DRB protocols.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX security symposium. vol. 17, pp. 335–348 (2008)
2. Azouvi, S., McCorry, P., Meiklejohn, S.: Winning the caucus race: Continuous leader election via public randomness. arXiv preprint arXiv:1801.07965 (2018)
3. Baigneres, T., Delerablée, C., Finiasz, M., Goubin, L., Lepoint, T., Rivain, M.: Trap me if you can-million dollar curve. IACR Cryptol. ePrint Arch. p. 1249 (2015)
4. Baum, C., David, B., Dowsley, R.: Insured mpc: Efficient secure computation with financial penalties. In: International Conference on Financial Cryptography and Data Security. pp. 404–420. Springer (2020)
5. Baum, C., David, B., Dowsley, R., Nielsen, J.B., Oechsner, S.: Tardis: A foundation of time-lock puzzles in uc. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 429–459. Springer (2021)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73 (1993)
7. Bentov, I., Gabizon, A., Zuckerman, D.: Bitcoin beacon. arXiv preprint arXiv:1605.04559 (2016)
8. Bhat, A., Shrestha, N., Kate, A., Nayak, K.: Randpiper-reconfiguration-friendly random beacons with quadratic communication. IACR Cryptol. ePrint Arch. **2020**, 1590 (2020)
9. Blum, M.: Coin flipping by telephone a protocol for solving impossible problems. ACM SIGACT News **15**(1), 23–27 (1983)
10. Boneh, D.: The decision diffie-hellman problem. In: International Algorithmic Number Theory Symposium. pp. 48–63. Springer (1998)
11. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 757–788. Springer International Publishing, Cham (2018)
12. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: International conference on the theory and application of cryptology and information security. pp. 514–532. Springer (2001)
13. Bonneau, J., Clark, J., Goldfeder, S.: On bitcoin as a public randomness source. IACR Cryptol. ePrint Arch. **2015**, 1015 (2015)
14. Bünz, B., Goldfeder, S., Bonneau, J.: Proofs-of-delay and randomness beacons in ethereum. IEEE Security and Privacy on the blockchain (IEEE S&B) (2017)

15. Buser, M., Dowsley, R., Esgin, M.F., Kermanshahi, S.K., Kuchta, V., Liu, J.K., Phan, R., Zhang, Z.: Post-quantum verifiable random function from symmetric primitives in pos blockchain. IACR Cryptol. ePrint Arch. **2021**, 302 (2021)
16. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. Journal of Cryptology **18**(3), 219–246 (2005)
17. Cascudo, I., David, B.: Scrape: Scalable randomness attested by public entities. In: International Conference on Applied Cryptography and Network Security. pp. 537–556. Springer (2017)
18. Cascudo, I., David, B.: Albatross: publicly attestable batched randomness based on secret sharing. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 311–341. Springer (2020)
19. Chavez-Saab, J., Henríquez, F.R., Tibouchi, M.: Verifiable isogeny walks: Towards an isogeny-based postquantum vdf. Cryptology ePrint Archive, Report 2021/1289
20. Cherniaeva, A., Shirobokov, I., Shlomovits, O.: Homomorphic encryption random beacon. IACR Cryptol. ePrint Arch. **2019**, 1320 (2019)
21. Clark, J., Hengartner, U.: On the use of financial data as a random beacon. EVT/WOTE **89** (2010)
22. Corestar: Corestar arcade: Tendermint-based byzantine fault tolerant (bft) middleware with an embedded bls-based random beacon (2019)
23. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., et al.: On scaling decentralized blockchains. In: International conference on financial cryptography and data security. pp. 106–125. Springer (2016)
24. DAOBet: Daobet to deliver on-chain random beacon based on bls cryptography (2019), https://daobet.org/blog/on-chain-random-generator/
25. Das, S., Krishnan, V., Isaac, I.M., Ren, L.: Spurt: Scalable distributed randomness beacon with transparent setup. IACR Cryptol. ePrint Arch. **2021**, 100 (2021)
26. Das, S., Xiang, Z., Ren, L.: Asynchronous data dissemination and its applications. IACR Cryptol. ePrint Arch. (2021)
27. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 66–98. Springer (2018)
28. Dimakis, A.G., Godfrey, P.B., Wu, Y., Wainwright, M.J., Ramchandran, K.: Network coding for distributed storage systems. IEEE Trans. Inf. Theory **56**(9), 4539–4551 (Sept 2010). https://doi.org/10.1109/TIT.2010.2054295
29. Dodis, Y.: Efficient construction of (distributed) verifiable random functions. In: International Workshop on Public Key Cryptography. pp. 1–17. Springer (2003)
30. Drake, J.: Minimal vdf randomness beacon. Ethereum Research (2018)
31. drand: Drand - a distributed randomness beacon daemon (2020), https://github.com/drand/drand
32. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)
33. Elrond, A.: Highly scalable public blockchain via adaptive state sharding and secure proof of stake (2019)
34. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: Continuous verifiable delay functions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 125–154. Springer (2020)
35. Fetch.ai.: Distributed verifiable random functions: an enabler of decentralized random beacons (2020), https://github.com/fetchai/research-dvrf

36. Galindo, D., Liu, J., Ordean, M., Wong, J.M.: Fully distributed verifiable random functions and their application to decentralised random beacons. IACR Cryptol. ePrint Arch. **2020**, 96 (2020)
37. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In: International Conference on Applied Cryptography and Network Security. pp. 156–174. Springer (2016)
38. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 295–310 (1999)
39. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th symposium on operating systems principles. pp. 51–68 (2017)
40. Goel, S., Robson, M., Polte, M., Sirer, E.: Herbivore: A scalable and efficient protocol for anonymous communication. Tech. rep., Cornell University (2003)
41. Goulet, D., Kadianakis, G.: Random number generation during tor voting. Tor's protocol specifications-Proposal **250** (2015)
42. Gurkan, K., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Aggregatable distributed key generation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 147–176. Springer (2021)
43. Haahr, M.: Random.org: True random number service. School of Computer Science and Statistics, Trinity College, Dublin, Ireland **10** (2010)
44. Han, R., Yu, J., Lin, H.: Randchain: Decentralised randomness beacon from sequential proof-of-work. IACR Cryptol. ePrint Arch. **2020**, 1033 (2020)
45. Hanke, T., Movahedi, M., Williams, D.: Dfinity technology overview series, consensus system. arXiv preprint arXiv:1805.04548 (2018)
46. Kelsey, J., Brandão, L.T., Peralta, R., Booth, H.: A reference for randomness beacons: Format and protocol version 2. Tech. rep., National Institute of Standards and Technology (2019)
47. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Annual International Cryptology Conference. pp. 357–388. Springer (2017)
48. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE symposium on security and privacy (SP). pp. 839–858 (2016)
49. Lenstra, A.K., Wesolowski, B.: A random zoo: sloth, unicorn, and trx. IACR Cryptol. ePrint Arch. **2015**, 366 (2015)
50. Li, Z., Tan, T.G., Szalachowski, P., Sharma, V., Zhou, J.: Post-quantum vrf and its applications in future-proof blockchain system (2021)
51. LoE: League of entropy : Decentralized randomness beacon (2019), https://www.cloudflare.com/it-it/leagueofentropy/
52. Loe, A.F., Medley, L., O'Connell, C., Quaglia, E.A.: A practical verifiable delay function and delay encryption scheme. Cryptology ePrint Archive (2021)
53. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf (2009)
54. Naor, O., Baudet, M., Malkhi, D., Spiegelman, A.: Cogsworth: Byzantine view synchronization. arXiv preprint arXiv:1909.05204 (2019)
55. Nguyen-Van, T., Nguyen-Anh, T., Le, T.D., Nguyen-Ho, M.P., Nguyen-Van, T., Le, N.Q., Nguyen-An, K.: Scalable distributed random number generation based on homomorphic encryption. In: 2019 IEEE International Conference on Blockchain (Blockchain). pp. 572–579. IEEE (2019)

56. Oraclize.it: Provable random number generator, https://provable.xyz
57. Pietrzak, K.: Simple verifiable delay functions. In: 10th innovations in theoretical computer science conference (itcs 2019) (2018)
58. Rabin, M.O.: Transaction protection by beacons. Journal of Computer and System Sciences **27**(2), 256–267 (1983)
59. Randao: Randao: A dao working as rng of ethereum, https://github.com/randao/randao., [Online; accessed 1-Nov-2021]
60. Rashmi, K.V., Shah, N.B., Gu, D., Kuang, H., Borthakur, D., Ramchandran, K.: A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster. In: 5th USENIX Workshop on Hot Topics in Storage and File Systems. USENIX (2013)
61. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. Journal of the society for industrial and applied mathematics **8**(2), 300–304 (1960)
62. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. . (1996)
63. Rotem, L.: Simple and efficient batch verification techniques for verifiable delay functions. Cryptology ePrint Archive (2021)
64. Schindler, P.: Hydrand, https://github.com/PhilippSchindler/hydrand
65. Schindler, P., Judmayer, A., Hittmeir, M., Stifter, N., Weippl, E.: Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. IACR Cryptol. ePrint Arch. **2020**, 942 (2020)
66. Schindler, P., Judmayer, A., Stifter, N., Weippl, E.: Hydrand: Efficient continuous distributed randomness. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 73–89. IEEE (2020)
67. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Annual International Cryptology Conference. pp. 148–164. Springer (1999)
68. Shumow, D., Ferguson, N.: On the possibility of a back door in the nist sp800-90 dual ec prng. In: Proc. Crypto. vol. 7 (2007)
69. Syta, E., Jovanovic, P., Kogias, E.K., Gailly, N., Gasser, L., Khoffi, I., Fischer, M.J., Ford, B.: Scalable bias-resistant distributed randomness. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 444–460. Ieee (2017)
70. Tomescu, A., Chen, R., Zheng, Y., Abraham, I., Pinkas, B., Gueta, G.G., Devadas, S.: Towards scalable threshold cryptosystems. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 877–893 (2020)
71. Wang, G., Nixon, M.: Randchain: Practical scalable decentralized randomness attested by blockchain. In: 2020 IEEE International Conference on Blockchain (Blockchain). pp. 442–449. IEEE (2020)
72. Wesolowski, B.: Efficient verifiable delay functions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 379–407. Springer International Publishing, Cham (2019)
73. Ye, M., Barg, A.: Explicit constructions of high-rate mds array codes with optimal repair bandwidth. IEEE Transactions on Information Theory **63**(4), 2001–2014 (2017)
74. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: Hotstuff: Bft consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. pp. 347–356 (2019)
75. Yurek, T., Luo, L., Fairoze, J., Kate, A., Miller, A.K.: hbacss: How to robustly share many secrets. IACR Cryptol. ePrint Arch. **2021**, 159 (2021)

A Secure DRB Protocol

A DRB protocol is said to be secure if for any probabilistic polynomial-time adversary $\mathcal{A}$ corrupting at most t parties in a round e, in a security game $\mathcal{G}$ played between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, $\mathcal{A}$ has negligible advantage.

1. $\mathcal{C}$ executes the setup and sends the public parameters of the system to $\mathcal{A}$.
2. $\mathcal{A}$ corrupts up to t participants and informs about t corrupted nodes to $\mathcal{C}$.
3. $\mathcal{C}$ creates the secret and public keys of honest nodes and sends the public keys of honest nodes to $\mathcal{A}$.
4. $\mathcal{A}$ sends the remaining public parameters (e.g. public keys) of t nodes to $\mathcal{C}$.
5. $\mathcal{C}$ and $\mathcal{A}$ runs the protocol execution interactively per round where:
   (a) $\mathcal{C}$ sends all the honest participants' messages to $\mathcal{A}$.
   (b) $\mathcal{A}$ decides on the delivery (sends / does not send) of the messages.
   (c) At the end of a round e, an honest node outputs the protocol transcript.
6. $\mathcal{C}$ samples a bit $b \in \{0, 1\}$ and sends either the DRB output based on transcript or a random element.
7. $\mathcal{A}$ makes a guess b' and the advantage of $\mathcal{A}$ is defined as $|\Pr[b = b'] - \frac{1}{2}|$.

B Publicly Verifiable Secret Sharing (PVSS)

In a PVSS scheme, a dealer shares a randomly selected secret s among a set of n nodes using an $(n, t+1)$ threshold access-structure. That means, secret s can be recovered from a set of $t+1$ valid shares.

Definition 6. *(PVSS): It is defined as a collection of following algorithms:*

- $\mathsf{Setup}(\lambda)$*: Given a security parameter* λ*, generates the public parameters* pp *and the public-private key-pair for each node, output the public parameter and public keys* (pp, pk)*.* pp *is an implicit input to all the other algorithms.*
- $\mathsf{Share}(s)$*: For a randomly chosen secret* s*, a dealer creates the secret shares for each node* $\vec{S} = (s_1, s_2, \ldots, s_n)$ *along with the encryption of the shares* $\vec{E} = (c_1, c_2, \ldots, c_n)$ *where* $c_i = Enc(s_i)$ *and proof of correct encryption* $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$*. It outputs* $(\vec{S}, \vec{E}, \vec{\pi})$*.*
- $\mathsf{Verify}(\vec{E}, \vec{\pi})$*: Given the encrypted shares and the proofs, any external* $\mathcal{V}$ *can non-interactively verifies if the sharing is correct. It outputs* 0 *or* 1*.*
- $\mathsf{Recon}(\vec{S})$*: Given valid set* $\vec{S} \subseteq \{s_1, s_2, \ldots, s_n\}^{t+1}$ *of* $t+1$ *decrypted shares, it reconstructs the secret and outputs* s*.*

In a DRB protocol involving a leader, once the setup phase is completed, for the round e, first a leader election algorithm $\mathsf{LeaderElec}(e, O_{e-1}, P_1, P_2, \ldots, P_n)$ is executed and a leader L_e is selected. The election algorithm can be *round-robin selection* or *sampling uniformly at random*. The leader L_e chooses a secret value s_{L_e} (either a new value or previously committed value in the previous round) and executes the PVSS scheme for secret s_{L_e}. At the end of round e, DRB output O_e is generated using the reconstructed secret and the previous round

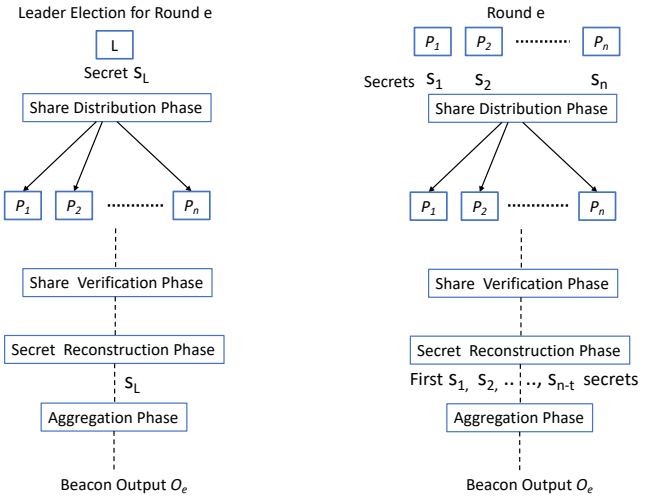

Fig. 2. PVSS-based DRB protocols with and without leader

(or rounds') output value. Figure 2 depicts leader and non-leader-based DRB protocols. In the first sub-figure, a leader is elected, followed by leader's secret is shared and beacon output is produced. In the second sub-figure, all participants randomly choose secrets at the start of the round and further share the encrypted shares of the secret to all the other participants. In the final stage, the first n-t reconstructed (or decrypted) shares are used to obtain beacon output.

Definition 7. *(PVSS-based Interactive Decentralized Randomness Beacon (I-DRB)) Given a set of participants $\mathcal{P} = (P_1, P_2, \ldots, P_n)$, a PVSS-based I-DRB without leader can be defined as a tuple $\mathcal{B}$ of polynomial-time algorithms:*
$\mathcal{B} = (\mathsf{Setup}, \mathsf{Share}, \mathsf{Verify}, \mathsf{Recon}, \mathsf{Aggregation})$

- $\mathsf{Setup}(e, \lambda)$*: Set the round $e = 1$. Run* $\mathsf{PVSS.Setup}(\lambda)$ *and generate public parameter pp and key-pairs (sk_i, pk_i) for each participant.*
- $\mathsf{Share}(e)$*: For a round e, each participant P_i runs* $\mathsf{PVSS.Share}(s_i)$ *for a randomly chosen value s_i from the input space and gets $(\vec{S}_i, \vec{E}_i, \vec{\pi}_i)$. P_i shares the encrypted shares and corresponding proofs $(\vec{E}_i, \vec{\pi}_i)$ with other participants.*
- $\mathsf{Verify}(e, \{\vec{E}, \vec{\pi}\})$*: Each party P_j runs the share verification algorithm* $\mathsf{PVSS.Verify}$ $(\vec{E}_i, \vec{\pi}_i); \forall i, i \neq j$ *on every shared secret. Let $\mathcal{C}$ be the set of first $n - t$ participants who have correctly shared their random secret values.*
- $\mathsf{Recon}(e, \{\vec{S}_i\})$*: Each party P_i in* C *opens the Shamir secret s_i and the randomness used, other participants $P_j; \forall j, j \neq i$ verify if it is consistent with sharing posted during* Share *phase. If a party P_i refuses to open its secret s_i, the secret is reconstructed by executing* $\mathsf{PVSS.Recon}(\vec{S}_i)$.
- $\mathsf{Aggregation}(e, \{s_i\})$*: Once the valid decrypted or reconstructed shares are available for the parties $P_i \in \mathcal{C}$. The beacon output is generated by executing a function f on input a set of valid shares $\{s_i\}$. This function f takes all the valid shares $\{s_i\}$ (additionally previous beacon outputs) as input and aggregates these input values to generate the beacon output O_e for round e.*

C Verifiable Delay Function (VDF)

Verifiable delay function $f : \mathcal{X} \to \mathcal{Y}$ was defined formally by Boneh et al. [11]. After the introduction of VDF, two new proposals [72] [57] were presented. A VDF has properties of *Sequentiality*, *Uniqueness* and ϵ-*Evaluation time*.

Definition 8. *(VDF): A VDF is defined as a tuple of following algorithms:*

- $\mathsf{Setup}(\lambda, T)$*: It is a randomized algorithm that takes security parameter* λ*, time parameter* T *and outputs public parameter* $pp := (\mathbb{G}, N, H, T)$*, where* $\mathbb{G}$ *is a finite abelian group of unknown order,* N *is an RSA modulus, and* $H : \mathcal{X} \to \mathbb{G}$ *is a hash function.*
- $\mathsf{Eval}(pp, x, T)$*: The evaluation algorithm applies* T *squarings in* $\mathbb{G}$ *starting with* $H(x)$ *and outputs the value* $y \leftarrow H(x)^{(2^T)} \mathsf{mod}\ N$*, along with a proof* π*.*
- $\mathsf{Verify}(pp, x, y, \pi, T)$*: The verification algorithm outputs a bit* $\in \{0, 1\}$*, given the input as public parameter* pp*, input value* x*, output value* y*, proof* π*, and time parameter* T*.*

D Verifiable Random Function (VRF)

VRF has properties of Uniqueness, Collision resistance and Pseudorandomness.

Definition 9. *(VRF): A VRF is defined as a tuple of following algorithms:*

- $\mathsf{KeyGen}(r)$*: On input value* r*, the algorithm generates a secret key* sk *and a verification key* vk*.*
- $\mathsf{Eval}(sk, M)$*: Evaluation algorithm produces pseudorandom output* O *and the corresponding proof* π *on input* sk *and a message* M*.*
- $\mathsf{Verify}(vk, M, O, \pi)$*: Verify algorithm outputs 1 if and only if the output produced by evaluation algorithm is* O *and it is verified by the proof* π *given the verification key* vk *and the message* M*.*

E Homomorphic Encryption (HE)

Definition 10. *(HE): An HE scheme is defined as a set of following alogorithms:*

- $\mathsf{Setup}(1^\lambda)$*: Given security parameter* λ*, Output global parameters* $params$*.*
- $\mathsf{KeyGen}(params)$*: Given global parameters* $param$*, output a public-private keypair* (pk, sk)*.*
- $\mathsf{Enc}(params, pk, \mu)$*: Given a message* $\mu \in R_\mathcal{M}$*, output a ciphertext* c*.*
- $\mathsf{Dec}(params, sk, c)$*: Given a ciphertext* c*, output a message* $\mu^* \in R_\mathcal{M}$*.*
- $\mathsf{Eval}(pk, f, c_1, ..., c_l)$*: Given the inputs as public key* pk*, a function* $f : R^l_\mathcal{M} \to R_\mathcal{M}$ *which is an arithmetic circuit over* $R_\mathcal{M}$*, and a set of* l *ciphertexts* $c_1, ..., c_l$*, output a ciphertext* c_f*.*

In the above scheme, the message space $\mathcal{M}$ of the encryption schemes is a ring $R_\mathcal{M}$, and the functions to be evaluated are represented as arithmetic circuits over this ring, composed of addition and multiplication gates. HE can be categorized into: *Partially HE* that supports only addition or multiplication; *Somewhat HE* that allows both operations but with limited times; *Fully HE* that supports arbitrary computation by allowing both operations with unlimited times.

F Hybrid DRB Protocols

There are many more DRB protocols. Some of these protocols use more than one crypto primitive to achieve all DRB properties with better efficiency/optimization.

- Mt. Random (PVSS + T(VRF))(eprint 2021/1096): It is a multi-tiered DRB protocol that combines PVSS, VRF, and Threshold VRF (TVRF) to construct a DRB with optimal efficiency and without compromising security guarantees of DRB. It is a flexible architecture for DRB where each tier runs a separate beacon based on PVSS, VRF, and TVRF, and output of one tier works as a seed for the next tier. Being constructed using different crypto-primitives, each tier differs in the provided randomness and complexity. Due to that, a high-level application can decide on which tier to use to obtain randomness.
- Harmony (VRF + VDF)(https://harmony.one/whitepaper.pdf): Harmony is a sharding-based, provably secure, and scalable blockchain. In Harmony, nodes compute local entropy by executing VRF using their secret keys. DRB output is computed using VDF where the input for the VDF is constructed from a threshold number of VRF evaluations from pairwise different nodes. DRB output is made pseudorandom by applying a random oracle on VDF output.
- CRAFT (TLP + VDF)(eprint 2020/784): Baum et. al first construct UC-secure publicly verifiable TLP and UC-secure VDF. To construct DRB, they replace the commitments with the UC-secure TLP in the standard commit-reveal coin-tossing protocol. Their construction achieves $\mathsf{O}(n)$ communication to generate DRB output. DRB output can be obtained as fast as the communication channel delay allows when nodes communicate their TLPs faster.
- VeeDo (STARK+VDF)(https://github.com/starkware-libs/veedo): It is based on STARK-based VDF. STARK is a post-quantum secure zero-knowledge proof protocol. VeeDo is a smart-contract-based DRB where a beacon smart contract and a verifier smart contract is placed on-chain. However, heavy computational parts involving VDF and STARK prover are kept off-chain. A VDF is run on a seed s from a block hash to compute the DRB output and a proof is computed using the STARK prover. The VDF output and the proof are sent to the on-chain contracts for verification and subsequently publishing.
- STROBE (RSA-based)(eprint 2021/1643): It is a history-generating DRB (HGDRB). It allows efficient generation of previous beacon outputs given only the current beacon value and public key. It is based on origin squaring based RSA approach of Beaver. It is well-suited for practical applications especially in streaming designs where it allows client software to generate game states by computing every missing beacon value and state. It is NIZK-free, concisely self-verifying and can be efficiently used in blockchain and voting systems.
- OptRand (Bilinear paring-based PVSS + NIZK)(eprint 2022/193): It is an optimally responsive DRB protocol. It employs a pairing-based PVSS scheme together with a NIZK proof system to produce DRB outputs. Despite the synchrony of the network, it can provide an optimal response and can progress. Therefore, OptRand can provide availability at actual network speed during optimistic conditions. It is reconfiguration-friendly and has low communication complexity and low latency while generating beacon outputs.