SoK: Public Randomness

Alireza Kavousi[1], Zhipeng Wang[2], and Philipp Jovanovic[1]

[1]University College London
[2]Imperial College London

Abstract

Public randomness is a fundamental component in many cryptographic protocols and distributed systems and often plays a crucial role in ensuring their security, fairness, and transparency properties. Driven by the surge of interest in blockchain and cryptocurrency platforms and the usefulness of such component in those areas, designing secure protocols to generate public randomness in a distributed manner has received considerable attention in recent years. This paper presents a systematization of knowledge on the topic of public randomness with a focus on cryptographic tools providing public verifiability and key themes underlying these systems. We provide concrete insights on how state-of-the-art protocols achieve this task efficiently in an adversarial setting and present various research gaps that may be suitable for future research.

1 Introduction

Public randomness is about generation and distribution of random values that are publicly verifiable and accessible by anyone after a certain barrier point. This is in stark contrast to the notion of private randomness, which has strict confidentiality requirements and is not supposed to be shared widely, *e.g.*, as used for cryptographic key generation. While some of the earliest known use cases for public randomness date back to antiquity, the concept is also a critical building block for various modern applications that have a vital need for transparency and fairness, like lotteries [95, 118], online games [65], blockchain sharding [90, 130, 133], and more.

Public randomness can be obtained through various means with the most straightforward one being to simply gather it from a centralized entity [87, 101]. This approach, however, has obvious disadvantages that no one can check whether the provided values have actually been generated randomly or the entity can simply withhold an output. In this paper we are concerned with randomness generation in a publicly verifiable manner and without relying on trusted third parties. Despite the apparent simplicity of the concept, it turns out producing high-quality random values that are unpredictable and unbiased, and ensuring their verifiability, is a non-trivial challenge requiring deep insights from distributed systems and cryptography.

As the need for trustworthy sources of public randomness continues to grow, a large volume of works utilizing different methods has appeared in academia and industry over the last decade or so. This paper contributes to the discussion by providing a comprehensive and systematic analysis of the notions, challenges, solutions, and techniques of state-of-the-art schemes. We acknowledge that there is concurrent work on the topic [48, 114] and argue that this work presents a complementary perspective as follows.

- We base our categorization on primitives with public verifiability in a leaderless or leader-based fashion, and further elaborate on key aspects including security, liveness, and scalability.
- We intentionally take an informal approach, giving us more room to cover various facets of the public randomness generation in detail.
- We provide key insights from existing works and outline research gaps throughout the paper to inspire future research.

Paper Organization. Section 2 introduces the required background, concepts, and primitives from cryptography and distributed systems. Section 3 presents the notion of distributed randomness beacon (DRB) and its relevant properties as a service for generating public randomness distributedly. Section 4 presents our systematization methodology and an overview on the four main tools to generate public randomness, which are publicly verifiable secret sharing (PVSS), verifiable random function (VRF), verifiable delay function (VDF), and public blockchain. These are then investigated in-depth in the subsequent sections. Finally, Section 9 concludes the paper with a discussion on further related properties and considerations. Also, throughout the paper we include plenty of insights and potential research gaps labeled by "Insight" and "Gap", respectively.

2 Background

Here we give a brief overview of the relevant preliminaries and primitives used in this paper.

System Model. There are n parties to execute the protocol at most t of which are corrupted by a computationally bounded adversary in security parameter λ. We may refer to f as the number of parties adversary *actually* corrupts, where $0 \leq f \leq t$. Corruption typically occurs in the form Byzantine failure, where the corrupted parties are free to do any adversarial behavior of their choice. By epoch, we mean a certain period of time divided discreetly. By multicast, we mean simply sending a message by a party to others. We may use some terms interchangeably throughout the paper with essentially the same meaning such as "faulty", "malicious", and "Byzantine". The adversary may be *rushing*, *i.e.*, it may wait and observe honest parties' messages before deciding on how to act next.

Network Propagation. There are two well-known communication types for exchanging messages between parties. First, point-to-point where each pair of parties is connected via a reliable authenticated channel that guarantees the confidentiality and integrity of the content. In this setting, parties have a priory knowledge about others involved in the protocol. Second, peer-to-peer, where the message is delivered by a sender to a random subgroup of parties (*i.e.*, its peers) in different steps until all the parties receive the message. In this setting, the parties do not necessarily know each other. Gossip-based protocols are notable examples where parties diffuse messages by passing them on to their peers.

Network Assumption. Based on the effects of adversary on message delivery between (honest) parties, three main assumptions are made regarding the underlying network. In the *synchronous* model, there is a finite known time-bound Δ for message delivery. The *partial synchronous* model relaxes this assumption by making the time-bound unknown. The *asynchronous* model just makes the minimal assumption of eventual delivery without specifying any time-bound. It is worth noting that there is an equivalent treatment for partial synchrony, which divides the system into two periods: an initial period of asynchrony followed by synchrony after some unknown point in time.

Consensus. The fundamental problem of consensus [92, 108] deals with enabling a group of n parties, each with an initial input value v_i to reach agreement on a common value v despite the presence of adversarial behavior by a threshold t of them. A working consensus protocol must satisfy the two main properties: *safety/consistency*, ensuring that all honest parties agree on the same decision; and *liveness*, ensuring that the protocol eventually terminates with honest parties reaching a valid decision. Different variants of consensus problems exist (refer to [68] for more details). A well-studied one, relevant to the focus of this paper, is Byzantine fault-tolerant state machine replication (BFT-SMR) [121], which aims to maintain consistency among a group of parties while processing ever-growing inputs.

Publicly Verifiable Secret Sharing (PVSS). A (t, n) secret sharing scheme [123] allows a dealer to distribute shares of a secret s among n parties, such that any gathering of size at most a threshold t of shares reveals no information about the secret while it can be uniquely revealed by any larger subset. To enable *public verification*, where anyone (even an external party) can verify the sharing phase by the dealer and reconstruction phase by the parties, PVSS [41, 122] incorporates cryptographic primitives including encryption, commitment, and non-interactive zero-knowledge proofs.

Distributed Key Generation (DKG). To carry out cryptographic evaluations (*e.g.*, encryption, signing) jointly, a DKG protocol shares a uniformly distributed random secret sk among n parties. It provides each party with a partial secret key sk_i for performing partial evaluations, a corresponding partial public key pk_i to verify the correctness of the partial evaluations, and a common public key pk to verify the correctness of the final evaluation. DKG is executed in a single-shot manner, meaning that it is only needed to run once to produce the required keys, which can then be used polynomially many times thereafter.

Verifiable Random Function (VRF). A VRF [102] allows one to produce a pseudorandom value along with a proof on an input using their own secret key sk such that anyone can verify its correctness using the corresponding public key pk. It can be considered as an asymmetric counterpart to a pseudorandom function (PRF) with an attached proof. With a DKG, it is possible to establish a threshold VRF, wherein only a proper subset of parties (*i.e.*, more than a threshold t) can jointly evaluate the function using their keys.

Verifiable Delay Function (VDF). A VDF [27,112,131] is an inherently sequential function that takes a predefined time T (*i.e.*, steps) to compute, even with a polynomial number of processors working in parallel. Given an input value x, it outputs a unique value y that can be verified efficiently by anyone in $poly(\log T, \lambda)$ time. Two well-known VDFs of [112,131] are based on repeated squaring, $y = x^{2^T}$, in a group of unknown order.

3 Distributed Randomness Beacon

3.1 Definition and Properties

The process of obtaining public randomness in a distributed manner is commonly captured in the literature via *distributed randomness beacon* (DRB), a service that outputs a continuous series of randomness with public verifiability. It is straightforward, however, to see a protocol producing public randomness can essentially turn into a DRB by executing multiple times.

Although recent works such as [23, 53] have taken a formal approach to the DRB problem by providing game-based security definitions, a significant portion of the existing literature has primarily relied on an informal treatment [119, 120, 128]. These works explore individual properties and

make arguments about their fulfillment. Due to the purpose of this paper, we refrain from such formulations and refer the reader to [48] for further details.[1]

Distributed Randomness Beacon (DRB). A DRB protocol enables a group of n distrustful parties to jointly generate a series of random values with the following properties: (1) *unpredictability*, meaning the adversary cannot predict the future beacon outputs given the knowledge of the existing ones; (2) *unbiasability*, meaning the adversary cannot impact an output to its advantage; (3) *availability*, meaning the protocol should continue making progress and produce valid outputs; and (4) *public-verifiability*, meaning the correctness of the result should be verifiable even by an external party.

Following the above well-established properties, we conceptualize the problem using three rather *orthogonal* challenges a DRB deals with. That is, *security*, capturing the secrecy of the produced values against adversarial behavior, *liveness*, capturing certain conditions that affect the progress of the protocol, and *scalability*, capturing the overhead of the protocol and its deployment at scale. We believe such distinction allows focusing on the core concerns accordingly. Needless to say, the interconnected aspect of these challenges makes a perfect distinction difficult and some level of intersection is inevitable.

Security. Unpredictability and unbiasability are two principal security properties of a DRB protocol. We elaborate on them as follows.

- Unpredictability. Arguably the most important property of a DRB is unpredictability, which ensures the inability of the adversary to predicate future beacon outputs given the knowledge of the already produced ones. Unpredictability can be satisfied in two flavors: *absolute* unpredictability and *probabilistic* unpredictability. The former puts a hard bound $d \geq 1$ on the number of epochs when the outputs become fully unpredictable, whereas the latter only guarantees the likelihood of obtaining the next outputs decreases exponentially with the number of epochs.

- Unbiasability. Ruling out the feasibility of any action by the adversary to influence the outputs to its advantage is captured by the unbiasability property. The adversarial impact on DRB can occur in different ways. One common type of such an impact is to *withhold* announcing contributions by an adversary after getting aware of other parties' contributions in the hope of rejecting an undesirable output, known as *last actor attack* [81].

The baseline security for a randomness beacon is unpredictability, and not all the proposed solutions can satisfy unbiasability [22,32,55]. The core definition of unbiasability essentially implies the beacon output should be uniformly distributed across the set of possible outputs. Existing DRB protocols achieve this quality at two major types: *uniform randomness*, ensuring the beacon output is a truly random value, and *pseudorandomness*, ensuring the beacon output is computationally indistinguishable from a truly random value. A long-lasting approach for generating randomness is via the so-called *commit-reveal* method which each party commits to some local random value (*i.e.*, private randomness) to later open it and compute the resulting outcome by running some operation, *e.g.*, adding all the proposals. While this approach provides unpredictability in its strongest type when only *one* single honest contribution (*i.e.*, true randomness) is involved, it fails to guarantee unbiasability as the adversary may act as the last actor and decide on its action (*i.e.*, to open or not) based on the view of the protocol. Intuitively, *secure* constructions with a commit-reveal paradigm output uniform randomness that is of importance for specific applications such as PRG seed.

DRB protocols often operate in epochs, where one of the parties may be selected as a leader to coordinate and advance the protocol to the next epoch [53, 119, 120]. The leader election process

[1] However, not all the required properties have yet been defined formally, such as liveness/availability.

is either *deterministic* when the sequence of leaders is known in advance[2], or is *probabilistic* when the next leader is chosen randomly. Although most of the existing DRB protocols can produce randomness with absolute unpredictability, the ones with probabilistic leader election yield the weaker notion, as there is always a possibility that some corrupted party is elected as the leader [24,119,120]. An immediate implication is the leader's ability to learn the next epoch's output prior to others, leading to achieving only d-unpredictability, for $d \geq 2$ epochs in the future. One way to address this issue is to transform the role of the leader from a sole contributor (to the beacon output) to only a coordinator [53].

Liveness. Ensuring the continual operation of the protocol under certain conditions is a decisive aspect of a DRB. The liveness of a distributed protocol is typically evaluated based on two key parameters: *network assumption*, defining limits to what extent the adversary is able to cause delay in message delivery, and *fault tolerance*, indicating the maximum number of corrupted (Byzantine) parties tolerated for successful protocol execution. The bulk of existing works on DRB focuses on the *synchronous* model. This, however, cannot faithfully cover real-world scenarios as the protocol might face severe outages like *denial of service* (DoS) attacks. As a result, the study of DRB protocols in *non-synchronous* models, such as partial synchronous and asynchronous models, has gained attention [35, 53]. The partially synchronous model aims to strike a balance by offering the advantages of the others. It introduces a notion of time for convenient analysis while providing a robust network definition to handle occasional outages. Significant progress has been made in the distributed system literature regarding fault tolerance under various network assumptions [59, 62]. In particular, the optimal fault tolerance for a BFT-SMR protocol is $t < n/2$ under the synchrony and is $t < n/3$ under the partial synchrony (and asynchrony). A DRB protocol with a set of parties on ever-growing inputs working towards establishing a totally ordered log of outputs resembles the consensus problem, particularly state machine replication (SMR) [121] with two ingredient properties of consistency and liveness.

Apart from the two principle aspects of network assumption and fault tolerance, there are two additional features relevant to liveness that are useful to consider in the context of DRB protocols. Namely, *guaranteed output delivery* (GOD), ensuring that all the honest parties obtain the beacon output at each epoch irrespective of the adversary's actions, and *responsiveness*, allowing the protocol to make progress at the actual network's speed rather than under a conservative delay Δ. Although GOD is deemed to be pivotal for the proper functionality of a DRB that may need to consistently feed the end users even in the face of non-Byzantine failures (*e.g.*, temporary disconnection), not all the existing secure proposals can achieve it [53]. Due to the lack of (known) time-bound in their formulations, non-synchronous network models inherently come with responsiveness that matters for getting higher throughput (*i.e.*, number of produced outputs per time unit). However, it has been shown that the synchronous model can only offer *optimal* responsiveness under certain conditions, including the presence of an honest leader and $f < n/4$ [107].

Scalability. The level of reliability achieved in a distributed computing system consequently impacts the scalability. In the context of DRBs, it amounts to providing randomness at a reasonable cost depending on the number of participating parties. Asymptotically, this is typically measured using the following terms: (1) *communication complexity*, which quantifies the total number of messages exchanged among parties during protocol execution; (2) *computation complexity*, which measures the amount of local work performed by each party per output; and (3) *verification complexity*, which evaluates the work required by an external party to verify the output. Moreover, accommodating *dynamic* participation of parties also matters for a scalable system. The presence of an expensive setup phase as part of the protocol poses a serious challenge to achieving this goal.

[2]For instance, in a *Round-Robin* election the leader of epoch r is party $i = r \bmod n$.

3.2 Applications

Having access to a trustworthy source of randomness is an essential part of many real-world systems. We now explore several application areas raised more attention in recent years.

Lotteries and Streaming Games. The need for a reliable source of randomness is paramount in lotteries and streaming games. Lotteries, whether traditional [61] or blockchain-based [65, 95, 118], rely on randomness to determine the winning numbers or participants. Using proper randomness is crucial to prevent any manipulation or bias that could compromise the integrity of the process. In the context of streaming games [67, 79], randomness plays a vital role in maintaining excitement and engagement by enabling randomly generated events, such as item drops, enemy spawns, or in-game challenges.

Blockchain Sharding. Sharding [90, 130, 133] is a promising solution for overcoming the scalability problems in blockchain. A sharding protocol aims to divide a large population into smaller groups of parties known as committees. By splitting the workload, committees can operate in parallel, enhancing performance and increasing throughput. In sharding protocols, a critical issue is how the participants are assigned to committees in a fair and unpredictable way [11]. By leveraging public randomness, participants can be randomly allocated to committees. This way of assignment serves as a defense against adversarial concentration by preventing malicious actors from predicting beforehand the valid set of identities supposed to carry out their tasks.

Single Secret Leader Election (SSLE). In proof-of-stake (PoS) blockchains with longest-chain consensus, leader election plays a critical role in maintaining the security and liveness of the system. It turns out deploying a single secret leader election (SSLE) mechanism, where the unique elected leader gets to know about their leadership prior to others and can claim it in a publicly verifiable manner, is crucial [12, 28]. First, it protects the leaders from various attacks, in particular DoS. Second, it obviates the need for a tie-breaking process to choose among possible set of leaders for a given time slot. The majority of the existing solutions for SSLE rely on public randomness to ensure security [84] or uniqueness [28].

Timelock Encryption. A critical characteristic for a proper randomness beacon is to satisfy a regular timing for the release of its outputs [87]. Drand [3], employing League of Entropy [1], precisely meets such requirement by currently producing a beacon output every 30 seconds. The recent work of [64] showed how to take advantage of such a feature to allow "encrypting to future" by constructing a timelock encryption [117] that has no sequential computation involved. In more detail, they observed that by viewing a beacon output – which is a threshold VRF on an epoch number – as a secret key in an identity-based encryption scheme [29], it is possible to encrypt a message to the future, with the public key (*i.e.*, identity) being the epoch number and corresponding secret key being the beacon output.

4 Systematization Methodology

Upon reviewing the existing literature, we realize that a representative method to categorize existing works is based on their *underlying tools*. Public verifiability, as a default property, imposes constraints on the cryptographic tools employed in these constructions. This, in turn, enables us to pinpoint four of these including PVSS, VRF, VDF, and public blockchain. The protocols deploying these tools can be further classified into *leaderless* and *leader-based* according to their communication patterns. The former requires all-to-all communication, whereas the latter designates a leader to communicate with others in a one-to-all (possibly together with an all-to-one) manner. For each

category, we then evaluate different protocol designs with regard to security, liveness, and scalability. This categorization enables us to classify most of the literature coherently.[3] While we put our focus on illustrative peer-reviewed papers, where appropriate, we point to some compelling ideas presented in other works. We also present a comparison between different DRBs in Table 1. In the following, we provide an overview of each category before doing an in-depth analysis later.

Protocols Using PVSS. The majority of research around generating public randomness is focused on using PVSS schemes that yield the strongest quality (*i.e.*, uniform randomness). The idea of using PVSS is built upon the well-known commit-reveal paradigm [26], making it robust against adversarial attacks due to its inherent threshold security, and publicly verifiable due to the use of non-interactive zero-knowledge proof (NIZK). A major limitation of this approach is its intense communication cost due to exchanging large-sized messages (*i.e.*, PVSS transcripts) in an all-to-all manner. To address this, various techniques have been proposed, including the appointment of a leader for coordination and transcript aggregation.

Protocols Using VRF. VRFs have gained popularity as a viable option for designing DRBs due to their simplicity and practicality. They are usually derived from signatures with uniqueness, a property ensuring that for each massage and public key, there exists only one valid signature. The security properties of the DRB can be linked to those of the underlying VRF, as the unpredictability and unbiasability of the DRB are implied by the unforgeability and uniqueness of the VRF, respectively. However, similar to the traditional commit-reveal mechanism, a VRF-based construction may be susceptible to bias if a party with prior knowledge of the beacon output chooses to abort the protocol. The common approach to deal with this issue is to make the construction thresholdize via deploying a distributed key generation (DKG) [109] in the setup phase. While this mitigates the bias concern, it introduces communication overhead and hampers efficient dynamic participation.

Protocols Using VDF. A promising way to produce public randomness is deploying time-based cryptography and in particular verifiable delay functions that have efficient public verification. VDF has found itself useful in various blockchain-related applications since its introduction by Boneh et al. [27]. In this context, it has been proposed mainly as a method to protect against last actor attack in a commit-reveal configuration by injecting a *computationally* guaranteed delay before releasing the output. Such a delay prevents the adversary from learning the output prior to the reveal phase. Interestingly, VDFs can also be used solely to generate public randomness, *e.g.*, through its continuous variant [60] or the version with trapdoor [119]. The security properties of the resulting DRB protocol stem from its inherently sequential characteristic and uniqueness.

Protocols Using Public Blockchain. A straightforward approach to obtaining public randomness is by extracting it from the available entropy in publicly available resources. One notable example of such resources is proof-of-work (PoW) blockchains where parties can access high entropy values using the intrinsic randomness lied in the mining process. Despite its simplicity, this approach has been shown to be vulnerable to different attacks that compromise the desired security properties [34].

5 Protocols Using PVSS

5.1 Security

The protocols in this category follow the commit-reveal paradigm used by the prominent work of [26], allowing two parties to jointly flip a coin and generate a uniformly random string. Due

[3] We highlight that our focus is merely on cryptographic solutions as the most prevalent ones.

to the inherent weakness of this basic approach for guaranteeing unbiased randomness, numerous works [23,41,42,53,89,120,128] have employed PVSS to enable the recovery of any committed secret, relying on an honest majority assumption. In fact, it requires a quorum of parties (involving at least one honest party, *i.e.*, $f+1$) to execute the protocol by sharing some secret with uniform distribution. Since more than a threshold t of parties are involved in the process of randomness generation, the existence of at least one honest contribution guarantees a *uniform* source of randomness for this type of DRB.[4] Moreover, anyone can use the information posted on a public bulletin board, which is typically assumed to exist, to verify the correctness of the protocol.

Syta et al. [128] present a series of randomness beacon protocols in an incremental way such that each one complements the previous in some respect. The first one, **RandShare** [128], runs in a single-shot manner and outputs a shared randomness to each party via collecting a proper agreed set of local randomness through VSS. The second one, **RandHound** [128], builds upon RandShare and introduces scalability improvements by randomly sharding parties into sub-groups. Within each sub-group, PVSS is executed to generate local contributions, which are then combined to produce a single beacon output. The third one, **RandHerd** [128] further enhances RandHound by augmenting it with collective signing [129] and threshold (Schnorr) signatures [127] to provide a continual sequence of beacon outputs.

SCRAPE randomness beacon [41] follows the idea originally proposed in **Ouroboros** [89] and works by having each party run a PVSS for a randomly distributed secret and reconstruct an aggregated randomness from more than a threshold of contributions. Note that the public verification property of PVSS obviates the need for running a consensus among parties to determine the set of parties with correct sharing. In SCRAPE PVSS, parties use Lagrange interpolation in the exponent to reconstruct a group element of the form $S = h^s$, where $s \in Z_q$ and $h \in G$. **ALBATROSS** [42] takes the idea behind SCRAPE a step further to construct a protocol generating a batch of beacon outputs at each epoch. They do so by deploying *packed Shamir secret sharing* [25] in SCRAPE PVSS. Packed Shamir secret sharing allows a dealer to share a vector of secrets of size l using one single polynomial of degree $t+l-1$ as in the standard scheme. The parties share a tuple of l random secrets in the commit phase and compute a set of l^2 beacon outputs by locally applying a randomness extractor [49] on the correctly revealed secrets. If some parties refuse to open their secret tuples, others jointly do so. Moreover, the security properties of the randomness extractors imply that as long as having a proper set of uniform input values of size $n-t$, no information will be inferred about the output by the adversary choosing t input coordinates. This is in line with the honest majority assumption of the underlying PVSS scheme. One can observe that in such protocols each epoch's beacon output is determined once the set of parties with valid sharing is known. Afterward, even if some parties in the set decide not to open their commitments in the hope of adversarially impacting the output of the protocol (*i.e.*, violating unbiasability), other parties can jointly reconstruct the secrets.

However, in some applications such as lottery obtaining all the random values in one go is not desirable as fresh randomness may be required. To address this issue, **GULL** [43] is introduced, which is a randomness beacon using PVSS that allows progressive release of sub-batch of beacon outputs at different steps depending on the need of the protocol. They do so by modifying the ALBATROSS in a way that instead of having parties reveal all their secrets in the reconstruction phase, they use threshold cryptography to encrypt parts of their secrets and open them gradually at further stages if needed. Thus, whenever some fresh uniformly random value is required, a new sub-batch will be revealed, which is significantly more efficient than re-executing the full ALBATROSS protocol.

[4]Note that the resulting beacon output has uniform distribution with computational security, due to the properties of PVSS.

Insight 1 *Unpredictability is the base level of security while pseudorandomness is strictly stronger and can imply it. However, we may have pseudorandom values that are not unpredictable, if a batch of them is generated as in ALBATROSS [42].*

Gap 1 *Producing a batch of uniform beacon outputs while maintaining their unpredictability as in GULL [43] is a subtle task. One alternative method to do so is by designing packed PVSS with gradual release of secrets, e.g., using techniques from [4].*

The authors in [120] propose **Hydrand**, a leader-based DRB where rather than having all parties perform PVSS per epoch, one single party (*i.e.*, leader) is designated to do so. In fact, either the leader opens their previously committed secret, or a threshold $t+1$ of the parties jointly reconstruct the secret using the corresponding shares. Adopting a leader-based style comes at the cost of weakening the unpredictability with two consequences. First, the leader gets to know in advance the output of the epoch in which they are selected as the leader. Second, if the adversary happens to control t consecutive leaders, it can pre-compute the next t random values ahead of all the honest parties. Therefore, Hydrand only achieves absolute unpredictability of the next $t+1$ epochs. To ensure an honest leader is indeed selected after $t+1$ successive epochs, Hydrand uses a leader election mechanism with the possibility of exempting the current leader to be selected in the next t epochs. **GRandPiper** roughly follows the same design model of Hydrand by having a leader derive the protocol and buffered PVSS shares to enable recovery in the presence of a Byzantine leader. The key idea behind **BrandPiper** [119] to improve upon GRandPiper in terms of achieving absolute unpredictability of the next epoch is to consume inputs from more than t parties for computing each beacon output. This ensures the presence of at least one honest contribution and takes away the opportunity of being the sole contributor from the leader.

The two recent works of **SPURT** [53] and **OptRand** [23] take advantage of the aggregation property of the PVSS transcript (for commitments and encrypted shares) to design a leader-based DRB, with the leader acting as an orchestrator instead of a sole contributor. This design rationale is promising as it offers absolute unpredictability of the next epoch due to the involvement of $t+1$ contributions in the beacon output. Moreover, even if a leader decides to abort, it will not affect the unbiasability property as this is a blind action without knowing the beacon output. Notice that a consensus protocol (*i.e.*, SMR) is required to get everyone to agree on a threshold set of contributions picked up by the leader.

Insight 2 *Corrupting leader violates unbiasability if either the leader is the sole contributor with no recovery mechanism, or they can abort after observing the output prior to others. Also, corrupting the next t leaders violates unpredictability if the leader is the only contributor.*

5.2 Liveness

The majority of works using PVSS are constructed over synchronous network [23,24,41,89,120,128]. This is to ensure that sharing of parties' secrets arrives at the recipient within a certain time-bound to guarantee the completion of each epoch. Since randomness beacon is a continually-running service, [41] argues about the necessity of having the additional property of guaranteed output delivery (GOD) [113] as a *strong* notion of liveness. This property guarantees that (honest) parties always complete each epoch with a beacon output no matter what the adversary does. In [41,42,89], GOD is achieved via the use of a public bulletin board to post PVSS sharing by each party that essentially has the same effect as using broadcast channel that is a essentially a variant of consensus protocol [41]. These protocols can accomplish the fault tolerance of $t < n/2$ which is optimal in the synchronous setting.

In a leader-based construction where a single party advances the protocol at each epoch, achieving GOD becomes rather complicated as a (faulty) leader can withhold the beacon output or even abort the protocol. To address this challenge, [23, 24, 120] utilize the idea of *buffering* of secrets. The intuition behind buffering is that each party commits to a randomly chosen secret using PVSS in advance (*i.e.*, more than t epochs before) and updates their commitment with a fresh random value when they are selected as the leader again. The purpose of such buffering is twofold. First, it ensures GOD as no matter what the Byzantine leader decides to do, the honest parties in the protocol can jointly reconstruct the already committed secret and may also remove that leader from the protocol. Second, it ensures all the honest parties open the same committed value as the underlying consensus may require $t + 1$ epochs to ensure consistency.[5] While [23, 24] are optimally resilient randomness beacons in synchrony, [120] can only satisfy fault tolerance of $t < n/3$ under the same network assumption. This limitation arises from their need for a *quorum intersection* as part of their consensus protocol to avoid equivocation by the leader.[6]

An immediate consequence of ensuring GOD is the deployment of broadcast channels [76, 113] that are only achievable in the synchronous network. Recently, SPURT [53] proposed to relax the strong notion of GOD to just ensure that the adversary cannot abort the protocol *after* learning the beacon output at each epoch. This relaxation allows for a reduction in the reliance on broadcast channels and the possibility of replacing them with variants of SMR protocols [6, 132]. Roughly speaking, the leader of an epoch employs the SMR protocol to get parties to agree on a constant-sized message with the remaining heavy parts of the aggregated data being sent through private channels to each party individually to enable verification. Note that no buffering is used in SPURT as the protocol does not aim at achieving GOD.

Insight 3 *The notion of fairness in secure multiparty computation [52] is weaker than GOD. It states that if the adversary learns the output, it will not be able to prevent others from doing so as well, similar to SPURT [53].*

Adopting SMR has the benefit of implementing under a partial synchronous network [44, 132]. This, in turn, opens the door for *responsiveness*, a property allowing the randomness beacon protocol to proceed at the actual network speed and produce outputs faster than sticking to the conservative synchrony condition. This is important for achieving higher throughput and enhancing resilience against DoS attacks. As responsiveness is a potential property of systems without strict time dependencies, the recent work of [23] aims at achieving optimal fault tolerance of $t < n/2$ in a synchronous setting while enjoying the feature of *optimistic responsiveness* [107, 126], obtaining responsiveness only in certain occasions when the leader is honest and $f < n/4$. This approach is realized through designing a BFT-SMR that supports responsive leader change by eliminating the timeout of $\Omega(\Delta)$ in BFT-SMR of [24] to detect equivocation before parties make a commit.

Insight 4 *The (partial) responsiveness in synchrony requires a reduction in the number of faults from $n/2$ to $n/4$, while (complete) responsiveness in partial synchrony can always tolerate $n/3$ faults.*

RandShare [129] is presented as an asynchronous protocol, aiming to achieve termination as a liveness property without making timing assumptions. However, as discussed in [120], RandShare implicitly relies on a notion of time such that the protocol cannot guarantee liveness under asynchrony. The RandHound [129] protocol is driven by a single client which may abort a protocol run and enforce a restart.

[5] BFT-SMR is weaker than Byzantine broadcast in the sense that it does not require committing an honest leader's proposal at each view [7].

[6] Given a set of n elements, two sub-sets of size $n - t$ must intersect at $n - 2t$ elements. So, $n > 3t$ is needed to derive a contradiction.

5.3 Scalability

Despite offering uniform randomness and useful properties such as transcript aggregation, protocols using PVSS face a significant challenge in terms of the overhead imposed on participating parties. At a high level, every single party is supposed to commit to a secret value and broadcast the result to others. However, this broadcasting process entails sending a message of size $O(n)$, with a communication complexity of $O(n^3)$ based on the lower bound of $\Omega(n^2)$ [57]. As a result, the overall complexity becomes $O(n^4)$. Additionally, each PVSS sharing requires a computation and verification overhead of $O(n)$. A linear public verification cost is inevitable due to the need for verifying the transcript of protocol execution [53].

From a practical point of view, communication complexity is the main concern over the applicability of this line of randomness beacons, as they are inherently communication intense. The interesting work of SCRAPE [41] used the observation by [98] regarding the equivalence of (t, n) Shamir secret sharing and the Reed-Solomon Code [115] to introduce a PVSS scheme with linear computational complexity. However, their randomness beacon requires all parties to perform PVSS, resulting in an overall communication complexity of $O(n^4)$. RandHound [128] makes use of sampling techniques to shard a system of n parties to the subgroups of size c. Running a randomness beacon protocol within each subgroup and combining the results into the final outcome reduces the communication complexity from the total of $O(n^4)$ to $O(c^2n)$. RandHerd [128] takes the previous approach a step further by taking advantage of collective signing [129] to generate a sequence of random values, reducing communication (and computation) complexity to $O(c^2logn)$ and verification complexity to $O(1)$. Although sharding is a pivotal technique to reduce the overhead, it will cause overall fault tolerance reduction to ensure an honest majority for each shard [53]. Using packed secret sharing enables ALBATROSS [42] to produce a batch of $O(n^2)$ random values instead of one at each epoch, reducing the amortized communication and computation complexity to $O(n^2)$ and $O(n)$, respectively. Notice that, the resulting asymptotic boost in complexity comes with decreasing the fault tolerance as a natural trade-off, *i.e.*, $t \leq (n - l)/2$.

Recent works take advantage of their leader-based design to lower the overheads, particularly communication complexity [23, 24, 53, 120]. This is an effective approach to dramatically mitigate the use of broadcast channels by replacing all-to-all with an all-to-one/one-to-all communication pattern. In SPURT [53], the leader only broadcasts a constant-sized message (*i.e.*, cryptographic digest) and sends the bigger part of the data (*e.g.*, aggregated transcripts) over private channels, leading to a quadratic communication complexity.

GRandPiper, RandPiper, and OptRand [23, 24] deploy a BFT-SMR protocol without any use of threshold signature to commit transcript of size $O(n)$, which could potentially lead to cubic communication complexity. To maintain the quadratic cost, they use error-correcting code [115] to encode the leader's proposal, and accumulators [104] to enable efficient equivocation checking. Making use of a type of SMR that does not use threshold signature (and therefore DKG) is necessary to allow the system (efficiently) to support dynamic change in the set of parties. They also present a concrete reconfiguration mechanism for their systems to maintain the resilience of the protocol after eliminating some Byzantine party.

Insight 5 *Supporting dynamic participation for underlying consensus enables the leader-based DRB protocols with PVSS to deal with the reconfiguration procedure smoothly.*

6 Protocols using VRF

6.1 Security

Verifiable random function (VRF) [102], can be considered as an asymmetric variant of a pseudo-random function (PRF) [75], with public verifiability of the output. Intuitively, its security properties state that given a sequence of VRF values, $r_0, r_1, \ldots, r_{n-1}$, for any $n \geq 1$, it should not be computationally possible to distinguish r_n from the output of a random function. Such properties make VRF a promising fit for generating public randomness. Although the original work of [102] proposed a rather complex design, it was later shown a straightforward approach to build VRF is through applying a hash function modelled as random oracle to a unique signature scheme [73, 81]. The uniqueness of the signature matters to guarantee the security properties of VRF. Two of the most well-known unique signatures in the literature are RSA [125] and BLS [30].

Insight 6 *Applying a hash function to a signature results in a PRF in the random oracle model. When the signature is unique, it actually is a VRF with proof being the signature.*

DRBs using VRF follow a range of blueprint designs. **Algorand** [73] and **Ouroboros Praos** [55] deploy an integrated DRB protocol as part of their systems with a focus on achieving efficiency for large-scale usage rather than getting high-quality randomness. They do so by letting each party set up their own VRF key-pairs. Algorand has the taste of a leader-based protocol where the output of VRF evaluation by the leader at each epoch determines the epoch's beacon value and the next leader. Ouroboros Praos takes a slightly different approach by having the XOR of all submitted VRFs by parties to be the epoch's beacon output. Due to its randomized leader-based design, Algorand has probabilistic unpredictability. As observed by [66], in the constructions where parties individually generate VRF key-pairs, the security of the beacon output relies on an honest generation of keys. In fact, the adversary may generate a malicious key that affects the security of the output (*i.e.*, unbiasability). To address this issue, the authors in [43] design a VRF construction by adopting that of [55], guaranteeing the security of the VRF output under malicious key generation.

Gap 2 *In a non-threshold setting, the security of the output relies on a key-pair that has been generated honestly. Designing a secure DRB using VRF while tolerating maliciously generated key-pairs is a research gap.*

Unfortunately, these protocols do not satisfy unbiasability property as they are subject to the last actor attack. To resolve this issue, there are two typical options. First, introducing a computational delay prior to the release of the beacon output as used in the design of **Harmony** [2]. Second, deploying VRF in a distributed setting with threshold security. When it comes to setting up a threshold cryptosystem, a distributed key generation (DKG) [109] is the first step that allows parties to obtain a common pubic key, an individual public key and its corresponding secret key. Motivated by the work of **Cachin et al.** [35], a number of protocols such as **Drand** [3], **DFINITY** [36, 81], and **Glow** [66] construct a DRB with continuously signing a common value (*e.g.*, epoch's number) in a threshold manner to create a series of randomness. To create a *chain* of randomness, the common value is typically considered to be the epoch number concatenated with the last epoch's output so that each beacon value uniquely determines the chain of randomness all the way to the earliest one. In a threshold setting, the uniqueness property for an underlying signature scheme additionally requires any set of partial signatures of size above threshold results in the *same* signature. This consequently leads to the consistency of the produced randomness, removing the need to run some form of consensus among participating parties on the beacon output.

Insight 7 *The uniqueness of the threshold signature scheme is a crucial property with two main implications. First, it circumvents running a consensus. Second, it allows parties to multicast their signature shares instead of broadcasting which is a stronger requirement.*

Insight 8 *The chained and unchained DRB of [3] are different in terms of what they sign, one the epoch number and the other its concatenation with the last beacon output. However, due to the underlying threshold security both achieve the same security properties.*

The underlying threshold security guarantees no adversary controlling at most t parties neither can predict the future beacon outputs nor bias them. To instantiate a threshold VRF and so ensure pseudorandomness, the beacon output is computed by applying a cryptographic hash function on the signature in the random oracle model [66, 81]. The basic security definition of a VRF can straightforwardly be extended to the threshold setting [66]. Due to the similarities in the concept of VRF and PRF, [66] adapts and revisits the definition of standard and strong pseudorandomness presented in [133] with respect to a threshold VRF. They provide a framework to build a DRB protocol from any threshold VRF instances in a secure way. At a high level, compared to the standard definition, the strong pseudorandomness preserves the security against an adversary with the additional power of getting partial evaluations on its challenges, selecting the corrupted parties' local secret keys, and influencing the public parameters computation. As observed by [66], all the constructions in [2, 55, 73] fail to satisfy the two types of pseudorandomness due to their avoidance of a threshold procedure.

Gap 3 *One caveat with using threshold VRF is that the initial seed cannot be used to generate unbiased randomness forever as the entropy is limited. It is worth exploring the process of quality degeneration of randomness over time and the way to handle it efficiently.*

Mt. Random [43] presents a DRB with three layers each one providing a different type of randomness in exchange for different security/performance trade-offs. More accurately, the first layer generates uniform randomness; the second layer generates pseudorandomness, and the last layer generates (bounded) biased randomness. They design such a construction by nicely combining different primitives including PVSS, threshold VRF, and VRF in each layer respectively. Moreover, each lower layer feeds the higher one with seeds to provide a consistent level of bias across the structures. Beaver et al. [17] proposed a DRB protocol called **STROBE** where the beacon output at each epoch x_r is computed by repeated RSA decryption of the previous epoch's output x_{r-1} in a distributed way. In fact, after running a setup phase assuming a dealer who generates the RSA modulus N and Shamir decryption key shares sk_i, each party disseminates their contribution x^{sk_i} and the epoch's output is computed by aggregating a threshold number of valid contributions through performing a Lagrange interpolation in the exponent. This construction is an extension to the work of [18] which was the first distributed randomness beacon producing values by repeated squaring of a random seed. Distributed generation of RSA modulus [45] is an old but still active line of research that can be used to get rid of the trusted dealer in this protocol.

Gap 4 *The main reason to use threshold VRF for DRB is to prevent biasing. Adopting a leader-based design with VRF while handling the bias with other routes helps to avoid a DKG. One possible approach would be deploying a digital signature with key extraction [9].*

Gap 5 *Inspired by the construction in [43] that outputs different types of randomness, it would be interesting to build a construction that generates beacon outputs under various thresholds, where a higher threshold could imply better randomness quality. Adopting the "Multiverse" DKG proposed in [15] to generate a unique signature could essentially lead to such DRB.*

6.2 Liveness

The DRB protocols in [55, 73] are built on top of their underlying distributed ledger, with the former using a BFT-type consensus in partial-synchrony with $t < n/3$ and the latter having parties communicate in a synchronous peer-to-peer fashion with $t < n/2$.[7] In a stand-alone DRB protocol like [88], the fact that parties evaluate their VFRs individually necessities deploying broadcast channels and consequently synchronous network settings.

Protocols deploying threshold VRF, however, do not need to use broadcast channels for producing their beacon outputs. This implicitly results in constructions that can be implemented in a *non-synchronous* setting while supporting the corresponding fault tolerance (*i.e.*, $t < 1/3$). Therefore, DFINITY [36, 81] is presented in partial-synchrony and Drand [3] in synchrony, despite having the same protocol flow. However, several existing DRB protocols using threshold VRF, such as [17, 43, 66], make synchronous assumptions due to the use of DKG in their setup phase [69]. The context of asynchronous DKG [5,54] has recently started receiving more attention which has a direct effect on constructing asynchronous DRB protocol. The protocol proposed by Cachin et al. [35] works independently of network delay and therefore is suitable for an asynchronous setting.

Insight 9 *Designing a DRB protocol in an asynchronous setting is tricky due to the fact that randomness itself is needed to get around FLP impossibility result [62]. With an asynchronous DKG, however, it is possible to generate public randomness via computing a threshold VRF in a single-shot manner and then repeat the process multiple times.*

One important consideration in protocols with a chain of randomness is that producing beacon output at each epoch is necessary for initiating the next one. This means if the protocol fails to output at any epoch (for any reason), the next epochs will also be influenced. Thus, satisfying GOD property is necessary for the correct operation of such protocols. Protocols like Algorand [73] and Ouroboros Praos [55] where parties individually evaluate their VRFs, cannot feature GOD due to the possibility of abortion.

6.3 Scalability

DRBs with individually set VRFs such as [73, 88, 89] are quite efficient in terms of computation and verification cost with a constant overhead of $O(1)$. However, they still need a quadratic communication complexity for broadcasting or deploying a public bulletin board.[8]

Running a DKG is the most expensive part of protocols using threshold VRF that dominates the communication and computation complexity. There are various DKG constructions in the literature among which [70, 77, 85, 91] are widely used. DFINITY uses the non-interactive distributed key generation of [77] with $O(n^3)$ communication complexity, while Drand employs [70] that incurs at least $O(n^3 \log n)$ overhead. As another example, to generate key materials for threshold BLS it still takes elaborate protocols with $O(n^4)$ communication complexity [85, 91]. Running such a setup phase is for once and when done, the parties can perform the rest of the process at a much lower cost. If needed, one can also use a publicly verifiable DKG [78] in the setup phase. Aggregation properties enable this DKG to offer lower communication and computation costs and can be used to instantiate verifiable unpredictable function (VUF), which essentially works as a randomness

[7] In the original work of [55] the network was assumed to be "semi-synchronous". Later it was shown that a longest-chain consensus requires synchronous assumption to work safely [116].

[8] Algorand [73] uses random committees of size c. This makes the communication complexity reduced to $O(cn)$, however, asymptotically remains quadratic if c depends on n.

beacon.[9] Excluding the DKG setup, the DRB protocols using threshold VRF are way more efficient compared to the ones using PVSS. However, all the protocols of [17, 35, 81] still have a quadratic communication complexity of $O(n^2)$ due to multicasting a partial signature by each party to others at each epoch. Each party then needs to verify the received set of messages, leading to a linear cost. An external verifier just needs to verify the final beacon output against the common public key, providing optimum public verifiability of $O(1)$. One major limitation of having DKG as the setup phase is an inability to support dynamic participation, due to the need for re-running the DKG whenever a new party joins or leaves.

7 Protocols Using VDF

7.1 Security

Although time-based primitives have been around for decades [117], the evolution of VDF in recent years [27,60,131] as an interesting variant has opened new doors to take advantage of their promising features to produce public randomness [27, 60, 80, 119]. The most common VDF constructions are those based on repeated squaring in groups of unknown order, like an RSA group, to enforce a guaranteed computational delay [112, 131]. The properties of VDF including uniqueness, sequentially, and public verifiability make it a solid option to construct a randomness beacon by simply using its output iteratively evaluated on an initial seed at regular points in time. It is worth mentioning that the sequentially of VDF implies unpredictability and its uniqueness together with timed-dependent evaluation implies unbiasability. However, a VDF provides efficient verification for a complete invocation, and not for every iteration. That is the main motivation behind the design of the Continuous VDF (cVDF) [60] that addresses this issue by introducing a primitive that enables efficient verification of a VDF at each iteration, which is independent of the time parameter. Thus, starting with an unpredictable seed, cVDF is iteratively applied to produce the epoch's output which is also used as the seed for the next epoch. As with VRF, one needs to apply a cryptographic hash function on the resulting VDF output to turn beacon outputs pseudorandom [60].

Gap 6 *The construction using cVDF in [60] is a centralized randomness beacon. It would be interesting to explore how to use such a primitive with iterative public verification to build a DRB. Using the notion of collaborative VDF [100] may be a direction to look into this further.*

The first concrete attempt towards constructing a DRB using VDF was due to the researchers at Ethereum Foundation [58]. Their proposal is based on applying a VDF on the aggregated local randomness from a set of parties through the commit-reveal process. The necessity of computing VDF prevents the adversary from learning the beacon output before revealing its input. **HeadStart** [93] is a recent randomness beacon protocol that follows a commit-reveal approach where a centralized organizer computes the VDF. To protect against a colluding organizer, it needs to compute the Merkle root of all the contributions and publishes the respective membership proofs prior to the release of the beacon output.

RandRunner [119] constructs a DRB protocol by cleverly making use of *trapdoor* VDF, enabling fast computation of VDF with some additional knowledge. The core idea behind the construction is to have each party designated as the epoch's leader efficiently compute VDF on the last beacon output using their trapdoor and broadcast the result to others. The crucial part is to ensure that the knowledge of the trapdoor does not give the leader an opportunity to threaten the uniqueness

[9] The DKG of [78] outputs group elements as secret keys and is not suitable for BLS signature that requires a field element.

of the VDF, and is merely used to accelerate the process of computation. To this end, given the security properties of [112] as the underlying VDF construction, a setup phase is required where each party must generate a NIZK [37] to show the correctness of the RSA modulus.[10] Observe that this procedure allows getting around an *interactive* setup phase for RSA modulus generation as each party just needs to *individually* set up their own VDF. An important consideration is the necessity of ensuring *strong* uniqueness for trapdoor VDF in case public parameters are adversarial generated, similar to the setting where parties set up their VRFs individually [66]. With a deterministic leader election, the protocol provides absolute unpredictability only after $d = \alpha \cdot t$ epochs which α denotes the adversary's computational advantage in evaluating VDFs compared to that of honest parties.

Gap 7 *As the only leader-based protocol using VDF, RandRunner [119] ensures absolute unpredictability of the next $t + 1$ epochs. More research needs to be conducted on how to reduce this preferably to one epoch.*

The latest attempt to design a DRB according to the folklore paradigm of commit-reveal is **Bicorn** [47], featuring an interesting approach to address the last actor attack efficiently. The subtle novelty of the approach is to enable the recovery of missed contributions just by doing *one* sequential computation, no matter how many parties abort. This is achieved by having each party commit to their contribution α_i as $c_i = g^{\alpha_i}$ with the resulting beacon output being $\prod_{i=1}^{n} h^{\alpha_i}$, where $h = g^{2^T}$ is a VDF evaluation with parameter T. In the event of abortion, this design allows anybody to compute the beacon output via $\prod_{i=1}^{n} (c_i)^{2^T}$, resembling a VDF computation on the product of commitment $\prod_{i=1}^{n} (c_i)$. Bicorn [47] has a similar flow as leaderless protocols using PVSS in the sense that they both follow a "commit-reveal-recover" paradigm [47], but the former handles the recovery with a slow computation and the latter does that with reconstruction due to its threshold security.

Insight 10 *Bicorn [47] shows the importance of the way beacon output is determined in a commit-reveal manner. Using an exponentiation operation results in an (optimistically) efficient protocol with optimal timed recovery, i.e., one slow computation for many aborts.*

7.2 Liveness

Time-based cryptography enables circumventing the result of [51], showing the impossibility of doing a secure coin flipping without an honest majority. That is, dishonest majority can be tolerated due to the possibility of timed recovery and security can be preserved due to the sequential nature of such time-dependent computation, even against parallel processors.

Although time-based cryptography can potentially relax the underlying network assumption and enable achieving the highest fault tolerance of $n - 1$ [48, 99], there exists some hurdle in realizing these exciting properties. Working in a synchronous setting seems to be critical for the current DRB protocols using VDF [47, 58, 93, 119] for two reasons. First, the reliance on time/timeout may be needed to indicate different stages of the protocol [47, 58, 93]. Second, the use of a public bulletin board or broadcast channel is assumed in [47, 119] to ensure consistency or security. More precisely, in Bicorn [47] parties should have access to a public bulletin board to consistently post their commitment in the commit phase, which as already mentioned has the same effect of implementing a Byzantine broadcast. In RandRunner [119], the corrupted leader may try to violate the unpredictability of the protocol by *selectively* disseminating the beacon output to a portion of parties, forcing others to go through slow computation to catch up. Deploying broadcast channels in the system is a countermeasure that consequently implies a fault tolerance of $t < n/2$. Due to the

[10] It should be the product of two safe primes of form $2p + 1$.

strong uniqueness of the used VDF that ensures the equality of the VDF evaluation normally and with a trapdoor, the only aspect of the protocol suffered from periods of asynchrony (*i.e.*, network partitions) is the unpredictability as anybody can compute the missed beacon output by a slow VDF computation, offering unbiasability and GOD.

7.3 Scalability

The major issue with this type of DRB is the need for performing sequential computation which is a highly energy-consuming task, introducing latency and dampening throughput. The idea of trapdoor VDF in [119] allows a fast beacon computation for the leader and efficient verification of $O(1)$ for verifiers. The same is the case in [47], except in leaderless fashion with $O(n)$ verification cost. The necessity of deploying (reliable) broadcast or public bulletin board in the system leads to a cubic communication complexity for [47] and quadratic communication complexity for [119] due to its leader-based style. Gossiping is an alternative approach that has lower complexity of $O(n \log n)$ but increases the latency. While the set of parties is known and fixed in [119], the two works of [47, 93] support *public* participation, allowing parties to efficiently come and go without knowing the set of parties in advance.

8 Protocols Using Public Blockchain

8.1 Security

Some of the earlier works for building randomness beacons were based on using a source of information as high entropy data that is publicly available [22, 32, 50, 94, 111]. High-quality randomness is typically extracted by applying a randomness extraction function to parts of the corresponding data. Among the existing public data structures, proof-of-work (PoW) blockchain is considered the most suitable one as it is always available (unlike financial markets in [50]) and inherently comes with additional security properties like the underlying Nakamoto consensus [103]. The randomness used in the process of mining a block (*e.g.*, PoW puzzle) can be utilized to extract a large number of unpredictable random bits. However, as mentioned in [34], two types of manipulation attacks can be launched: (1) a miner could withhold proposing a valid block just because it does not lead to the desired randomness [11]; (2) due to the network latency it is possible that forks occur. One might also consider an attack scenario to affect the beacon output just by manipulating the network to prevent or delay the propagation of a particular block producing an undesirable output. Although imposing financial penalties or slashing is a well-known way to restrict miners from such manipulation attacks [10, 13],[12] the possible gain from attacks may be unbounded while any penalty is bounded. A detailed security analysis on the ability of malicious miners is carried out in [22], showing the impossibility of deriving even one single truly random bit when the attacker has a considerable fraction of the total computing power. On the other hand, when the attacker has a limited computing power that is not enough for block production, generating truly random bits is feasible. The work of [105] presents a model for uniform randomness extraction over a public blockchain that has each party outputs a public value together with secret values to the other parties in a multi-round fashion.

As common in the DRB literature, incorporating a delay function is a working method to protect against a malicious miner via imposing a delay period only after which the output is achievable. [34, 94] proposed two protocols for augmenting such delay functions based on computing modular

[11] It is costly as they require to spend considerable time and computational efforts on this action.

[12] We note that slashing is an established method for a proof-of-stake (PoS) setting where the actors have some stake already deposited in the system.

square root and iterating a pseudorandom permutation (*e.g.*, block cipher) or hash function as compositionally-sequential functions. Notice that despite having a (randomized) leader-based style with the miner being the epoch's leader, these protocols can provide absolute unpredictability, even for the current epoch's beacon output, thanks to intrinsic randomness lied in the system (plus the use of delay). **Randchain** [80] combines security properties of Nakamoto consensus and delay functions to address two main issues in a PoW blockchain-based system including biasability, and unfairness, *i.e.*, parties with high computational power dominating the randomness generation. It introduces a primitive called SeqPoW which is a puzzle that, unlike the typical PoW puzzle, cannot be solved faster using multiple parallel processors. To put it another way, SeqPoW is a cryptographic puzzle that takes a random and unpredictable number of sequential steps to solve. Similar to VDFs, sequentiality in SeqPoW also implies unpredictability.

Insight 11 *Proof-of-work mechanism in Nakamoto consensus prevents equivocation by the leader (e.g., miner) via making it costly, somewhat resembling the use of (threshold) signature in BFT-type consensus.*

8.2 Liveness

As Nakamoto consensus works safely only in synchrony [116], the randomness beacon protocol built on top of it inherent the same network assumption [32, 34, 80]. More precisely, a non-synchronous network condition (partial-synchronous or asynchronous) leads to arbitrarily long network partitions, violating the consistency of the system. Moreover, these protocols can tolerate a (computationally) honest majority of $t < n/2$.

8.3 Scalability

The properties of DRBs using public blockchain rely on their underlying distributed ledger and cannot be treated in a stand-alone regime. In [32, 34, 80] the party who first finds the solution to a puzzle, publishes the result *globally* to the peer-to-peer network. In such a network, the message is delivered by a sender to a random subgroup of parties (*i.e.*, its peers) in different steps until all the parties receive the message. Thus, the parties do not necessarily know each other in contrast to the network with point-to-point channels. Although the availability of a (public) distributed ledger facilitates the process, one may argue that posting to a blockchain could be more costly compared to implementing broadcast channels. That is, the former usually contains a large population of parties that are not necessarily involved in the randomness generation, but the latter is only concerned with a limited set of participating parties. Verification should be done efficiently just by checking the correctness of the solution, resulting in $O(1)$ cost. In these protocols, the participating parties may dynamically change over time, featuring public participation.

9 Discussion

We now look at some aspects of DRB protocols that are worthy enough to be highlighted independently from the systematization already provided.

Adaptive Security. There are two widely known strategies that an adversary can adopt to corrupt parties: *static corruption*, where the set of corrupted parties is fixed and known to the adversary before the protocol begins; and *adaptive corruption*, where the adversary can corrupt any party it wishes based on its view of the protocol. The adversary's capability is still limited to corrupt only a certain number of parties and they remain corrupted until the end of protocol [40, 96]. In [106],

Table 1: Comparison of distributed randomness beacons (DRBs).

Category	Protocol	Security		Liveness		Scalability			Setup	Dynamic	Adap.	GOD	Resp.
		Unpre.	Unbias.	Net.	Faults	Comm.	Comp.	Veri.					
PVSS	**RandShare*** [128]	✓	✓	async.	$n/3$	$O(c^2n)$	$O(c^2n)$	$O(c^2n)$	CRS	✓	✓	✓	✓
	RandHound [128]	✓	✓	sync.	$n/3$	$O(c^2n)$	$O(c^2n)$	$O(c^2n)$	CRS	✓	✗	✓	✗
	RandHerd [128]	✓	✓	sync.	$n/3$	$O(c^2 \log n)$	$O(c^2 \log n)$	$O(1)$	DKG	✗	✗	✗	✗
	Ouroboros [89]	✓	✓	sync.	$n/2$	$O(n^4)$	$O(n^3)$	$O(n^3)$	CRS	✓	✗	✗	✗
	SCRAPE [41]	✓	✓	syn.	$n/2$	$O(n^4)$	$O(n^2)$	$O(n^2)$	CRS	✓	✗	✗	✗
	Hydrand [120]	$t+1$	✓	sync.	$n/3$	$O(n^2)$	$O(n)$	$O(n)$	CRS	✓	✗	✗	✗
	ALBATROSS [42]	✓	✓	sync.	$n/2$	$O(n^2)$	$O(n)$	$O(1)$	CRS	✓	✗	✗	✗
	GULL [43]	✓	✓	sync.	$n/2$	$O(n^4)$	$O(n^2)$	$O(n^2)$	DKG	✗	✗	✗	✗
	GRandPiper [24]	$t+1$	✓	sync.	$n/2$	$O(n^2)$	$O(n^2)$	$O(n)$	CRS††	✓	✗	✓	✓
	BRandPiper* [24]	✓	✓	sync.	$n/2$	$O(n^3)$	$O(n^2)$	$O(n^2)$	CRS††	✓	✓	✓	✓
	SPURT [53]	✓	✓	p.sync.	$n/3$	$O(n^2)$	$O(n)$	$O(n)$	CRS	✓	✗	✗	✓
	OptRand [23]	✓	✓	sync.	$n/2$	$O(n^2)$	$O(n)$	$O(n)$	CRS††	✓	✗	✓	✓
VRF	**Cachin et al.** [35]	✓	✓	async.	$n/3$	$O(n^2)$	$O(n)$	$O(1)$	DKG	✗	✗	✓	✓
	DFINITY [81]	✓	✓	p.sync.	$n/3$	$O(n^2)$	$O(n)$	$O(1)$	DKG	✗	✓	✓	✓
	Drand [3]	✓	✓	sync.	$n/2$	$O(n^2)$	$O(n)$	$O(1)$	DKG	✗	✓	✓	✗
	Algorand [73]	$\Omega(t)$	✗	p.sync.	$n/3$	$O(cn)$	$O(1)$	$O(1)$	CRS	✗	✓	✗	✓
	Ouroboros Praos [55]	$\Omega(t)$	✗	sync.	$n/2$	$O(n^2)^\dagger$	$O(n)$	$O(n)$	CRS	✓	✓	✗	✗
	Harmony [2]	$\Omega(t)$	✓	p.sync.	$n/3$	$O(n^2)$	VDF	$O(n)$	CRS	✓	✗	✗	✗
	Glow [66]	✓	✓	sync.	$n/2$	$O(n^2)$	$O(n)$	$O(1)$	DKG	✓	✗	✓	✗
	STROB [17]	✓	✓	sync.	$n/2$	$O(n^3)$	$O(n)$	$O(1)$	DKG	✗	✗	✓	✗
	Kiayias et al. [88]	✓	✓	sync.	$n/2$	$O(n^3)$	$O(n)$	$O(1)$	CRS	✓	✓	✗	✗
VDF	**RandRunner** [119]	$t+1$	✓	sync.	$n/2$	$O(n^2)$	VDF	$O(1)$	CRS	✓	✗	✓	✓
	HeadStart [47]	✓	✓	sync.	n	$O(n)^\dagger$	$O(n)$	$O(1)$	-	✓	✗	✗	✗
	Bicorn [93]	✓	✓	sync.	n	$O(n^3)$	$O(n)$	$O(n)$	-	✓	✓	✓	✗
Blockchain	**Bitcoin** [32]	✓	✗	sync.	$n/2$	$O(1)^\dagger$	PoW	$O(1)$	CRS	✓	✓	✗	✗
	Proof-of-Delay [34]	✓	✓	sync.	$n/2$	$O(1)^\dagger$	VDF	$O(\log \lambda)$	CRS	✓	✓	✗	✗
	RandChain [80]	✓	✓	sync.	$n/2$	$O(1)^\dagger$	PoW	$O(1)$	CRS	✓	✓	✗	✗

* =: VSS is used; † =: Public bulletin board is assumed; †† =: Private setup is assumed.

two levels of adaptiveness are considered for an adversary, namely *fully* and *mildly*. The former refers to a strong adversary that may corrupt the victims instantly while it takes some time for the latter to corrupt a new party. Most of the public randomness protocols in the literature consider the static adversary. However, providing proven security against an adversary enjoying adaptiveness is rather delicate and requires more work. RandRunner [119] offers absolute unpredictability of the next $t+1$ epochs for both static (worst-case) and adaptive adversary while it offers probabilistic unpredictability against a *mild* adaptive adversary since the protocol can make progress before the adversary gets a chance to corrupt new leaders.

Insight 12 *In a leader-based protocol with the leader solely contributing to the beacon output, an adaptive adversary can corrupt up to t consecutive leaders and violate the unpredictability. So, the adaptive adversary essentially acts as a static adversary in a worst-case scenario.*

As mentioned in [88], the single or multi-round design of the protocol plays a critical role in providing adaptive security. A single-round protocol where each party only speaks once during each epoch can provide adaptive security as the best the adversary can do is to try and randomly corrupt parties before they contribute; otherwise, any corruption would be useless. This model is known and formalized under the notion of YOSO, you only speak once [21, 71]. Therefore, the important observation in designing an adaptively secure leader-based protocol is that the adversary should not be able to detect the next leader ahead of broadcasting the result. With this in mind, Algorand [73] provides probabilistic unpredictability in the presence of an adaptive adversary. The work of **Kiayias et al.** [88] can be thought of as a parallelization of Algorand where instead of having a leader at each epoch, all parties individually compute the VRF on a common seed and broadcast it to others. Eventually, the hash of the least k submissions (*i.e.*, VRFs with smallest values) determines the round's beacon output. This approach rules out the possibility of an adversary controlling the whole set of contributors to breach the security as it is shown the probability of occurring such an event – the k-th smallest adversarial contribution being less than the honest one – is bounded

with the appropriate choice of parameters, ensuring at least one honest contribution is included to make the beacon output secure. As a result, the protocol is secure against an adaptive adversary corrupting up to $t < n/2$ parties.

Gap 8 *It is interesting to explore how to construct an adaptively secure DRB protocol with absolute unpredictability, where the leader solely contributes to the beacon output at each round. Such protocol should likely resemble the Nakamoto-style structure, where by the time the adversary decides to corrupt a party, they have already revealed the contribution.*

Since we are not yet aware of the suitable proof techniques showing the adaptive security of the existing PVSS schemes, it is rather challenging to argue about the adaptiveness of protocols using PVSS despite not knowing concrete attack against them [53]. It is clear, however, to argue that in such multi-round protocols, parties go through at least two logical steps of commit, and reveal. So, to provide security against an adaptive adversary using more than t contributions from distinct parties is inevitable. This is the way BrandPiper [24] achieves adaptive security, employing the communication-wise efficient VSS scheme of [86] with more than t contributions involved in producing the beacon output at each epoch.

Gap 9 *Investigating the adaptive security of PVSS is a vital research question. This, in turn, leads to analyzing the adaptiveness of resulting DRB protocols.*

Leaderless protocols using VRF, in particular those based on threshold signature, can be viewe as a single-round protocol excluding the one-time DKG phase. So, adaptiveness should be investigated in the combination of the two sub-protocols. As realized in [40], one major problem with adaptive security for threshold scheme is constructing a simulator to simulate the adversary's view in the real execution of the protocol as it is hard to predict which subset of parties is corrupted.[13] The authors in [8, 40] use some techniques such as erasing the secrets, rewinding, and zero-knowledge proof to achieve an adaptively secure threshold RSA signature on top of a static-secure one. A requirement of such transformation is the refreshment of partial secret keys after each signature generation that additionally leads to proactive security [83, 97], a level of security required when parties are under threat of losing/leaking their secret keys. Very recently, Bacho and Loss [14] introduced an adaptive security proof in algebraic group model (AGM) [63] for threshold BLS signature which could essentially bring the respective randomness beacon protocols the joy of adaptiveness.

Simulation-based Security. It is not difficult to see that a DRB protocol is actually a secure multi-party computation (MPC) [74, 96]. Treating protocols based on some specific properties, like the majority of existing works, poses the threat of not covering all the required ones. This consequently demands paying more attention to the grounded paradigm of *real/ideal simulation* which is a well-known security formulation in the context of secure computation. By defining an ideal functionality that acts as a trusted entity faithfully carrying out the computations, a secure system is defined by comparing the real-world execution of the protocol and the execution within the presence of the ideal functionality. This approach apart from having the advantage of capturing the security concerns, allows moving towards *composable security* [38, 39] which is an important but overlooked necessity for protocols producing public randomness. Such protocols, even when designed in a stand-alone manner, often are deployed within a larger system and may have interactions with other sub-protocols to provide required fresh randomness. To the best of our knowledge, only recently a few rare attempts has been made in this direction [16, 42].

[13] Unless we assume that the simulator knows all the private shares of parties which do not make sense.

Insight 13 *Looking at DRB as a type of secure multiparty computation together with its resemblance to SMR, allows arguing about its properties formally by adopting a security definition from the former and a liveness definition from the latter.*

History Generation. STROBE [17] put forth an interesting property for a DRB regarding efficient (re)generation of the beacon history, given the current epoch's output. This novel feature is of importance in applications that require a high-throughput stream of randomness and are likely to suffer from occasional disconnections, *e.g.*, online gaming. So, at epoch r, it is possible to generate all the previous beacon outputs $\{x_1, \ldots, x_{r-1}\}$ using x_r and public key. This also implies *self-verification*, enabling verification of each output only against the previous ones with no need for any NIZK proof. This property is the direct result of using a (threshold) RSA signature where the secret key is the multiplicative inverse of the public key, allowing to trace the chain of randomness back by iterative encryptions. Despite the possibility of generating the history, it gets more computationally expensive (*i.e.*, grows exponentially) as the number of epochs to (re)generate increases.[14] With threshold BLS, it is possible to create a unique chain of randomness, and with threshold RSA, it is possible to have continual back and forth on such a chain.

Insight 14 *In a (plain) secret sharing, the shares do not carry any self-verifying information. But, in STROBE [17] this is the case thanks to the underlying chain of RSA decryption.*

Insight 15 *As a trade-off, the history generation property diminishes randomness quality from being truly random or pseudorandom to just being unpredictable as it is not possible to get the full history from a fresh (truly/pseudo)random value.*

Setup Assumption. DRB protocols typically need to use some form of setup assumption for initializing the process. This can be either for efficiency purposes (*e.g.*, bootstrapping) or to enable the implementation of elaborate operations. One well-known example of such a setup is PKI which is a functionality F whose responsibility is to relay parties' public keys to others. Depending on whether such functionality F outputs public or private values to parties, we can refer to it as a *public* or *private* setup. In settings with a public setup, all parties receive the same value from the functionality. This can be some group specification (*e.g.*, group generator) or an initial seed, known as a common reference string (CRS). DRB protocols using PVSS and VDF can be implemented with such a public setup. We remark that deploying VDF also needs a setup phase for its underlying group of unknown order. However, a sidestep would be either using class groups of imaginary quadratic fields [33] or trapdoor VDF augmented with a NIZK [119]. On the other hand, protocols with a private setup need to either rely on a trusted party or run an MPC protocol (*e.g.*, DKG) to generate secret values that must be kept hidden during the protocol execution. This process is an efficiency bottleneck and makes the *re-configuration* problematic [24], *i.e.*, parties cannot be replaced easily once the setup gets executed.

Cryptographic Assumption. Underlying cryptographic assumptions play an important role in analyzing the security of the resulting DRB protocol. One well-known assumption used in a range of randomness beacon protocols is random oracle [19]. Particularly, the beacons with pseudorandom outputs [3, 43, 55, 66, 81, 119] make use of such assumption to turn the unpredictable beacon values to the pseudorandom ones that are indistinguishable from a uniform distribution. This assumption is also used in the PVSS scheme of [41, 122], allowing more efficient constructions. Another popular assumption in the context of randomness beacons is decisional Diffie-Hellman (DDH) [31] commonly

[14] It is possible to deploy some techniques to decrease the proof size and the running time. For more details see [17].

served as the standard assumption for underlying PVSS and DKG schemes [41–43, 46]. Some protocols also may rely on stronger assumptions that are less standard and their security properties are not studied substantially. For instance, SCRAPE in the plain model uses pairings, relying on the hardness assumption of decisional bilinear squaring (DBS) [82].[15] SPURT [53] makes modifications to the pairing-based PVSS of SCRAPE to turn its security assumption to the more standard decisional bilinear Diffie-Hellman (DBDH) assumption [31]. Also, repeated squaring (RSW) [117] is the common hardness assumption for DRB protocols using VDF.

Gap 10 *Although the leaderless DRB of [41] works with a version of PVSS in the random oracle model, all the existing leader-based protocols need to use pairings. Designing a leader-based protocol without pairings considerably boosts performance.*

Post-quantum Security. Looking over DRB literature lets us know that the emergence of quantum computers could be a real threat to the existing protocols due to their cryptographic hardness assumptions. In particular, bilinear-map-based PVSS is susceptible to quantum attacks [124], making the corresponding DRB protocols insecure. Gentry, Halevi, and Lyubashevsky [72] recently proposed a proactive and non-interactive PVSS scheme in which the underlying encryption scheme is based on the learning with errors (LWE) problem [110]. Although the adoption of LWE encryption in their lattice-based scheme is primarily motivated by scaling PVSS in large-scale systems (*e.g.*, where committees may scale to hundreds or thousands of parties), the secrecy of their PVSS is preserved even against quantum attackers. HERB [46] builds a randomness beacon protocol with additive homomorphic threshold encryption where a group of parties encrypt their local randomness and any threshold of participant can retrieve the aggregated beacon value. A potential advantage of this construction is its ability to resist quantum attacks by replacing fully homomorphic lattice-based schemes, supporting distributed key generation and threshold decryption [20].

Gap 11 *To the best of our knowledge, there exists no concrete effort in the literature to design a post-quantum secure DRB protocol. Deploying lattice-based PVSS [72] or isogeny-based VDF [56] are two possible tools to do so.*

Acknowledgments. The authors would like to thank Ewa Syta and Renas Bacho for helpful discussions.

References

[1] League of entropy. `https://en.wikipedia.org/wiki/League_of_Entropy`.

[2] Team harmony, technical whitepaper - version 2.0. `https://harmony.one/whitepaper.pdf`.

[3] Team drand, drand project website. `https://drand.love`, 2020.

[4] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, and G. Stern. Bingo: Adaptively secure packed asynchronous verifiable secret sharing and asynchronous distributed key generation. *Cryptology ePrint Archive*, 2022.

[5] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 363–373, 2021.

[15] SCRAPE PVSS scheme is proposed in two versions, one in the random oracle model under DDH assumption and one in the plain model under DBS assumption.

[6] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 106–118. IEEE, 2020.

[7] I. Abraham, K. Nayak, L. Ren, and Z. Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 331–341, 2021.

[8] J. F. Almansa, I. Damgård, and J. B. Nielsen. Simplified threshold rsa with adaptive and proactive security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 593–611. Springer, 2006.

[9] O. Alpos, Z. Wang, A. Kavousi, S. Y. Chau, D. Le, and C. Cachin. Dske: Digital signature with key extraction. *Cryptology ePrint Archive*, 2022.

[10] M. Andrychowicz and S. Dziembowski. Distributed cryptography based on the proofs of work. *Cryptology ePrint Archive*, 2014.

[11] Z. Avarikioti, A. Desjardins, L. Kokoris-Kogias, and R. Wattenhofer. Divide & scale: Formalization and roadmap to robust sharding. In *Structural Information and Communication Complexity: 30th International Colloquium, SIROCCO 2023, Alcalá de Henares, Spain, June 6–9, 2023, Proceedings*, pages 199–245. Springer, 2023.

[12] S. Azouvi and D. Cappelletti. Private attacks in longest chain proof-of-stake protocols with single secret leader elections. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 170–182, 2021.

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

[14] R. Bacho and J. Loss. On the adaptive security of the threshold bls signature scheme. *Cryptology ePrint Archive*, 2022.

[15] L. Baird, S. Garg, A. Jain, P. Mukherjee, R. Sinha, M. Wang, and Y. Zhang. Threshold signatures in the multiverse. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2057–2073. IEEE Computer Society, 2022.

[16] C. Baum, B. David, R. Dowsley, R. Kishore, J. B. Nielsen, and S. Oechsner. Craft: C omposable r andomness beacons and output-independent a bort mpc f rom t ime. In *Public-Key Cryptography–PKC 2023: 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7–10, 2023, Proceedings, Part I*, pages 439–470. Springer, 2023.

[17] D. Beaver, K. Chalkias, M. Kelkar, L. K. Kogias, K. Lewi, L. de Naurois, V. Nicolaenko, A. Roy, and A. Sonnino. Strobe: Stake-based threshold random beacons. *Cryptology ePrint Archive*, 2021.

[18] D. Beaver and N. So. Global, unpredictable bit generation without broadcast. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 424–434. Springer, 1993.

[19] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[20] R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *Theory of Cryptography: 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings 7*, pages 201–218. Springer, 2010.

[21] F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin. Can a public blockchain keep a secret? In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, pages 260–290. Springer, 2020.

[22] I. Bentov, A. Gabizon, and D. Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016.

[23] A. Bhat, A. Kate, K. Nayak, and N. Shrestha. Optrand: Optimistically responsive distributed random beacons. *Cryptology ePrint Archive*, 2022.

[24] A. Bhat, N. Shrestha, Z. Luo, A. Kate, and K. Nayak. Randpiper–reconfiguration-friendly random beacons with quadratic communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3502–3524, 2021.

[25] G. R. Blakley and C. Meadows. Security of ramp schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 242–268. Springer, 1984.

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

[27] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

[28] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco. Single secret leader election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 12–24, 2020.

[29] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, pages 213–229. Springer, 2001.

[30] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 416–432. Springer, 2003.

[31] D. Boneh and V. Shoup. A graduate course in applied cryptography. *Recuperado de https://crypto. stanford. edu/˜ dabo/cryptobook/BonehShoup_0_4. pdf*, 2017.

[32] J. Bonneau, J. Clark, and S. Goldfeder. On bitcoin as a public randomness source. *Cryptology ePrint Archive*, 2015.

[33] J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, 1988.

[34] B. Bünz, S. Goldfeder, and J. Bonneau. Proofs-of-delay and randomness beacons in ethereum. *IEEE Security and Privacy on the blockchain (IEEE S&B)*, 2017.

[35] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3):219–246, 2005.

[36] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup, and D. Williams. Internet computer consensus. *Cryptology ePrint Archive*, 2021.

[37] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 107–122. Springer, 1999.

[38] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*, 13(1):143–202, 2000.

[39] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[40] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In *Annual International Cryptology Conference*, pages 98–116. Springer, 1999.

[41] I. Cascudo and B. David. Scrape: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017.

[42] I. Cascudo and B. David. Albatross: publicly attestable batched randomness based on secret sharing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 311–341. Springer, 2020.

[43] I. Cascudo, B. David, O. Shlomovits, and D. Varlakov. Mt. random: Multi-tiered randomness beacons. In *Applied Cryptography and Network Security: 21st International Conference, ACNS 2023, Kyoto, Japan, June 19–22, 2023, Proceedings, Part II*, pages 645–674. Springer, 2023.

[44] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[45] M. Chen, C. Hazay, Y. Ishai, Y. Kashnikov, D. Micciancio, T. Riviere, A. Shelat, M. Venkitasubramaniam, and R. Wang. Diogenes: lightweight scalable rsa modulus generation with a dishonest majority. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 590–607. IEEE, 2021.

[46] A. Cherniaeva, I. Shirobokov, and O. Shlomovits. Homomorphic encryption random beacon. *Cryptology ePrint Archive*, 2019.

[47] K. Choi, A. Arun, N. Tyagi, and J. Bonneau. Bicorn: An optimistically efficient distributed randomness beacon. *Cryptology ePrint Archive*, 2023.

[48] K. Choi, A. Manoj, and J. Bonneau. Sok: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 75–92. IEEE Computer Society, 2023.

[49] B. Chor, O. Goldreich, J. Hasted, J. Freidmann, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 396–407. IEEE, 1985.

[50] J. Clark and U. Hengartner. On the use of financial data as a random beacon. In *2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 10)*, 2010.

[51] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369, 1986.

[52] R. Cohen and Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, 2017.

[53] S. Das, V. Krishnan, I. M. Isaac, and L. Ren. Spurt: Scalable distributed randomness beacon with transparent setup. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2502–2517. IEEE, 2022.

[54] S. Das, T. Yurek, Z. Xiang, A. Miller, L. Kokoris-Kogias, and L. Ren. Practical asynchronous distributed key generation. *Cryptology ePrint Archive*, 2021.

[55] B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.

[56] L. De Feo, S. Masson, C. Petit, and A. Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25*, pages 248–277. Springer, 2019.

[57] D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985.

[58] J. Drake. Minimal VDF randomness beacon - Sharding, 2018. `https://ethresear.ch/t/minimal-vdf-randomness-beacon/3566`.

[59] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[60] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass. Continuous verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–154. Springer, 2020.

[61] T. Fagerström. Lotteries in communities of sessile organisms. *Trends in Ecology & Evolution*, 3(11):303–306, 1988.

[62] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[63] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 33–62. Springer, 2018.

[64] N. Gailly, K. Melissaris, and Y. Romailler. tlock: practical timelock encryption from threshold bls. *Cryptology ePrint Archive*, 2023.

[65] S. M. Gainsbury and A. Blaszczynski. How blockchain and cryptocurrency technology could revolutionize online gambling. *Gaming Law Review*, 21(7):482–492, 2017.

[66] D. Galindo, J. Liu, M. Ordean, and J.-M. Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 88–102. IEEE, 2021.

[67] P. Gałka and A. Strzelecki. How randomness affects player ability to predict the chance to win at playerunknown's battlegrounds (pubg). *The Computer Games Journal*, 10:1–18, 2021.

[68] J. Garay and A. Kiayias. Sok: A consensus taxonomy in the blockchain era. In *Cryptographers' track at the RSA conference*, pages 284–318. Springer, 2020.

[69] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 295–310. Springer, 1999.

[70] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.

[71] C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov. Yoso: You only speak once: Secure mpc with stateless ephemeral roles. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II*, pages 64–93. Springer, 2021.

[72] C. Gentry, S. Halevi, and V. Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part I*, pages 458–487. Springer, 2022.

[73] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.

[74] O. Goldreich. *Foundations of Cryptography, Volume 2*. Cambridge university press Cambridge, 2004.

[75] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[76] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 2019.

[77] J. Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.

[78] K. Gurkan, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu. Aggregatable distributed key generation. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*, pages 147–176. Springer, 2021.

[79] R. Halprin and M. Naor. Games for extracting randomness. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–12, 2009.

[80] R. Han, J. Yu, and H. Lin. Randchain: Decentralised randomness beacon from sequential proof-of-work. *IACR Cryptol. ePrint Arch.*, 2020:1033, 2020.

[81] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

[82] S. Heidarvand and J. L. Villar. Public verifiability from pairings in secret sharing schemes. In *International Workshop on Selected Areas in Cryptography*, pages 294–308. Springer, 2008.

[83] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 100–110, 1997.

[84] G. Kadianakis. Whisk: A practical shuffle-based ssle protocol for ethereum. *Ethereum Research.(Jan. 13, 2022). Retrieved Sept*, 5:2022, 2022.

[85] A. Kate and I. Goldberg. Distributed key generation for the internet. In *2009 29th IEEE International Conference on Distributed Computing Systems*, pages 119–128. IEEE, 2009.

[86] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, pages 177–194. Springer, 2010.

[87] J. Kelsey, L. T. Brandão, R. Peralta, and H. Booth. A reference for randomness beacons: Format and protocol version 2. Technical report, National Institute of Standards and Technology, 2019.

[88] A. Kiayias, C. Moore, S. Quader, and A. Russell. Efficient random beacons with adaptive security for ungrindable blockchains. *Cryptology ePrint Archive*, 2021.

[89] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[90] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[91] E. Kokoris Kogias, D. Malkhi, and A. Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1767, 2020.

[92] L. LAMPORT, R. SHOSTAK, and M. PEASE. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[93] H. Lee, Y. Hsu, J.-J. Wang, H. C. Yang, Y.-H. Chen, Y.-C. Hu, and H.-C. Hsiao. Headstart: Efficiently verifiable and low-latency participatory randomness generation at scale. In *Network and Distributed System Security (NDSS) Symposium*, volume 2022, 2022.

[94] A. K. Lenstra and B. Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017.

[95] D.-Y. Liao and X. Wang. Design of a blockchain-based lottery system for smart cities applications. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 275–282. IEEE, 2017.

[96] Y. Lindell. Secure multiparty computation (mpc). *Cryptology ePrint Archive*, 2020.

[97] S. K. D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song. Churp: dynamic-committee proactive secret sharing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2369–2386, 2019.

[98] R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

[99] L. Medley, A. F. Loe, and E. A. Quaglia. Sok: Delay-based cryptography. *Cryptology ePrint Archive*, 2023.

[100] L. Medley and E. A. Quaglia. Collaborative verifiable delay functions. In *Information Security and Cryptology: 17th International Conference, Inscrypt 2021, Virtual Event, August 12–14, 2021, Revised Selected Papers 17*, pages 507–530. Springer, 2021.

[101] M.Haahr. Random.org: True random number service, 2010.

[102] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.

[103] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[104] L. Nguyen. Accumulators from bilinear pairings and applications. In *Cryptographers' track at the RSA conference*, pages 275–292. Springer, 2005.

[105] J. B. Nielsen, J. Ribeiro, and M. Obremski. Public randomness extraction with ephemeral roles and worst-case corruptions. In *Annual International Cryptology Conference*, pages 127–147. Springer, 2022.

[106] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. *Cryptology ePrint Archive*, 2016.

[107] R. Pass and E. Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*, pages 3–33. Springer, 2018.

[108] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[109] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*, pages 129–140. Springer, 2001.

[110] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*, pages 554–571. Springer, 2008.

[111] C. Pierrot and B. Wesolowski. Malleability of the blockchain's entropy. *Cryptography and Communications*, 10(1):211–233, 2018.

[112] K. Pietrzak. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[113] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85, 1989.

[114] M. Raikwar and D. Gligoroski. Sok: Decentralized randomness beacon protocols. In *Information Security and Privacy: 27th Australasian Conference, ACISP 2022, Wollongong, NSW, Australia, November 28–30, 2022, Proceedings*, pages 420–446. Springer, 2022.

[115] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[116] L. Ren. Analysis of nakamoto consensus. *Cryptology ePrint Archive*, 2019.

[117] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. 1996.

[118] K. Sahitya and B. Borisaniya. D-lotto: the lottery dapp with verifiable randomness. In *Data Science and Intelligent Applications: Proceedings of ICDSIA 2020*, pages 33–41. Springer, 2021.

[119] P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, and E. Weippl. Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. 2021.

[120] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl. Hydrand: Practical continuous distributed randomness. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy*, 2020.

[121] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[122] 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.

[123] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[124] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[125] V. Shoup. Practical threshold signatures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 207–220. Springer, 2000.

[126] N. Shrestha, I. Abraham, L. Ren, and K. Nayak. On the optimality of optimistic responsiveness. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 839–857, 2020.

[127] D. R. Stinson and R. Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *Australasian Conference on Information Security and Privacy*, pages 417–434. Springer, 2001.

[128] E. Syta, P. Jovanovic, E. Kokoris-Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable bias-resistant distributed randomness. In *38th IEEE Symposium on Security and Privacy*, May 2017.

[129] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping authorities" honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 526–545. Ieee, 2016.

[130] G. Wang, Z. J. Shi, M. Nixon, and S. Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[131] B. Wesolowski. Efficient verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 379–407. Springer, 2019.

[132] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.

[133] M. Zamani, M. Movahedi, and M. Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 931–948, 2018.