Efficient Random Beacons with Adaptive Security for Ungrindable Blockchains

Aggelos Kiayias[1,2], Cristopher Moore[3], Saad Quader[4], and Alexander Russell[4,2]

[1]Univeristy of Edinburgh
[2]IOHK
[3]Santa Fe Institute
[4]University of Connecticut

June 2020

Abstract

We describe and analyze a simple protocol for n parties that implements a *randomness beacon*: a sequence of high entropy values, continuously emitted at regular intervals, with sub-linear communication per value. The algorithm can tolerate a $(1-\epsilon)/2$ fraction of the n players to be controlled by an *adaptive* adversary that may deviate arbitrarily from the protocol. The randomness mechanism relies on verifiable random functions (VRF), modeled as random functions, and effectively stretches an initial λ-bit seed to an arbitrarily long public sequence so that (i) with overwhelming probability in k—the security parameter—each beacon value has high min-entropy conditioned on the full history of the algorithm, and (ii) the total work and communication required per value is $O(k)$ cryptographic operations.

The protocol can be directly applied to provide a qualitative improvement in the security of several proof-of-stake blockchain algorithms, rendering them safe from "grinding" attacks.

1 Introduction

We revisit the well-studied topic of randomness generation in a distributed setting of n players with an adversary who controls a minority of the players. Our particular goal is motivated by a flurry of recent work on generating random beacons for proof-of-stake blockchains; these are distributed protocols that critically rely on the ability of a committee of participants to generate history-independent clean randomness. More broadly, we consider the question of generating randomness for a long-lived distributed protocol that proceeds in *epochs*: Each epoch of the protocol requires clean randomness, is responsible for carrying out a distributed computation of interest, and must furthermore generate the randomness to carry out the next epoch. We study the particular demands of randomness generation in this setting, develop and analyze an efficient protocol for the problem, and indicate how the protocol can be used to provide a striking qualitative improvement in the security provided by proof-of-stake blockchains such as Snow White [5], Ouroboros Praos [10], and Tezos [15].

The simplest formulation that arises in our setting is described by n players $P_1, \ldots, P_n$, each of which is assigned an independent and uniformly random string r_i. A random subset of players is selected, and *honest* players simply shout out their strings for all to hear, while *adversarial* players—after examining the values of the honest players—may selectively choose whether or not to broadcast their string. A function F is then applied to the concatenation; what can be said about the randomness content of the result? This question immediately calls to mind the classical perfect information model of Ben-Or and Linial [3]: the adversarial coalition may delay its action until the honest players have submitted their strings, and the resulting randomness is determined by an application of a fixed function.

Our objective of high entropy and sublinear communication make such classical techniques unsuitable. First, given the sublinear complexity requirement, the number of adversarial parties exceeds the number of parties who may contribute to the generation of a particular beacon value; one then has to guard against the possibility that the adversary can induce a protocol execution for which a certain beacon value is completely determined by adversarial parties, violating its unpredictability. To go a step further, in light of adaptive corruptions it must be difficult to even *predict* if a party will materially contribute to the protocol. To circumvent this problem, we will allow a common random string to be used as a protocol setup. Even in such a case, however, a significant obstacle remains; if any party is active in more than a single stage of the protocol, the adaptive adversary can corrupt that party. For this reason single-round protocols will play a special role in the construction and analysis: in such protocols, given that each player speaks only once, the power of adaptive corruption can be mitigated.

The model and the figure of merit. To be more precise about the problem, we begin with n players $P_1, \ldots, P_n$ in a public-key setting where we have the luxury of pre-established public/private key pairs for the players. The protocol is provided a common random seed $s \in \{0,1\}^\lambda$, where λ will act as a security parameter of the system, and must generate a common result $s' \in \{0,1\}^\lambda$. The goal is to ensure that s' is sufficiently random, conditioned on the seed s, even in the face an adaptive adversary who may adaptively corrupt a coalition of $[(1-\epsilon)/2]n$ of the players for constant $\epsilon \in (0,1)$. While the players are provided a synchronous broadcast network, the adversary is *rushing*: it may observe the broadcast values of the honest players prior to determining the behavior of adversarial parties in the same broadcast round.

Our goal is a resulting random value s' that is sufficiently close to uniform to support a generic distributed process $\mathbf{\Pi}()$ that runs at regular intervals. Specifically, consider a process $\mathbf{\Pi}(r)$ that requires a random value $r \in \{0,1\}^\lambda$ for operation and has failure probability ε_p; we wish to iterate the process T times. Of course, ideal iteration would carry out the sequence $\mathbf{\Pi}(r_1), \ldots, \mathbf{\Pi}(r_T)$, where each r_i is an independently uniform string provided by a perfect external randomness "beacon"; this would yield error probability $\approx \varepsilon_p T$ (by the union bound). Suppose now that we have a randomness generation mechanism $B : \{0,1\}^\lambda \rightsquigarrow \{0,1\}^\lambda$ with the property that for uniformly chosen $s \in \{0,1\}^\lambda$ and a parameter $k > 0$, the output $s' = B(s)$ satisfies

$$\Pr_s[H_\infty(s' \mid s) \leq \lambda - k] \leq 2^{-Ck} \tag{1}$$

for a constant $C > 1$, even if a fraction $(1-\epsilon)/2$ of the players deviate arbitrarily and adaptively. Consider then the sequence of seeds $s_1, \ldots, s_T$, where s_1 is uniform and each subsequent $s_i = B(s_{i-1})$. Recalling that the probability of an event $E \subset \{0,1\}^\lambda$ can only be inflated by a factor 2^k when evaluated with respect to a random variable with min-entropy $\lambda - k$ (vis-a-vis its probability with respect to the uniform distribution) it follows immediately that, except with probability $T2^k 2^{-Ck} = T2^{-(C-1)k}$, each one of these seeds has min-entropy $\lambda - k$ (even when conditioned on the prior history). Furthermore, by the same reasoning, the error probability arising from using this sequence of T random seeds to iterate the process $\mathbf{\Pi}(\cdot)$ T times is no more than

$$\left(2^{-(C-1)k} + 2^k \varepsilon_p\right) T\,.$$

Observe the critical role played by the constant C: If $C \leq 1$ the quality bound of (1) is insufficient to guarantee strong iteration. If $C > 1$, by appropriately tuning k one achieves resulting error $\approx T\varepsilon_p^{1-1/C}$. If $C = 2$, for example, one can achieve error $O(T\sqrt{\varepsilon_p})$. This somewhat informal picture is made precise in Section 3.

The beacon protocol, B. Our principal contribution is a simple protocol **B** that achieves $C > 1$. The protocol is *one-round*, calling for each player to generate and broadcast a "nonce" $w_i \in \{0,1\}^\lambda$; as above, honest players broadcast their nonces while the adversarial players, acting in concert, may now each broadcast a nonce that depends on the honest observed values or choose not to broadcast a nonce at all. Finally, a function $F(s; w_1, \ldots, w_n) \in \{0,1\}^\lambda$ is applied to the broadcast nonces with the convention that $w_i = \perp$ if P_i did not broadcast. We wish to control $H_\infty(F(s; w_1, \ldots, w_n) \mid s)$, the min-entropy conditioned on the seed.

The protocol **B** depends on two cryptographic objects: a cryptographic hash function $\mathsf{H}(\cdot)$ and a family of verifiable random functions (Definition 1). Specifically, each player P_i is associated with a public/private key

pair for a *verifiable random function* $F_i : \{0,1\}^\lambda \to \{0,1\}^\lambda$. Such a function appears random to all parties, can only be evaluated by the private key holder (P_i), but admits a publicly verifiable proof of evaluation: that is, after evaluating the function to find that $F_i : x \mapsto y$, P_i can generate a publicly verifiable proof of this fact. As indicated above, the protocol is determined by a parameter k, and proceeds as follows:

1. With the seed s, each player evaluates $F_i(s) = w_i$. These values, w_i, along with proofs that $F_i(s) = w_i$ are broadcast to all players.

2. All values w_i received (with correct proofs) are sorted lexicographically. The first k values are hashed together to produce the new seed s'.

As indicated above, we assume a broadcast channel but permit the adversary to select the contributions of his players after observing the w_i of the honest players.

Reducing broadcasts. The algorithm admits a simple optimization that yields a significant improvement in message efficiency; we call this *the optimized version of* **B**. As written, the protocol **B** calls for n broadcast messages, one per party. However, as only those w_i appearing among the first k positions in the lexicographically sorted list of w_i will contribute to the final value, player P_i can safely remain silent if $w_i \geq \ell 2^\lambda / n$, for an appropriately chosen threshold $\ell > k$. More precisely, the probability that the first k elements of the sorted sequence are not all less than $\ell 2^\lambda / n$ is $\exp(-\Omega(\ell) + O(k))$. As we will take k to scale with the security parameter, this yields a protocol with message complexity that scales with the security parameter rather than n.

One effect of this application of verifiable random functions is to significantly limit the behavior of the adversarial players: essentially, each adversarial player may only choose whether or not to submit his w_i. It follows immediately that the adversary has no more than $2^{(1-\epsilon)n/2}$ choices for the resulting seed s' and thus $H_\infty(s) \geq \lambda - (1-\epsilon)n$. Of course, this bound is far too weak for our purposes; for one thing, it scales with n.

Remarks on the analysis. The analysis of the protocol establishes an inequality of the form (1) where k is precisely the number of "lowest" nonces used by the algorithm. The analysis provides this guarantee even in the face of an adaptive adversary controlling $[(1-\epsilon)/2] \cdot n$ of the parties.

Note that while the algorithm itself critically uses cryptographic objects, we treat these as idealized black boxes in the analysis; in particular, both the hash function and the verifiable random functions are modeled as random functions. Thus the analysis is essentially a probabilistic affair, analyzing the moments of a particular discrete process and the order statistics of the related Erlangian distribution.

The blockchain application; forestalling grinding attacks. Proof-of-stake blockchains have precisely the structure described above, and it is in this context that such beacon generation algorithms have been most closely studied. Blockchains such as Ouroboros Praos, Snow White, and Tezos all use randomness beacons on which *grinding attacks* can be launched. This means that the resulting beacon value R is subject to a resampling attack by the adversary: specifically, while R is drawn from a high entropy distribution, the beacon generation protocol permits the adversary to redraw R as many times as he pleases before settling on one he likes. Thus the security guarantees of these protocols have a "grinding" term that bounds above the number such adversarial resamplings: the most convenient way to do this is to define $R = \mathsf{H}(S)$, for a cryptographically strong hash function H queried at an input S that the adversary partially controls, and bound from above the number of adversarial evaluations of the hash function. This has a rather unpleasant effect: while these are proof-of-stake blockchains, one still requires a bound on the adversarial hashing power to establish security. The analysis in [22] overcomes this difficulty by giving an exact analysis of the grinding term in the "hash the blockchain" beacons used in Ouroboros Praos [10] and Snow White [5] in the synchronous setting; this eliminates the need to rely on the hashing power of the adversary. However, this analysis requires quite strong bounds on adversarial stake ratio—rough 9.5%—to give acceptable guarantees.

Our simple beacon protocol can be directly incorporated into these blockchain protocols by the standard device: the broadcast channel is implemented using blockchain consensus mechanism itself. (See, e.g., the general notion of "input endorsers" in Ouroboros [18].) This straightforward transformation, while preserving the other properties of the blockchains, removes the grinding term from their security analysis. Indeed, in the parlance above, one can select k as a function of the security parameter and achieve an explicit polynomial effect on the

insecurity of the protocol. For deployed protocols (e.g., Tezos [15] and Ouroboros Praos [10]), this is especially favorable because the fundamental parameters of the protocol (i.e., epoch length, settlement time, etc.) can be determined merely as a function of the stake of the adversary. Note that our beacon allows these protocols to tolerate an adversary who holds less than 50% stake. In contrast, the existing "hash the blockchain" beacons in Praos and Snow White become insecure if the adversary holds more than 9.5% stake [22].

Prior to a detailed treatment of the related work, it is worth emphasizing that a major distinction between our effort and previous work is our insistence that the protocol is efficient and secure against an adaptive adversary. Indeed, protocols based on publicly-verifiable secret sharing schemes (e.g., Ouroboros [18]) can achieve perfect randomness with two rounds. The emphasis on one-round solutions arises because of adaptive security and the fact that the length of the randomness generation algorithm is an important design parameter for these proof-of-stake blockchains. (It determines the recency of the distribution of stake that is used to define the leader election distribution and, as a result, the bound on the rate at which stake can move in the system.) It is worth mentioning that these systems implement a broadcast channel with a particular application of their low-level consensus mechanism, and this means that a single round of broadcast may take significant time to settle—days in some cases. Thus the distinction between a one-round protocol and even a two-round protocol has additional practical significance. We remark, additionally, that our one-round protocols are extremely simple.

Our contribution. Recall that our objective is an n-player beacon protocol with the following properties: (*i*) it is an iterated single-round coin-flipping protocol; (*ii*) it is secure against an adaptive adversary who may control less than half the players (or, equivalently, in the proof-of-stake setting, control less than a 50% stake); (*iii*) it is allowed to use a broadcast channel and cryptography; (*iv*) the setup and communication complexity is sublinear in n; (*v*) for some parameter k (which is independent of n), the loss in the output min-entropy is $O(k)$ except with probability $e^{-\Omega(k)}$.

The optimized version of our beacon protocol **B** has all these properties. To our best knowledge, no other beacon does the same:

1. Algorand [9]—one of the main inspirations for this paper—provides single-round coin-flipping but guarantees high min-entropy with only a constant probability. Our approach can be viewed as a kind of parallelization of the Algorand technique; see below.

2. The coin-flipping in Ouroboros and RandHound [25] are multi-round and insecure against an adaptive adversary who controls a minority stake or a minority coalition, respectively. DFINITY [16] uses one-round coin-flipping but, like Ouroboros, it is secure only against a static adversary.

We present a more comprehensive literature survey in Section 1.1.

1.1 Related work

Below, we survey the coin-flipping and beacon literature in more detail.

Collective coin-flipping as a Boolean function. The (single round) collective coin-flipping in the full information model is a classic problem introduced in the seminal work of Ben-Or and Linial [3]. In this problem, n-players communicate over a single broadcast channel. First, each honest player submits a uniformly random bit, and then each adversarial player (strategically) submits an arbitrary bit. The output of the protocol is a single bit, computed as a Boolean function F applied to the submitted bits. If F is the majority function, an adversarial coalition of size $O(\sqrt{n})$ can arbitrarily influence the output of the protocol [3]. Later, Kahn, Kalai, and Linial [17] showed that for *every* Boolean function F in this setting, there is an adversarial coalition of size $O(n/(\log n))$ that can bias the output. It was conjectured by Friedgut [12] that Boolean functions on the continuous cube $[0,1]^n$ (or equivalently, the discrete cube $[n]^n$) can be biased by a coalition of size $o(n)$. See [11] and [26] for advances on this line of research. As for multi-round coin-flipping protocols, Russell, Saks, and Zuckerman [23] showed that any $o(\log^* n)$-round coin-flipping protocol can be biased by a coalition of size $o(n)$.

Cryptography is a powerful tool for collective coin-flipping. While the most directly relevant related work is AlgoRand—which uses a mechanism similar to ours—we survey some other work first to set the stage.

PVSS-based beacons. Publicly verifiable secret sharing (PVSS) techniques have been successfully used to design unbiasable, multi-round coin-flipping protocols, such as Ouroboros [18], RandHound [25], Scrape [8], and HydRand [24]. The coin-flipping in these protocols have two (logical) rounds: first, a round where the players broadcast commitments to their inputs, followed by a round where these commitments are revealed. An advantage of these protocols is that they can provide extremely strong guarantees on the quality of the randomness—when the protocols are secure, they produce output values that are indistinguishable from uniform. A difficulty is that they demand $O(n^2)$ messages to be broadcast, which is intractable in practice. A standard technique to mitigate this is to first elect a small committee of players which then carry out the full PVSS. Of course, such a procedure is immediately subject to attacks by an adaptive adversary who can corrupt the committee once it is determined.

VDF-based beacons. Verifiable Delay Functions (VDF) [7, 27, 21] may be used to construct unpredictable, multi-round coin-flipping protocols (for a sketch of such a possible construction see [1]). Using VDFs in this context is orthogonal to our techniques: a VDF can delay the revelation of the beacon outcome to temper the opportunities the adversary has to grind the beacon output. In general, tuning the hardness parameter of the VDF to a high level aids this security objective but naturally interferes with the availability (or liveness) of the beacon. For this reason, a fine-grained analysis of the randomness properties of the beacon is a precondition to the effective use of a VDF in a practical beacon algorithm. Moreover, employing a VDF will require a less cryptographic assumption (moderate hardness). We leave the combination of VDF techniques with our randomness generation algorithm as an interesting future research direction.

Threshold signatures. DFINITY [16] uses a one-round coin-flipping protocol based on non-interactive (t, n)-Threshold BLS signatures [6]. In the setup phase, a random subset of n players—a "group"—is selected from a universe of N players and key-pairs for the group members are established. After the setup, each player broadcasts his share and any player can recover the (unique) beacon output from any t shares. The main disadvantage with respect to our techniques is of course the complexity of the group setup which requires $\Omega(n^2)$ communication for a single distributed key generation process between n players.

Algorand. Algorand [9] uses a Byzantine Agreement-based multi-round coin-flipping protocol where the players are equipped with verifiable random functions (VRF). While the multi-round beacon output is uniform except with a negligible probability, the beacon output of a single-round is uniform with only constant probability—thus conditional min entropy is small. As mentioned above, our protocol can be viewed as a parallelization of the AlgoRand technique; indeed, a single-round of AlgoRand coin-flipping is equivalent to setting $k = 1$ in our protocol. In contrast, our one-round protocol guarantees a "small" loss in min-entropy except with a negligible probability.

VRF-based eventual consensus Proof-of-Stake blockchains. Many proof-of-stake blockchain protocols, such as Ouroboros Praos [10], Genesis [2], and Snow White [5], use VRFs in their coin-flipping protocols. (VRFs are formally defined in Definition 1.) This is a multi-round protocol where each "block" contains a nonce. The coin-flipping output is a cryptographic hash of the XOR of the nonces recorded in the "common prefix" of all blockchains held by the honest players at the end of the epoch. (Thus, these protocols do not use a broadcast channel.) We can call these beacon protocols "hash the blockchain" beacons. It was shown in the Ph.D. thesis [22] that the min-entropy loss for this beacon in a single epoch (in the synchronous communication setting) grows linearly in the security parameter if the adversarial stake is 9.5% or higher.

Nakamoto-style PoW beacons. Bentov et al. [4] establish a lower bound for a class of Nakamoto-style proof-of-work based beacons and discuss approaches for a game-theoretic analysis of beacon protocols.

1.2 Paper outline

We present our model and the beacon protocol in Section 2 and present the analysis in Section 3. The impact of our results in the eventual consensus domain is discussed in Section 4.

2 Model and definitions; the protocol

As described in the introduction, we adopt a distributed model of computation with n players $P = \{P_1, \ldots, P_n\}$. Communication is synchronous, providing reliable broadcast. The model further provides cryptographic setup assumptions, assuming public/private key pairs for any cryptographic primitives of interest and an initial random string. Despite these modeling simplifications, the protocol we develop has direct applications to the blockchain protocols discussed in the introduction as they can provide reliable broadcast as a side effect of eventual consensus; see Section 4.

We adopt an adaptive rushing adversary, capable of corrupting a coalition of $[(1-\epsilon)/2] \cdot n$ players where $\epsilon \in (0,1)$. Broadcast is tipped in the adversary's favor: During any broadcast round, adversarial parties may wait to hear from all honest parties before deciding on the values they contribute to the (otherwise synchronous) broadcast round. We do not permit the adversary to suppress messages sent by honest parties, something in line with the modeling of information propagation in blockchain protocols, see e.g., [14, 13].

While we make use of two cryptographic primitives, we treat them both as idealized random oracles; in particular, the analysis and discussion are in an information-theoretic style. We use λ to denote the security parameter of the protocol: our goal throughout is insecurity of the form $\exp(-\Theta(\lambda))$.

- The protocol relies on a cryptographically secure hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$; this we simply model as a uniformly random function.
- The protocol relies on a family of *verifiable random functions*, defined below.

Definition 1 (Verifiable Random Function). *A family $\mathcal{F}$ of functions $F : \{0,1\}^\ell \to \{0,\}^k$ is a family of VRFs if there exist algorithms* $(\mathsf{gen}, \mathsf{prove}, \mathsf{verify})$ *so that the following holds:* $\mathsf{gen}(1^k)$ *outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*;* $\mathsf{prove}_{\mathsf{sk}}(x)$ *outputs a pair* $(F_{\mathsf{sk}}(x), \pi_{\mathsf{sk}}(x))$ *where $F_{\mathsf{sk}} \in \mathcal{F}$, $F_{\mathsf{sk}}(x)$ is the function value, and $\pi_{\mathsf{sk}}(x)$ is the proof of correctness; and* $\mathsf{verify}_{\mathsf{pk}}(x, y, \pi_{\mathsf{sk}}(x))$ *efficiently verifies that $y = F_{\mathsf{sk}}(x)$ using the proof $\pi_{\mathsf{sk}}(x)$, outputting 1 if y is valid and 0 otherwise. Additionally, we require the following properties:*

1. Uniqueness: *No values* $(\mathsf{pk}, x, y, y', \pi, \pi')$ *can satisfy both* $\mathsf{verify}_{\mathsf{pk}}(x, y, \pi) = 1$ *and* $\mathsf{verify}_{\mathsf{pk}}(x, y', \pi') = 1$ *unless $y = y'$.*
2. Provability: *If* $(y, \pi) = \mathsf{prove}_{\mathsf{sk}}(x)$ *then* $\mathsf{verify}_{\mathsf{pk}}(x, y, \pi) = 1$.
3. Pseudorandomness: *To all probabilistic polynomial-time (PPT) algorithm which runs* $\mathsf{poly}(k)$ *steps when its first input is 1^k and does not query the* prove *oracle on x, the distribution of $F_{\mathsf{sk}}(x)$ appears uniform in $\{0,1\}^k$, except with a probability negligible in k.*

For the purposes of our exposition, an ideal functionality of the form below is sufficient: with each player is associated a uniformly random function $F_i : \{0,1\}^\lambda \to \{0,1\}^\lambda$. While the function is uniquely determined (by the player's public key), it can only be evaluated by the player P_i (with her private key). P_i can further produce, after evaluating the function to discover that $F_i : x \mapsto y$, a proof $\pi_i(x)$ of this fact, which can be reliably checked by the other players.

The protocol. Let $\epsilon \in (0,1)$. As our basic protocol is a one-round affair we can simplify our presentation as follows.

THE RANDOMNESS GENERATION PROTOCOL $\mathbf{B}_k(s)$

1. All parties are presented a uniformly random string $s \in \{0,1\}^\lambda$.

2. Based on s, the adversary may corrupt a subset $A \subset P$ of the players of size $[(1-\epsilon)/2] \cdot n$.

3. Honest players announce their nonce values $F_i(s)$ (along with a proof of evaluation).

4. Adversarial players may choose whether to announce their nonce values $F_i(s)$ (along with a proof of evaluation).

5. The result of the protocol can be privately determined as $\mathsf{H}(\eta^{(1)}\| \cdots \|\eta^{(k)})$ where $\eta^{(1)}, \ldots, \eta^{(k)}$ are the first k nonce values in lexicographic order.

As remarked in the introduction, for efficiency considerations, we also consider the *optimized* version of the protocol, where P_i only broadcasts its nonce η_i if $\eta_i \leq \ell 2^\lambda / n$ for a parameter $\ell > 2k$. By a Chernoff bound, except with probability $\exp(-\Omega(\ell))$ this has the effect of reducing the number of broadcast messages to $O(\ell)$ without affecting the result (as the lowest k nonces will appear below this threshold). In our regime of interest, one can take $\ell = ck$ for a constant $c > 2$.

Observe that as a one-round protocol, the adversary's single opportunity to corrupt players appears after s is announced, at which point they are indistinguishable (as we assume ideal VRFs, public keys leak no information about the F_i). In an iterative setting, the situation is rather more complicated because an adversary—with the view of a collection of corrupt parties—has additional information that may be useful for the selection of the input string s. We discuss this in detail in the next section. In any case, as it will follow that the s_i selected for each instance of the protocol in the iterative setting are distinct with overwhelming probability, the values $F_i(s) = \eta_i$ of the honest parties may be treated as independent uniform values in any instance.

3 The analysis

Notation. The *min entropy* of a random variable X, denoted $H_\infty(X)$, is $-\log \max_v \Pr[X = v]$; throughout we let log denote the base two logarithm. When X is conditioned on an event (such as the random variable Y taking the value y), we write $H_\infty(X \mid Y = y)$ or, simply, $H_\infty(X \mid y)$ when Y can be inferred.

To approach the analysis, consider the n players P partitioned into two sets: H, the honest players, and A, the adversarial players. We assume that after adversarial corruption $|A| < (1-\epsilon)n/2$ for a constant $\epsilon \in (0,1)$ we shall fix throughout the analysis.

Our goal is to show that with high probability in $s \in \{0,1\}^\lambda$, the min-entropy of the resulting hash value is close to λ, as in (1). We will actually show something slightly stronger: with high probability the resulting distribution is the result of adversarial choice among some g different uniform values—such a distribution has min-entropy $\lambda - \log g$. We precisely define this notion:

Definition 2 (g-optativity). *Let X be a random variable taking values in $\{0,1\}^\lambda$. We say that $C : (\{0,1\}^\lambda)^g \to \{0,1\}^\lambda$ is a* choice function *if $C(u_1, \ldots, u_g) \in \{u_1, \ldots, u_g\}$ for any tuple $(u_1, \ldots, u_g)$. We say that X is* g-optative *if there is a choice function $C : (\{0,1\}^\lambda)^g \to \{1, \ldots, g\}$ so that*

$$X = C(U_1, \ldots, U_g),$$

where the U_i are uniform, independent random variables in $\{0,1\}^\lambda$. Thus a random variable is g-optative if it can be expressed as the result of an "adversary" selecting from a population of g independent and uniformly random values.

Observe that if $X = C(U_1, \ldots, U_g)$ then for any $x \in \{0,1\}^\lambda$,

$$\Pr[X = x] \leq \Pr[x \in \{U_1, \ldots, U_g\}] \leq g/2^\lambda$$

by the union bound. Thus $H_\infty(X) \geq \lambda - \log(g)$. To conclude:

Fact 3. *Let X be a g-optative random variable taking values in $\{0,1\}^\lambda$. Then $H_\infty(X) \geq \lambda - \log(g)$.*

Our goal is the following theorem, which describes the behavior of a single instance of the beacon algorithm.

Theorem 4. *For any $\epsilon > 0$, there are constants $a > 1$ and $c > 1$ so that, for sufficiently large k and n, if s is an a^k-optative random variable in $\{0,1\}^\lambda$ and $s' = \mathbf{B}(s)$ denotes the result of the beacon generation algorithm with seed s and an adaptive adversary capable of corrupting $(1-\epsilon)n/2$ of the players, then s' is also a^k-optative, except with probability c^{-k}.*

A survey of the proof. Let $\eta : P \to \{0,1\}^\lambda$ be the function $\eta(P_i) = \eta_i$ that collects together the nonces; let $\eta_A : A \to \{0,1\}^\lambda$ be the restriction to the adversarial parties and let η_H the restriction to the honest parties. We will use the notation $\eta^{(1)}, \eta^{(2)}, \ldots$ to denote the nonces of η in sorted order and use the same convention for η_A and η_H.

Consider the view of the adversary after Step 2 of the protocol, so that he observes s and η_A. We are interested in the quantity $s' = \mathbf{B}_k(s)$ and study this by identifying two cases of interest. If it should happen that the k smallest nonces among (the range of) η_A are in fact the k smallest nonces among all the η_i, we say that a *catastrophe* has occurred—conditioned on this event (and the values taken by s and η_A), the random variable s' has zero min-entropy (as we will only assume that H is fresh on values that contain an honestly generated nonce). Otherwise, the adversary is faced with the full sorted list of all η_i, and must choose which of his minions will contribute their own nonces. Here an interesting combinatorial quantity arises: *the number of distinct subsets of nonces which the adversary can induce by silencing an arbitrary subset of his players and forming the subset of the smallest k remaining nonces.* Considering that a random function is applied to the resulting concatenation of nonces, each such subset corresponds to a uniformly random element of $\{0,1\}^\lambda$. The adversary may choose freely among these, which results in a g-optative random variable (where g is the number of such subsets). The proof concludes with a (second moment) tail bound on g which is strong enough to achieve the desired properties discussed in the introduction.

In the next two sections we carry out these analyses and then conclude with the main security statement for the beacon mechanism $\mathbf{B}$.

3.1 The catastrophe

Continuing with the notation above, we let $\eta_A^{(k)}$ denote the kth smallest adversarial nonce and $\min(\eta_H)$ the smallest honest nonce. Then the catastrophe is that $\eta_A^{(k)} < \min(\eta_H)$.

Theorem 5. *Suppose that a fraction $p = (1-\epsilon)/2 < 1/2$ of the players are adversarial. Then*

$$\Pr_\eta[\eta_A^{(k)} \le \min(\eta_H)] = 2(\mathrm{e}^{-1/\mathrm{e}} + o(1))^k\,.$$

Moreover, for any constant $a > 1$, there are constants $\rho_a > 0$ and $c > 1$ so that

$$\Pr_\eta[\eta_A^{(k)} \le \rho_a k 2^\lambda/n] = (a + o(1))^{-k} \qquad \textit{and} \qquad \Pr_\eta[\min(\eta_H) > \rho_a k 2^\lambda/n] = (c + o(1))^{-k}\,.$$

Proof. Let n be the total number of players, with pn adversarial players and $(1-p)n$ honest ones. It's convenient to treat the nonces as real-valued, and scale them so that they are uniformly random in the interval $[0, pn]$. The adversary's k smallest nonces are asymptotically the arrivals of a Poisson process with rate 1: that is, their kth smallest nonce $r = \eta_A^{(k)}$ is an Erlang random variable, i.e., the sum of k independent exponential random variables $x_1, \ldots, x_k$ each with mean 1. We can derive this by starting with the exact beta probability distribution for r,

$$P(r) = \frac{(pn-1)!}{(pn-k)!\,(k-1)!}\left(\frac{x}{pn}\right)^{k-1}\left(1-\frac{x}{pn}\right)^{pn-k}\,. \tag{2}$$

Using $\binom{n}{k} = (1 - O(k^2/n))n^k/k!$ and $1 - x = (1 - O(x^2))\mathrm{e}^{-x}$, if $x \le k$ this becomes

$$P(r) = \left(1 \pm O\left(\frac{k^2}{pn}\right)\right)\frac{\mathrm{e}^{-x}x^{k-1}}{(k-1)!}\,, \tag{3}$$

where the right-hand side (without the error term) is the Erlang distribution. (We note in passing that $P(r)$ is exactly Erlang if the number of adversarial players is Poisson with mean pn rather than being fixed.)

For now let us neglect the error term in (3), and derive a lower tail bound on r assuming it is Erlang. Namely, for any $0 < \rho < 1$, the probability that r is less than ρ times its expectation k is bounded by

$$\Pr[r < \rho k] \leq \left(\rho e^{1-\rho}\right)^k . \tag{4}$$

The proof is standard, but we include it here for completeness. Since the x_i are independent, the moment generating function of r is simply the kth power of the m.g.f. of the exponential distribution,

$$\mathbb{E}[e^{\lambda r}] = \prod_{i=1}^{k} \mathbb{E}[e^{-\lambda x_i}] = \left(\int_0^\infty dx\, e^{-x} e^{-\lambda x}\right)^k = \left(\frac{1}{1+\lambda}\right)^k .$$

For any $\lambda > 0$, Markov's inequality gives

$$\Pr[r < \rho k] = \Pr[e^{-\lambda r} > e^{-\lambda \rho k}] \leq \frac{\mathbb{E}[e^{-\lambda r}]}{e^{-\lambda \rho k}} = \left(\frac{e^{\lambda\rho}}{1+\lambda}\right)^k .$$

The right-hand side is minimized when $\lambda = -1 + 1/\rho$, and simplifying gives (4). We note that (4) can also be derived using the fact that the number t of adversarial nonces less than ρk is Poisson with mean ρk, and $r < \rho k$ if and only if $t \geq k$.

Since there are $(1-p)n \geq pn$ honest nonces and they are also uniform in $[0, pn]$, the probability that none of them fall below ρk is

$$\left(1 - \frac{\rho k}{pn}\right)^{(1-p)n} \leq \left(1 - \frac{\rho k}{pn}\right)^{pn} \leq e^{-\rho k} .$$

We conclude that for $0 < \rho < 1$,

$$\begin{aligned} \Pr[\eta_A^{(k)} \leq \min(\eta_H)] &\leq \min_\rho \Pr[\eta_A^{(k)} < \rho k \text{ or } \rho k \leq \min(\eta_H)] \\ &\leq \left(1 \pm O\left(\frac{k^2}{pn}\right)\right)(\min_\rho(\rho e^{1-\rho})^k + e^{-\rho k} , \end{aligned}$$

by the union bound and where we have restored the error term from (3). Taking $\rho = 1/e$ balances these terms, giving the bound $2e^{-k/e}$. Moreover, for any fixed constant $a > 1$, by choosing ρ suitably small one can ensure that $\rho e^{1-\rho} < 1/a$; this yields the second statement of the theorem. □

3.2 The grinding term

We return now to the "non-catastrophic" case. The adversary is faced with the sorted list of all the nonces $\eta^{(1)}, \eta^{(2)}, \ldots$ and may choose which of his parties will actually contribute their nonces to the protocol. Depending on this selection, the smallest k contributed nonces are hashed together to give the final result of the protocol. In this non-catastrophic case, regardless of the adversary's selection, at least one of the included nonces will be honest and the resulting hash value will be a fresh, uniform value. (Here we neglect the probability of a collision among the random values submitted to the hash function over the lifetime of the protocol.) Thus the major question of interest is the number of possible subsets the adversary can produce by this process.

To study this question, with the sorted sequence of nonces $\eta^{(1)}, \eta^{(2)}, \ldots$ we may associated a bit sequence $s_1, s_2, \ldots$ with the convention that $s_i = 0$ if $\eta^{(i)}$ is honest, and 1 otherwise. Note that the s_i are i.i.d. random variables with $\Pr[s_i = 1] = p = (1-\epsilon)/2$. Viewing the subsets that can emerge by the adversary's choice as subsets of these indices yields the following definition.

Definition 6. *Given an infinite sequence $s \in \{0,1\}^{\mathbb{N}}$, a subsequence $u = (s_{i_1}, s_{i_2}, \ldots, s_{i_k})$ with $0 \leq i_1 < i_2 < \cdots < i_k$ is* legal *if $j \in \{i_1, \ldots, i_k\}$ for all $j \leq i_k$ with $s_j = 0$. That is, it must include all the* 0*s to the left of its rightmost element.*

Theorem 7. *Let $s \in \{0,1\}^{\mathbb{N}}$ be an infinite Bernoulli sequence with density $p < 1/2$. That is, for all $i \in \mathbb{N}$ the s_i are independent, $s_i = 1$ with probability p and $s_i = 0$ with probability $1-p$. Let N_k be the number of legal subsequences of length k. Then there are constants a, b such that $a < b$ and a constant k_0 such that*

$$\Pr\big[N_k > a^k\big] < b^{-k} \tag{5}$$

for all $k > k_0$. Moreover, if $p < 0.319$ this holds with $a < 1/p$.

Proof. We will use Chebyshev's inequality, i.e., Markov's inequality applied to the second moment,

$$\Pr\big[N_k > a^k\big] < \frac{\mathbb{E}[N_k^2]}{a^{2k}}\,. \tag{6}$$

It seems tricky to compute the second moment $\mathbb{E}[N_k^2]$ directly. Instead, let $N_\ell^{(2)}$ denote the number of ordered pairs (u, v) of legal subsequences whose lengths sum to ℓ. Clearly

$$N_k^2 \le N_{2k}^{(2)}\,, \tag{7}$$

so the modified second moment $\mathbb{E}[N_{2k}^{(2)}]$ is an upper bound on $\mathbb{E}[N_k^2]$.

Now consider the generating function

$$g(x,y) = \mathop{\mathbb{E}}_{s} \sum_{\ell}^{\infty} N_\ell^{(2)} z^\ell = \sum_{\ell}^{\infty} \mathbb{E}[N_\ell^{(2)}] z^\ell\,.$$

We can calculate $g(z)$ from a fixed point equation. Clearly the distribution of s is unchanged if we shift it to the right, setting s_{i+1} to s_i, and choose s_0 from the Bernoulli distribution. If $s_0 = 0$, we can extend an existing pair (u, v) of legal subsequences by prepending s_0 to both of them, giving a factor of z^2 to the generating function; if $s_0 = 1$, we can append s_0 to either one independently, giving a factor of $(1+z)^2$; and in either case we can create a new pair (u, v) where both strings are empty, adding 1. This gives the linear recurrence

$$\begin{aligned} g(z) &:= 1 + g(x,y) \mathop{\mathbb{E}}_{s_0} \begin{cases} z^2 & s_0 = 0 \\ (1+z)^2 & s_0 = 1 \end{cases} \\ &= 1 + \big((1-p)z^2 + p(1+z)^2\big) g(z)\,, \end{aligned}$$

where we used the fact that s_0 is Bernoulli with expectation p. The fixed point of this recurrence is

$$g(z) = \frac{1}{1 - z^2 - p(1+2z)}\,.$$

which we can also derive as a geometric series.

Now, $g(z)$ is rational. Its unique positive pole, which is simple, is

$$z_0 = \sqrt{p^2 - p + 1} - p\,.$$

It follows from standard techniques in generating functions [28] that the coefficient $\mathbb{E}[N_\ell^{(2)}]$ of z^ℓ is bounded by

$$\mathbb{E}[N_\ell^{(2)}] \le C z_0^{-\ell} \tag{8}$$

for some constant C.

Combining (8) with (6) and (7), for any $a > 0$ the probability that $N_k > a^k$ is bounded by

$$\Pr\big[N_k > a^k\big] \le C(a z_0)^{-2k}\,.$$

For sufficiently large k, the right-hand side is less than b^{-k} if $b < (az_0)^2$, so we can set $a < b$ whenever $a > 1/z_0^2$. Finally, we can have $a < 1/p$ if

$$p < z_0^2 . \tag{9}$$

This holds whenever p is less than the unique real root of the cubic equation $p^3 - p^2 + p - 1/4 = 0$, which according to Cardano's formula is

$$\frac{1}{6}\left(2 - \frac{8}{\sqrt[3]{3\sqrt{57}-1}} + \sqrt[3]{3\sqrt{57}-1}\right) = 0.319448\ldots$$

This completes the proof. □

3.3 The beacon security bound

Finally, we return to the security of beacon protocol itself. We wish to show the following:

Theorem 8 (Theorem 4, restated)**.** *For any $\epsilon > 0$, there are constants $a > 1$ and $c > 1$ so that, for sufficiently large k and n, if s is an a^k-optative random variable in $\{0,1\}^\lambda$ and $s' = \mathbf{B}(s)$ denotes the result of the beacon generation algorithm with seed s and an adaptive adversary capable of corrupting $(1-\epsilon)n/2$ of the players, then s' is also a^k-optative, except with probability c^{-k}.*

Proof. The proof involves three random variables η_A, η_H, and H. For convenience, we simply assume that H takes independent, uniform values on all strings of the form $\eta_{i_1}\|\eta_{i_2}\|\ldots$, so long as at least one of these η_i was generated by an honest player. (To be completely precise here, one would have to maintain a term of the form (number of queries)$^2/2^\lambda$ to account for the possibility of a collision during the execution of the protocol, which we ignore.)

In general, in light of Theorem 7, there are constants $1 < a < b$ for which the probability that the adversary can choose between more than a^k different resulting subsets of nonces is no more than b^{-k} if s is chosen uniformly. As s is in fact is a^k-optative, the probability that there are more than a^k possible subsets is no more than $a^k \cdot b^{-k} = (a/b)^k$. Let $c_1 = b/a$.

Thus, unless a catastrophe occurs—and there is a subset of entirely adversarial nonces—the resulting distribution is a^k-optative as H is uniform on each of these nonce concatenations. It remains to ensure that the catastrophe does not occur. Note that by Theorem 5, except with probability $(b + o(1))^{-k}$ in the selection of η_A (with uniform s), a catastrophe cannot occur (in selection of the η_H) except with probability $(c_2 + o(1))^{-k}$. We say that such η_A is *safe*—that is, the conditional probability of a catastrophe in selection of η_H is small. Again using the fact that s is a^k optative, we find that with probability $a^k/(b + o(1))^k$, η_A is safe in this sense. For such safe η_A, the probability of a catastrophe (in the selection of η_A) is no more than $(c_2 + o(1))^{-k}$ for appropriate c_2. Taking $c = \min(c_1, c_2)$ completes the theorem. □

Secure iteration. Finally, consider the behavior of the protocol in the iterated setting. As discussed in the introduction, $\mathbf{B}_k$ is used to generate a sequence of beacons $s_0, s_1, \ldots, s_T$. We assume that s_0 is uniform and thus a^k-optative. By induction, let us assume that s_t is a^k-optative and consider the challenge of an adversary attacking the $t+1$st instance of $\mathbf{B}_k$. When this adversary selects s_t (from among the a^k different random possibilities), it is aware of the VRF functions F_i for as many as $[(1-\epsilon)/2]n$ of the parties and may use this information to cleverly select the next beacon. As in the proof of the theorem above, with each of these potential input beacons we may ask if (i) the number of finally induced subsets (that is N_k) will be too large (that is, more than a^k) and (ii) if the values η_A (which the adversary may evaluate!) are not *safe* in the parlance of the proof. Note that these events only occur with probability $\approx b^{-k}$ and hence that even with the opportunity to select the input beacon from a^k possibilities with probability c^{-k} the circumstances are under control—η_A is safe and there are few candidate sets. As η_A is safe, the probability of a catastrophe in η_H is c^{-k} and unless this occurs the resulting distribution of s_{t+1} will again be a^t-optative, as desired.

We conclude that except with probability Tc^{-k}, each beacon value has min entropy $\lambda - k\log a$, as desired.

4 Applications to eventual consensus

The eventual consensus paradigm notably includes the Proof-of-Work (PoW) setting, such as Bitcoin [19, 13], and the Proof-of-Stake setting, such as Ouroboros [18], Praos [10], and Snow White [5]. (There are several other variants which we do not mention.) A broadcast channel can be implemented via the *input endorsement* mechanism; see Ouroboros [18], FruitChains [20], and Garay et al. [13]. Roughly speaking, for an integer parameter d, these mechanisms ensure that except with probability $\exp(-\Omega(d))$, if an honest participant broadcasts a message in this channel, it will be observed by all honest observers that are at least $2d$ rounds ahead in the future.

In the proof-of-stake setting, the execution of these blockchain protocols proceeds in *epochs*, each epoch comprised of a fixed number of rounds (say R). The coin-flipping protocol inside an epoch is seeded by the output of the same protocol in the previous epoch. Specifically, recall the coin-flipping protocol $\mathbf{B}_k = \mathbf{B}_k(s, n, \epsilon, \lambda)$ from Section 2. Now consider the following beacon protocol which outputs the strings $\eta_1, \eta_2, ...$, as follows:

1. As a one-time initialization, all parties are presented with the uniformly random string η_0 and an integer $t \in [R]$.
2. At the tth round in every epoch $e = 1, 2, ...$, the players collectively compute $\eta_e = \mathbf{B}_k(\eta_{e-1}, n, \epsilon, \lambda)$.

Improving existing PoS beacons. Although Praos is secure against an adaptive adversary, their analysis has a "grinding term" q as described in Section 1. Specifically, they assume that an adversarial player can make at most q queries to the random oracle per round. This is reminiscent of the adversarial hashing power in proof-of-work and hence, undesirable in the proof-of-stake setting. We remark that Snow White has a similar grinding term as well.

If these PoS blockchains use the beacon protocol mentioned above, no such assumptions have to made: Indeed, Theorem 7 would immediately guarantee that the beacon's min-entropy is high as long as the adversary controls a minority coalition.

References

[1] Minimal vdf randomness beacon. `https://ethresear.ch/t/minimal-vdf-randomness-beacon/3566`. Accessed: January 28, 2019.

[2] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 913–930, 2018.

[3] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 408–416. IEEE, 1985.

[4] Iddo Bentov, Ariel Gabizon, and David Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016.

[5] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. page 919, 2016. URL `http://eprint.iacr.org/2016/919`.

[6] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.

[7] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

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

[9] Jing Chen and Silvio Micali. Algorand, 2016.

[10] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology – EUROCRYPT 2018*, pages 66–98, 2018.

[11] Y Filmus, L Hambardzumyan, H Hatami, P Hatami, and D Zuckerman. Biasing boolean functions and collective coin-flipping protocols over arbitrary product distributions. In *46th International Colloquium on Automata, Languages and Programming (ICALP)*, 2019.

[12] Ehud Friedgut. Influences in product spaces: KKL and BKKKL revisited. *Combinatorics, Probability and Computing*, 13(1):17–29, 2004.

[13] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[14] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, 2015. URL `https://doi.org/10.1007/978-3-662-46803-6_10`.

[15] LM Goodman. Tezos—a self-amending crypto-ledger white paper. *URL: https://www. tezos. com/static/papers/white paper. pdf*, 2014.

[16] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system, 2018.

[17] J Kahn, G Kalai, and N Linial. The influence of variables on boolean functions. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 68–80, 1988.

[18] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388, 2017.

[19] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

[20] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, page 315–324, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349925. doi: 10.1145/3087801.3087809. URL `https://doi.org/10.1145/3087801.3087809`.

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

[22] Saad Quader. *Security of Proof-of-Stake Blockchains*. PhD thesis, University of Connecticut, U.S., 2021.

[23] Alexander Russell, Michael Saks, and David Zuckerman. Lower bounds for leader election and collective coin-flipping in the perfect information model. *SIAM Journal on Computing*, 31(6):1645–1662, 2002.

[24] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. Hydrand: Efficient continuous distributed randomness. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 32–48.

[25] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 444–460. Ieee, 2017.

[26] Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

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

[28] Herbert S Wilf. *generatingfunctionology*. AK Peters/CRC Press, 3 edition, 2005.