ICPG TechnologyOverviewSeries

ConsensusSystem

Rev.1

Timo Hanke, Mahnush Movahedi and Dominic Williams

ABSTRACT

The DFINITY blockchain computer provides a secure, performant and flexible consensus mechanism. While first defined for a permissioned participation model, the consensus mechanism itself can be paired with any method of Sybil resistance (e.g. proof-of-work or proof-of-stake) to create an open participation model. DFINITY's greatest strength is unfolded in the most challenging proof-of-stake case.

At its core, DFINITY contains a decentralized randomness beacon which acts as a verifiable random function (VRF) that produces a stream of outputs over time. The novel technique behind the beacon relies on the existence of a unique-deterministic, non-interactive, DKG-friendly threshold signatures scheme. The only known examples of such a scheme are pairing-based and derived from BLS [3, 10].

The DFINITY blockchain is layered on top of the DFINITY beacon and uses the beacon as its source of randomness for leader selection and leader ranking. A "weight" is attributed to a chain based on the ranks of the leaders who propose the blocks in the chain, and that weight is used to select between competing chains. The DFINITY blockchain is further hardened by a notarization process which dramatically improves the time to finality and eliminates the nothing-at-stake and selfish mining attacks.

DFINITY's consensus algorithm is made to scale through continuous quorum selections driven by the random beacon. In practice, DFINITY achieves block times of a few seconds and transaction finality after only two confirmations. The system gracefully handles temporary losses of network synchrony including network splits, while it is provably secure under synchrony.

1 PROLOGUE

DFINITY is a decentralized network design whose protocols generate a reliable "virtual blockchain computer" running on top of a peer-to-peer network upon which software can be installed and can operate in the tamperproof mode of smart contracts. The goal is for the virtual computer to finalize computations quickly (using short block times and by requiring only a small number of blocks as "confirmations"), to provide predictable performance (by keeping the time between confirmations approximately constant), and for computational and storage capacity to scale up without bounds as demand for its services increases (using novel validation mechanisms and sharding systems discussed in our other papers). The protocols must be secure against an adversary controlling less than a certain critical proportion of its nodes, must generate cryptographic randomness (which is required by advanced decentralized applications) and must maintain a decentralized nature as it grows in size to millions of nodes.

DFINITY will be introduced in a series of technology overviews, each highlighting an independent innovation in DFINITY such as the consensus backbone, smart contract language, virtual machine, concurrent contract execution model, daemon contracts, peer-to-peer networks and secure broadcast, governance mechanism and scaling techniques. The present document will focus on the consensus backbone and cryptographic randomness.

DFINITY has an unbiasable, verifiable random function (VRF) built-in at the core of its protocol. The VRF not only drives the consensus, it will also be the foundation for scaling techniques such as sharding, validation towers, etc. Moreover, the VRF produced by the consensus layer is available to the application layer, i.e., to the smart contracts and virtual machine. In this way, the consensus backbone is intertwined with many of the other topics.

2 INTRODUCTION

DFINITY's consensus mechanism has four layers as depicted in Fig. 1. The first layer provides registered and Sybil-resistant client identities. On the second layer is a decentralized random beacon. On the third layer is a blockchain that is driven by the random beacon through a probabilistic mechanism for leader ranking. On the fourth layer is a decentralized notary that provides timestamping and publication guarantees, and is ultimately responsible for near-instant finality. DFINITY's consensus layers and other key aspects of the consensus mechanism can be summarized in the following main categories.

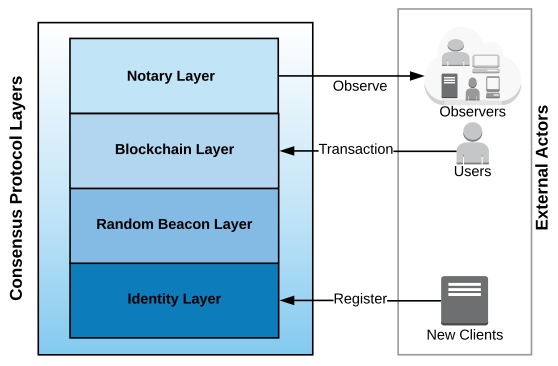


Figure 1: DFINITY's consensus mechanism layers. 1. Identity layer: provides a registry of all clients. 2. Random Beacon layer: provides the source of randomness (VRF) for all higher layers including applications (smart contracts). 3. Blockchain layer: builds a blockchain from validated transactions via the Probabilistic Slot Protocol driven by the random beacon. 4. Notarization layer: provides fast finality guarantees to clients and external observers.

1st layer: Identities and Registry. The active participants in the Dfinity network are called *clients*. All clients in Dfinity are registered, i.e., have permanent, pseudonymous identities. The registration of clients has advantages over the typical proof-of-work blockchains where it is impossible to link different blocks to the same miner. For example, if registration requires a security deposit, a misbehaving client would lose its entire deposit, whereas a miner in a typical proof-of-work blockchain would only forego the block reward during the time of misbehavior. As a result, the penalty for misbehavior can be magnitudes larger for registered identities than it can be for unregistered identities. This is particularly important as blockchains can track unbounded external value that exceeds the value of the native token itself. Moreover, Dfinity supports open membership by providing a protocol to register new clients via a stake deposit with a lock-up period. This is the responsibility of the first layer.

2nd layer: Random Beacon. The random beacon in the second layer is an unbiasable, verifiable random function (VRF) that is produced jointly by registered clients. Each random output of the VRF is unpredictable by anyone until just before it becomes available to everyone. This is a key technology of the Dfinity system which relies on a threshold signature scheme with the properties of uniqueness and non-interactivity. The BLS signature scheme is the only practical[1] scheme that can provide these features, and Dfinity has a particularly optimized implementation of BLS built in [2, 11]. Using a threshold mechanism for randomness creation solves the fundamental "last actor" problem. Any decentralized protocol for creating public randomness without a threshold mechanism suffers from the problem that the last actor in that protocol knows the next random value and can decide to abort the protocol.

3rd layer: Blockchain and fork resolution. The third layer deploys the "probabilistic slot protocol" (PSP). This protocol ranks the clients for each height of the chain, in an order that is derived deterministically from the unbiased output of the random beacon for that height. A weight is then assigned to block proposals based on the proposer's rank such that blocks from clients at the top of the list receive a higher weight. Forks are resolved by giving favor to the "heaviest" chain in terms of accumulated block weight – quite similar to how traditional proof-of-work consensus is based on the highest accumulated amount of work. The first advantage of the PSP protocol is that the ranking is available instantaneously, which allows for a predictable, constant block time. The second advantage is that there is always a single highest-ranked client which allows for a homogenous network bandwidth utilization. Instead, a race between clients would favor a usage in bursts.

4th layer: Notarization and near-instant finality. Finality of a given transaction means a system-wide consensus that a given transaction has been irreversibly executed. While most distributed systems require rapid transaction finality, existing blockchain techniques are unable to provide it. Dfinity deploys the novel technique of block notarization in its fourth layer to speed up finality. A notarization is a threshold signature under a block created jointly by registered clients. Only notarized blocks can be included in a chain. Of all the block candidates that are presented to a client for notarization, the client only notarizes the highest-ranked one with respect to a publicly verifiable ranking algorithm driven by the random beacon. It is important to emphasize that notarization is not consensus because it is possible, due to adverse timing, for more than one block to get notarized at a given height. This is explicitly tolerated and an important difference to other proof-of-stake proposals that apply full Byzantine agreement at every block. Dfinity achieves its high speed and short block times exactly because notarization is *not* full consensus. However, notarization can be seen as optimistic consensus because it will frequently be the case that only one block gets notarized. Whether this is the case can be detected after one subsequent block plus a relay time (cf. Theorem 9.3). Hence, whenever the broadcast network functions normally a transaction is final in the Dfinity consensus after two notarized confirmations plus a network traversal time.

We like to emphasize that a notarization in Dfinity is not primarily a validity guarantee but rather a timestamp plus a proof of publication. The notarization step makes it impossible for the adversary to build and sustain a chain of linked, notarized blocks in secret. For this reason, Dfinity does not suffer from the selfish mining attack [4] or the nothing-at-stake problem.

Threshold Relay and Network Scalability. Dfinity's consensus is designed to operate on a network of millions of clients. To enable scalability to this extent, the random beacon and notarization protocols are designed such that they can be safely and efficiently delegated to a committee. A *committee* is a randomly sampled subset of all registered clients that deploys a threshold mechanism (for safety) that is moreover non-interactive (for efficiency).

In Dfinity, the active committee changes regularly. After having temporarily executed the protocol on behalf of all clients, the committee relays the execution to another pre-configured committee. We call this technique "Threshold Relay" in Dfinity.

Consistency vs availability. It is worth noting that network splits are implicitly detectable by Dfinity and are handled conservatively. This is a consequence of the random sampling of committees. If the network splits in two halves of more or less the same size, this will automatically cause the random beacon to pause within a few blocks so that none of the sides can continue. The random beacon will automatically resume once the network reconnects. If the network splits in a way that one component is significantly larger than half of the network, the protocol may continue in that one large component but will pause in all other components.

Network splits can not only occur when the communication is interrupted. Another important and even more realistic case is when there are multiple implementations of the Dfinity client and they disagree due to the exposures of a bug. Dfinity handles this case gracefully. If there are two clients in evenly widespread use and they start to disagree, then both clients will pause. If there are many evenly spread clients and one starts to disagree from all the others, then the network will likely continue and only the isolated client will pause. This is exactly the desired behavior in the given scenarios. Other blockchains do not handle this case well and the occurrence of such an event poses a real threat to them. The reason is that these chains put too much emphasis on availability rather than consistency.

[1] RSA-based alternatives exist but suffer from an impracticality of setting up the threshold keys without a trusted dealer.

Paper organization. § 3 presents a high-level view of the protocol. § 4 specifies our system, communication and threat models and introduces relevant notations. § 5-7 describe the probabilistic slot protocol and random beacon protocol in detail. § 8.1 introduces the Threshold Relay technique which allows the protocols to be safely executed by pre-configured committees rather than by all replicas. § 8.2 describes the open participation model which allows members to join and leave the protocol over time. Finally, § 9 provides the security and correctness proofs for the DFINITY protocol.

3 A HIGH-LEVEL VIEW OF THE CONSENSUS PROTOCOL

Roles. The DFINITY peer-to-peer network consists of clients connected by a broadcast network over which they can send messages to everyone. Clients fulfill three active functions: (a) participate in the decentralized random beacon, (b) participate in the decentralized notary, (c) propose blocks. Clients also observe blocks and build their own view of the finalized chain.

Committees and Threshold Relay. To improve scalability, the random beacon and notary are run by a *committee*. In a small scale network the committee can be the set of all clients. In a large scale network, the committee is smaller than the set of all clients and changes from round to round (i.e., from block to block). The random beacon output in one round chooses the committee for the next round according to the *threshold relay* technique described in § 8.1. The committee size is configured based on a failure probability calculation (see § 4.2.4).

Block ranking. If we abstract away the decentralized aspect of the random beacon and notary then the consensus protocol is depicted in Fig. 2. The protocol proceeds in rounds such that there is a one-to-one correspondence between the round number and the position (called *height*) in the chain. At the beginning of round r, the randomness beacon produces a fresh, verifiable random value and broadcasts it to the network (Fig. 2, step 1). The random beacon output for round r, denoted by ξ_r, determines a priority ranking of all registered clients. Any client can propose a block but a higher priority level of the client means a higher chance that the block will get notarized and that block makers in the subsequent round will build on top of it.

Notarization. Once a client sees a valid ξ_r, it pools transactions collected from users into a block candidate and sends it to the notary (Fig. 2, step 2). The notary waits for a specific constant time (BlockTime) to receive the proposed blocks. Then, the notary runs the ranking mechanism based on the random beacon, chooses the highest-ranked block, and signs and broadcasts it (Fig. 2, step 3). As soon as clients receive a notarized block, they use it to extend their copies of the blockchain thereby ending round r in their respective views. Finally, the random beacon broadcasts ξ_{r+1} which marks the beginning of a new round.

Decentralized Random Beacon. The random beacon protocol is completely decentralized and operated by all clients in the committee together. Nevertheless from the outside (i.e., looking only at the outputs produced and the timing of the outputs), the beacon

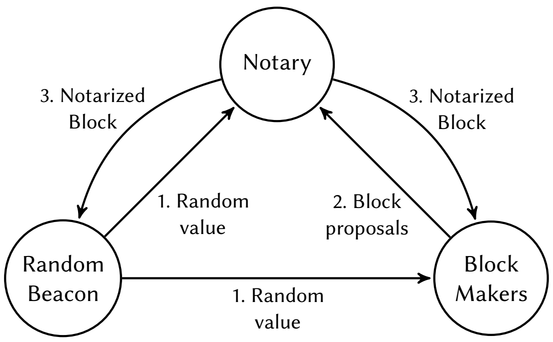


Figure 2: High-level overview on the system components. 1. The random beacon in round r produces random output ξ_r. 2. The block maker(s) selected deterministically by ξ_r propose block(s) for round r. 3. The decentralized notary notarizes the block(s) of the preferred block maker(s). 4. The random beacon advances to round $r+1$ upon seeing a notarized block from round r.

behaves like a trusted third party. We emphasize that the committee does not need to run a Byzantine agreement protocol for every output that the beacon produces. Instead, agreement on each of the beacon's output is automatic because of the uniqueness property of our threshold signature scheme. This explains how the random beacon can run at such high speed, and thereby the DFINITY blockchain can achieve such a low block time.

Decentralized Notary. As was the case for the random beacon, the notary is completely decentralized and operated by all clients in the committee together and its behavior as a whole can be equated to a trusted third party. However, unlike the random beacon, the notary seeks to agree on live input – a block – rather than on a pseudo-random number. There is no "magic" cryptography available for this, so a full Byzantine agreement protocol would be the only option. But instead of doing that, the DFINITY notary merely runs an optimistICPGrotocol which achieves consensus "under norm al operation" though may sometimes notarize more than one blo ck per round. If this happens, DFINITY's chain ranking algorithm will resolve the fork and finality can be achieved in a subsequent no rmal round. The optimistICPG rotocol is non-interactive and fast , hence the notary can run at the same speed as the random beaco n

4 MODELS AND PRELIMINARIES

4.1 System Model

4.1.1 Replicas. From now on we speak of clients as *replicas* and label them $1, 2, \ldots \in \mathbb{N}$. Let U be the finite set of labels of all replicas, called the *universe*. Each replica $i \in U$ has a public/private key pair $(\mathrm{pk}_i, \mathrm{sk}_i)$. We assume the set $\{\mathrm{pk}_i \mid i \in U\}$ is known and agreed upon[2] by all $i \in U$.

4.1.2 Authentication. Each protocol message is signed by the replica that issues the message. The replicas only accept and act upon a message if the message is signed by one of the $\mathrm{sk}_i, i \in U$.

4.1.3 Groups. At any given time, some or all $i \in U$ are arranged into one or more subsets $G_1, G_2, \ldots \subseteq U$ called *groups*, of which

[2]In practice, the agreement is achieved by registering all replicas on the blockchain with their public key.

a single one, the *committee*, is active to drive progress and ensure consensus. We assume all groups G_j have the same size n. The number n is a system parameter called the *group size*.

4.1.4 Synchrony. For the practical use of Dfinity we assume a *semi-synchronous* network by which we mean that the network traversal time can be modeled by a random variable Y whose probability distribution is known. The Dfinity protocol then chooses two system-wide timeout constants $\mathsf{BlockTime}$ and T based on the distribution of Y and the security parameter of the system. In the formal security analysis in § 9 we give proofs for the synchronous case in which an upper bound Δ for Y is known.

The two constants are responsible for liveness ($\mathsf{BlockTime}$) and safety (T) of the system, respectively. Timeout clocks are triggered based on local *events*, i.e. received messages. The protocol does not depend on a global time nor does it assume synchronized clocks between the replicas.

The system evolves in *rounds*. Replicas advance to the next round based on events. The rounds are not expected to be in sync across different replicas.

4.2 Threat Model

4.2.1 Byzantine replicas. A replica that faithfully follows the protocol is called *honest* and all other replicas are called *Byzantine*. A Byzantine $i \in U$ may behave arbitrarily, e.g., it may refuse to participate in the protocol or it may collude with others to perform a coordinated attack the system.

4.2.2 Adversarial strength. For any $G \subseteq U$ let $f(G)$ denote the number of Byzantine replicas in G.

Assumption 1. *There is $\beta > 2$ such that*

$$|U| > \beta f(U). \qquad (4.1)$$

The value $1/\beta$ is called the *adversarial strength*. In practice, Assumption 1 is achieved through economic incentives in conjunction with a form of Sybil resistance.[3]

4.2.3 Honest groups. Let n be the group size. Then a group G is called *honest* if

$$n > 2f(G). \qquad (4.2)$$

The protocols described in § 5-7 rely on

Assumption 2. *Each group G used in the system is honest.*

4.2.4 Random samples. Given Assumption 1, the universe U itself is honest. Each group $G \subseteq U$ used in the system is a random sample of size n drawn from U. Given n, the probability Prob[G honest] can be calculated as follows:

Proposition 4.1. *Let $\mathrm{CDF}_{\mathrm{hg}}(x, n, M, N)$ denote the cumulative distribution function of the hypergeometrICPGrobability distributio n where N is the population size, M is the number of successes*[4] *in t he population, n is the sample size and x is the maximum numbe r of successes allowed per sample. Then*

$$\mathrm{Prob}[G\ \textit{honest}] = \mathrm{CDF}_{\mathrm{hg}}(\lceil n/2 \rceil - 1, n, \lfloor |U|/\beta \rfloor, |U|). \qquad (4.3)$$

[3] Sybil resistance is achieved for example by requiring a stake deposit for each replica. Then Assumption 1 translates to the assumption that at least a $1/\beta$ fraction of stake deposits were made by honest participants.

[4]In our application M is the number of Byzantine replicas in U.

Given an acceptable *failure probability* ρ, we can solve (4.3) for the minimal group size $n = n(\beta, \rho, |U|)$ such that

$$\mathrm{Prob}[G \text{ honest}] > 1 - \rho$$

for each random sample $G \subseteq U$ with $|G| = n$. The result for the example value $|U| = 10^4$ and different values for ρ and β are shown in Figure 3 below.

$-\log_2 \rho$	$n(\beta, \rho, 10^4)$		
	$\beta = 3$	$\beta = 4$	$\beta = 5$
40	405	169	111
64	651	277	181
80	811	349	227
128	1255	555	365

Figure 3: Minimal group size for $|U| = 10^4$. Example values for the minimal group size $n(\beta, \rho, 10^4)$ for adversarial strength $1/\beta$ and failure probability ρ.

As the population size increases to infinity the hypergeometric distribution converges to the binomial distribution. Thus, as $|U|$ increases to infinity we get

Proposition 4.2. *Let $\mathrm{CDF}_{\mathrm{binom}}(x, n, p)$ denote the cumulative distribution function of the binomial probability distribution where p is the success probability per draw, n is the sample size and x is the maximum number of successes allowed per sample. Then*

$$\mathrm{Prob}[G\ \textit{honest}] \geq \mathrm{CDF}_{\mathrm{binom}}(\lceil n/2 \rceil - 1, n, 1/\beta). \qquad (4.4)$$

Given ρ, we can solve (4.4) for n and get the minimal group size $n(\beta, \rho)$ such that $n(\beta, \rho) \geq n(\beta, \rho, |U|)$ for all values of $|U|$.

The result for different values for ρ and β are shown in Figure 4 below. As one can see, within the range of interest for ρ, the group size is approximately linear in $-\log_2 \rho$. The resulting group sizes are practical for the protocols described in this paper.[5]

$-\log_2 \rho$	$n(\beta, \rho)$		
	$\beta = 3$	$\beta = 4$	$\beta = 5$
40	423	173	111
64	701	287	185
80	887	363	235
128	1447	593	383

Figure 4: Minimal group size for arbitrarily large U. Example values for the minimal group size $n(\beta, \rho)$ for adversarial strength $1/\beta$ and failure probability ρ.

4.2.5 Adaptive adversary. We assume that the adversary is *mildly adaptive*. This means the adversary may adaptively corrupt groups but this corruption takes longer than the activity period of the group.

[5] The main protocols described in this paper are so-called "non-interactive". Group sizes of 1,000 have been tested in implementations and were proven to be unproblematic. Dfinity plans to launch its network with group sizes in the order of 400.

4.3 CryptographICPGrimitives

4.3.1 Hash function. We assume we have a collision-resistant hash function H with digests of bit-length l where l matches the security parameter κ.

4.3.2 Pseudo-random numbers. We also assume we have a cryptographically secure pseudo-random number generator PRG which turns a seed ξ into a sequence of values $\mathrm{PRG}(\xi, i)$ for $i = 0, 1, \ldots$.

4.3.3 Pseudo-random permutations. The sequence $\mathrm{PRG}(\xi, i)$ can be used as input to the Fisher-Yates shuffle [9, Algorithm 3.4.2P] to produce a random permutation of U. The result is an bijective map $\{1, \ldots, |U|\} \to U$ which we denote by $\mathsf{Perm}_U(\xi)$.

4.3.4 Diffie-Hellman. We assume that the adversary is bounded computationally and that the computational Diffie-Hellman problem is hard for the elliptic curves with pairings in [2].

4.4 Dfinity's Block Chain

We now define formally the concept of a blockchain in Dfinity.

4.4.1 Blocks.

Definition 4.3. A *block* is either a special *genesis block* or a tuple (p, r, z, d, o) where $p \in \{0,1\}^l$ is the hash reference to the previous block, $r \in \mathbb{N}$ is the round number, $z \in \{0,1\}^*$ is the notarization of the previous block, $d \in \{0,1\}^*$ is the data payload ("transactions" and "state"), $o \in U$ is the creator (or "owner"). A *notarization* is a signature on the previous block created by a "notary". For a block $B = (p, r, z, d, o)$ we define

$$\operatorname{prv} B := p, \quad \operatorname{nt} B := z, \quad \operatorname{rd} B := r, \quad \operatorname{dat} B := d, \quad \operatorname{own} B := o.$$

We emphasize that a block contains the notarization z of the previous block in the chain that it references.

4.4.2 Chains.

Definition 4.4. By a *chain* C we mean a finite sequence of blocks $(B_0, B_1, \ldots, B_r)$ with $\operatorname{rd} B_i = i$ for all i, $\operatorname{prv} B_i = H(B_{i-1})$ for all $i > 0$, and $\operatorname{nt} B_i$ a valid signature of B_{i-1} for all $i > 0$. The first block B_0 is a genesis block. The last block B_r is called the *head* of C. We define

$$\operatorname{len} C := r + 1, \quad \operatorname{gen} C := B_0, \quad \operatorname{head} C := B_r.$$

Since blocks in a chain are linked through cryptographic hashes, a chain is an authenticated data structure. A chain is completely determined by its head by virtue of

Proposition 4.5. *It is computationally infeasible to produce two chains $C \neq C'$ with* $\operatorname{head} C = \operatorname{head} C'$.

Definition 4.6. We write $C(B)$ for the uniquely defined chain C with $\operatorname{head} C = B$. Given two chains C, C' we write $C \leq C'$ if C is a prefix of C'.

Assume from now on that all chains have the same genesis block B_0.

Definition 4.7. For any non-empty set S of blocks we denote by $C(S)$ the largest common prefix of all chains $C(B)$ with $B \in S$.

The chain $C(S)$ is defined because every $C(B), B \in S$, contains the genesis block. For any sets of blocks S, T with $S \subseteq T$ we have $C(T) \leq C(S)$. Suppose $\operatorname{prv} S := \{\operatorname{prv} B \mid B \in S \setminus \{B_0\}\} \neq \emptyset$. Then

$$C(\operatorname{prv} S) \leq C(S). \tag{4.5}$$

5 PROBABILISTIC SLOT PROTOCOL AND NOTARIZATION

As was explained in § 3, each protocol round runs through the steps of producing a random beacon output (1), producing block proposals (2), and producing block notarizations (3). Since more than one block can get notarized, these steps alone do not provide consensus. This is where the *probabilistic slot protocol* (PSP) steps in.

Based on *block weight*, the PSP allows replicas to decide which chain to build on when they propose a new block. Over time, this leads to probabilistic consensus on a chain prefix, where the probability of finality increases the more "weight" is added to a chain. This is analogous to proof-of-work chains, where the probability of finality increases the more "work" is added to a chain. However, Dfinity does not stop here and does not rely on this probabilistic type of finality decisions. PSP is only used to guide the block proposers. For finality, Dfinity applies a faster method utilizing a *notarization* protocol.

For this section, we assume the random beacon (which we introduce later in § 7) is working without failure and provides all replicas with a new, unbiased random value ξ_r at the start of each round r. Figure 5 shows how the protocol alternates between extending the blockchain and extending the random beacon chain and demonstrates how the random beacon, block proposer and notary advance in lockstep.

For the exposition of the present section, however, the decentralized nature and precise inner workings of the random beacon are irrelevant. Hence we simply regard the sequence ξ_r as given without making further assumptions about it.

Regarding the threat model, we assume (4.2) for all groups, as stated in Assumption 2. However, for the description and understanding of the notarization protocol it is sufficient to assume that there is only a single group consisting of the universe of replicas U and that

$$|U| > 2f(U). \tag{5.1}$$

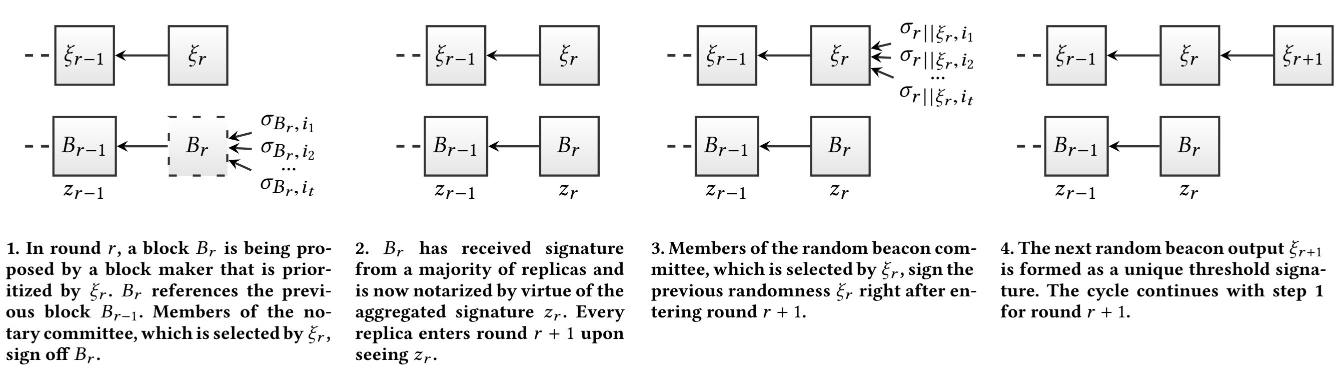
For simplicity of exposition we do adopt this view. It may then be apparent that the protocol described in this section can be delegated to any honest committee or sequence of honest committees.

5.1 Block Rank and Chain Weight

Based on ξ_r, the protocol assigns a *rank* to each $i \in U$ and the rank of the proposer defines the *weight* of a block as follows.

Definition 5.1 (Replica Ranking). The *ranking permutation* for round r is defined as $\pi_r := \mathsf{Perm}_U(\xi_r)$. The *rank* of $i \in U$ in round r is defined as $\pi_r(i)$.

Definition 5.2 (Block Ranking). The *rank* of a block B is defined as $\operatorname{rk} B := \pi_r(\operatorname{own} B)$ where $r = \operatorname{rd} B$. We say B has *higher priority* level than B' if $\operatorname{rk} B < \operatorname{rk} B'$.



ξ_{r-1}
ξ_r
ξ_{r+1}
B_{r-1}
B_r
z_{r-1}
z_r
σ_{B_r, i_1}
σ_{B_r, i_2}
σ_{B_r, i_t}
$\sigma_r || \xi_r, i_1$
$\sigma_r || \xi_r, i_2$
$\sigma_r || \xi_r, i_t$
1. In round r, a block B_r is being proposed by a block maker that is prioritized by ξ_r. B_r references the previous block B_{r-1}. Members of the notary committee, which is selected by ξ_r, sign off B_r.
2. B_r has received signature from a majority of replicas and is now notarized by virtue of the aggregated signature z_r. Every replica enters round $r+1$ upon seeing z_r.
3. Members of the random beacon committee, which is selected by ξ_r, sign the previous randomness ξ_r right after entering round $r+1$.
4. The next random beacon output ξ_{r+1} is formed as a unique threshold signature. The cycle continues with step 1 for round $r+1$.

Figure 5: Alternation between the random beacon chain and the notarized blockchain.

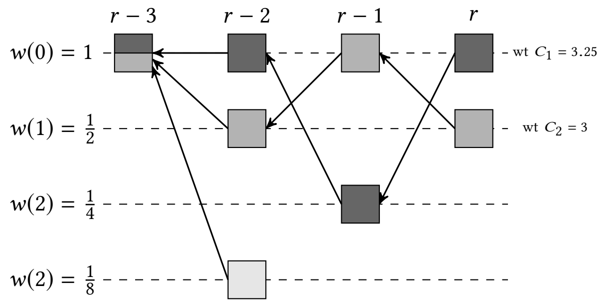


Figure 6: Block proposals. The weight of each block is** 1, 1/2, 1/4, 1/8, . . ., **based on the rank of its proposer. Each chain accumulates the weights of its blocks (here shown only for blocks starting at round $r-3$). A replica with this view will resolve between the two forks that are still active at round r and will choose to build on the heavier one: C_1.

If an adversary equivocates then there will be multiple blocks for the same round with the same rank.

We assume the protocol has defined a monotonically decreasing function w. In particular, for DFINITY we instantiate w as $w(x) = 2^{-x}$.

Definition 5.3 (Block Weight). The *weight* of a block B is defined as $\mathrm{wt}\, B := w(\mathrm{rk}\, B)$.

Definition 5.4 (Chain Weight). The *weight* of a chain $C = (B_0, \ldots, B_r)$ is defined as $\mathrm{wt}\, C := \sum_{h=0}^{r} \mathrm{wt}\, B_h$. We call C *heavier* than another chain C' if $\mathrm{wt}\, C > \mathrm{wt}\, C'$.

5.2 Block Proposals

At each round each replica can propose a block. To do so, in round $r+1$, the replica selects the heaviest valid[6] chain C with $\mathrm{len}\, C = r$ in its view (cf. Fig. 6). The replica then considers all new transactions that it has received from users. The new proposed block B_r references head C and is composed of the selected transactions. The replica broadcasts B_r in order to request notarization from the notary committee.

[6] Validity of blocks is explained after Def. 5.5. A chain is valid if and only if all its blocks are valid.

5.3 Block Notarization

The goal of notarization is to enforce that chains are only being built from blocks that were published during their respective round, not later. In other words, notarization prevents that an adversary can build a private conflicting chain and reveal it later. Blocks that are revealed too late cannot get notarized anymore, so that the *timely publication* of block proposals is enforced. A notarization is therefore regarded as a timestamp as well as a proof of publication.

The protocol guarantees to notarize at least one of multiple proposed chain heads for the current round. It attempts to notarize exactly one chain head for each round but does not guarantee that. Therefore, notarization does not imply consensus nor does it require consensus.

When participating in the notarization protocol the replicas are only concerned about extending at least one valid chain, not about which chain wins (which is the subject of the *finalization* in § 6).

Definition 5.5. A *notarization* of block B is an aggregated signature by a majority subset of U on the message B. We call a block *signed* if it has received at least one signature and *notarized* if it has received a notarization. A *notarized block* is a block concatenated with a notarization of itself.

As described in Alg. 1 below, each replica in each round r collects all valid block proposals from all replicas (including from itself) for a fixed time frame, the so-called BlockTime. A proposed block B is considered valid for round r if $\mathrm{rd}\, B = r$ and there is a valid block B' such that

(1) $\mathrm{prv}\, B = H(B')$ and $\mathrm{rd}\, B' = \mathrm{rd}\, B - 1$,
(2) $\mathrm{nt}\, B$ is a notarization of B',
(3) $\mathrm{dat}\, B$ is valid.[7]

After BlockTime, the replica signs all highest priority blocks for the current round that it has received and broadcasts a signature message for this block to the entire network.

More than one block can have the highest priority but only if the block maker has equivocated. In this case all equivocated block proposals will get signed. This is not an issue because each honest block makers in the next round will only build on one of the blocks.

[7] Having a validity criteria for dat B is optional and not required for the consensus protocol. Depending on the application of the blockchain, for example, dat B can be configured to be valid only if dat B represents valid transactions and a valid state transition from dat B'. From the perspective of the consensus protocol dat B is arbitrary data.

The replica continues to sign all highest priority block proposals as it receives more block proposals after BlockTime. When a notarization for the current round has been observed then the replica advances to the next round.

Algorithm 1 – Block Notarization

Goal: Notarize at least one block for the current round.

```
 1: Initialize chain with the genesis block
 2: $r \leftarrow 1$                                   ▹ initialize round number
 3: while true do
 4:     Wait(BlockTime)
 5:     while no notarization for round $r$ received do
 6:         $\mathcal{B} \leftarrow$ set of all valid round-$r$ block proposals so far
 7:         for All $B \in \mathcal{B}$ with minimal rk $B$ do
 8:             if $B$ not already signed then
 9:                 $\sigma \leftarrow$ Sign($B$)
10:                 Broadcast($\sigma$)
11:             end if
12:         end for
13:     end while
14:     $r \leftarrow r + 1$
15: end while
```

5.4 Properties of notarization

5.4.1 Liveness. From the description above it should be clear that Algorithm 1 cannot deadlock – even in the presence of an adversary. The fact that each replica continues to sign the highest priority block proposal until a notarization for the current round is observed is sufficient to ensure that at least one block gets notarized in the current round. Eventually this will happen because (5.1) holds and the ranking establishes a well-ordering on the set of block proposals. After observing the first notarization for the current round it is safe to stop signing because the observed notarization is being re-broadcasted and will eventually reach all honest replicas. Thus, all honest replicas will advance to the next round.

The above argument relies on propagation assumptions for block proposals and notarizations. We will analyze these in detail, including the relay policies involved, and provide a formal proof for liveness in § 9.

5.4.2 Honestly signed blocks.

Definition 5.6. A block is called *honestly signed* if it has received at least one signature from an honest replica.

Note that an honest replica i only issues a signature on a block B if B was the highest priority proposal visible to i at some point in the round after BlockTime expired. The concept of honestly signed blocks is a theoretical one, used to argue about the security properties of the notarization protocol. It is not possible to tell if a given signature was issued by an honest or Byzantine replica. Hence it is not observable whether a signed block is an honestly signed block or not.

5.4.3 Timely publication.

Definition 5.7. An artifact of round r was *timely published* (within d rounds) if it was broadcasted while at least one honest replica was in round $\leq r$ (resp. in round $\leq r + d$).

As a rule, honest replicas re-broadcast every block that they sign. Hence,

Only timely published blocks can be honestly signed. (5.2)

Given a signed block, it is not possible to tell whether the block was timely published or not because it is not possible to tell if the signature was honest or not. This is different for notarized blocks as we will explain next.

5.4.4 Notarization withholding. By (5.1), any majority subset of replicas contains at least one honest replica. Thus, we have:

Only honestly signed blocks can get notarized. (5.3)

We emphasize that an adversary can withhold its own signatures under an honestly signed block which can lead to a situation where an honestly signed block does not appear to be notarized to the public. However, the adversary can use its own signatures to produce and reveal a notarization at any later point in time.

5.4.5 Enforcing publication. By (5.3) and (5.2),

Only timely published block proposals can get notarized. (5.4)

However, despite (5.4), notarizations can be withheld. To show that withholding notarizations is harmless we introduce another theoretical concept:

Definition 5.8. A notarization z is *referenced* if there is a notarized block B with $\text{nt}\, B = z$.

Note that publishing a block, B, implicitly publishes the notarization of the previous block, nt B, because nt B is contained in B. Hence, (5.4) implies

Only notarizations that are timely published within 1 round can get referenced. (5.5)

Obviously, in a surviving chain all notarizations are referenced. Thus, the publication of both block proposals and notarizations is enforced. An adversary cannot build a private chain because a chain can only survive if:

- All its blocks were timely published.
- All its notarizations were timely published within 1 round.

5.4.6 Consensus. We stated above that a chain can only survive if all its notarizations were timely published within 1 round. This means a replica looking at the notarizations of round r can restrict itself to a certain time window. All notarizations for round r received after that window are necessarily irrelevant for the surviving chains. This fact is the key to the finalization algorithm in § 6 below. See Figure 7 for an example.

There are multiple ways that this fact can lead to a consensus point. Note that it is not necessary for consensus that a single block gets notarized in a round, nor that a single notarization can get referenced. It is sufficient for a consensus point that all notarizations that were received within the time window (indirectly) reference the same block one (or more) rounds back.

5.4.7 *Normal operation.*

Definition 5.9. A round has *normal operation* if only one block for that round gets notarized.

Algorithm 1 strives to achieve normal operation by enforcing the BlockTime waiting period, and by giving preference to the highest priority block proposal.

If the highest priority block maker is honest and BlockTime is large enough then the highest priority block proposal will arrive at all honest replicas before BlockTime expires. This means that only one block can get notarized in this round. Hence, assuming that BlockTime is chosen correctly, Algorithm 1 will achieve normal operation in every round in which the highest priority block maker is honest.

We will analyze in detail what it means that BlockTime is "large enough" (see § 9), taking into account the inner workings of the broadcast network such as the relay policy. We will show that if the network traversal time is bounded by Δ then Alg. 1 is correct if BlockTime $\geq 3\Delta$ (Cor. 9.16, Prop. 9.24, Prop. 9.27).

Note that every round with normal operation creates consensus on the unique block notarized in that round. However, normal operation is a theoretical concept that cannot be observed due to the possibility of notarization withholding. Luckily, as we saw in § 5.4.6, normal operation is not necessary for consensus.

6 FINALIZATION

Replicas use the finalization procedure described in Alg. 2 to identify points of consensus. For this process, it suffices to observe only the notarized blocks, i.e. block proposals and individual signatures under block proposals can be ignored. The finalization protocol is passive and independent from the notarization protocol. Since it can be carried out by anyone (outside of the replicas) who has access to the notarized blocks, we speak of *observers* in this section rather than of replicas.

6.1 Description

Algorithm 2 makes the assumption that the observer receives all round-($r-1$) notarizations that can get referenced before time T after having received the first notarization for round r. This assumption is equivalent to the correctness of Alg. 2 and is proved for all replicas in Theorem 9.18.

The general idea of Alg. 2 is as follows: We continuously collect all notarized blocks and bucket them according to their round number. Let $\mathcal{N}_r$ be the bucket for all notarized blocks for round r.

Multiple buckets can be filled concurrently. For example, a second block may go into $\mathcal{N}_r$ even when $\mathcal{N}_{r+1}$ is already non-empty. However, a block cannot be validated without knowing its predecessor. Therefore, we assume that for every pair of blocks that reference each other, the predecessor is processed first. As a consequence $\mathcal{N}_r$ must receive its first element before $\mathcal{N}_{r+1}$ does.

By our initial assumption, for each round r, there is a time when we can rule out the receipt of any further notarized blocks for $\mathcal{N}_r$ that can get referenced. At that time we "finalize" round r because we know that $\mathcal{N}_r$ already contains all chain tips that can possibly survive beyond round r. Therefore, we can output the longest common prefix $C(\mathcal{N}_r)$ as being final.

Algorithm 2 – Finalization

Goal: Build the finalized chain from observing notarized blocks.

```
MAIN:
1:  $\mathcal{N}_i \leftarrow \emptyset$ for all $i$        ▷ Empty buckets for notarized blocks
2:  $\mathcal{N}_0 \leftarrow \{B_0\}$        ▷ Consider genesis block "notarized"
3:  $C \leftarrow (B_0)$        ▷ Finalized chain
4:  $r \leftarrow 1$        ▷ Current round
5:  while true do
6:      while $\mathcal{N}_r = \emptyset$ do
7:          $B \leftarrow$ incoming notarized block
8:          Store $B$ in $\mathcal{N}_{\mathrm{rd}\,B}$
9:      end while
10:     Schedule the call FINALIZE$(r-1)$ at time $T$ from now
11:     $r \leftarrow r+1$
12: end while

FINALIZE$(h)$:
1:  if $h > 0$ then
2:      $C \leftarrow C(\mathcal{N}_h)$
3:  end if
```

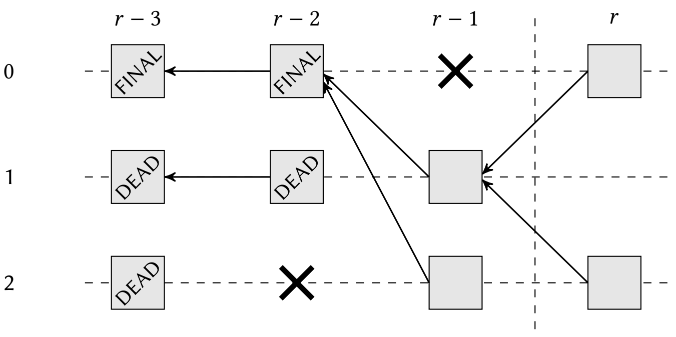


Figure 7: Finalizing blocks. The diagram shows a view of notarized blocks at time T after first seeing a notarized block for round r (i.e. at time T after ending round r and beginning round $r+1$). According to the correctness assumption, (6.1)*, no additional blocks that receive an inbound arrow can appear for the empty positions on the left of the vertical dashed line. Therefore, we mark positions* $(r-2, 2)$ *and* $(r-1, 0)$ *with a cross. The* FINALIZE$(r-1)$ *procedure outputs the longest common prefix, marked "FINAL", of all chains defined by blocks from round* $r-1$ *as their chain tips. As a consequence the blocks marked "DEAD" are now excluded from ever becoming final.*

6.2 Properties of Finalization

We emphasize that there are some notable differences between the use of BlockTime and T: BlockTime is agreed upon and part of the protocol specification, whereas each observer can specify its own T. The notarization protocol requires only BlockTime, not T. The finalization protocol requires only T, not BlockTime.

The following assumption about Alg. 2 is called the *correctness assumption*:

At the time when FINALIZE(h) is being executed, $\mathcal{N}_h$ contains all round-h blocks that can get referenced. (6.1)

Proposition 6.1. *Suppose* (6.1) *holds. Then the chain C in Alg. 2 is append-only.*

The assertion justifies to call the chain C in Alg. 2 *finalized.*

Proof. Let C_h, C_{h+1} be the chains returned by $\textsc{Finalize}(h)$ and $\textsc{Finalize}(h+1)$, respectively, in an execution of Alg. 2. Let $\mathcal{N}_h^t$ denote the set $\mathcal{N}_h$ at time t. Obviously, $\mathcal{N}_h^t$ can grow with increasing t. Let t_0, t_1 be the times when $\textsc{Finalize}(h), \textsc{Finalize}(h+1)$ are called, respectively. By (6.1), none of the blocks added to $\mathcal{N}_h$ after t_0 can get referenced. In other words, we have $\mathrm{prv}(\mathcal{N}_{h+1}) \subseteq \mathcal{N}_h^{t_0}$ regardless of the time at which $\mathcal{N}_{h+1}$ is considered. Thus,

$$C_h = C(\mathcal{N}_h^{t_0}) \leq C(\mathrm{prv}(\mathcal{N}_{h+1}^{t_1})) \overset{(4.5)}{\leq} C(\mathcal{N}_{h+1}^{t_1}) = C_{h+1}.$$

□

We will show that if the network traversal time is bounded by Δ then (6.1) holds if $T \geq 2\Delta$ (Thm. 9.18, Prop. 9.25). Notably, this result does not make any assumptions about BlockTime, i.e. the result holds even if the notaries choose an incorrect value for BlockTime.

There is alternative version of Alg. 2 which does not require the parameter T. In line 10, instead of calling $\textsc{Finalize}(r-1)$ at time T from now, the alternative calls $\textsc{Finalize}(r-2)$ immediately. This can also guarantee (6.1) according to Cor. 9.19 but it requires an assumption about the value of BlockTime used by the notaries.

7 DECENTRALIZED RANDOMNESS BEACON

The *decentralized random beacon* protocol (DRB) allows replicas to agree on a verifiable random function (VRF) and to jointly produce one new output of the VRF in every round. By a VRF we mean a commitment to a deterministic, pseudo-random sequence $(\xi_r)_{r\geq 0}$ for which each output ξ_r is unpredictable given the knowledge of all prior outputs $\xi_0, \dots, \xi_{r-1}$ and for which each output ξ_r is verifiable for correctness by anyone against the commitment. In particular, the VRF outputs are unbiasable due to their deterministic pseudo-random nature. In our decentralized protocol, the output ξ_r shall not be predictable by the adversary before at least one honest replica has advanced to round r.

Regarding the threat model, we assume (4.2) for all groups, as stated in Assumption 2. For simplicity of exposition we describe the random beacon protocol for a single group G with $|G| = n$ and $n > 2f(G)$. The protocol can then be adapted to be executed by changing groups as described in § 8.1 below.

Our DRB protocol uses unique t-of-n threshold signatures (see § 7.1) created by the group G as the source of randomness. The adversary cannot predict the outcome of such a signature if $f \leq t-1$ and cannot prevent its creation if $f \leq n-t$. If the adversary could abort the protocol by preventing a signature from being created, then any restart or fallback mechanism would inevitably introduce bias into the output sequence.[8] We treat the two failures (predicting and aborting) equally. Therefore we require $t \in [f+1, n-f]$. Note that if we set $n = 2t-1$ then both conditions are equivalent to $f \leq t-1$.

[8] Several existing proposals in the literature are susceptible to bias due to a single party aborting the protocol. For example Algorand [8, § 5.2] describes a fallback mechanism which inevitably introduces bias. RANDAO [1] relies on game-theoretic incentives to keep malicious actors from aborting the protocol. In practice, however, the gain for a malicious actor from biasing the randomness is unbounded whereas the penalty for aborting is bounded.

The threshold signature scheme used in the DRB protocol is set up using a distributed key generation mechanism (see 7.1.4) which does not rely on trusted parties. We start by providing the background information on threshold cryptography that we use.

7.1 Background on Threshold Cryptography

7.1.1 Threshold Signatures. In a (t,n)-threshold signature scheme, n parties jointly set up a public key (the *group public key*) and each party retains an individual secret (the *secret key share*). After this setup, t out of the n parties are required and sufficient for creating a signature (the *group signature*) that validates against the group public key.

7.1.2 Non-interactiveness. A threshold signature scheme is called *non-interactive* if the process of creating the group signature involves only a single round of one-way communication for each of the t participating parties. Typically in a non-interactive scheme, each participating party creates a signature share using its individual secret and sends this signature share to a third party. Once the third party has received t valid shares it can recover the group signature without any further interaction. For example, ECDSA can be turned into a threshold signature scheme ([6]) but it does not have the property of non-interactivity.

7.1.3 Uniqueness. A signature scheme is called *unique* if for every message and every public key there is only one signature that validates successfully. This property applies to single signature schemes and threshold signature schemes alike. But in the setting of a threshold scheme it has the additional requirement that the signature must not depend on the subset of t parties that participated in creating the signature. In other words, in a unique threshold signature scheme, regardless of who signs, the resulting group signature will always be the same.

"Unique" is a property that is strictly stronger than "deterministic". A signature scheme is called *deterministic* if the signing function does not use randomness. Note that "unique" is a property of the verification function whereas "deterministic" is a property of the signing function. Unique implies deterministic but not conversely. For example, DSA and ECDSA can be made deterministic by redefining the signing function in a way that it derives its so-called "random k-value" deterministically via a cryptographic hash function from the message plus the secret key instead of choosing it randomly. However, this technique cannot be used to make DSA or ECDSA unique because one cannot expose the k-value to the verification function.

7.1.4 Distributed Key Generation (DKG). For a given (t,n)-threshold signature scheme, a DKG protocol allows a set of n parties to collectively generate the keys required for the scheme (i.e. the group public key and the individual secret key shares) without the help of a trusted party.

Note that a DKG protocol is more than a secret sharing protocol. In a secret sharing protocol the secret shares can be used to recover the group secret, but this can be done only once. After everyone has learned the group secret the shares are not reusable. In a DKG the shares can be used repeatedly for an unlimited number of group signatures without ever recovering the group secret key explicitly.

DKG protocols are relatively straight-forward for discrete-log based cryptosystems and typically utilize multiple instances of a verifiable secret sharing protocol (VSS). Dfinity uses the "Joint-Feldman DKG"[9] as described in [7].

7.2 The BLS signature scheme

The only known signature schemes that have a unique, non-interactive threshold version and allow for a practical, efficient DKG are the pairing-based schemes derived from BLS [3]. BLS was introduced by Boneh, Lynn, and Shacham in 2003 and related work can be found in [10]. We shall use the original BLS scheme throughout.

7.2.1 BLS functions. Assuming we have generated a secret/public key pair (sk, pk), BLS provides the following functions:

(1) $\mathsf{Sign}(m, \mathsf{sk})$: Signs message m using secret key sk and returns signature σ.
(2) $\mathsf{Verify}(m, \mathsf{pk}, \sigma)$: Verifies the signature σ for message m against the public key pk and returns true or false.

Under the hood, BLS uses a non-degenerate, bilinear pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

between cyclic subgroups $\mathbb{G}_1, \mathbb{G}_2$ of suitable elliptic curves points with values in a group of units $\mathbb{G}_T$. We shall write all groups multiplicatively in this paper. For each group, we fix an arbitrary generator: $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, g_T \in \mathbb{G}_T$. We also assume a hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1$ with values in $\mathbb{G}_1$.

The secret keys are scalars, the public keys are elements of $\mathbb{G}_2$ and the signatures are elements of $\mathbb{G}_1$. The function $\mathsf{Sign}(m, \mathsf{sk})$ computes $H_1(m)^{\mathsf{sk}}$ and $\mathsf{Verify}(m, \mathsf{pk}, \sigma)$ tests whether $e(\sigma, g_2) = e(H_1(m), \mathsf{pk})$.

7.2.2 Threshold BLS. We refer to the threshold version of BLS as *TBLS*. The same functions Sign and Verify that are defined for BLS also apply to the key/signature shares and group keys/signatures in TBLS. We assume all participating parties in the (t, n)-DKG are numbered $1, \dots, n$. After having run the DKG as in 7.1.4, the (t, n)-TBLS provides additionally the function:

(1) $\mathsf{Recover}(i_1, \dots, i_t, \sigma_{i_1}, \dots, \sigma_{i_t})$: Recover the group signature σ from the signature shares $\sigma_{i_j}, j = 1, \dots, t$, where σ_{i_j} is provided by the party $i_j \in \{1, \dots, n\}$.

Because of the uniqueness property, the output of Recover does not depend on which t shares from the group are used as inputs. Recover computes a "Lagrange interpolation" for points in $\mathbb{G}_1$. The indices $i_1, \dots, i_t$ must be pairwise different for the Recover function to succeed.

7.3 Randomness Generation

The randomness generation consists of a) a one-time setup in which a DKG is run and b) a repeated signing process in which the outputs are produced. The DKG is slow and requires agreement whereas the repeated signing is non-interactive and fast.

7.3.1 Setup. When setting up a threshold signature scheme, we do not want to rely on any trusted third party. Therefore, the group G runs a DKG for BLS to set up the group public key and the secret key shares during the initialization of the blockchain system. The threshold t is a parameter of the setup.

Once the DKG has finished successfully, it outputs a public verification vector $V_G \in \mathbb{G}_2^t$, and leaves each replica $i \in G$ with its secret key share $\mathsf{sk}_{G,i}$. The verification vector V_G gets committed to and recorded in the blockchain, for example in the genesis block.

Let $V_G = (v_0, \dots, v_{t-1})$. The group public key is $\mathsf{pk}_G = v_0 \in \mathbb{G}_2$. The secret key sk_G corresponding to pk_G is not known explicitly to anyone in G but can be used implicitly through $\mathsf{sk}_{G,i}$. The verification vector V_G can be used to recover the public key share $\mathsf{pk}_{G,i} \in \mathbb{G}_2$ corresponding to $\mathsf{sk}_{G,i}$ via "polynomial" substitution

$$\mathsf{pk}_{G,i} = \prod_{k=0}^{t-1} v_k^{i^k} \in \mathbb{G}_2.$$

Hence, all signature shares produced by i can be publicly verified against the information V_G and i. The group public key pk_G can be used to verify the output of Recover.

7.3.2 Signing process. Recall that a replica enters round r upon seeing the first notarization for round $r-1$. At the beginning of its round r, replica $i \in G$ computes the signature share

$$\sigma_{r,i} = \mathsf{Sign}(r \parallel \xi_{r-1}, \mathsf{sk}_{G,i}),$$

where ξ_{r-1} is the random value of round $r-1$. To bootstrap, ξ_0 has been set to a nothing-up-my-sleeve number, e.g. the hash of the string "Dfinity". Replica i then broadcasts $(i, \sigma_{r,i})$.

Any replica who receives this data can validate $(i, \sigma_{r,i})$ against the public information V_G as described in 7.3.1 above. If valid then the replica stores and re-broadcasts $(i, \sigma_{r,i})$. As soon as a replica has received at least t different valid signature shares, it runs $\mathsf{Recover}(i_1, \dots, i_t, \sigma_{r,i_1}, \dots, \sigma_{r,i_t})$ to compute the group signature $\sigma_{G,r}$. Finally, the random output ξ_r for round r is computed as the hash of $\sigma_{G,r}$.

We emphasize that the signing process is non-interactive. Any third party can do the recovery after a one-way communication of sufficiently many shares.

8 SCALABILITY

8.1 Threshold Relay

For reasons of scalability the notarization and random beacon protocols from § 5 and § 7 are executed by groups of size n rather than by all replicas in U. Otherwise the message complexity would be unbounded as the total number of replicas grows.

The groups, also called *committees* here, are random samples of size n from the whole population U. The group size n is a system parameter that is chosen according to the failure probability analysis of § 4.2.4. A large enough group size ensures that – up to an acceptable failure probability – every group used in the system is honest (Assumption 2).

The mechanism by which Dfinity randomly samples replicas into groups, sets the groups up for threshold operation, chooses

[9] It is known from [6] that the adversary can bias the distribution of public keys generated by the Joint-Feldman DKG. However, the bias generally does not weaken the hardness of the DLP for the produced public key ([6, § 5]). Therefore, with the simplicity of our protocol in mind, we use the original, unmodified Joint-Feldman DKG even though variations are available that avoid the bias.

the current committee, and relays from one committee to the next is called *threshold relay*.

8.1.1 Group Derivation. Let n be the group size. The groups are derived from a random seed ξ where the j-th derived group is

$$\mathsf{Group}(\xi, j) := \mathsf{Perm}_U(\mathsf{PRG}(\xi, j))(\{1, \ldots, n\}). \tag{8.1}$$

At the start of the system, we choose a number m and a seed ξ and form groups

$$G_j := \mathsf{Group}(\xi, j), \quad j = 1, \ldots, m. \tag{8.2}$$

Each G_j runs the DKG described in Section 7.3 to create group keys pk_{G_j} which are then stored in the genesis block.

8.1.2 Committee Selection. The sequence (ξ_i) is bootstrapped by defining an initial value for ξ_0. Then, in round r, we choose

$$G^{(r)} := G_j, \quad j := \xi_r \bmod m \tag{8.3}$$

as the committee for round r. The same committee can be used for the notarization and the random beacon protocols of the same round.

In the random beacon protocol, the members of $G^{(r)}$ jointly produce the output ξ_r, which is then used to select the next committee $G^{(r+1)}$. Since activity is relayed from one group to the next, we call the mechanism "threshold relay".

8.2 Open Participation

It is impractical to assume that the set of all replicas is known from the start of the protocol, especially in DFINITY's public chain. This section describes how the protocol adopts an open participation model in which new replicas can join and existing replicas can leave the system.

8.2.1 Epochs. We divide the rounds into non-overlapping *epochs* of length l where l is a system parameter and is fixed. The block produced in the first round of each epoch is a *registry block* (also called *key frame*) and contains a summary of all new registrations and de-registrations of replicas that happened during the previous epoch that just ended. Note that the summary is a deterministic result of all the blocks in the preceding epoch so that the block maker of the key frame has no opportunity to censor registrations. The first round of the very first epoch is DFINITY's genesis block which is also a key frame.

8.2.2 Registration of Replicas. A replica can request to join the network (i.e. *register*) or leave the network (i.e. *de-register*) by submitting a special transaction for that purpose. The transaction is submitted to the existing replicas for inclusion in the chain just like any other user transaction. A registration transaction contains the public key of the new replica plus an *endorsement* proving that it was allowed to form. Depending on the underlying Sybil resistance method, the endorsement is, e.g., the proof of a locked-up stake deposit, the solution to a proof-of-work puzzle tower, or the certification by a central, trusted authority.

8.2.3 Registration of Groups. The random beacon output of the first round in an epoch e defines the composition of all the groups that are allowed to newly enter the system during this epoch. A system parameter, $m_{\max}$, governs how many different groups can form during an epoch.

Let r be the first round of epoch e. For each $j \leq m_{\max}$ the j-th *candidate group* is defined as $G = \mathsf{Group}(\xi_r, j)$. The members of G run a DKG to establish a group public key pk_G. If the DKG succeeds then the members create a registration transaction for G which contains the tuple $x = (e, j, \mathsf{pk}_G)$. After x is signed by a super-majority of G, any member can submit x for inclusion in the blockchain. The validity of the signature under x is publicly verifiable against the information already on the blockchain, i.e., the pool U of active replicas and the random beacon output ξ_r that defined the group. A registration transaction x is only valid if it is included in a block that lies within the epoch e.

If the DKG fails or x fails to get a super-majority signature from G or x is not included in the blockchain within epoch e then G cannot register. An adversary can cause the registration to fail. For example, if the super-majority is defined as a $2/3$-majority then an adversary controlling $\geq 1/3$ of G can deny the signature under x. However, due to variance, this will happen only to some of the group candidates. For example, an adversary controlling $< 1/3$ of U will control $< 1/3$ in at least half of all groups.

Groups are de-registered automatically when they expire after a fixed number of epochs defined by a system parameter.

8.2.4 Delayed Activity. If the registration of a new identity (replica or group) is included in the chain in epoch e, then the newly registered entity becomes active in epoch $e + 2$. Thus, there is always a gap of at least l rounds between the registration of a new entity and the first activity of that new entity. This sequence of events is shown in Fig. 8.

The gap is required to ensure that all registrations of new entities are finalized before they can be allowed to have any influence on the random beacon. The minimum value of l can be derived from the growth property of the finalized chain that is proved in Prop. 9.24 below.

DFINITY uses a value for l that is far greater than the minimum required, because we want to limit the rate at which key frames are produced, in order to reduce load on so-called observing "light clients".

9 SECURITY ANALYSIS

In this section, we show that the DFINITY protocol provides us with a robust and fast distributed public ledger abstraction. Any ledger must satisfy the following two fundamental properties which we will derive from lower-level properties in § 9.4.

Definition 9.1 (Ledger properties).

a) *Persistence.* Once a transaction is included into the finalized chain of one honest replica, it will then be included in every honest replica's finalized chain.
b) *Liveness.* All transactions originating from honest parties will eventually be included in the finalized chain.

What distinguishes DFINITY from other ledgers is the property of *near-instant finality.* This property is formalized by the following two definitions and theorem.

Definition 9.2 (Number of Confirmations). We say a transaction has *n confirmations* if it is contained in a notarized block B_r and there is a chain of notarized blocks of the form $(\ldots, B_r, \ldots, B_{r+n-1})$.

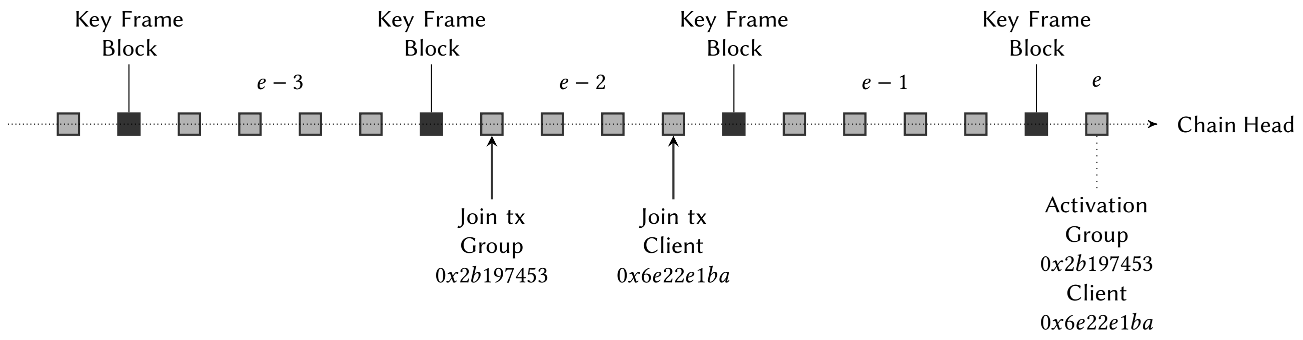

Key Frame Block
Key Frame Block
Key Frame Block
Key Frame Block
$e-3$
$e-2$
$e-1$
e
Chain Head
Join tx Group $0x2b197453$
Join tx Client $0x6e22e1ba$
Activation Group $0x2b197453$ Client $0x6e22e1ba$

Figure 8: Epochs and Registration. The chain is divided into epochs defined by the round numbers of the blocks. A client joins by submitting a special transaction into a block which also locks up a stake deposit. A group joins by successfully executing a DKG (distributed key generation) and submitting the result as a join transaction into the blockchain. Clients and groups become actively involved in the protocol only after a gap of at least 1 epoch between their join transaction and their first activity.

Note that the definition refers to any notarized blocks known to the replica, not necessarily finalized blocks.

THEOREM 9.3 (MAIN THEOREM). *Under normal operation in round r, every transaction included in a block for round r is final after two confirmations plus the maximum network roundtrip time 2Δ.*

From the perspective of an arbitrary observer, the Main Theorem means the following. Suppose an observer sees a transaction x that has received two confirmations, i.e. a notarized block B_r for round r containing x and another notarized block B_{r+1} with prv $B_{r+1} = B_r$. If round r experienced normal operation then, at time 2Δ after the observer received the notarization for B_{r+1}, the finalization algorithm (Alg. 2) is guaranteed to append B_r to the observer's final chain. We assume here that T in Alg. 2 is set to 2Δ. The proof of the Main Theorem will occupy § 9.3 below.

We will provide proofs for the synchronous model where an upper bound for the network traversal time Δ is known. We assume that processing times for messages are included in the network traversal time.

9.1 Broadcast and Processing

The security analysis must take into account the behavior of the broadcast network, which implements a gossip protocol. In particular, the relay policy that is applied to gossiping is going to be essential for the provability of our results.

Replicas continuously receive new protocol artifacts, e.g. block proposals, signatures under block proposals, notarizations, notarized blocks or random beacon outputs. As soon as an artifact is determined to be valid it is immediately relayed ("gossiped") to the replica's peers if it falls under the relay policy defined below.

Definition 9.4 (Relay Policy). All honest replicas relay the following artifacts

a) for the current round: valid block proposals and valid signatures under block proposals,
b) for any round: notarizations and notarized blocks.

We say an artifact has *saturated the network* if it has been received by all honest replicas.

We emphasize that saturating the network is a global condition that is only of theoretical value in our security arguments. The replicas cannot observe whether an artifact has saturated the network or not. Saturating the network does not constitute a reliable broadcast.

Artifacts can be received out of order, e.g. a signature or a notarization for a block can be received before the block. If an artifact x is received before an artifact that is referenced by x then x can not be validated. For this reason, all honest replicas first queue any incoming artifact x until all artifacts referenced by x have also been received. Only then is x processed. In particular, an honest replica i relays an artifact x only if it possesses all artifacts referenced by x. Hence, a peer j of i who receives x from i can then request any artifact that is referenced by x that j does not already possess. This is an artifact synchronization process which happens transparently in the background and is completed before j processes x. Therefore, throughout the paper, we take for granted that if an artifact x is received then all artifacts referenced by x have also been received.

Signatures under block proposals are collected in a background process and, once a majority is available for a given block proposal, are aggregated into a notarization which is then treated in the same way as if it was received from outside. Block proposals and notarizations are collected in the background and made available to Alg. 1 and 2.

Our relay policy and network assumptions (see § 9.2.1 below) guarantee the following property:

Any artifact that falls under b) and is processed by an honest replica will eventually saturate the network. (9.1)

Property (9.1) does not hold for artifacts relayed under policy a) because a replica in the middle of the broadcast path may have advanced to the next round in which case the artifact will be considered old and will be dropped.

Suppose an honest replica i has processed a block proposal B and considered it valid. Then i must possess prv B and a notarization of prv B. We emphasize that the honest replica i re-broadcasts the notarized block prv B in this case. By (9.1), this behavior guarantees:

If a block B is honestly signed then the notarized block prv B eventually saturates the network. (9.2)

Property (9.2) does not hold for B itself or for individual signatures under B, for these artifacts do not fall into category b) of Def. 9.4.

9.2 Timing and Progress

This section makes statements about the relative timing of events that happen at different replicas. We do not assume normal operation in any round and therefore have to consider the possibility of multiple notarizations $z_r, z'_r, \ldots$ being created and broadcasted for the same round r.

9.2.1 Preliminaries. We assume that a message broadcasted by an honest replica at time t reaches every honest replica before $t + \Delta$ (i.e. at a time $< t + \Delta$). Since processing times are not in the scope of our analysis, we assume all processing times to be zero. This applies to the creation as well as to the validation of all messages including block proposals, signatures, notarizations, random beacon shares, and random beacon outputs. As a consequence, for example, when a replica i receives a random beacon output ξ_r at time t then it broadcasts its block proposal for round r at the same time t. Or, when a random beacon member i receives a notarization z_r for round r at time t then i broadcasts its random beacon share for round $r + 1$ immediately at the same time t.

Definition 9.5. Let $\tau_i(A)$ denote the time at which replica i sees event A where A is one of the following: a random beacon output ξ_r, a block proposal B_r, or a notarization z_r. We set $\tau_i(r) := \tau_i(z_{r-1})$ where z_{r-1} is the first notarization for round $r - 1$ that i receives.

Thus, $\tau_i(r)$ is the time when replica i enters round r. To study when the first honest or last honest replica sees an event, we define:

Definition 9.6.

$$\underline{\tau}(A) := \min_{i \text{ honest}} \tau_i(A), \quad \bar{\tau}(A) := \max_{i \text{ honest}} \tau_i(A).$$

For example, $\underline{\tau}(r)$ is the time when the first honest replica enters round r and $\bar{\tau}(r)$ is the time when the last honest replica enters round r. Finally, it is also of interest when an event can first be seen or constructed by the adversary. Therefore, we define:

Definition 9.7.

$$\underline{\tau}^*(A) := \min_i \tau_i(A).$$

For example, $\underline{\tau}^*(\xi_r)$ is the earliest time that the adversary can construct the random beacon output ξ_r.

We will prove in Cor. 9.16 below that the protocol makes continuous progress, i.e. that all values $\tau_i(r)$ are finite. As the reader may verify, the statements made in this section up until Cor. 9.16 also hold (trivially) in the case that any of the values $\tau_i(A)$ are infinite.

Lemma 9.8. *For all rounds r we have:*

$$\bar{\tau}(r) \le \underline{\tau}(r) + \Delta \tag{9.3}$$

and, for any round-r event A under Def. 9.4a),

$$\underline{\tau}(A) + \Delta \le \underline{\tau}(r + 1) \Longrightarrow \bar{\tau}(A) \le \underline{\tau}(A) + \Delta \tag{9.4}$$

Proof. Let i be an honest replica and let z_{r-1} be a notarization for round $r - 1$ such that $\tau_i(z_{r-1}) = \underline{\tau}(r)$. By Def. 9.4b), z_{r-1} is relayed across the network and reaches any other honest replica j by $\tau_i(r) + \Delta$. This proves (9.3).

Now let A be any event that falls under the relay policy in Def. 9.4a) for round r. If $\underline{\tau}(A) + \Delta \le \underline{\tau}(r + 1)$ then the same argument applies. Indeed, the assumption means that all replicas along the broadcast path will still be in round r, thus will relay A according to Def. 9.4a). This proves (9.4). □

9.2.2 Maximal Progress.

Lemma 9.9 (Notarization, "Fast" Bound). *For all rounds r we have:*

$$\underline{\tau}(r) + \mathsf{BlockTime} \le \underline{\tau}^*(r + 1). \tag{9.5}$$

Proof. Honest replicas participate in a notarization z_r for round r only after they have been in round r for at least BlockTime (cf. Alg. 1). The inequality (9.5) means that at least one honest replica is required before anyone can see a notarization z_r for round r. □

Proposition 9.10 (Maximal Progress). *Suppose* BlockTime $\ge \Delta$. *Then the round number of any honest replica increases at most every* BlockTime $- \Delta$. *Moreover, at any point in time, the difference between the round numbers of two honest replicas can be at most* 1.

Proof. From (9.3) and (9.5) together we get:

$$\bar{\tau}(r) + (\mathsf{BlockTime} - \Delta) \le \underline{\tau}^*(r + 1).$$

This implies both statements. □

Corollary 9.11 (Safe Broadcast). *Suppose* BlockTime $\ge \Delta$. *For all rounds r and any round-r event A under Def. 9.4a) we have:*

$$\underline{\tau}(A) \le \underline{\tau}(r) + (\mathsf{BlockTime} - \Delta) \Longrightarrow \bar{\tau}(A) \le \underline{\tau}(r) + \mathsf{BlockTime}. \tag{9.6}$$

The interpretation is that if a round-r event is broadcasted by $\underline{\tau}(r) + (\mathsf{BlockTime} - \Delta)$ then it is guaranteed to saturate the network, and this happens by $\underline{\tau}(r) + \mathsf{BlockTime}$.

Proof. From $\underline{\tau}(A) \le \underline{\tau}(r) + (\mathsf{BlockTime} - \Delta)$ we conclude

$$\underline{\tau}(A) + \Delta \le \underline{\tau}(r) + \mathsf{BlockTime} \overset{(9.5)}{\le} \underline{\tau}(r + 1).$$

Thus, by (9.4),

$$\bar{\tau}(A) \le \underline{\tau}(A) + \Delta \le \underline{\tau}(r) + \mathsf{BlockTime}.$$

□

9.2.3 Normal Operation.

Lemma 9.12 (Beacon, "Slow" Bound). *For all rounds r we have:*

$$\bar{\tau}(\xi_r) \le \underline{\tau}(r) + 2\Delta \tag{9.7}$$

Proof. Each honest random beacon member i broadcasts its random beacon share for ξ_r at $\tau_i(r)$ (cf. § 7.3.2). Thus, any other honest replica will receive all honest random beacon shares for ξ_r by $\bar{\tau}(r) + \Delta$ and thus will recover ξ_r by that time, i.e.

$$\bar{\tau}(\xi_r) \le \bar{\tau}(r) + \Delta.$$

The assertion follows after applying (9.3). □

We now assume BlockTime $\ge 3\Delta$ for the rest of the subsection. The timing of events with BlockTime $= 3\Delta$ is illustrated in Fig. 9.

Lemma 9.13 (Block, "Slow" Bound). *Suppose* BlockTime $\ge 3\Delta$. *For each round r and each honest block proposal B_r we have:*

$$\bar{\tau}(B_r) \le \underline{\tau}(r) + \mathsf{BlockTime}. \tag{9.8}$$

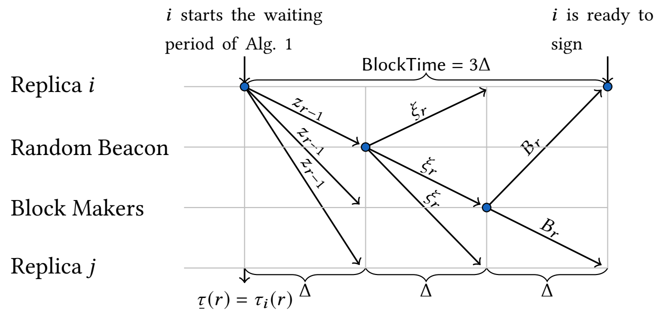


Figure 9: Events Timings. Let replica i be the first honest replica to enter round r, at time $\tau_i(r) = \underline{\tau}(r)$, upon receipt (or construction) of a notarization z_{r-1} for round $r-1$. Let j be any other honest replica for illustration. Replica i broadcasts z_{r-1} to everyone where it arrives before $\underline{\tau}(r)+\Delta$. Thus, all honest replicas start their round r before $\underline{\tau}(r)+\Delta$. Immediately after starting round r, all honest random beacon members create and broadcast their random beacon share to everyone where it arrives before $\underline{\tau}(r)+2\Delta$. Immediately after receiving ξ_r, each honest block maker broadcasts its block proposals B_r to everyone where it arrives before $\underline{\tau}(r)+3\Delta$. Replica i finishes its waiting period at $\tau_i(r)+\mathsf{BlockTime} = \underline{\tau}(r)+3\Delta$ and proceeds to sign the highest priority proposal B_r in its view. Replica j finishes its waiting period later, at $\tau_j(r)+\mathsf{BlockTime}$, and does the same.

Proof. Since B_r is proposed by an honest replica i, it is broadcasted immediately when i receives ξ_r. (Note that we generally ignore processing times throughout the section, including the block creation time.) This means $\underline{\tau}(B_r) \le \bar{\tau}(\xi_r)$, hence

$$\underline{\tau}(B_r) \le \bar{\tau}(\xi_r) \overset{(9.7)}{\le} \underline{\tau}(r) + 2\Delta \le \underline{\tau}(r) + (\mathsf{BlockTime} - \Delta).$$

The assertion follows from Cor. 9.11 for $A = B_r$. □

Proposition 9.14 (Normal Operation). *Suppose* BlockTime $\ge 3\Delta$. *If the highest priority replica in round r is honest, then round r has normal operation.*

Proof. Let i be the highest priority replica of round r and suppose i is honest. Then i proposes exactly one block B_r for round r. Lemma 9.13 implies that for each honest notary member j we have $\tau_j(B_r) \le \tau_j(r) + \mathsf{BlockTime}$. By Alg. 1, replica j waits BlockTime after entering round r and then signs B_r and only B_r because B_r's proposer i has the highest possible priority.

Since notarization requires the participation of at least one honest replica, and all honest replicas sign only B_r, B_r is the only block that can possibly get notarized. In other words, round r can only be ended by a notarization of B_r, which guarantees that the signatures under B_r saturate the network and B_r indeed gets notarized. □

9.2.4 Minimal Progress.

Proposition 9.15 (Minimal Progress). *Suppose* BlockTime $\ge 3\Delta$. *Suppose that in a given round r the replica i with $\pi_r(i) = d$ is honest. Then*

$$\underline{\tau}(r+1) \le \underline{\tau}(r) + \mathsf{BlockTime} + (d+2)\Delta. \tag{9.9}$$

Proof. Set $x_0 := \underline{\tau}(r) + \mathsf{BlockTime}$. Let B_r be the unique proposal made by i for round r. By Lemma 9.13, all honest notary members receive B_r by time x_0.

For each time $x \ge x_0$, we consider the set S_x of valid block proposals for round r that have been received by at least one honest replica. More formally, S_x consists of those valid round-r proposals B that satisfy $\underline{\tau}(B) \le x$. For example, $B_r \in S_{x_0}$. We then define

$$f(x) := \min\{\operatorname{rk} B \mid B \in S_x\}.$$

For example, $f(x_0) \le d$ because B_r has rank d.

For $x \ge x_0$, the function f has values in the non-negative integers and is monotonically decreasing. Since $f(x_0) \le d$, it follows that there is a time $x_0 \le x_1 \le x_0 + d\Delta$ such that $f(x_1) = f(x_1 + \Delta)$. (Otherwise, if $f(x+\Delta) \le f(x) - 1$ for all $x_0 \le x \le x_0 + d\Delta$ then $f(x_0 + (d+1)\Delta) < 0$, a contradiction.)

We claim $\underline{\tau}(r+1) \le x_1 + 2\Delta$. This proves (9.9) since

$$\begin{aligned} x_1 + 2\Delta &\le x_0 + (d+2)\Delta \\ &= \underline{\tau}(r) + \mathsf{BlockTime} + (d+2)\Delta. \end{aligned}$$

If a notarization for a block different from B arrives at any replica before $x_1 + 2\Delta$ then the claim is already proven. We assume w.l.o.g. that this is not the case. Note that under this assumption, (9.4) applies to the events B and any signature under B.

Let $B \in S_{x_1}$ with $\operatorname{rk} B = f(x_1)$. Since $B \in S_{x_1}$ we have $\underline{\tau}(B) \le x_1$. Hence, by (9.4),

$$\bar{\tau}(B) \le x_1 + \Delta.$$

The facts $\bar{\tau}(B) \le x_1 + \Delta$ and $\operatorname{rk} B = f(x_1 + \Delta)$ together mean that B has minimal rank in every honest replica's view at time $x_1 + \Delta$. Also, $x_1 + \Delta \ge \underline{\tau}(r) + \mathsf{BlockTime} + \Delta \ge \bar{\tau}(r) + \mathsf{BlockTime}$, i.e. $x_1 + \Delta$ is past the BlockTime waiting period for every honest replica. Thus, all honest replicas have broadcasted a signature for B by $x_1 + \Delta$. By (9.4), all honest replicas receive $f+1$ signatures by $x_1 + 2\Delta$, i.e. $\bar{\tau}(r+1) \le x_1 + 2\Delta$. □

We remark without proof: In the case $d = 0$ the bound can be improved to $\underline{\tau}(r+1) \le \underline{\tau}(r) + \mathsf{BlockTime} + \Delta$. The resulting bounds are then strict for all d.

As a corollary, by induction, we conclude that the protocol makes continuous progress:

Corollary 9.16. *Suppose* BlockTime $\ge 3\Delta$. *For all rounds r and all honest replicas i, $\tau_i(r)$ is finite.*

9.3 Near-Instant Finality

Finality is provided by Alg. 1. This section first proves the correctness of this algorithm and then shows as the main theorem that finality is achieved quickly under normal operation.

Recall:

- According to the relay policy the notarization will reach all other honest replicas.
- As was stated in (9.2), if a block is honestly signed then the notarized previous block will saturate the network.
- if a block is honestly signed then $\operatorname{nt} B$ is re-broadcasted under the policy Def. 9.4b).

Lemma 9.17. *Suppose z_{r-1} is a referenced notarization for round $r-1$. Then,*

$$\bar{\tau}(z_{r-1}) \le \bar{\tau}(r+1) + \Delta. \tag{9.10}$$

Proof. Let B_r be a notarized block with nt $B_r = z_{r-1}$. Since B_r received an honest signature, we have $\underline{\tau}(B_r) \le \bar{\tau}(r+1)$. Since B_r contains z_{r-1}, we have $\underline{\tau}(z_{r-1}) \le \underline{\tau}(B_r)$. This implies $\bar{\tau}(z_{r-1}) \le \underline{\tau}(z_{r-1}) + \Delta \le \bar{\tau}(r+1) + \Delta$. □

Theorem 9.18 (Correctness of Finalization). *Suppose $T \ge 2\Delta$. For every honest replica, before the replica executes Finalize(h) in Algorithm 2 it has received all notarizations for round h that can get referenced.*

The assertion is precisely the correctness assumption (6.1).

Proof. Suppose z_{r-1} is a referenced notarization for round $r-1$. From (9.10) and (9.3), we get

$$\bar{\tau}(z_{r-1}) \le \underline{\tau}(r+1) + 2\Delta. \tag{9.11}$$

In particular, for any honest replica i,

$$\tau_i(z_{r-1}) \le \tau_i(r+1) + 2\Delta. \tag{9.12}$$

This shows that any notarization for round $r-1$ that arrives at i after $\tau_i(r+1)+2\Delta$ cannot get referenced. Since Alg. 2 calls Finalize($r-1$) at time $\tau_i(r+1) + T$ and $T \ge 2\Delta$, this proves the claim. □

Corollary 9.19. *Suppose* BlockTime $\ge 2\Delta$. *Suppose z_{r-1} is a referenced notarization for round $r-1$. Then,*

$$\bar{\tau}(z_{r-1}) \le \underline{\tau}(r+2). \tag{9.13}$$

Proof. By (9.5), $\underline{\tau}(r+1) + \text{BlockTime} \le \underline{\tau}(r+2)$. Hence, by (9.11), $\bar{\tau}(z_{r-1}) \le \underline{\tau}(r+1) + 2\Delta \le \underline{\tau}(r+2)$. □

Provided that BlockTime $\ge 2\Delta$, (9.13) provides an alternative criteria for when to execute Finalize($r-1$) in Alg. 2 that does not require T. Instead of waiting for T into round $r+1$, the finalization procedure can simply wait for round $r+1$ to end in the observer's view.

Theorem 9.20 (Main Theorem). *Under normal operation in round r, every transaction included in a block for round r is final for an observer after two confirmations plus the maximum network roundtrip time* 2Δ.

Proof of Theorem 9.3. Suppose round r has normal operation, i.e. only one block B_r gets notarized for round r. We assume the observer has chosen $T = 2\Delta$. At time T after seeing a notarization for round $r+1$, the observer will finalize round r. Since $\mathcal{N}_r$ (the bucket of all received notarized blocks for round r) contains only B_r, B_r is appended to the final chain at this time. □

9.4 Chain Properties

Recall that each replica at the end of each round has its own view of an append-only finalized chain (cf. Alg. 2). We consider the following properties regarding state and content of the finalized chain C.

Definition 9.21 (Chain properties).

a) *Growth with parameter k.* Each honest replica's finalized chain at the end of their round r has length $\ge r - k$.
b) *Consistency.* If C, C' are the finalized chains of two honest parties, taken at any point in time, then C is a prefix of C' or vice versa.
c) *Quality with parameter l and μ.* Out of any l consecutive blocks from the finalized chain of an honest replica, at least μl blocks were proposed by an honest replica.

Proposition 9.22. *Persistence follows from the properties of chain consistency and chain growth.*

Proof. Suppose a transaction is included in the finalized chain of one honest replica i, say in the block at round r. Given any other honest replica j, by the growth property, j's finalized chain will eventually reach length r as well. At that point, the consistency property guarantees that j's block at round r is identical to the one in i's finalized chain. □

Proposition 9.23. *Liveness follows from the properties of chain quality and chain growth.*

Proof. A transaction originating from an honest replica is picked up by all other honest parties and included in their block proposals.[10] The growth property guarantees that the finalized chain will eventually grow arbitrarily long. The chain quality property applies and guarantees that there will eventually be a block in the finalized chain that was proposed by an honest replica. □

9.4.1 Chain Growth.

Proposition 9.24 (Chain Growth). *Suppose* BlockTime $\ge 3\Delta$. *Accepting a failure probability of ρ, the Chain Growth property holds with parameter $k = \lceil -\log_\beta \rho \rceil$.*

Proof. We first look at the property assuming normal operation in round r. In this case, the chain will be finalized up to and including round r at the end of round $r+1$ plus 2Δ according to Alg. 2. In terms of round numbers, based on Proposition 9.10, each round including round $r+2$ takes at least BlockTime $- \Delta > 2\Delta$, thus at the end of round $r+2$, we can finalize round r so that the finalized chain will have length $r+1$. This means the property holds with $k = 1$.

Whenever the highest priority block maker for r is honest then round r has normal operation (Prop. 9.14). Thus, in general, normal operation happens with probability at least $1 - 1/\beta$. Then, with probability at least $1 - (1/\beta)^k$ the property holds with parameter k. Thus, if we equate this probability with the minimal desired success probability of $1 - \rho$ and solve for k we get $k = \lceil -\log_\beta \rho \rceil$. □

9.4.2 Chain Consistency.

Proposition 9.25 (Chain Consistency). *Suppose $T \ge 2\Delta$. Suppose two honest replicas i, i' output the finalized chains C, C' upon executing Finalize(h), Finalize(h'), respectively. Then $C \le C'$ or $C' \le C$.*

Proof. Assume w.l.o.g $h' \le h$. Let $\mathcal{N}_h, \mathcal{N}'_h$ be the sets of Algorithm 2 at the time when Finalize(h) is being executed by i, i', respectively. Since $h' \le h$, by Prop. 6.1, $C' \le C(\mathcal{N}'_h)$. Let X be the set of all round-h notarizations that can get referenced. Note that X is globally defined after all honest replicas have ended their round $h+1$ (though X is not known to anyone). By Thm. 9.18 (correctness

[10] There may be reasons why a transaction can not be included in a block proposal even if the proposer is honest. Those reasons, such as limited block space, are not part of our definition of "liveness" and are not considered here.

of finalization), we have $X \subseteq \mathcal{N}_h, \mathcal{N}'_h$. The set X is furthermore non-empty because FINALIZE(h) is only called once some round-h notarizations are actually referenced. Hence, $C = C(\mathcal{N}_h) \leq C(X)$ and $C' \leq C(\mathcal{N}'_h) \leq C(X)$. The fact that C, C' are prefixes of a common chain proves the claim. □

9.4.3 Chain Quality. Whether the highest priority replica in round r is adversarial can be modelled as a Bernoulli trial X_r with success probability $f(U)/|U|$. Since the adversary cannot bias the random beacon, the trials X_r for different r are independent. Therefore, an execution of the protocol comes with a Bernoulli process $X_1, X_2, \ldots$ that models the success of the adversary in gaining preference on the proposed block.

Following Garay et.al. [5], we define an execution of the protocol as *typical* if the Bernoulli process does not deviate too much from its expectation. For any set of rounds S we define the random variable $X(S) := \sum_{i \in S} X_i$.

Definition 9.26. An execution is (ϵ, η)-*typical* if, for any set S of consecutive rounds with $|S| \geq \eta$,

$$\left| \frac{X(S) - E(X(S))}{E(X(S))} \right| < \epsilon$$

The idea is that a) we can guarantee chain quality in all typical executions, and b) any execution is typical with all but negligible probability. In practice, "negligible probability" is defined by the security parameter κ and the parameters ϵ, η are a function of κ.

PROPOSITION 9.27 (CHAIN QUALITY). *Suppose* BlockTime $\geq 3\Delta$. *In a (ϵ, η)-typical execution the Chain Quality property holds for $\mu = (1 - 1/\beta)(1 - \epsilon)$ and $l = \eta$.*

PROOF. Whenever $X_r = 0$ there will be only one notarization for r (Prop. 9.14). This notarization will be for the honest proposer's block, so the honest proposal is guaranteed to be in the finalized chain. Therefore the chain quality is at least $(1 - 1/\beta)(1 - \epsilon)$ for a (ϵ, η)-typical execution if $l \geq \eta$. □

ACKNOWLEDGMENTS

We would like to thank Robert Lauko, Marko Vukolic, Arthur Gervais, Bryan Ford, Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris, Cristina Basescu and Nicolas Gailly for helpful comments and discussions.

REFERENCES

[1] RANDAO: A DAO working as RNG of Ethereum. https://github.com/randao/randao, 2017.

[2] J.-L. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves. *Pairing*, 6487:21–39, 2010.

[3] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532, London, UK, UK, 2001. Springer-Verlag.

[4] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[5] J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[6] R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, pages 156–174. Springer, 2016.

[7] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. *Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings*, chapter Secure Distributed Key Generation for Discrete-Log Based Cryptosystems, pages 295–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[8] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies.

[9] D. E. Knuth. The art of computer programming. volume 1: Fundamental algorithms. volume 2: Seminumerical algorithms. *Bull. Amer. Math. Soc*, 1997.

[10] B. Libert, M. Joye, and M. Yung. Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. *Theoretical Computer Science*, 645:1–24, 2016.

[11] S. Mitsunari. Barreto-Naehrig curve implementation and BLS. https://github.com/dfinity/bn, 2017.