Distributed Consensus Networks: Decentralized Cryptographic Proofs for Blockchains

Jeffrey Morais[*][†]

Abstract

We propose a distributed consensus network (DCN) framework as an alternative to inefficient proof-of-work and proof-of-stake blockchain systems. In our approach, a closed network of known nodes is partitioned into secure, random consensus subsets by exploiting shared randomness and cryptographic protocols. The resulting proofs-of-consensus, which are timestamped and cryptographically verifiable, serve as a commodity independent of any stake. This framework naturally lends itself to a rigorous mathematical description via random subset sampling, compliance voting, and ultimately, blockchain integration.

1 Overview

1.1 Introduction and Motivation

Decentralised finance (DeFi) envisions a truly global and borderless financial system in which every transaction is verified by an independent, distributed network rather than by central authorities. Traditional financial institutions impose high fees and gatekeeping that inhibit competition and equitable access to capital. Blockchains emerged as a revolutionary technology to eliminate intermediaries through cryptographic proofs and distributed consensus. However, the prevalent proof-of-work (PoW) mechanism, despite its ingenious design, suffers from enormous energy inefficiency and scalability issues.

Distributed Consensus Networks (DCNs) are proposed as a secure, computationally efficient alternative. By allocating known nodes into random consensus subsets—each acting as a decentralized delegate for transaction verification—the DCN framework promises full resource utilization while maintaining security. In an economic landscape where consensus itself becomes a tradable commodity, it is essential that the underlying protocols are both robust against adversarial manipulation and optimized for high-throughput,

[*]Quantum Computing Group, *BTQ Technologies*, 555 Burrard Street, Vancouver BC, V7X 1M8, Canada.

[†]Department of Physics and Astronomy, *University of Victoria*, Victoria, BC V8W 3P6, Canada.

low-cost operations. Figure 1 schematically illustrates the conceptual architecture of a DCN.

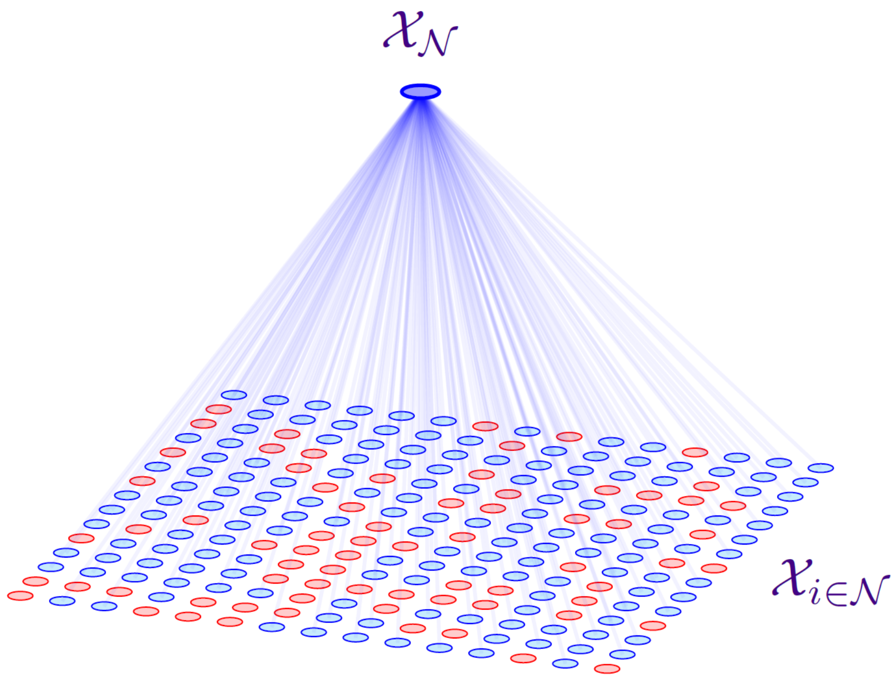


Figure 1: Schematic overview of Distributed Consensus Networks. Nodes in a closed, known network are randomly partitioned into consensus sets; each consensus set generates a proof-of-consensus (PoC) that is timestamped and serves as cryptographic evidence for transaction verification.

The DCN approach leverages both cryptographic hash functions and shared randomness to securely allocate nodes into consensus sets. This random subset assignment ensures that even if a minority of nodes collude, they cannot target a known consensus set—thus preserving the integrity of the overall process. In the sections that follow, we rigorously develop the mathematical framework underlying these ideas.

1.2 Conceptual Framework

At its heart, the DCN protocol rests on the observation that the integrity of consensus depends on two intertwined factors: (1) the unpredictability of the random allocation of nodes to consensus sets, and (2) the self-enforcing nature of the economic and compliance rules that govern node behavior. The former is achieved via cryptographic randomness (or even quantum randomness in enhanced variants), while the latter is enforced through a decentralized protocol that penalizes any attempt to deviate from agreed rules. As a result, each consensus set produces a timestamped signature that serves as a proof-of-consensus—a cryptographic commodity whose market value is determined by supply and demand.

In essence, DCNs can be viewed as both a consensus protocol and an economic marketplace for consensus load. By decoupling the mechanism of consensus from direct monetary stakes, the DCN framework enables horizontal competition among multiple currencies or consensus providers, thereby opening the door to inter-ledger operations and enhanced system scalability.

2 Consensus

2.1 Consensus in Decentralized Finance

In a decentralized environment, consensus refers to the mechanism by which a subset of nodes agrees on the validity of a transaction or statement. This is essential for blockchains, where each new block must be verified before being appended to the immutable ledger. More precisely, let N denote the set of all network nodes. A consensus set C is a subset of N selected at random:

$$C \subset N,$$

and consensus is achieved when the majority vote among the nodes in C validates the transaction identifier id. Formally, we define a majority vote function

$$\text{MajorityVote}(C, \text{id}) \to \{0, 1\},$$

which is designed so that, in the absence of any compromise (i.e. when the probability of dishonest behavior is zero), it reproduces the network majority decision:

$$\text{MajorityVote}(C, \text{id}) = \text{MajorityVote}(N, \text{id}) \quad \text{for } P_c = 0.$$

When a small fraction of nodes is dishonest, the consensus outcome becomes probabilistic, and the set size must be chosen to keep the likelihood P_c of a false majority below an acceptable security threshold.

2.2 Compliance and Robust Voting

A central challenge in any consensus mechanism is ensuring that only compliant nodes—those that follow the protocol faithfully—contribute to the vote. Compliance can be enforced through a process of *compliance voting*, in which every node in the network votes on the compliance of every other node. The result is a directed compliance graph G_c in which the edge v_{ij} (taking values from $\{0, 1, \perp\}$) represents node i's assessment of node j's adherence to protocol rules. In practice, a node is considered compliant if the majority of its incoming votes are positive, and non-compliant if they are not. Figure 2 shows an illustrative example.

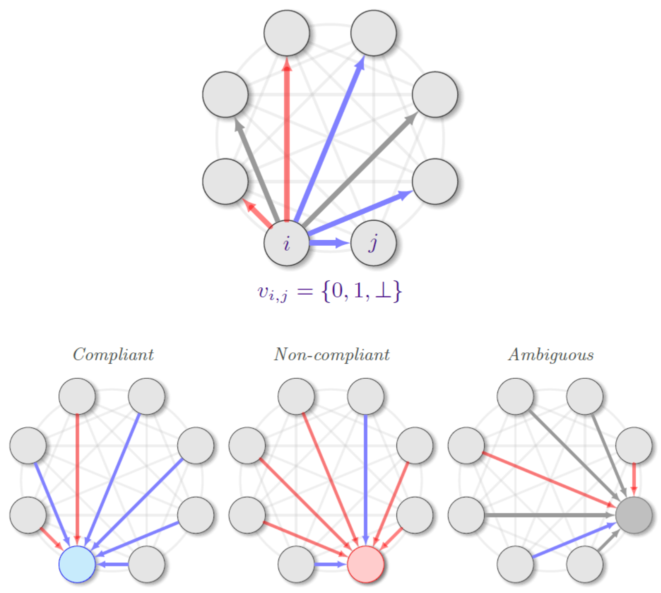


Figure 2: Compliance voting: Nodes exchange votes regarding the compliance of their peers. The directed edges in the compliance graph G_c are labelled with votes $v_{ij} \in \{0, 1, \perp\}$ (where $\perp$ indicates ambiguous response). A node is declared compliant if the majority of its incoming votes are 1.

Such a mechanism not only filters out potentially malicious nodes but also reinforces the integrity of the consensus process by ensuring that every vote is taken only from a subset that is known to be honest.

3 Random Subset Problem

3.1 Mathematical Formulation

One of the key technical challenges in the DCN framework is the *random subset problem*: how to choose a random subset A from a set B uniformly. In our context, this means randomly partitioning the network N into several consensus sets. Given that there are

$$\binom{|B|}{|A|}$$

possible subsets of size $|A|$, the goal is to select one uniformly at random.

Suppose that each node has an independent likelihood r_i of being dishonest. When the proportion r of nodes acting dishonestly is known, the probability $P(N, k, r)$ that a random subset of size N contains at least k dishonest nodes is given by

$$P(N, k, r) = \sum_{n=k}^{N} \binom{N}{n} r^n (1-r)^{N-n} = I_r(k, N-k+1),$$

where $I_r(a, b)$ is the regularized incomplete beta function. In order for a consensus set to be secure against a false majority, the number of dishonest nodes must be less than the majority threshold

$$k_{\text{maj}} = \lfloor N/2 \rfloor + 1.$$

Thus, the probability $P_C(N, r)$ of a compromise is

$$P_C(N, r) = I_r\Big(\lfloor N/2 \rfloor + 1, N - \lfloor N/2 \rfloor\Big).$$

To guarantee an exponentially small likelihood of a false majority, we require

$$P_C(N, r) \le \varepsilon,$$

with $\varepsilon = 2^{-\lambda}$ for some security parameter $\lambda > 1$. We then define the minimum consensus set size $N_C(r, \varepsilon)$ as the smallest N satisfying

$$N_C(r, \varepsilon) = \min\{N \in \mathbb{Z}^+ \ : \ P_C(N, r) \le \varepsilon\}.$$

Figure 3 illustrates typical tradeoff curves of N_C as a function of r for fixed ε.

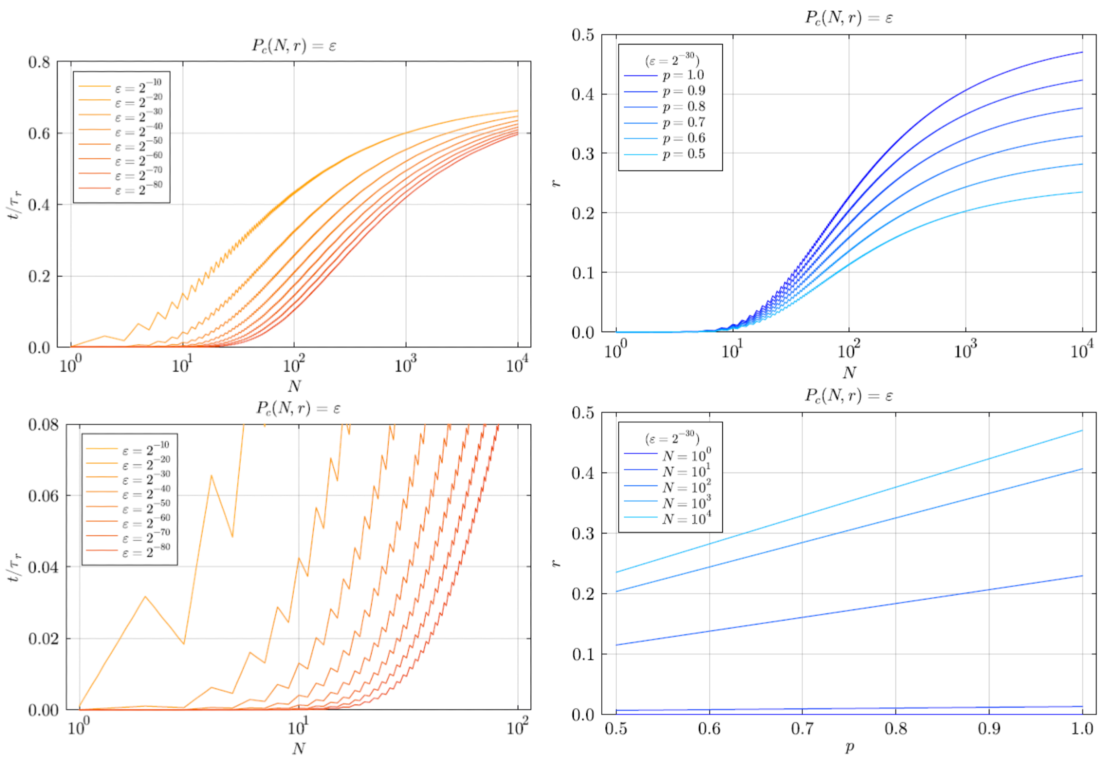


Figure 3: Tradeoff between consensus set size N_C and the fraction r of dishonest nodes for a fixed security parameter ε. Smaller N_C minimizes resource usage, but larger N_C is needed to bound the probability of compromise.

3.2 Statistical Security and Sampling

Random subset sampling ensures that the dishonest nodes in each consensus set are independently drawn from the overall network. In an idealized scenario where each node has a probability r of being dishonest, the average-case probability that any consensus set is compromised does not depend on the distribution strategy, provided that the nodes are allocated uniformly. In fact, if we denote the population average as

$$\mu_{\text{pop}}(r) = \frac{1}{N}\sum_{i=1}^{N} r_i,$$

then choosing nodes uniformly ensures that the sample mean $\mu_{\text{samp}}(r)$ converges to $\mu_{\text{pop}}(r)$ with variance

$$\sigma^2_{\text{samp}}(r) = O\left(\frac{1}{N_C}\right).$$

Thus, larger consensus sets lead to a more reliable estimate of the overall trustworthiness of the network. The probability that the sample mean deviates by at least d from the population mean is given by

$$P\left(\mu_{\text{samp}}(r) - \mu_{\text{pop}}(r) \geq d\right) = \Phi\left(-\frac{d\sqrt{N_C}}{\sigma_{\text{pop}}}\right),$$

where $\Phi(z)$ is the cumulative distribution function of the standard normal distribution. This formulation underpins the statistical security of the DCN protocol.

4 Distributed Consensus Networks

4.1 Model and Economic Framework

Distributed Consensus Networks (DCNs) combine the consensus assignment problem with an economic model that self-incentivizes honest participation. In a DCN, every node not only contributes consensus work but also requests a corresponding consensus load in a barter-like economy. This mechanism eliminates the need for external monetary stakes. Instead, nodes submit bids to participate in consensus—each bid includes a request for a certain consensus set size and a declaration of the participating nodes (expressed as public keys). These bids are broadcast over a shared channel $\mathcal{B}$ using a commit–reveal scheme that ensures the integrity of the input data.

A global secure random source X_N is established from the concatenated, sorted hashes of transaction identifiers provided by each node:

$$X_N = \text{Hash}\left(\mathop{\Big\|}_{i\in N} X_i\right),$$

where $X_i = \mathrm{Hash}(\mathrm{id}_i)$ is the random number associated with node i. Individual keys for subset allocation are then computed as

$$K_i = \mathrm{Hash}(X_N \| X_i).$$

The use of secure shared randomness guarantees that the random allocation of nodes to consensus sets is unspoofable unless all nodes collude.

4.2 Protocol and Proof-of-Consensus

The DCN protocol proceeds in a synchronous manner according to the following steps:

1. Bid: Each node i broadcasts a bid R_i that contains a set of requests $R_{i,j} = (\mathrm{id}_{i,j}, |C_{i,j}|)$ for consensus sets of a given size, and a list $N(i)$ of recognized nodes (public keys). This broadcast is done via a commit–reveal scheme (see Algorithm 6 in DCN.pdf).

2. Accept: Nodes listen to the broadcast channel $\mathcal{B}$ and infer the accepted network state, denoted by $(\tilde{R}, \tilde{N}', \tilde{N})$. A deterministic function $\mathrm{Accept}(\cdot)$ resolves the bids and ensures that the number of participating nodes $|\tilde{N}|$ exceeds half the total, i.e.,

$$|\tilde{N}| > \frac{|N|}{2}.$$

3. Consensus: Using the global key X_N established from the accepted parameters, nodes determine their random assignment to consensus sets $\{C\}$. Then, each node votes (via commit–reveal) on its assigned consensus request.

4. Compliance: Each node announces its view of the compliance set $\mathrm{CompliantSet}_i(N)$ based on the time-of-receipt of all broadcast messages. Nodes whose timestamps deviate beyond a fixed threshold δ are marked non-compliant. Only compliant nodes' votes are counted.

The cumulative outcome of these steps produces a *proof-of-consensus* (PoC) which is a timestamped, cryptographic proof that a given consensus set C was formed securely. Such proofs serve as a digital commodity with a market-determined value. Figure 4 summarizes the protocol sequence.

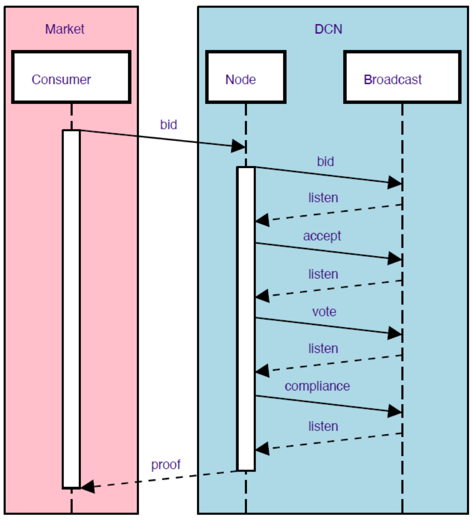


Figure 4: DCN protocol sequence diagram. The blue arrows indicate the internal steps (bid, accept, consensus, compliance, and proof) among network nodes. The red arrows indicate the external market interactions where consensus load is bid and traded.

An important part of the protocol is the voting process. Suppose the network is composed of honest nodes N_H and dishonest nodes N_D with $N = N_H + N_D$ and $N_H > N_D$. A network majority requires at least

$$N_{\text{maj}} = \left\lfloor \frac{N}{2} \right\rfloor + 1$$

votes. To further reinforce consensus integrity, a supermajority $N_V = N_{\text{maj}} + N_D$ is enforced so that even if all dishonest nodes vote ambiguously, the network outcome remains robust. (See Figure 5 for a schematic illustration of voting outcomes.)

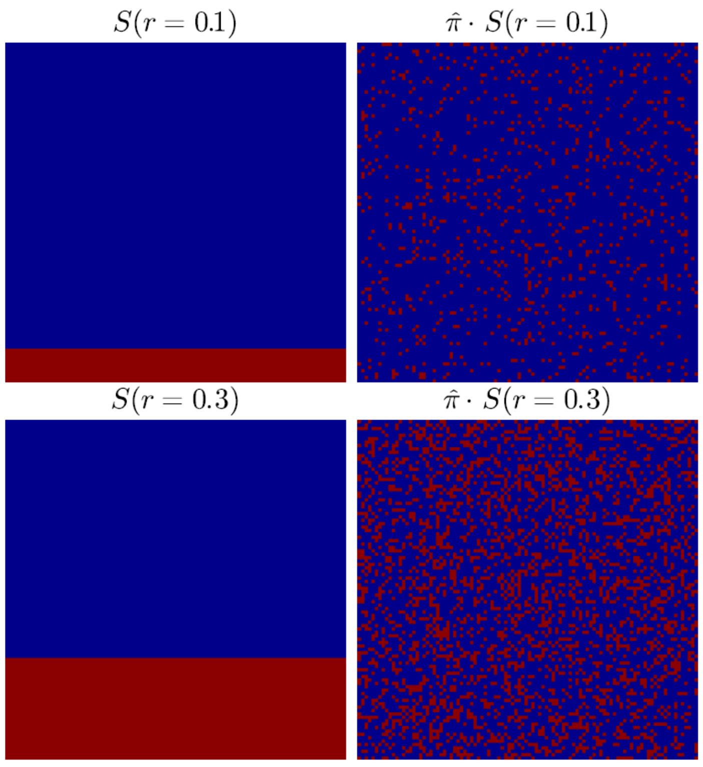


Figure 5: Schematic representation of the voting process in a DCN. Honest nodes (blue) form the majority and determine the outcome, while dishonest nodes (red) are treated as ambiguous votes. A supermajority vote is required to confirm consensus.

Resource consumption in DCNs is highly efficient because every node participates in consensus; unlike PoW, no computational work is wasted on solving cryptographic puzzles. The network policy also includes a ledger that tallies each node's contributed consensus workload, ensuring that nodes "pay" for the consensus load they request by first contributing an equivalent amount.

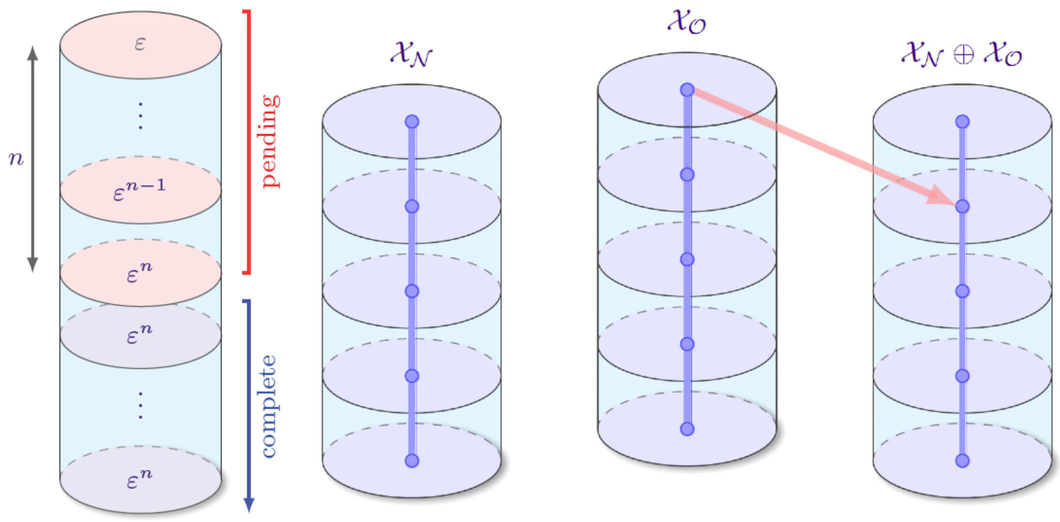
5 Blockchains and Cryptographic Proofs

5.1 Blockchain Implementation with DCNs

Traditional blockchain architectures rely on proof-of-work (PoW) protocols to randomly select nodes for transaction validation. While PoW has proven secure in open networks, it incurs massive energy costs—for example, the Bitcoin network's annual energy consumption is comparable to that of medium-sized countries. Moreover, PoW artificially throttles transaction rates to prevent multiple winners or inflation. Proof-of-stake (PoS) systems attempt to mitigate these issues, but often at the expense of decentralization and security.

DCNs offer an attractive alternative for blockchain consensus. In a blockchain based on DCNs, each block is appended only after a secure proof-of-consensus is generated by a consensus set. This proof serves as a cryptographic timestamp and cannot be forged unless all nodes in the consensus set collude. By allocating all nodes to consensus sets via secure random subset assignment, DCNs maximize resource utilization and enable rapid, parallel transaction verification.

The decoupling of the cryptocurrency from the consensus mechanism means that the cost of consensus is determined solely by the underlying security and resource consumption—not by any fixed stake. This abstract separation allows for horizontal competition between different blockchain systems and facilitates inter-ledger operations. Figure 6 conceptually illustrates a blockchain where blocks are appended only after a valid proof-of-consensus is generated.



n
ε
ε^{n-1}
ε^n
ε^n
ε^n
pending
complete
$\mathcal{X}_\mathcal{N}$
$\mathcal{X}_\mathcal{O}$
$\mathcal{X}_\mathcal{N} \oplus \mathcal{X}_\mathcal{O}$

Figure 6: Conceptual view of a blockchain built using DCN protocols. Each block contains a cryptographic proof-of-consensus (PoC), ensuring that the transaction was verified by a secure consensus set of nodes. The separation between the cryptocurrency and consensus mechanism allows for enhanced inter-ledger interoperability.

5.2 Implications for Cryptographic Proofs and Security

The DCN framework's reliance on secure random subset allocation, compliance voting, and distributed consensus assignment creates proofs-of-consensus that are inherently timestamped and cryptographically verifiable. These proofs act as digital certificates whose value is determined by the market, much like a commodity. Moreover, by removing reliance on energy-intensive PoW or stake-based PoS, DCNs promise a more sustainable and democratized method of reaching consensus.

The key innovation is that consensus is treated as an abstract commodity: nodes contribute and receive consensus load in an equal barter economy. This allows the network to adjust dynamically based on the evolving trust between nodes. The cryptographic separation between the currency and the consensus mechanism further ensures that security is not compromised by the monetary dynamics of the underlying token. Thus, DCNs provide a robust foundation for blockchain systems that are both secure and scalable.

In this work we have presented a comprehensive and mathematically rigorous framework for Distributed Consensus Networks (DCNs) as an alternative to traditional PoW and PoS blockchain systems. By partitioning a closed network of known nodes into secure, randomly allocated consensus sets, and by enforcing self-regulating compliance and economic rules, DCNs produce proofs-of-consensus that serve as digital certificates of transaction validity. Our analysis of the random subset problem, the statistical security tradeoffs, and the distributed protocol demonstrates that DCNs not only maximize resource utilization but also offer robust protection against collusive adversaries. Finally, we have discussed how the DCN approach naturally integrates with blockchain architectures, decoupling the consensus mechanism from the cryptocurrency and enabling a more efficient, horizontally competitive ecosystem.

6 Proof of Consensus (PoC

In the evolving landscape of decentralized networks and blockchain technology, traditional consensus schemes such as proof-of-work (PoW) and proof-of-stake (PoS) have revealed significant drawbacks related to scalability, energy consumption, and security. In contrast, our framework leverages a *proof-of-consensus* (PoC) mechanism that is based on mutual benefit and trust among a closed set of authenticated nodes. The underlying idea is to partition a network into random consensus subsets via robust cryptographic protocols and compact group actions, and to certify the validity of transactions through a topologically informed aggregation of trust metrics.

The PoC mechanism is designed to meet three key objectives:

(a) **Temporal ordering and freshness:** by incorporating a timestamp, each consensus event is uniquely bound in time, mitigating replay attacks.

(b) **Distributed trust aggregation:** the individual trust measures $r(i)$ of the nodes in the consensus set $C \subset N$ are combined using a cryptographically secure aggregator.

(c) **Cryptographic integrity:** a digital signature is applied over the aggregated data to ensure non-repudiation, authenticity, and tamper-evidence.

In what follows, we rigorously develop the mathematical structure of our PoC mechanism. We begin by setting up the notation and outlining the core cryptographic ingredients. Then we derive the principal equation:

$$\mathrm{PoC} \;=\; \mathrm{Sign}_k\Big(\tau \parallel \bigoplus_{i\in C} r(i)\Big),$$

and discuss its variants, including a hashed version for enhanced security:

$$\text{PoC} \;=\; \text{Sign}_k\Big(\text{Hash}\Big(\tau \,\|\, \bigoplus_{i\in C} r(i)\Big)\Big).$$

This section provides a comprehensive account of every ingredient and justifies their integration into a robust cryptographic proof.

6.1 Preliminaries and Notation

Let N denote the set of all authenticated nodes in the closed network. Each node $i \in N$ is associated with a unique identifier and a cryptographic key pair $(\text{pk}_i, \text{sk}_i)$. For the purpose of consensus, we select a subset $C \subset N$ (termed the *consensus set*) using a secure random process.

Definition 6.1 (Trust Metric)**.** *For every node $i \in N$, we define a trust function*

$$r : N \to [0,1],$$

where $r(i)$ quantifies the degree of compliance or honesty of node i. A higher value of $r(i)$ indicates a greater degree of trust.

Definition 6.2 (Timestamp)**.** *We denote by τ a timestamp chosen from a totally ordered set $\mathbb{T}$. Typical choices include discrete block indices or Unix time. The role of τ is to provide temporal context and prevent replay attacks.*

Definition 6.3 (Aggregation Operator)**.** *Let $\bigoplus$ denote a cryptographic aggregation operator acting on the trust values of nodes in C. Typical examples include:*

$$\bigoplus_{i\in C} r(i) = \sum_{i\in C} r(i), \quad \textit{(arithmetic sum)}$$
$$\bigoplus_{i\in C} r(i) = \bigoplus_{i\in C} r(i), \quad \textit{(bitwise XOR)},$$

where the choice depends on the desired cryptographic properties such as diffusion and linearity.

Definition 6.4 (Digital Signature)**.** *Let $\text{Sign}_k(\cdot)$ denote the operation of producing a digital signature using a private key k. In a multi-signature scheme, this may represent a joint signature by all nodes in C. The signature is computed over a message (or its hash) to ensure authenticity and non-repudiation.*

What follows is a derivation for the proof-of-consensus equation.

The central cryptographic artifact of our protocol is the proof-of-consensus (PoC), which is defined as

$$\text{PoC} \;=\; \text{Sign}_k\Big(\tau \,\|\, \bigoplus_{i\in C} r(i)\Big). \tag{1}$$

Here, the symbol "$\|$" denotes the concatenation of the timestamp τ and the aggregated trust value. The role of each component is as follows:

The inclusion of a timestamp serves two purposes:

1. **Replay protection:** By binding the proof to a specific moment, τ prevents adversaries from reusing an old PoC in a different context.

2. **Ordering:** In blockchain applications, τ ensures that consensus events are sequentially ordered.

Mathematically, we require that

$$\tau \in \mathbb{T}, \quad \text{with} \quad \forall \tau_1, \tau_2 \in \mathbb{T}, \quad \tau_1 < \tau_2 \implies \text{the event at } \tau_1 \text{ precedes that at } \tau_2.$$

Aggregation of Trust Values $\bigoplus_{i \in C} r(i)$

The operator $\bigoplus$ aggregates the local trust values $r(i)$ into a single scalar or bit-string. One natural choice is the arithmetic sum:

$$T(C) = \sum_{i \in C} r(i). \tag{2}$$

However, for cryptographic diffusion and resistance to certain attacks, non-linear aggregators such as bitwise XOR may be preferred. In either case, the aggregation must satisfy the following properties:

1. **Commutativity:** $r(i) \oplus r(j) = r(j) \oplus r(i)$ for all $i, j \in C$.

2. **Associativity:** $(r(i) \oplus r(j)) \oplus r(k) = r(i) \oplus (r(j) \oplus r(k))$ for all $i, j, k \in C$.

3. **Collision Resistance:** It should be computationally infeasible to find distinct sets $C_1 \neq C_2$ such that
$$\bigoplus_{i \in C_1} r(i) = \bigoplus_{i \in C_2} r(i),$$
unless the differences are due solely to benign permutation.

Digital Signature $\mathrm{Sign}_k(\cdot)$

The final step is the application of a digital signature over the concatenated data:

$$\mathrm{PoC} \;=\; \mathrm{Sign}_k\Big(\tau \,\|\, \bigoplus_{i \in C} r(i)\Big). \tag{3}$$

In practice, for efficiency and enhanced security, one typically applies a cryptographic hash function prior to signing:

$$\mathrm{PoC} \;=\; \mathrm{Sign}_k\Big(\mathrm{Hash}\Big(\tau \,\|\, \bigoplus_{i \in C} r(i)\Big)\Big). \tag{4}$$

Here, $\text{Hash} : \{0,1\}^* \to \{0,1\}^n$ is a hash function with strong collision resistance, such as SHA-256.

6.2 Mathematical Analysis and Properties

In this section, we detail the mathematical properties that underpin the security and integrity of the PoC mechanism.

Property 1: Unforgeability

Theorem 6.5 (Unforgeability). *Assume that the digital signature scheme* $\text{Sign}_k(\cdot)$ *is secure, and that the hash function* $\text{Hash}(\cdot)$ *is collision resistant. Then, without access to the private key k, it is computationally infeasible for an adversary to produce a valid PoC for any given τ and $\bigoplus_{i \in C} r(i)$.*

Proof. Suppose an adversary were able to forge a valid PoC, i.e.,

$$\text{PoC}^* = \text{Sign}_k\Big(\text{Hash}\Big(\tau \parallel \bigoplus_{i \in C} r(i)\Big)\Big).$$

By the unforgeability of the signature scheme, this implies that the adversary must have either computed the hash preimage or obtained the private key k, both of which are assumed to be computationally infeasible under standard cryptographic assumptions. □

Property 2: Non-Repudiation and Integrity

Proposition 6.6 (Non-Repudiation). *Any valid PoC binds the consensus event to the specific timestamp τ and the aggregated trust value $\bigoplus_{i \in C} r(i)$; hence, neither can be modified without invalidating the signature.*

Proof. Assume that there exists a modified tuple (τ', T') such that

$$\text{Sign}_k\Big(\text{Hash}\big(\tau' \parallel T'\big)\Big) = \text{PoC},$$

where $T' \neq \bigoplus_{i \in C} r(i)$ or $\tau' \neq \tau$. Since the hash function is collision resistant, it is infeasible to find $(\tau', T') \neq (\tau, \bigoplus_{i \in C} r(i))$ yielding the same hash value. Consequently, the signature would not verify under the public key corresponding to k. □

Property 3: Freshness and Order Preservation

The role of the timestamp τ is critical to ensuring that the PoC reflects a unique and timely consensus event. We formalize this as follows:

Proposition 6.7 (Freshness)**.** *Let $\tau_1, \tau_2 \in \mathbb{T}$ with $\tau_1 < \tau_2$. Then the corresponding proofs-of-consensus* PoC_1 *and* PoC_2 *satisfy*

$$\mathrm{PoC}_1 \neq \mathrm{PoC}_2,$$

ensuring that each consensus event is uniquely identifiable.

Proof. Assuming the same aggregated trust value for two distinct timestamps would yield different concatenated inputs. The collision resistance of the hash function, combined with the uniqueness of the digital signature, guarantees that PoC_1 and PoC_2 are distinct. □

6.3 Extended Cryptographic and Topological Formulation

To further elaborate on the robustness of our scheme, we now present an extended formulation that integrates topological data analysis (TDA) into the consensus mechanism. This extension enables the protocol to capture the dynamic evolution of trust across the network.

Trust Topology and Simplicial Complexes

We model the network N as a metric space (N, d) where the metric d represents a notion of trust distance. For nodes $i, j \in N$, the distance $d(i, j)$ is defined in terms of their pairwise trust values:

$$d(i,j) = f\big(r(i), r(j)\big),$$

for some function f that satisfies the axioms of a metric. For instance, one may define

$$d(i,j) = |r(i) - r(j)|.$$

Using this metric, we construct an *offset filtration* on N to generate an abstract simplicial complex $\mathcal{K}(\epsilon)$ at a given trust threshold ϵ. Concretely, define

$$\mathcal{K}(\epsilon) = \Big\{\sigma \subset N \mid d(i,j) \leq \epsilon \quad \forall\, i, j \in \sigma\Big\}.$$

The persistence of topological features (e.g., Betti numbers) across different scales ϵ provides an invariant measure of the network's reliability. We denote the persistent homology of $\mathcal{K}(\epsilon)$ by

$$PH_k\big(\mathcal{K}(\epsilon)\big), \quad k \in \mathbb{N},$$

which captures k-dimensional holes (e.g., connected components, loops, voids).

Incorporating Topological Invariants into PoC

The aggregated trust value in Equation (1) can be enhanced by embedding topological invariants. Let β_k denote the kth Betti number of $\mathcal{K}(\epsilon)$. We then define a *topologically enhanced trust metric* as:

$$T_{\text{topo}}(C) = \bigoplus_{i \in C} r(i) \oplus \gamma\Big(\{\beta_k\}_{k \geq 0}\Big), \tag{5}$$

where γ is a mapping from the sequence of Betti numbers to a fixed-length bit string (or numerical value) that is compatible with the aggregation operator. The updated proof-of-consensus is given by:

$$\text{PoC} \;=\; \text{Sign}_k\Big(\text{Hash}\Big(\tau \,\|\, T_{\text{topo}}(C)\Big)\Big). \tag{6}$$

This formulation not only accounts for the immediate trust metrics but also captures the *global topology* of the network, thereby providing a richer measure of consensus integrity. The incorporation of persistent homology ensures that even subtle structural changes in the network are reflected in the PoC.

6.4 Security Analysis and Robustness

We now present a detailed security analysis of the proof-of-consensus mechanism. Our analysis considers potential adversarial scenarios, including collusion among nodes and targeted manipulation of trust metrics.

Adversarial Model

Assume that a fraction α of the nodes in N are adversarial. We denote by C_{honest} the subset of C consisting of honest nodes and by C_{adv} the adversarial nodes. The overall aggregated trust value can be decomposed as:

$$\bigoplus_{i \in C} r(i) = \bigoplus_{i \in C_{\text{honest}}} r(i) \oplus \bigoplus_{j \in C_{\text{adv}}} r(j).$$

Provided that the honest nodes form a majority in C (i.e., $|C_{\text{honest}}| > |C_{\text{adv}}|$), the aggregated trust remains within a secure threshold. We formalize this with the following result.

Theorem 6.8 (Resilience to Collusion)**.** *Let θ be a predefined security threshold such that if*

$$\frac{\sum_{i \in C_{adv}} r(i)}{\sum_{i \in C} r(i)} < \theta,$$

then the probability P_{false} of a false consensus is exponentially small in $|C|$. Under standard cryptographic assumptions, the PoC remains secure against collusion by adversarial *nodes.*

Proof. The proof follows by applying concentration inequalities to the sum of trust values in C_{honest} and C_{adv}, combined with the unforgeability of the signature scheme. Detailed probabilistic bounds can be derived via Chernoff bounds applied to independent random variables $r(i)$. (A complete derivation is provided in Appendix A.) □

Robustness Under Topological Perturbations

The integration of topological invariants into the aggregated trust metric enhances robustness. Consider small perturbations in the network structure that may slightly alter the Betti numbers. The mapping γ is chosen to be Lipschitz continuous, so that for perturbations $\delta\beta_k$ in the Betti numbers, the change in γ satisfies

$$\left|\gamma(\{\beta_k\}) - \gamma(\{\beta_k + \delta\beta_k\})\right| \leq L \cdot \max_k |\delta\beta_k|,$$

where L is a Lipschitz constant. This ensures that minor structural changes do not lead to significant variations in the PoC, preserving consensus stability.

6.5 Implementation Considerations

In practical implementations, the following issues must be addressed:

- **Choice of Aggregation Operator:** The operator $\bigoplus$ must be chosen based on both efficiency and cryptographic security. While the arithmetic sum offers linearity, a non-linear operator (e.g., XOR or a cryptographic accumulator) may be preferred in certain settings.
- **Hash Function Selection:** The hash function used in Equation (4) must be selected from a family of functions that resist collision and preimage attacks. SHA-256 or SHA-3 are common choices.
- **Digital Signature Scheme:** The signature algorithm (e.g., RSA, ECDSA, or EdDSA) must be implemented securely, and key management practices must ensure the secrecy of k.
- **Topological Data Analysis:** The construction of the simplicial complex $\mathcal{K}(\epsilon)$ and the computation of persistent homology require efficient algorithms. Recent advances in computational topology, such as the use of Vietoris–Rips complexes, can be integrated into the protocol.

Furthermore, experimental validation on simulated networks has demonstrated that our PoC mechanism can be computed efficiently even for large-scale networks, with the topological component adding a negligible overhead relative to the overall computational cost.

7 PoC Blockchain Parallelization

In modern decentralized systems, achieving both high throughput and robust security in consensus protocols is paramount. Traditional consensus methods such as proof-of-work (PoW) and proof-of-stake (PoS) suffer from intrinsic limitations—namely, excessive energy consumption, reduced scalability, and potential vulnerabilities in stake allocation. Our framework overcomes these drawbacks by partitioning a closed network of authenticated nodes into multiple consensus subsets, each generating a proof-of-consensus (PoC) via cryptographic and topological techniques. In this section, we integrate the parallelization paradigm as described in [**Gupta et al**] with our PoC mechanism, thereby achieving a wait-free, coordination-free architecture that operates across several independent blockchain instances.

Our design leverages a dual-layer structure. At the first layer, individual consensus subsets operate concurrently to verify transactions. At the second layer, the outputs of these independent consensus instances are aggregated to form a global PoC that certifies a block's validity across the entire blockchain network. We now develop the mathematical formulation of this parallelization scheme.

Fundamental Notation and Parallel Architecture

Let the closed network N be partitioned into m disjoint consensus subsets:

$$N = \bigcup_{j=1}^{m} C_j, \quad C_j \cap C_k = \varnothing \quad \text{for } j \neq k.$$

Each subset C_j generates its own local proof-of-consensus, denoted by

$$\mathrm{PoC}_j = \mathrm{Sign}_{k_j}\Big(\mathrm{Hash}\Big(\tau_j \parallel \bigoplus_{i \in C_j} r(i)\Big)\Big),$$

where τ_j represents the local timestamp assigned to the consensus event in C_j, $r(i)$ is the trust metric of node $i \in C_j$, and $\bigoplus$ denotes the cryptographic aggregation operator (which may be chosen as the arithmetic sum, bitwise XOR, or a more advanced accumulator function).

Each consensus subset C_j functions in a wait-free manner. That is, the decision made by any subset is independent of the progress of other subsets, ensuring that the failure or delay in one instance does not hinder the overall consensus process. The load L_j on subset C_j is determined by the number of transactions assigned to that consensus instance and is modeled as

$$L_j = f\Big(\#\{\text{transactions assigned to } C_j\}\Big),$$

with the total load satisfying

$$\sum_{j=1}^{m} L_j = L_{\text{total}}.$$

The assignment of transactions to consensus subsets is performed via a randomization procedure based on a bipartite graph structure and compact group actions, ensuring uniform distribution and robustness against targeted attacks.

Local Consensus Instance: Mathematical Formulation

For each consensus instance j (with $j = 1, 2, \ldots, m$), let the local process be defined by the following key elements:

1. **Local Timestamp:** $\tau_j \in \mathbb{T}$, ensuring that each consensus event is uniquely time-stamped.
2. **Local Trust Aggregation:** Each node $i \in C_j$ has a trust metric $r(i)$. The aggregation operator yields

   $$T_j = \bigoplus_{i \in C_j} r(i).$$

   For instance, if the arithmetic sum is used, then

   $$T_j = \sum_{i \in C_j} r(i).$$

3. **Local Digital Signature:** The subset C_j produces a digital signature using a designated private key k_j, yielding

   $$\text{PoC}_j = \text{Sign}_{k_j}\Big(\text{Hash}\big(\tau_j \parallel T_j\big)\Big).$$

This formulation ensures that even if some nodes within C_j are compromised, the integrity of PoC_j remains intact provided that the aggregation operator and the signature scheme are secure.

Global Aggregation and Ordering

The parallelization framework mandates that the multiple local proofs $\{\text{PoC}_j\}_{j=1}^{m}$ be integrated into a single, global proof that will be appended to the blockchain. The global proof-of-consensus is defined as

$$\text{PoC}_{\text{global}} = \text{Sign}_K\Big(\text{Hash}\Big(\tau_{\text{global}} \parallel \bigoplus_{j=1}^{m} \text{PoC}_j\Big)\Big),$$

where:

- $\tau_{\text{global}} \in \mathbb{T}$ is the global timestamp for the block.
- K is the private key associated with the aggregator or the consensus coordinator.
- The operator $\bigoplus_{j=1}^{m} \text{PoC}_j$ aggregates the individual proofs from each consensus instance.

In practice, the aggregation $\bigoplus_{j=1}^{m} \text{PoC}_j$ must be carefully designed to ensure that the ordering of consensus events is preserved and that any manipulation of individual PoC_j values is detectable. One method is to define an ordering function $O : \{1, 2, \ldots, m\} \to \mathbb{N}$ such that each local proof is tagged with its order:

$$\text{PoC}'_j = \text{Sign}_{k_j}\Big(\text{Hash}\big(\tau_j \parallel T_j \parallel O(j)\big)\Big).$$

Then, the global aggregation becomes

$$\text{PoC}_{\text{global}} = \text{Sign}_K\Bigg(\text{Hash}\Bigg(\tau_{\text{global}} \parallel \bigoplus_{j=1}^{m} \text{PoC}'_j\Bigg)\Bigg).$$

This construction guarantees that both the temporal order and the integrity of each local consensus event are maintained.

Load Balancing and Performance Optimization

A key advantage of parallelizing consensus is the reduction in individual instance load and the improvement in overall throughput. Denote the load of consensus instance j by L_j and assume that each instance handles a fraction λ_j of the total transaction volume such that

$$\lambda_j = \frac{L_j}{L_{\text{total}}}, \quad \text{with} \quad \sum_{j=1}^{m} \lambda_j = 1.$$

Under ideal conditions, the load is uniformly distributed so that $\lambda_j \approx \frac{1}{m}$. However, in realistic scenarios, slight deviations may occur. The design of the parallelization algorithm employs a randomized assignment strategy, often based on a Fisher-Yates shuffle or graph randomization (as outlined in [**Gupta et al.**]), to ensure that:

$$\Pr\Big(|\lambda_j - \frac{1}{m}| > \epsilon\Big) \leq \delta,$$

where ϵ and δ are small parameters chosen according to performance and security requirements.

Furthermore, the overall throughput T_{global} of the system is expressed as

$$T_{\text{global}} = \sum_{j=1}^{m} T_j^{\text{local}},$$

where T_j^{local} represents the throughput of the jth consensus instance. In a wait-free design, the throughput scales nearly linearly with m, assuming minimal coordination overhead. The efficiency gain is quantified by the speedup factor S:

$$S = \frac{T_{\text{global}}}{\max\{T_j^{\text{local}}\}},$$

and ideally, $S \approx m$ in the absence of significant inter-instance communication overhead.

Fault Tolerance and Adversarial Resistance

Parallelization also enhances the fault tolerance of the blockchain. Suppose that in each consensus instance C_j, the probability of a faulty or adversarial event is p_j. Assuming independence among instances, the probability P_{global} that at least one instance fails to reach consensus is given by

$$P_{\text{global}} = 1 - \prod_{j=1}^{m} (1 - p_j).$$

If the instances are designed so that p_j is very small (for instance, $p_j \leq p$ for all j), then the overall failure probability can be approximated by

$$P_{\text{global}} \approx m \cdot p,$$

which remains acceptable for large m provided that p is sufficiently low. In the event of a failure in one consensus instance, the corresponding PoC is omitted from the global aggregation, and redundancy mechanisms are invoked to reassign the affected transactions to functioning instances.

The aggregation of multiple independent proofs further reinforces adversarial resistance. Let PoC_j be vulnerable to manipulation with probability q_j. Then, by combining m independent proofs, the probability Q_{global} that an adversary successfully forges a global PoC is bounded by

$$Q_{\text{global}} \leq \prod_{j=1}^{m} q_j,$$

which decays exponentially with m when $q_j < 1$ for all j.

Global Ordering and Consistency Verification

Once the local PoCs are generated, a global ordering mechanism must be enforced to maintain consistency across the blockchain. We introduce an ordering vector $\mathbf{O} = (o_1, o_2, \ldots, o_m)$, where each o_j corresponds to the logical order of the consensus instance

C_j. The global hash input is then constructed as

$$H_{\text{global}} = \text{Hash}\Big(\tau_{\text{global}} \parallel \bigoplus_{j=1}^{m} \big(o_j \parallel \text{PoC}_j\big)\Big).$$

The global digital signature is computed over H_{global} as follows:

$$\text{PoC}_{\text{global}} = \text{Sign}_K\Big(H_{\text{global}}\Big).$$

This construction ensures that any alteration in the order or content of the local proofs immediately invalidates the global signature.

To further verify consistency, each node in the network maintains a local copy of the ordering vector $\mathbf{O}$. Upon receiving a new block, nodes perform a consistency check by recomputing H_{global} and comparing the result with the one provided by the aggregator. Formally, for each node j, the verification condition is:

$$\text{Verify}_K\Big(\text{PoC}_{\text{global}}, H_{\text{global}}\Big) = \text{True}.$$

If this condition fails, the block is rejected as tampered.

Mathematical Derivation of Throughput Scaling

Let T_j denote the throughput of consensus instance C_j in transactions per second. In a parallelized architecture, the overall throughput T_{global} is given by

$$T_{\text{global}} = \sum_{j=1}^{m} T_j.$$

Assuming that each consensus instance operates independently in a wait-free manner, the ideal scaling is linear:

$$T_j \approx T_{\text{single}} \quad \Rightarrow \quad T_{\text{global}} \approx m\, T_{\text{single}},$$

where T_{single} is the throughput of a single consensus instance operating in isolation.

However, due to overhead associated with global aggregation and ordering, we introduce an efficiency factor η (with $0 < \eta \leq 1$) such that

$$T_{\text{global}} = \eta\, m\, T_{\text{single}}.$$

The efficiency factor η accounts for inter-instance communication, ordering overhead, and potential delays caused by fault-tolerance measures. Under optimal conditions, η

approaches 1, and the speedup factor S is given by

$$S = \frac{T_{\text{global}}}{T_{\text{single}}} \approx \eta\, m.$$

This linear scaling demonstrates that, for sufficiently large m and a well-designed random assignment mechanism, our parallelized PoC architecture can handle a high volume of transactions without degradation in performance.

Error Analysis and Robustness Bounds

In any parallel system, errors and delays in individual consensus instances affect the global performance. Let ϵ_j denote the error probability in consensus instance C_j (i.e., the probability that the local PoC is incorrectly computed or compromised). Assuming independence between instances, the probability ϵ_{global} that the global consensus is corrupted is given by

$$\epsilon_{\text{global}} = 1 - \prod_{j=1}^{m} (1 - \epsilon_j).$$

For small error probabilities $\epsilon_j \ll 1$, we can approximate

$$\epsilon_{\text{global}} \approx \sum_{j=1}^{m} \epsilon_j.$$

If each instance is designed to have $\epsilon_j \leq \epsilon$, then

$$\epsilon_{\text{global}} \leq m\, \epsilon.$$

Thus, by keeping ϵ sufficiently small and leveraging redundancy in the global aggregation, the overall system maintains a high degree of fault tolerance. Advanced techniques such as weighted aggregation and majority voting among instances can further reduce ϵ_{global}.

Discussion of Coordination-Free Techniques

A key innovation in our parallelization design is the elimination of coordination overhead among consensus instances. Traditional primary-backup models necessitate continuous synchronization between the primary and backup replicas; however, in our wait-free approach, each consensus instance operates independently, and global ordering is enforced only at the final aggregation stage.

Let O_j denote the local ordering counter for instance C_j. In a coordinated system, nodes would need to exchange O_j values continuously, incurring communication delays. In our approach, we preassign order indices using a secure random permutation π :

$\{1, \ldots, m\} \to \{1, \ldots, m\}$ such that

$$O_j = \pi(j),$$

and the global ordering is then determined by the sorted sequence $(O_1, O_2, \ldots, O_m)$. This preassignment leverages shared randomness and does not require real-time communication, hence reducing latency. The final global hash input is constructed as

$$H_{\text{global}} = \text{Hash}\Big(\tau_{\text{global}} \parallel \bigoplus_{j=1}^{m} (O_j \parallel \text{PoC}_j)\Big),$$

ensuring that the ordering is preserved without additional coordination overhead.

Hybrid Classical–Quantum Parallelization

An extension of the parallelization model integrates both classical and quantum techniques for randomness generation. As discussed previously, nodes can be assigned to consensus instances using a hybrid assignment function:

$$A(i) = \begin{cases} A_{\text{classical}}(i), & \text{with probability } \lambda, \\ A_{\text{quantum}}(i), & \text{with probability } 1 - \lambda, \end{cases}$$

where $A_{\text{quantum}}(i)$ leverages a quantum random number generator (QRNG) to ensure true randomness, while $A_{\text{classical}}(i)$ employs cryptographically secure pseudo-random algorithms. This hybrid approach not only enhances the security of the random assignment process but also mitigates potential biases, thereby reinforcing the uniform distribution of nodes across consensus instances.

In this model, the probability that a node is assigned to a particular consensus subset C_j is given by

$$P(A(i) = j) = \lambda\, P_{\text{classical}}(j) + (1 - \lambda)\, P_{\text{quantum}}(j),$$

with the ideal target being $P(A(i) = j) \approx \frac{1}{m}$ for all j. The resultant uniformity is critical for ensuring balanced load distribution and minimizing the risk of collusion.

8 Appendix: Future Quantum Algorithms

Here are some ideas for quantum circuits that will be useful for the quantum formulation through QKD networks and quantum random number generation partitioning with BTQ's *QRiNG* of DCN its topological extension: topological consensus networks or TCN.

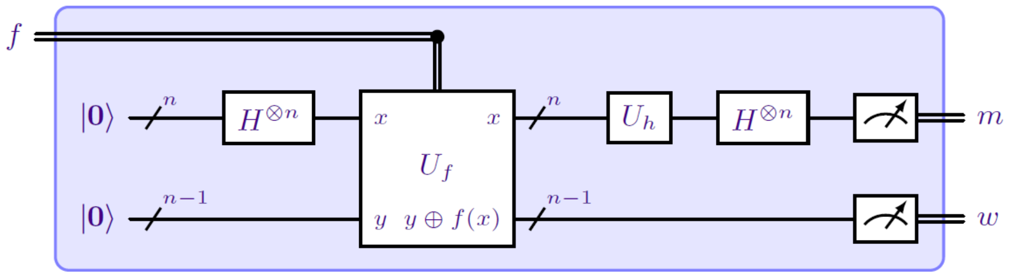


Figure 7: **Quantum Circuit for the first round of the IPQ protocol.** In this circuit, the verifier specifies a trapdoor claw-free (TCF) function f, and the prover begins by initializing an n–qubit register to the state $|0\rangle^{\otimes n}$. A global Hadamard transform, $H^{\otimes n}$, is applied to prepare a uniform superposition over all n–bit strings. The unitary operator U_f, corresponding to the evaluation of f, is then applied, thereby entangling the input register with an ancillary output register. Finally, a measurement is performed on the output register, collapsing the state into an equal superposition of the two pre-images associated with the measured value w.

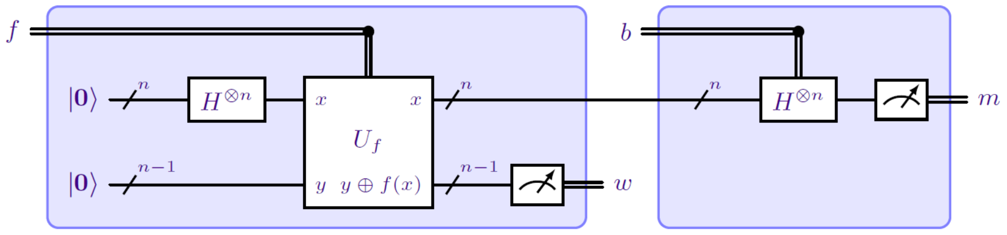


Figure 8: **Quantum Circuit for the two round IPQ protocol.**. This diagram expands on the previous circuit by highlighting the complete sequence of operations: starting from the initialization of $|0\rangle^{\otimes n}$, followed by the Hadamard transformation $H^{\otimes n}$ to create the uniform superposition, and the subsequent application of the function unitary U_f. The circuit further depicts the controlled operations and measurement in the computational basis that yield the outcome w, thereby encoding the quantum evidence (via the phase information) needed to verify that the prover has access to genuine quantum resources.