Rubato: Provably Post-Quantum Secure and Batched Asynchronous Randomness Beacon

Linghe Yang*, Jian Liu*, Jingyi Cui*, Guangquan Xu*, Yude Bai†, Wei Wang‡
*College of Intelligence and Computing, Tianjin University
†School of Software, Tiangong University
‡Ministry of Education Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University
{yanglinghe, jianliu, cuijingyi, losin, baiyude}@tju.edu.cn, wei.wang@xjtu.edu.cn

Abstract*—Distributed Randomness Beacons (DRBs) provide secure, unbiased random numbers for decentralized systems. However, existing protocols face critical limitations. Most rely on cryptographic assumptions which are vulnerable to quantum attacks, risking long-term security in asynchronous networks where unbounded delays may allow attackers time to exploit these weaknesses. Many achieve low beacon generation rates, often below 100 beacons per minute in moderate-scale networks (e.g., *Spurt* IEEE S&P'22), hindering their use in applications requiring high-throughput randomness. Additionally, traditional Verifiable Secret Sharing (VSS)-based DRBs, using a share-consensus-reconstruct paradigm, are unsuitable for asynchronous networks due to circular dependencies between beacon generation and consensus. Given these limitations, we propose *Rubato*, the first provably post-quantum secure DRB for asynchronous environments, incorporating a lattice-based batched Asynchronous Verifiable Secret Sharing scheme (bAVSS-PQ). *Rubato* supports batching of $\mathcal{O}(\lambda^2)$ secrets with communication complexity $\mathcal{O}(\lambda n^3 \log n)$ and tolerates Byzantine faults in up to one-third of the nodes. Integrated with DAG-based consensus protocols like *Bullshark* or *Tusk*, its epoch-staggered architecture resolves circular dependencies, enabling efficient and secure randomness generation. Evaluations across 10 to 50 nodes show *Rubato* generates 5200 to 350 beacons per minute with per-beacon latencies of 11.60 to 96.37 milliseconds, achieving a consensus throughput of 186,088 transactions per second with a latency of 16.78 seconds at 30 nodes. *Rubato* offers robust post-quantum security and high performance for small-to-medium-scale decentralized systems.

1. Introduction

Distributed Randomness Beacons (DRBs) provide unpredictable, bias-resistant random numbers in decentralized systems, addressing the risks of single points of failure and manipulation inherent in centralized sources like Random.org [11]. By distributing trust across multiple nodes, DRBs ensure resilience against Byzantine faults, making them critical for applications such as fair validator selection in Proof-of-Stake blockchains [12], [13], [14], secure electronic voting [15], randomized mechanisms in decentralized finance (e.g., auctions and lotteries) [16], [17], secure multiparty computation [18], [19], and maintaining liveness of consensus protocols in asynchronous networks [20], [21], [22], [23], [24]. These applications demand DRBs that are secure, high-throughput, and robust across diverse network conditions.

DRBs leverage various cryptographic primitives, each with distinct trade-offs in efficiency, network assumptions, and security. Verifiable Delay Functions (VDFs) offer low communication complexity ($\mathcal{O}(\lambda n)$) but require synchronous networks with bounded delays, rendering them impractical for asynchronous environments [25], [26]. Threshold signature schemes, such as BLS [27] and ECDSA [28] variants, provide robust security in synchronous or partially synchronous networks but rely on Distributed Key Generation (DKG), which incurs significant communication and computational overhead whenever nodes' participation changes [29], [30], [31], [32], [33]. In contrast, Verifiable Secret Sharing (VSS) schemes [34], [35] require only a Public Key Infrastructure (PKI) and a Common Reference String (CRS) for setup, support asynchronous operation through AVSS variants, and enable high-throughput batching, making them well-suited for generating randomness at scale in asynchronous networks [4], [5], [7], [8], [9], [36]. These advantages motivate our focus on VSS-based DRBs for achieving secure and efficient randomness generation.

VSS-based DRBs operate by having nodes share random secrets via VSS, achieving consensus on $t+1$ secrets from nodes whose VSS instances have terminated, reconstructing these secrets, and aggregating them into a beacon, assuming n nodes with up to $t < n/3$ Byzantine faults. Despite their flexibility, existing VSS-based DRBs face critical limitations. Many VSS protocols lack batching capabilities, resulting in low beacon generation rates, limiting their applicability in high-throughput scenarios [1], [2], [3], [4], [8], [37]. Furthermore, the share-consensus-reconstruct paradigm introduces circular dependencies in asynchronous networks, as the FLP impossibility theorem necessitates randomness for consensus, conflicting with the DRB's reliance on consensus [38]. Existing solutions mitigate this through Monte-Carlo termination, ensuring deterministic termination but risking inconsistent outputs with small probability [6], [7], or Las Vegas termination, guaranteeing consistency but

Protocol	Network Model	Fault Tolerance	Termination Flavor	Adaptive Security	Post-quantum Security	Communication Complexity	Computational Complexity	Cryptographic Primitives	Cryptographic Assumption	Setup
RandPiper [1]	sync.	1/2	D	✓	✗	$O(\lambda n^3)$	$O(n^2)$	VSS	q-SDH	SRS
Optrand [2]	sync.	1/2	D	✗	✗	$O(\lambda n^2)$	$O(n^2)$	PVSS	q-SDH & SXDH	SRS & PKI
RandFlash [3]	sync.	1/3	D	✗	✗	$^{\dagger}O(n \log n)$	$O(n^2)$	PVSS	DDH	CRS & PKI
Spurt [4]	psync.	1/3	D	✗	✗	$O(\lambda n^2)$	$^{\flat}O(n)$	PVSS	DBDH	CRS & PKI
Rondo [5]	psync.	1/3	D	✓	✗	$O(\lambda n^2 \log n)$	$O(n)$	bAVSS-PO	DLog	CRS & PKI
Freitas et al. [6]	async.	1/3	MC	✗	✓	$O(\lambda n^3 \log n)$	$O(n^3)$	APVSS	DLog	Secure channels
Hashrand [7]	async.	1/3	MC	✓	$^{\sharp}$✓	$^{\ddagger}O(\lambda c n^2 \log n)$	$O(n \log n)$	bAwVSS	pRO	$^{\natural}$Secure channels
RandShare [8]	async.	1/3	LV	✗	✗	$O(\lambda n^3)$	$O(n^3)$	PVSS	DLog	DKG
AsyRand [9]	async.	1/3	LV	✗	✗	$O(\lambda n^2)$	$O(n)$	PVSS	DLog	BRB
Rubato (This work)	async.	1/3	LV	✓	$^{\diamond}$✓✓	$O(\lambda n^3 \log n)$	$^{\star}O(n^2)$	bAVSS-PQ	SIS	$^{\natural}$CRS & PKI

TABLE 1: Comparison of VSS-based randomness beacon protocols. †RandFlash's $O(n \log n)$ is the general case, with a worst case of $O(n^2)$. ‡HashRand's $O(\lambda c n^2 \log n)$ uses an AnyTrust sample of size c; with deterministic consensus, it becomes $O(\lambda n^3 \log n)$. $^{\sharp}$HashRand uses a programmable Random Oracle (pRO), with post-quantum security reliant on the pRO, achieving computational security. $^{\flat}$Spurt elects a leader every n beacons, amortizing $O(n^2)$ computation to $O(n)$. $^{\diamond}$✓✓ indicates provably post-quantum secure. $^{\natural}$Rubato and HashRand additionally assume reliable broadcast [10] for Merkle root and DAG vertex dissemination. *Verification complexity is $O(n \log n)$ **Complexity**: Complexities reflect per-beacon generation. **Termination Flavor**: Deterministic (D) beacons terminate in deterministic time under synchronous or partially synchronous networks; in asynchronous networks, Monte-Carlo (MC) beacons terminate deterministically but may output inconsistent beacons with small probability (adjustable); Las Vegas (LV) beacons ensure consistency without deterministic termination.

lacking deterministic termination [9], [39], [40].

Moreover, the quantum vulnerability of VSS-based DRBs, coupled with their other limitations, underscores the urgent need to address emerging quantum threats. Historically, DRBs relied on cryptographic assumptions robust against classical adversaries, but the rise of quantum computing introduces severe risks. Shor's algorithm solves discrete logarithm problems in polynomial time ($\mathcal{O}((\log N)^3)$) using $\mathcal{O}(\log N)$ qubits, rendering schemes based on threshold signatures and most VSS protocols vulnerable [27], [28], [34], [41]. Most VSS-based DRBs rely on assumptions like Bilinear Diffie-Hellman or Discrete Logarithm, which are susceptible to quantum attacks, compromising long-term security in asynchronous networks where unbounded delays exacerbate this risk. Advances like IBM's 127-qubit Eagle processor highlight the growing feasibility of quantum attacks [42]. In contrast, classical algorithms like the General Number Field Sieve require subexponential time, making quantum threats imminent [43]. NIST's post-quantum standardization underscores the need for quantum-resistant cryptography [44]. Traditional DRBs, designed when quantum computing was not a practical concern, are unprepared for these emerging threats.

To mitigate these quantum vulnerabilities, post-quantum VSS schemes have emerged, but their limitations reveal the need for a more scalable and efficient approach. Early lattice-based VSS schemes, grounded in Learning With Errors (LWE) and Short Integer Solution (SIS) problems, suffered from inefficient batch processing, high communication costs, and large proof sizes, with most lacking batching support [45], [46], [47]. Lightweight random oracle-based schemes like bAwVSS [7] support batching but lack provable security without Quantum Random Oracle Model assumptions [48], [49], [50]. These limitations highlight the need for scalable, batch-capable lattice-based VSS to ensure robust post-quantum security [51].

These challenges—quantum vulnerabilities in VSS-based DRBs, low beacon generation rates due to limited batching, and circular dependencies in asynchronous networks—underscore the need for a DRB that combines post-quantum security, high-throughput batching, and robust operation in asynchronous networks. Addressing these requirements is essential to enable secure and scalable randomness generation for decentralized systems facing diverse network conditions and emerging cryptographic threats.

1.1. Our Contribution

We propose **Rubato**, a post-quantum secure distributed randomness beacon tailored for asynchronous networks. Rubato integrates a lattice-based batched Asynchronous Verifiable Secret Sharing (bAVSS-PQ) scheme, based on the SIS problem, with a high-throughput DAG-based consensus protocol. Leveraging an SIS-based polynomial commitment scheme [52], Rubato ensures provable post-quantum security and supports efficient batching of $\mathcal{O}(\lambda^2)$ secrets with $\mathcal{O}(\lambda n^3 \log n)$ communication complexity. Its epoch-staggered design for the collaboration of DRB protocol with DAG, certificate numbering, and waiting mechanisms resolve circular dependencies and prevent deadlocks, achieving liveness with negligible failure probability. Table 1 compares Rubato with other VSS-based protocols. This paper

is organized as follows: system model, protocol design, security analysis, performance evaluation, discussion, and conclusion.

2. System Model

This section formalizes the system model, threat model, and problem definition for asynchronous distributed randomness beacon protocol.

2.1. Network and Computational Model

We consider a distributed system comprising n nodes, denoted $\mathcal{N} = \{1, 2, \ldots, n\}$. The network is *asynchronous*, with no upper bound on message delivery delays, allowing messages to arrive out of order. However, messages between honest nodes are guaranteed to be delivered eventually through reliable broadcast and point-to-point communication.

Nodes may exhibit heterogeneous computational capabilities, affecting the performance of cryptographic operations, such as proof generation in bAVSS-PQ. We assume sufficient network bandwidth to support reliable broadcast with negligible message loss probability. The system operates under a probabilistic polynomial-time (PPT) adaptive adversary, detailed in the threat model.

The protocol relies on the *Short Integer Solution (SIS)* problem [53] for constructing post-quantum secure polynomial commitments in bAVSS-PQ, ensuring resistance against quantum adversaries. No additional cryptographic assumptions are required unless explicitly stated for auxiliary components (e.g., Fiat-Shamir heuristic challenges).

2.2. Threat Model

An adaptive PPT adversary $\mathcal{A}$ may corrupt up to $f < n/3$ nodes, dynamically choosing which nodes to corrupt at any point during the protocol's execution, controlling their actions, and accessing their internal states. The remaining $n - f \geq 2n/3$ nodes are honest, strictly adhering to the protocol. The adversary can:

- Introduce arbitrary delays or reorder messages, subject to the constraint that messages between honest nodes are eventually delivered.
- Perform polynomial-time quantum computations, necessitating post-quantum cryptographic primitives.
- Coordinate Byzantine nodes to send inconsistent messages or deviate from the protocol.

The adversary aims to compromise Protocol's security properties, including unpredictability, bias-resistance, agreement, and liveness.

2.3. Distributed Randomness Beacon Definition

A distributed randomness beacon (DRB) protocol operates among n nodes in an asynchronous network, producing a sequence of tuples $\langle e, i, b_{e,i} \rangle$, where:

- $e \in \mathbb{N}$ is the epoch index, representing a batch processing period.
- $i \in \{1, 2, \ldots, \theta\}$ is the intra-epoch index, with $\theta \leq \kappa m$ as the batch size per epoch, where κm is the parameter in the commitment scheme.
- $b_{e,i} \in \mathcal{D}$ is a random value from a predefined domain $\mathcal{D}$.

Each epoch involves a single invocation of a secret sharing scheme to share a batch of θ secrets, with a subset of beacons reconstructed sequentially ($i \in \{1, 2, \ldots, B\}$, $B \leq \theta$) within the epoch, supporting the consensus protocol's leader election. The remaining beacon outputs ($i \in \{B+1, \ldots, \theta\}$) are available for external applications.

A DRB protocol comprises two subprotocols:

- PREP$(e, \mathbf{s})$: Invoked by each node to share a batch of random numbers $\mathbf{s} = \{s_1, s_2, \ldots, s_\theta\}$ for epoch e, generating certificates $\mathcal{C}_e$, each consisting of $2f + 1$ signatures from nodes verifying the shared secrets, for consensus via the underlying consensus protocol.
- OPEN(e, i): Invoked to reconstruct and output the beacon $b_{e,i}$ for index i in epoch e, based on consensus-agreed $\{\mathcal{C}_e\}$.

A DRB satisfies the following properties, ensuring Las Vegas-style agreement (guaranteed consistency with probabilistic termination) despite asynchronous delays and batch processing:

- **Unpredictability**: For any future beacon $\langle e'', i'', b_{e'',i''} \rangle$ (where $e'' > e'$ or $e'' = e'$ and $i'' > i'$ relative to prior beacons $\langle e, 1, b_{e,1} \rangle, \ldots, \langle e', i', b_{e',i'} \rangle$), no PPT adversary $\mathcal{A}$ can predict $b_{e'',i''}$ with probability exceeding negl(λ):

$$\Pr\left[b'_{e'',i''} = b_{e'',i''} \mid \langle e, 1, b_{e,1} \rangle, \ldots, \langle e', i', b_{e',i'} \rangle\right] \leq \text{negl}(\lambda),$$

  where $b'_{e'',i''}$ is $\mathcal{A}$'s guess.
- **Bias-resistance**: Each bit $b_{e,i}(k)$ (for $k \in \{1, \ldots, |b_{e,i}|\}$) of a beacon $b_{e,i}$ is indistinguishable from a uniform random bit. For any PPT adversary $\mathcal{A}$ with access to prior beacons, the probability of predicting $b_{e,i}(k)$ deviates from $\frac{1}{2}$ by at most negl(λ):

$$\left| \Pr\begin{bmatrix} b_{e,i}(k) = 1 \mid \langle e, 1, b_{e,1} \rangle, \\ \ldots, \langle e, i-1, b_{e,i-1} \rangle \end{bmatrix} - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

- **Agreement**: All honest nodes outputting $\langle e, i, b_{e,i} \rangle$ agree on the same $b_{e,i}$ with probability $1 - \text{negl}(\lambda)$.
- **Liveness**: If all honest nodes invoke PREP$(e, \mathbf{s})$ for epoch e, and then invoke OPEN(e, i), every honest node eventually outputs $\langle e, i, b_{e,i} \rangle$ for each $i \in \{1, \ldots, \theta\}$ with probability $1 - \text{negl}(\lambda)$.

2.4. Problem Definition

We aim to design an asynchronous distributed randomness beacon protocol, satisfying the properties in Section 2.3, addressing the following challenges:

- **Circular Dependency**: Beacon generation requires the consensus protocol's agreement on $f+1$ bAVSS-PQ certificates $\mathcal{C}$, while the consensus protocol's liveness depends on beacons for leader election. A mechanism must resolve this dependency to ensure protocol termination.
- **Deadlock Prevention**: bAVSS-PQ's certificate generation may lag behind the consensus protocol's epoch progression, risking epoch without available $\{\mathcal{C}\}$. The protocol must ensure certificate availability to prevent deadlocks.
- **Strong Post-quantum Security**: The protocol must employ strong primitives to resist quantum adversaries, ensuring long-term security for beacon generation.
- **High throughput Beacon Generation**: The protocol must support high-throughput generation of beacons per epoch, with B beacons for the consensus protocol's waves and $\theta - B$ for external applications.

TABLE 2: Notations for Rubato.

Notation	Description
n	Number of nodes.
f	Maximum Byzantine faults, $f < n/3$.
t	Polynomial degree, set to f.
λ	Security parameter.
$\mathcal{C}_e$	Certificate for epoch e, $2f+1$ signatures.
$\mathrm{pk}_i, \mathrm{sk}_i$	Node i's key pair in bAVSS-PQ.
θ	Max secrets per epoch.
$\{s_j(\cdot)\}_{j\in[\theta]}$	Polynomials with the secret as the constant term.
$s_{d,j}$	j-th secret shard from dealer d.
$\mathbf{A}$	Random Matrix in $\mathbb{Z}_q^{m\times L\log q}$ for SIS.
$\mathbf{G}$	Gadget matrix in $\mathbb{Z}_q^{L\times L\log q}$.
$\mathbf{f}$	Coefficient vector for polynomials.
$\mathbf{t}$	Commitment vector from PC.Commit.
$\mathbf{u}_i$	Evaluation vector, $\mathbf{u}_i[j] = s_j(i)$.
π_i	Proof for node i, $\{(\mathbf{y}_k, \mathbf{v}_k)\}_{k\in[0,\ell]}$.
$\mathbf{x}_j$	Evaluation point vector in j-th iteration.
m	Lattice matrix $\mathbf{A}$ row dimension.
ℓ	Depth of r-ary tree in PC.
r	Branching factor in PC tree.
κ	Statistical security parameter.
q	Modulus for $\mathbb{Z}_q$ in lattice.
β	Short vector norm bound in PC.
L	Coefficient vector length, $L = r^{\ell+1}\kappa m$.
$\otimes$	Kronecker product.
B	Beacons for the consensus protocol, $B \le \theta$.
$\mathcal{D}$	Beacon value domain.
$b_{e,i}$	Beacon for epoch e,index i.

Our objective is to develop Rubato, integrating bAVSS-PQ with a post-quantum secure consensus protocol (e.g., DAG-based Bullshark [23], Tusk [22]), achieving post-quantum security, high throughput (comparable to Bullshark's 125,000 transactions per seconds), and robust liveness in asynchronous networks. The protocol must ensure negligible failure probabilities and support high-throughput beacon generation scenarios.

3. Protocol Design

This section presents Rubato, our proposed protocol that integrates a batched asynchronous verifiable secret sharing (bAVSS-PQ) scheme with a DAG-based consensus protocol to construct a post-quantum secure, high-throughput distributed randomness beacon. To illustrate its operation, we describe the workflow of Rubato in Fig. 1. Table 2 summarizes the notations used throughout this paper, covering network parameters, protocol variables, and cryptographic constructs for all components.

We elaborate on Rubato in three subsections: Subsection 3.1 describes the Beacon Protocol, Subsection 3.2 elaborates on the bAVSS Workflow, and Subsection 3.3 details how to integrate DAG Consensus Mechanism to our beacon protocol.

3.1. Beacon Protocol

Rubato operates in epochs, using bAVSS-PQ to share and reconstruct secret batches and the DAG-based consensus protocol to agree on a common core of certificates. An epoch-based structure decouples random number generation from consensus, while a certificate epoch-numbering mechanism and waiting strategy prevent deadlock due to mismatched bAVSS-PQ and consensus speeds.

The protocol's workflow, formalized in Algorithm 1, is event-driven, handling certificate generation, consensus decisions, and beacon requests asynchronously. We describe its workflow through three core *functional components*: *Secret Sharing*, *Certificate Agreement*, and *Beacon Generation*. Each component responds to specific events, and their interactions produce random beacons across epochs. While a single beacon's lifecycle logically progresses from sharing to agreement to reconstruction, the system processes multiple epochs and requests in a pipelining manner. Below, we detail each component and its role in the protocol.

- **Secret Sharing:** At the start of the protocol, each node, acting as a dealer, calls Rubato.PREP$(0, \mathbf{s})$ to share a batch of $\theta \le \kappa m$ secrets chosen randomly among $\mathbb{Z}_q$ for epoch $e = 0$ (Line 2), enabling beacon and global coin for the first DAG epoch ($e = 1$). Upon receiving a certificate $\mathcal{C}_e$ (containing $2f+1$ signatures) gathered from bAVSS-PQ, the node relays it to the DAG-based consensus protocol for agreement, or to an auxiliary consensus mechanism for epoch $e = 0$ (Lines 3–4), since DAG has not been booted yet.
- **Certificate Agreement:** Upon receiving a decided certificate $\mathcal{C}_e$ from consensus for epoch e, the node stores it and checks if $f+1$ certificates in this epoch have been collected (Lines 5–6). Once this threshold is met, epoch e is marked as decided, and

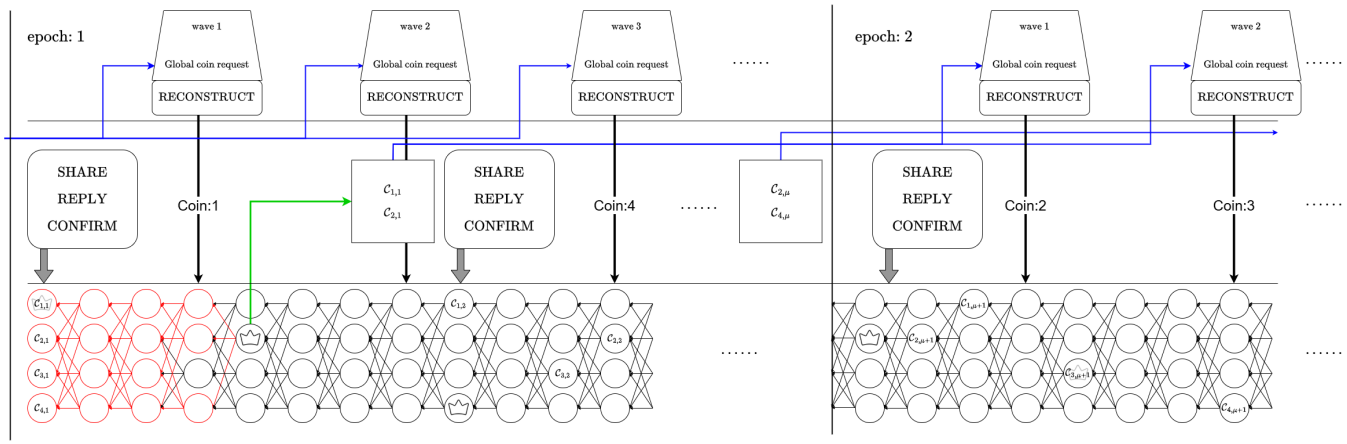


Figure 1: Overview of a node's protocol view in Rubato, abstracted into three conceptual layers for clarity. The bottom layer, illustrated with a 4-round-per-wave DAG, handles consensus, where nodes submit bAVSS certificates $\mathcal{C}_{d,e}$ (dealer d, epoch e) via vertices. The middle layer performs bAVSS (share, reply, confirm) and certificate consensus, committing μ sets of $f+1$ $\mathcal{C}$ per epoch. The top layer reconstructs randomness using the prior epoch's $\mathcal{C}$, in each wave's last round, selecting the leader vertex from the first round via global coin requests.

Algorithm 1 Rubato: Asynchronous Distributed Randomness Beacon Protocol

1: **Initialize:** n nodes, $f < n/3$, epoch $e \leftarrow 0$, $\theta \leq \kappa m$ beacons/epoch, $B \leq \theta$ for DAG consensus waves
2: *Dealer* d: PREP$(0, \mathbf{s})$: Start bAVSS-PQ SHARING for epoch $e = 0$ with θ random numbers $\{s_{d,j}\}_{j\in[\theta]} \in \mathbb{Z}_q$ ▷ Initiate secret sharing for epoch 0 to support DAG epoch 1
3: **Upon** receiving certificate $\mathcal{C}_e$ from bAVSS-PQ for epoch e ▷ Handle certificate generated by bAVSS-PQ
4: Send $\mathcal{C}_e$ to the DAG-based consensus protocol (or auxiliary consensus if $e = 0$) ▷ Forward certificate for agreement; use auxiliary consensus for $e = 0$
5: **Upon** receiving decided $\mathcal{C}_e$ from consensus protocols for epoch e ▷ Process decided certificate from consensus
6: Store $\mathcal{C}_e$ for epoch e; if $f+1$ $\mathcal{C}_e$ collected, mark e as decided ▷ Ensure $f+1$ certificates for epoch decision
7: PREP$(e+1, \mathbf{s})$: Start bAVSS-PQ SHARING for epoch $e+1$ with $\{s_{d,j}\}_{j\in[\theta]} \in \mathbb{Z}_q$ ▷ Trigger sharing for next epoch
8: **Upon** calling OPEN(e, i) ▷ Handle global coin or beacon request for epoch e using epoch $e-1$ certificates
9: **if** $e-1$ not decided **then**
10: Reply: "No common core" ▷ Indicate epoch $e-1$ not ready
11: **else**
12: Return cached beacon or request bAVSS-PQ $\langle \text{RECON}, \mathcal{C}_{e-1}, i\rangle$ for $f+1$ decided $\mathcal{C}_{e-1}$
13: **end if**
14: **Upon** receiving $f+1$ reconstructed secrets $\{s_{d,i}\}_{d\in\{\text{dealers of}(\{\mathcal{C}_{e-1}\})\}}$ for $(e-1, i)$ from bAVSS-PQ
15: Cache $\sum_d s_{d,i}$ as beacon $b_{e,i}$ ▷ Aggregate $f+1$ secrets to form beacon
16: Send $b_{e,i}$ to the consensus protocol if $i \leq B$, else to consumer

Rubato.PREP for epoch $e+1$ is triggered to maintain protocol progress (Line 7).

- **Beacon Generation:** This component produces randomness in response to Rubato.OPEN(e, i) from the consensus protocol (for global coins) or external consumers, using certificates from epoch $e-1$. If epoch $e-1$ is not decided, the node replies "No common core" (Lines 9–10); otherwise, it returns a cached beacon $b_{e,i}$ or initiates reconstruction by requesting bAVSS-PQ with decided certificates $\{\mathcal{C}_{e-1}\}_{f+1}$ (Line 12). The node aggregates $f+1$ reconstructed secrets $\{s_{d,i}\}_{d\in\{\text{dealer of}(\mathcal{C}_{e-1})\}}$ from bAVSS-PQ to form the beacon $b_{e,i}$, which is cached and sent to the consensus protocol (if $i \leq B$) or the consumer (Lines 14–16). Actually, to optimize computational efficiency, we concatenate g beacons, denoted as $\hat{b}_{e,j}$, and apply a hash function to produce the final output: $b_{e,i} = H(\hat{b}_{e,(i-1)g+1}, \ldots, \hat{b}_{e,ig})$, where the modulus q satisfies $\log q = \lambda/g$. The security of this mechanism is shown in Section 4.

The Rubato protocol resolves the circular dependency between beacon generation and consensus by using epoch $e-1$ certificates to produce beacons for epoch e. Beacon generation for wave (e, i) relies on epoch $e-1$'s decided certificates (Lines 11–13), while the consensus protocol uses these beacons for liveness (e.g., fallback leader election in

Bullshark). For the initial epoch $e = 1$, epoch $e = 0$ certificates are decided via an auxiliary consensus mechanism, since the DAG is not booted yet. Although the communication complexity of the auxiliary consensus is typically high (e.g., [54]), it is executed only once. This epoch-offset design maintains the logical progression of a single beacon's lifecycle (sharing in epoch $e-1 \rightarrow$ agreement in epoch $e-1 \rightarrow$ generation for epoch e) while enabling concurrent processing of multiple epochs to support high throughput.

3.2. Batched Asynchronous Verifiable Secret Sharing

Algorithm 2 bAVSS-PQ: Post-quantum Batched AVSS Protocol

1: **Parameters:** $n \geq 3t+1$, public keys $\{\mathrm{pk}_i\}_{i\in[n]}$, polynomial commitment scheme PC, reliable broadcast channel RBC
2: **Input:** Dealer d's secret set $S_d = \{s_{d,j}\}_{j\in[\theta]}$, node i's signing key sk_i
3: **SHARING PHASE**
4: *Dealer* d: Sample θ t-degree polynomials $\{s_j(\cdot)\}_{j\in[\theta]}$ with $s_j(0) = s_{d,j}$
5: $\mathbf{t} \leftarrow \mathrm{PC.Commit}(s_j(\cdot))$ for $j \in [\theta]$ ▷ Commit to polynomials
6: **for** $i \in [n]$ **do**
7: $(u_i, \pi_i) \leftarrow \mathrm{PC.Open}(i)$ ▷ Generate shares and proofs
8: Build Merkle tree over $\{u_{k,j}\}_{k\in[n]}$
9: Get $\mathrm{root}_{L,j}$, $\mathrm{proof}_{i,j}$ for $j \in [\theta]$
10: Send $\langle \mathrm{SHARE}, \mathbf{t}, u_i, \pi_i, \{\mathrm{proof}_{i,j}\}_{j\in[\theta]}\rangle$ to node i
11: **end for**
12: Broadcast $\{\mathrm{root}_{L,j}\}_{j\in[\theta]}$ via RBC ▷ Ensure share consistency
13: **REPLY PHASE**
14: *Node* i: On $\langle \mathrm{SHARE}, \mathbf{t}, u_i, \pi_i, \{\mathrm{proof}_{i,j}\}_{j\in[\theta]}\rangle$, $\mathrm{root}_{L,j}$:
15: **if** $\mathrm{PC.Verify}(\mathbf{t}, i, u_i, \pi_i) = 1$
16: and $\mathrm{Merkle.Verify}(\mathrm{root}_{L,j}, u_{i,j}, \mathrm{proof}_{i,j}) = 1$, $\forall j \in [\theta]$ **then**
17: Send $\langle \mathrm{REPLY}, \mathrm{sign}(\mathrm{sk}_i, \mathbf{t})\rangle$ to dealer d
18: **end if**
19: **CONFIRM PHASE**
20: *Dealer* d: Collect $2t+1$ valid signatures into $\mathcal{C}$, output $\mathcal{C}$
21: **RECONSTRUCTION PHASE**
22: *Node* i: On $\langle \mathrm{RECON}, \mathcal{C}, j\rangle$
23: Select shares $\{u_{i,j}\}$ from dealer d of $\mathcal{C}$ ▷ Use shares from certified dealer
24: Broadcast $\langle \mathrm{RECON}, d, j, u_{i,j}, \mathrm{proof}_{i,j}\rangle$
25: **Upon** receiving $\langle \mathrm{RECON}, d, j, u_{k,j}, \mathrm{proof}_{k,j}\rangle$ from node k
26: **if** $\mathrm{Merkle.Verify}(\mathrm{root}_{d,j}, u_{k,j}, \mathrm{proof}_{k,j}) = 1$ **then**
27: Collect $u_{k,j}$ ▷ Verify and store valid share
28: **end if**
29: Interpolate $s_{d,j}$ from $t+1$ valid shares $u_{k,j}$

We present the workflow of our batched asynchronous verifiable secret sharing (bAVSS-PQ) scheme, the cryptographic core of Rubato. bAVSS-PQ enables robust secret sharing with strong commitment, supporting up to $\theta \leq \kappa m$ secrets per batch with deterministic outputs.

The bAVSS-PQ workflow, formalized in Algorithm 2, operates in four phases: *sharing*, where the dealer disseminates secret shares; *reply*, where nodes validate shares; *confirm*, where the dealer collects signatures; and *reconstruction*, where nodes recover secrets.

3.2.1. bAVSS-PQ Workflow. Algorithm 2 outlines the bAVSS-PQ protocol, inspired by the structure of Rondo's Breeze. It proceeds as follows:

- **Sharing Phase (Lines 3–12):** The dealer d samples θ t-degree polynomials $\{s_j(\cdot)\}_{j\in[\theta]}$ with constant terms $s_{d,j}$, commits to them using the polynomial commitment scheme (PC) to produce commitment $\mathbf{t}$, and generates evaluation shares $u_{i,j} = s_j(i)$ and proofs π_i for each node i. To ensure share consistency, a Merkle tree is built over shares, with roots $\{\mathrm{root}_{L,j}\}_{j\in[\theta]}$ broadcast via a reliable broadcast channel (RBC). Each node i receives $\langle \mathrm{SHARE}, \mathbf{t}, u_i, \pi_i, \{\mathrm{proof}_{i,j}\}_{j\in[\theta]}\rangle$ and the corresponding roots.
- **Reply Phase (Lines 13–18):** Upon receiving a share message and Merkle roots, node i verifies the polynomial commitment using $\mathrm{PC.Verify}$ and checks Merkle proofs. If valid, it signs $\mathbf{t}$ with its secret key sk_i and sends a reply to the dealer, enforcing strong commitment.
- **Confirm Phase (Lines 19–20):** The dealer collects $2t+1$ valid signatures to form a certificate $\mathcal{C}$, which serves as validation data proving the secret's fixation. The certificate is output for consensus in Rubato (Algorithm 1 Line 3).
- **Reconstruction Phase (Lines 21–29):** Upon a reconstruction request for secret $s_{d,j}$, node i selects share $u_{i,j}$ from the dealer who proposed $\mathcal{C}$, and broadcasts it with Merkle proof. Nodes collect $t+1$ valid shares, verified via Merkle proofs, and reconstruct $s_{d,j}$ using Lagrange interpolation.

3.2.2. Polynomial Commitment Workflow. Building upon the framework of [52], Algorithm 3 presents a lattice-based non-interactive polynomial commitment scheme tailored for our bAVSS-PQ protocol. In contrast to discrete logarithm-based commitments, this scheme leverages the hardness of the Shortest Integer Solution (SIS) problem to achieve post-quantum security with transparent setup. The scheme supports the commitment of $\theta \leq \kappa m$ univariate polynomials, each of degree at most $t+1 \leq r^{\ell+1}$, represented as coefficient vectors $\mathbf{f} \in \mathbb{Z}_q^{r^{\ell+1}\kappa m}$. For succinct proofs, it employs FRI-style folding techniques [55], as adopted by [52], which extend the folding approach of Bulletproofs [56] to the lattice setting through lattice homomorphism. The workflow of the scheme is detailed below:

- **Commitment (Lines 4–12):** The dealer transforms θ t-degree polynomials into the coefficient vector $\mathbf{f}$ using TRANSFORMPOLYNOMIALS (Algorithm 3 Line 5), padding with zeros to length $r^{\ell+1}\kappa m$ if $\theta < \kappa m$ or $t+1 < r^{\ell+1}$. The $\mathrm{PC.Commit}$ function recursively folds the input through ℓ levels, starting with the initial $\mathbf{f}$. In each level, it computes a short

Algorithm 3 Polynomial Commitment Scheme for bAVSS-PQ

1: **Public Parameters:** $m, \ell, r, \kappa, q, \beta$, $\mathbf{A} \in \mathbb{Z}_q^{m \times L \log q}$, $\mathbf{G} \in \mathbb{Z}_q^{L \times L \log q}$, $L = r^{\ell+1}\kappa m$, $H : \{0,1\}^* \to \{0,1\}^{r\kappa \times \kappa}$
2: **Notations:** Batch size θ, $\mathbf{f} \in \mathbb{Z}_q^L$, $\mathbf{t} \in \mathbb{Z}_q^{\kappa m}$, $\mathbf{u}_i \in \mathbb{Z}_q^{\kappa m}$, $\pi_i = \{(\mathbf{y}_k, \mathbf{v}_k)\}_{k\in[0,\ell]}$, $\mathbf{x}_j \in \mathbb{Z}_q^r$
3: ---
4: **Algorithm** PC.Commit$(\{s_j(\cdot)\}_{j\in[\theta]})$ *Output:* $\mathbf{t}$
5: $\mathbf{f} \leftarrow$ TRANSFORMPOLYNOMIALS$(\{s_j(\cdot)\}_{j\in[\theta]})$, $\{\mathbf{s}^{(i)} \leftarrow \text{None}\}_{i\in[0,\ell]}$ ▷ Initialize coefficient vector and short vectors
6: **function** FOLDCOMMITMENT($\mathbf{f}, depth, \{\mathbf{s}^{(i)}\}$)
7: **if** $depth < 0$ **then return** $\mathbf{f}$
8: **end if**
9: $\mathbf{s} \leftarrow$ solution to $\mathbf{G} \cdot \mathbf{s} = \mathbf{f} \mod q$ with $\|\mathbf{s}\|_\infty \le \beta$, $\mathbf{s}^{(depth)} \leftarrow \mathbf{s}$, $\mathbf{t} \leftarrow (\mathbf{I}_{r^{depth}\kappa} \otimes \mathbf{A}) \cdot \mathbf{s} \mod q$ ▷ Compute folded commitment
10: **return** FOLDCOMMITMENT($\mathbf{t}, depth - 1, \{\mathbf{s}^{(i)}\}$) ▷ Recursively fold until depth 0
11: **end function**
12: **return** $\mathbf{t} \leftarrow$ GENERATEPROOF$(\mathbf{f}, \ell, \{\mathbf{s}^{(i)}\})$, Store $\leftarrow (\mathbf{f}, \{\mathbf{s}^{(i)}\}_{i\in[0,\ell]})$ ▷ Store for opening, return commitment
13: ---
14: **Algorithm** PC.Open(i) *Output:* $(\mathbf{u}_i, \pi_i)$
15: $(\mathbf{f}, \{\mathbf{s}^{(k)}\}_{k\in[0,\ell]}) \leftarrow$ Store, $\{\mathbf{x}_j\}_{j\in[0,\ell]} \leftarrow$ TRANSFORMEVALUATIONPOINT(i, ℓ, r) ▷ Retrieve stored data, generate evaluation vectors
16: $\mathbf{u}_i \leftarrow \prod_{j=0}^{\ell}(\mathbf{I}_{r^j\kappa m} \otimes \mathbf{x}_{\ell-j}^\top) \cdot \mathbf{f}$, $\pi_i \leftarrow \emptyset$ ▷ Compute polynomial evaluations $u_{i,j} = s_j(i)$
17: **function** GENERATEPROOF($\mathbf{f}, \{\mathbf{s}^{(k)}\}, \{\mathbf{x}_j\}, \mathbf{t}, \mathbf{u}_i, \text{depth}$)
18: **if** depth $= \ell$ **then**
19: $\pi_i \leftarrow \pi_i \cup \{(\mathbf{s}^{(0)}, \mathbf{f})\}$, **return** ▷ Final round
20: **end if**
21: $\mathbf{y} \leftarrow \mathbf{s}^{(0)}, \mathbf{v} \leftarrow \prod_{k=1}^{\ell-\text{depth}}(\mathbf{I}_{r^k\kappa m} \otimes \mathbf{x}_{\ell-\text{depth}-k}^\top) \cdot \mathbf{f}$, $\pi_i \leftarrow \pi_i \cup \{(\mathbf{y}, \mathbf{v})\}$ ▷ Compute proof components
22: $\mathbf{C} \leftarrow H(\mathbf{t}, \mathbf{u}_i, \{\mathbf{x}_j\}_{j=0}^{\ell-\text{depth}}, \mathbf{y}, \mathbf{v})$ ▷ Fiat-Shamir challenge for non-interactivity
23: **for** $k \in [0, \ell - \text{depth} - 1]$ **do**
24: $\mathbf{s}^{(k)} \leftarrow (\mathbf{C}^\top \otimes \mathbf{I}_{r^{k+1}m\log q}) \cdot \mathbf{s}^{(k+1)}$ ▷ Update short vectors via folding
25: **end for**
26: $\mathbf{t}' \leftarrow (\mathbf{C}^\top \otimes \mathbf{I}_m) \cdot \mathbf{G}_{r\kappa m} \cdot \mathbf{y}, \mathbf{u}_i' \leftarrow (\mathbf{C}^\top \otimes \mathbf{I}_m) \cdot \mathbf{v}, \mathbf{f} \leftarrow (\mathbf{C}^\top \otimes \mathbf{I}_{r^{\ell-\text{depth}}m}) \cdot \mathbf{f}$ ▷ Compute new folded commitment and evaluation
27: GENERATEPROOF($\mathbf{f}, \{\mathbf{s}^{(k)}\}, \{\mathbf{x}_j\}, \mathbf{t}', \mathbf{u}_i', \text{depth} + 1$) ▷ Recurse to next depth
28: **end function**
29: GENERATEPROOF($\mathbf{f}, \{\mathbf{s}^{(k)}\}, \{\mathbf{x}_j\}, \mathbf{t}, \mathbf{u}_i, 0$) **return** $(\mathbf{u}_i, \pi_i)$ ▷ Generate and return proof
30: ---
31: **Algorithm** PC.Verify$(\mathbf{t}, \mathbf{u}_i, i, \pi_i)$ *Output:* Boolean
32: $\{\mathbf{x}_j\}_{j\in[0,\ell]} \leftarrow$ TRANSFORMEVALUATIONPOINT(i, ℓ, r)
33: **function** VERIFYPROOF($\mathbf{t}, \mathbf{u}_i, \{\mathbf{x}_j\}, \pi_i, \text{depth}$)
34: $(\mathbf{y}_{\text{depth}}, \mathbf{v}_{\text{depth}}) \leftarrow \pi_i$ ▷ Extract proof components
35: **if** $\|\mathbf{y}_{\text{depth}}\|_\infty > (r\kappa)^{\text{depth}}\beta$ or $(\mathbf{I}_\kappa \otimes \mathbf{A}) \cdot \mathbf{y}_{\text{depth}} \neq \mathbf{t}$ or $(\mathbf{I}_{\kappa m} \otimes \mathbf{x}_{\ell-\text{depth}}^\top) \cdot \mathbf{v}_{\text{depth}} \neq \mathbf{u}_i$ **then**
36: **return** false ▷ Check norm, commitment, and evaluation
37: **end if**
38: **if** depth $= \ell$ **then return** true ▷ Accept in final round
39: **end if**
40: $\mathbf{C} \leftarrow H(\mathbf{t}, \mathbf{u}_i, \{\mathbf{x}_j\}_{j=0}^{\ell-\text{depth}}, \mathbf{y}_{\text{depth}}, \mathbf{v}_{\text{depth}})$ ▷ Fiat-Shamir challenge
41: $\mathbf{t}' \leftarrow (\mathbf{C}^\top \otimes \mathbf{I}_m) \cdot \mathbf{G}_{r\kappa m} \cdot \mathbf{y}_{\text{depth}}, \mathbf{u}_i' \leftarrow (\mathbf{C}^\top \otimes \mathbf{I}_m) \cdot \mathbf{v}_{\text{depth}}$ ▷ Compute next folded commitment and evaluation
42: **return** VERIFYPROOF($\mathbf{t}', \mathbf{u}_i', \{\mathbf{x}_j\}_{j=0}^{\ell-\text{depth}-1}, \pi_i, \text{depth} + 1$) ▷ Recurse to next depth
43: **end function**
44: **return** VERIFYPROOF($\mathbf{t}, \mathbf{u}_i, \{\mathbf{x}_j\}, \pi_i, 0$) ▷ Return verification result

vector $\mathbf{s}$ satisfying $\mathbf{G} \cdot \mathbf{s} = \mathbf{f}' \mod q$, where $\mathbf{f}'$ is the output of the previous level (or the initial $\mathbf{f}$ for the first level), and produces a folded commitment $\mathbf{t}$.

- **Opening (Lines 14–29):** For an evaluation point i, PC.Open computes evaluations $\mathbf{u}_i \in \mathbb{Z}_q^{\kappa m}$, where $\mathbf{u}_i[j] = s_j(i)$ for $j \in [\theta]$, and generates a proof π_i. Note that the evaluation point i is transformed into vectors $\mathbf{x}_j \in \mathbb{Z}_q^r$ for $j \in [0, \ell]$ by TRANSFORMEVALUATIONPOINT to support univariate polynomial evaluation. For example, when $\ell = 2$, $r = 3$, the vectors are $\mathbf{x}_0 = (1, i, i^2)$, $\mathbf{x}_1 = (1, i^3, i^6)$, $\mathbf{x}_2 = (1, i^9, i^{18})$. The proof is constructed recursively via FRI-style folding: in each round, the prover sends a short vector $\mathbf{y}$ using the opening $\mathbf{s}^{(\ell-\text{depth})}$ updated in the former round and an evaluation vector $\mathbf{v}$, then computes a fiat-shamir heuristic challenge $\mathbf{C}$. The challenge $\mathbf{C}$ randomly folds the commitment $\mathbf{t}$, evaluation $\mathbf{u}_i$, and coefficient vector $\mathbf{f}$ for the next round. In the final round, the prover sends $\mathbf{s}^{(0)}$ and $\mathbf{f}$.
- **Verification (Lines 31–44):** PC.Verify recursively validates the commitment, evaluation, and norm constraints. For each round, it checks the norm of $\mathbf{y}_{\text{depth}}$,

the commitment consistency $(\mathbf{I}_\kappa \otimes \mathbf{A}) \cdot \mathbf{y}_{\text{depth}} = \mathbf{t}$, and the evaluation correctness $(\mathbf{I}_{\kappa m} \otimes \mathbf{x}_{\ell-\text{depth}}^\top) \cdot \mathbf{v}_{\text{depth}} = \mathbf{u}_i$. It generates the same challenge $\mathbf{C}$ as in $\mathrm{PC.Open}$ with identical inputs, ensuring consistency across rounds. The folded commitment $\mathbf{t}'$ and evaluation $\mathbf{u}_i'$ are computed for the next round.

3.3. DAG-based Consensus

Our Byzantine Fault Tolerance State Machine Replication (BFT-SMR) protocol adapts a Directed Acyclic Graph (DAG)-based consensus protocol to integrate with Rubato. This section provides a comprehensive overview of the consensus mechanism, detailing its operation, and how it ensures the liveness of Rubato. The protocol constructs a DAG to disseminate transactions and certificates, achieving consensus without extra communication beyond DAG construction, as described in [57]. We embed bAVSS-PQ certificates $\mathcal{C}_e$ from Rubato into DAG vertices, leveraging randomness for leader selection while retaining the consensus protocol's optimal $O(n)$ amortized communication complexity, fairness, and liveness.

3.3.1. DAG Structure and Operation. The consensus protocol operates by constructing a DAG, where each vertex represents a message broadcast by a node via reliable broadcast (r_bcast). The DAG is maintained locally by each node p_i as DAG_i, an array of vertex sets jed by round number $r \in \mathbb{N}$, i.e., $DAG_i[r]$ contains vertices associated with round r. The structure and operation of the DAG are as follows:

Vertex Structure: Each vertex v contains:

- *Transactions*: A block of transactions proposed by the source node.
- *Metadata*: The round number ($v.round$), the source node ($v.source$), and, in our adaptation, a bAVSS-PQ certificate $\mathcal{C}_e$ (or $\emptyset$ if none).
- *Edges*: References to parent vertices.

DAG Advancement: As described in Algorithm 4, the DAG progresses in rounds, driven by two key events in Algorithm 4:

- *Vertex Delivery (Lines 8–22)*: When a node p_i receives a valid vertex v via reliable broadcast (r_deliver$_i(v, r, p_j)$), if all referenced parent vertices (strong and weak edges) are in DAG_i, the vertex is added to $DAG_i[r]$; otherwise, it is stored in a buffer. After each addition, the node checks the buffer for vertices whose parents are now available, adding them to DAG_i.
- *Round Advancement (Lines 24–39)*: When $|DAG_i[r]| \geq 2f + 1$, the TryAdvanceRound procedure is triggered, indicating a quorum of vertices in round r. The node checks if the current epoch e has an available certificate ($\mathcal{C}_e$) in the queue. In the first round of every epoch, if the queue is empty and $ec \leq e$ (latest certificate epoch is behind), the node waits until a certificate is enqueued (Line 27). This mechanism ensures the latest certificate will be committed in this epoch. The node then creates a new vertex for round $r+1$, embedding a dequeued $\mathcal{C}_e$ if available, updates ec, and broadcasts the vertex via r_bcast$_i$.

Consensus Mechanism: Consensus is achieved through leader election and vertex ordering:

- *Leader Election*: The DAG is organized into waves, each consisting of four rounds (or two rounds in case of Tusk [22]). Each wave has a leader vertex, selected via a global perfect coin (provided by Rubato's beacons), ensuring liveness and unpredictability. The leader is chosen from vertices in the first round of the wave, with a probability of at least $2/3$ of selecting an honest node's vertex, as there are at least $2f + 1$ honest nodes out of $3f + 1$ total nodes [57].
- *Vertex Ordering*: Once a leader is committed, nodes order its causal history (all vertices reachable via strong or weak paths) deterministically. To ensure total order, nodes retroactively check for prior leaders (in earlier waves) with strong paths from the current leader, committing them in reverse order. This process (invoked after adding vertices, Line 13) guarantees that all honest nodes deliver the same sequence of vertices.

3.3.2. Integration with Rubato. Our adaptation extends the DAG to handle bAVSS-PQ certificates. Key modifications include:

- **Certificate Queue**: Rubato certificates $\mathcal{C}_e$ are enqueued, triggering round advancement when unblocking the queue.
- **Vertex Creation**: New vertices embed a dequeued $\mathcal{C}_e$ if available, updating the latest certificate epoch ec; otherwise, $\mathcal{C}_e = \emptyset$.
- **Synchronization**: If $ec \leq e$ and the queue is empty, the node waits for a certificate, aligning bAVSS-PQ certificate production with DAG progression.

The fast production rate of bAVSS-PQ ensures continuous certificate broadcasts, while slower rates trigger waiting, preserving certificate order. The protocol retains the consensus's liveness, achieving high throughput and fairness, as all honest vertices are eventually ordered.

4. Security Analysis

This section analyzes the security of Rubato against an adaptive probabilistic polynomial-time (PPT) quantum adversary corrupting up to $f < n/3$ nodes in an asynchronous network. Detailed proofs of the following theorems are provided in Appendix.

Theorem 1 (Security of bAVSS-PQ)**.** *The bAVSS-PQ protocol (Algorithm 2) satisfies the following properties:*

Algorithm 4 Modified DAG-based Consensus with Rubato for Party p_i

Local variables:
  $DAG_i[]$, round $\leftarrow 1$, $e \leftarrow 0$, $ec \leftarrow -1$, queue $\leftarrow \emptyset$, buffer $\leftarrow \emptyset$ ▷ DAG, round, epoch, queue, buffer
Struct vertex:
  source, transactions, $\mathcal{C}_e$ ▷ Vertex with certificate

Upon receive $\mathcal{C}_e$ from Rubato for epoch e ▷ Get Rubato certificate
  queue $\leftarrow$ queue $\cup \{\mathcal{C}_e\}$ ▷ Enqueue $\mathcal{C}_e$
Upon $r\_\text{deliver}_i(v, r, p_j)$ ▷ Receive vertex
  if v.source $= p_j \wedge v$.round $= r$ **then**
    If v's parents in DAG_i, add to DAG_i, else to buffer ▷ Add to DAG
    for $v' \in$ buffer **do**
      Check if v''s parents in DAG_i; if yes, add to DAG_i, remove from buffer ▷ Retry and order
      Order Vertex v' ▷ Choose leader via global coin request and call a_deliver(v')
    end for
    if $|DAG_i[r]| \geq 2f + 1$ **then**
      TryAdvanceRound ▷ Advance round
    end if
  end if
Upon $a\_\text{deliver}_i(v)$ ▷ Output ordered vertex
  if $v.\mathcal{C}_e \neq \emptyset$ **then**
    Send $\mathcal{C}_e$ to Rubato ▷ Feedback decided certificate
  end if

procedure TryAdvanceRound
  $e \leftarrow$ get_epoch(round) ▷ Update epoch
  if $ec \leq e \wedge is\_first\_round\_of\_epoch(round) \wedge$ queue $= \emptyset$ **then** ▷ Wait for certificate from bAVSS
    Poll queue until non-empty ▷ Wait for certificate
  end if
  round $\leftarrow$ round $+ 1$
  $v \leftarrow$ CreateVertex($round$) ▷ New vertex
  if queue $\neq \emptyset$ **then**
    $\mathcal{C}_e \leftarrow$ queue.dequeue() ▷ Dequeue $\mathcal{C}_e$
    $v.\mathcal{C}_e \leftarrow \mathcal{C}_e$
    $ec \leftarrow$ epoch of $\mathcal{C}_e$ ▷ Update epoch
  else
    $v.\mathcal{C}_e \leftarrow \emptyset$ ▷ Empty certificate
  end if
  $r\_\text{bcast}_i(v, \text{round})$ ▷ Broadcast vertex
end procedure

- ***Liveness****: If the dealer d is honest during the sharing phase, all honest nodes eventually complete the reply phase, a certificate $\mathcal{C}$ is formed; if at least $t + 1$ honest nodes have validated their shares and initiated the reconstruction phase, all honest nodes eventually complete reconstruction, producing some output.*
- ***Strong Commitment****: Upon formation of a valid certificate $\mathcal{C}$ for dealer d, whether it is honest or not, once all honest nodes complete the reconstruction phase, they reconstruct consistent secrets $s_{d,j} \in \mathbb{Z}_q$.*
- ***Privacy****: If the dealer d remains honest throughout the protocol and no honest node initiates reconstruction, an adversary corrupting at most f nodes gains no information about the secret set $\{s_{d,j}\}_{j\in[\theta]}$.*

Theorem 2 (Security of Polynomial Commitment)**.** *The lattice-based polynomial commitment scheme (Algorithm 3) satisfies the following properties for univariate polynomials of degree at most $t + 1 \leq r^{\ell+1}$, with probability at least $1 - negl(\lambda)$:*

- ***Completeness****: For an honest prover committing to polynomials $\{s_j(\cdot)\}_{j\in[\theta]}$ and generating a proof π_i for evaluation at point i, the verification procedure $\text{PC.Verify}(\mathbf{t}, \mathbf{u}_i, i, \pi_i)$ accepts.*
- ***Binding****: Under the Short Integer Solution (SIS) assumption, no probabilistic polynomial-time (PPT) adversary can produce two distinct polynomial sets $\{s_j(\cdot)\}_{j\in[\theta]} \neq \{s'_j(\cdot)\}_{j\in[\theta]}$ that yield the same commitment $\mathbf{t}$.*
- ***Knowledge Soundness****: Under the SIS assumption, no PPT adversary can forge a proof π_i for an evaluation $\mathbf{u}_i$ at point i without knowing the polynomial coefficients, except with soundness error at most $\frac{\ell r \kappa}{2^\kappa} \leq negl(\lambda)$.*
- ***Hiding****: For any polynomial vector $\mathbf{f} \in \mathbb{Z}_q^{r^{\ell+1}\kappa m}$, the commitment $\mathbf{t}$ and proof $\{\pi_i\}$ reveals no information about $\mathbf{f}$, .*

Theorem 3 (Liveness of DAG-based Consensus and Integration)**.** *The consensus protocol (Algorithm 4) and its integration with Rubato satisfy liveness: for each epoch e, honest nodes eventually output $f+1$ certificates $\{\mathcal{C}_e\}$ with probability $1-negl(\lambda)$, provided at least $f+1$ honest nodes invoke PREP$(e, \mathbf{s})$.*

Theorem 4 (Indistinguishability)**.** *For any PPT adaptive adversary $\mathcal{A}$ corrupting at most $f < n/3$ nodes in an asynchronous network with $n \geq 3f+1$, $\mathcal{A}$ cannot distinguish the Rubato beacon output $b_{e,i}$ (Algorithm 1) from a uniformly random element in $\mathbb{Z}_q$ with advantage greater than $negl(\lambda)$.*

Note that Rubato remains secure against an adaptive adversary in the no-erasures model [58], where $\mathcal{A}$ can access the entire tape (i.e., the complete historical state, including all secrets, shares, and messages) of up to f corrupted nodes. The proof shows that even with this access post-sharing, $\mathcal{A}$ gains no advantage. In contrast, HashRand [7] achieves adaptive security only in the secure erasures model.

Theorem 5 (Unpredictability, Bias-resistance, Agreement and Liveness of Beacon Protocol)**.** *The Rubato beacon protocol satisfies all the properties in Section 2.3 with probability $1 - negl(\lambda)$.*

Theorem 6 (Hash-Based Domain Randomness)**.** *Let $H : \mathbb{Z}_q^g \to \mathcal{D}$ be a cryptographic hash function modeled as a random oracle, where q is a prime such that $\log_2 q \approx \lambda/g$, and λ is the security parameter. Let $\hat{b} \in \mathbb{Z}_q$ be independently and uniformly random elements. Then, $b_{e,i} = H(\hat{b}_{e,(i-1)g+1}, \ldots, \hat{b}_{e,ig})$ is indistinguishable from a uniformly random element in $\mathcal{D}$.*

5. Evaluation

We evaluate the performance of our Rubato protocol, integrated with bAVSS-PQ and Consensus protocols, focusing on beacon generation, consensus throughput, and latency. Our experiments compare the post-quantum secure lattice-based scheme against Rondo's non-post-quantum Breeze [5], which relies on discrete logarithm assumptions. We assess scalability across 10, 20, 30, and 50 nodes, measuring beacon output rate, amortized beacon latency, consensus throughput, and latency, alongside bAVSS-PQ's cryptographic overhead.

5.1. Experimental Setup

Our experiments are conducted on Amazon Web Services (AWS) using c4.4xlarge instances (16 vCPUs, 30 GB memory, high network performance) in a geo-distributed setup with nodes deployed across multiple regions to simulate realistic network conditions (ap-northeast-1, us-west-1, eu-north-1, sa-east-1, ca-central-1). We evaluate committee sizes of 10, 20, 30, and 50 nodes. The implementation is in Rust, leveraging `tokio` as the asynchronous runtime. For bAVSS-PQ, we use `DiLithium` for signatures, `rs-merkle` for Merkle trees, `nalgebra` for algebraic operations, and `rayon` for parallel computation, with polynomial commitments implemented in-house. For Breeze, we use `ed25519-dalek` for signatures and `curve25519-dalek` for commitments. Cryptographic overhead measurements for bAVSS-PQ are performed on an Intel i9 CPU (2.50 GHz, 32 GB memory). Forking from Bullshark[1], we have made the source code of our experiments publicly available.[2]

5.2. Beacon Generation Performance

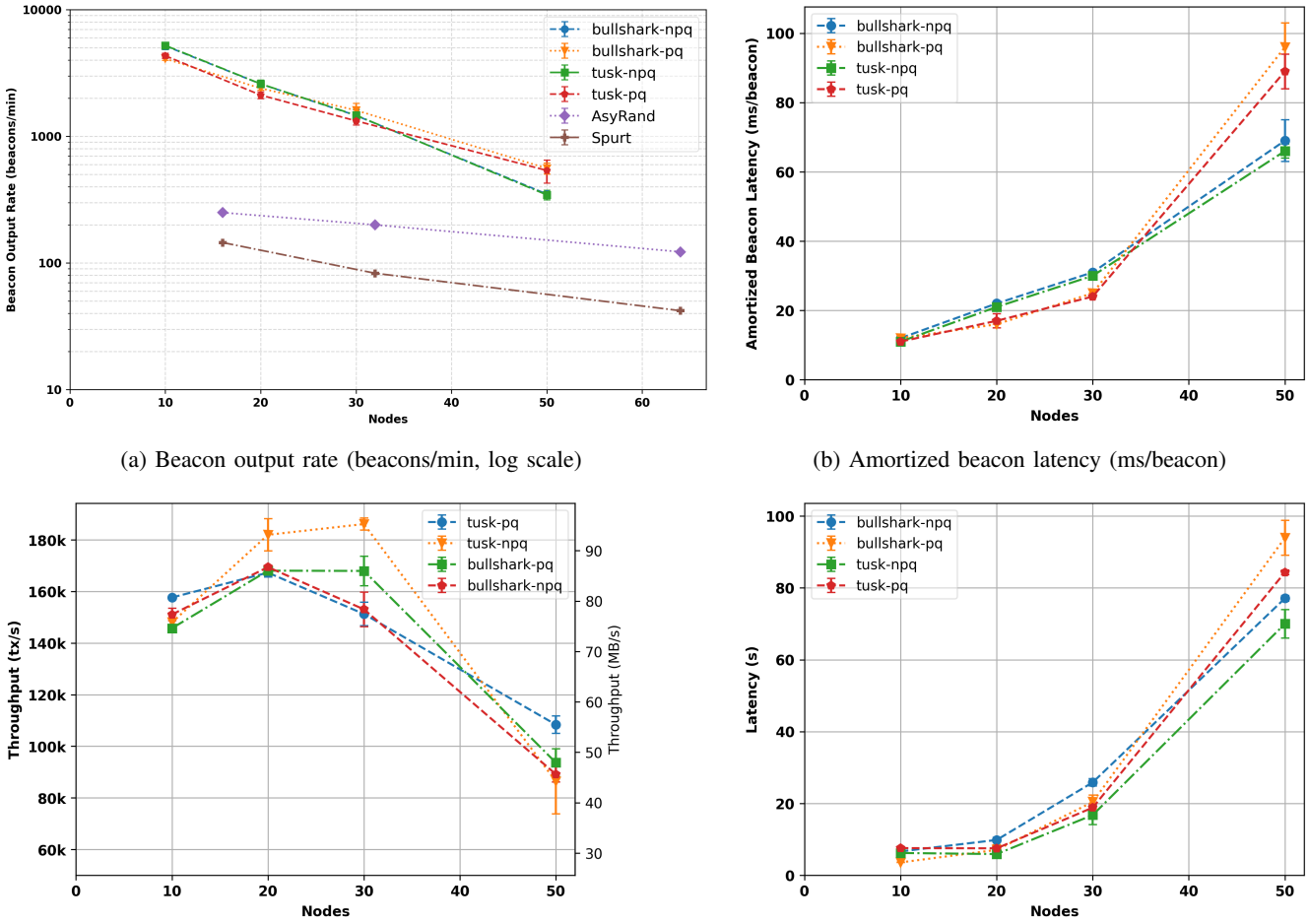
Figure 2a shows the beacon output rate (beacons/min) for our DRB protocol, Rubato, integrated with Bullshark and Tusk, using post-quantum bAVSS-PQ (pq) and non-post-quantum Breeze (npq), compared to baselines AsyRand and Spurt. Rubato achieves 4100–5200 beacons/min at 10 nodes, decreasing to 350–560 beacons/min at 50 nodes due to coordination overhead. Rubato sustains high rates through batched AVSS, efficiently sharing multiple secrets for on-demand reconstruction, with Bullshark and Tusk enabling fast consensus on certificate sets and clients' transactions. The npq variants slightly outperform pq due to Breeze's lower cryptographic overhead (ECDSA vs. DiLithium), but bAVSS-PQ's post-quantum security incurs minimal rate reduction (e.g., 14% for Bullshark at 50 nodes), ensuring robust secure applications. AsyRand and Spurt's lower rates reflect simpler protocols, less suited for high-frequency beacon generation. The rate decline with node count stems from increased communication overhead, particularly for Rubato's $O(\lambda n^3 \log n)$ complexity compared to AsyRand/Spurt's $O(\lambda n^2)$. For larger committees ($n > 100$), AsyRand/Spurt may surpass Rubato due to better scalability.

Figure 2b presents the amortized beacon latency (millisecond/beacon), measuring the time per beacon from sharing to reconstruction for 1232 output beacons (out of 2432 total beacons per epoch). Rubato, using batched AVSS, achieves low latency, with Tusk-pq at 11.60 ms/beacon for 10 nodes, increasing to 89.14 ms/beacon at 50 nodes, and Bullshark-pq reaching 96.37 ms/beacon at 50 nodes. Breeze's npq variants exhibit higher latency, estimated at 76 ms/beacon for Bullshark-npq at 50 nodes. Rubato's bAVSS-PQ, supported by fast Bullshark/Tusk consensus, ensures low latency, ideal for high-frequency beacon generation.

5.3. Consensus Performance

Figures 2c and 2d illustrate the consensus throughput (tx/s) and latency (s) of Bullshark and Tusk, integrated with Rubato's beacon, using bAVSS-PQ (pq) and Breeze (npq), at a fixed input rate of 200,000 tx/s with nodes and workers co-deployed on the same host. The throughput peaks at 20–30 nodes (e.g., Tusk-npq: 186,088 tx/s at 30 nodes), driven by batched AVSS and low beacon latency, but declines at 50 nodes (e.g., Bullshark-pq: 93,628 tx/s) due to increased beacon latency slowing leader commits. Latency is low at small

1. https://github.com/asonnino/narwhal.git.
2. https://github.com/linghe-yang/narwhal.git



(a) Beacon output rate (beacons/min, log scale)

(b) Amortized beacon latency (ms/beacon)

(c) Consensus throughput (tx/s)

(d) Consensus latency (s)

Figure 2: Performance metrics for Beacon and Consensus.

TABLE 3: Performance Metrics of BAVSS-PQ

Metric	Number of Nodes			
	10	**20**	**30**	**50**
Commit Time/Beacon (ms)	0.61	1.63	2.99	4.35
Proof Time/Beacon (ms)	0.32	2.45	7.12	16.17
Verify Time/Beacon (ms)	0.52	0.83	1.20	1.41
Commit Size/Beacon (kB)	0.313	0.625	0.938	1.563
Proof Size/Beacon (kB)	4.125	6.188	8.250	10.313

committee, with Bullshark-pq at 3.59 s for 10 nodes, rising to 93.92 s at 50 nodes. Rubato's efficiency, balancing post-quantum security and performance, suits high-throughput applications.

5.4. bAVSS-PQ Cryptographic Overhead

Table 3 summarizes the performance metrics of bAVSS-PQ across varying node counts, using polynomial commitments with coefficients $n = 128$, security parameter $\kappa = 76$, $\log q = 32$, and reconstruction parameter $g = 4$, producing 2432 beacons per epoch. Polynomial degrees are 3, 8, 15, and 24 for 10, 20, 30, and 50 nodes, respectively. Commit time per beacon increases linearly from 0.61 ms at 10 nodes to 4.35 ms at 50 nodes. Proof generation time exhibits a quadratic increase, growing from 0.32 ms to 16.17 ms, driven by the complexity of polynomial commitments as node count and polynomial degree rise. Verification time scales linearly and remains low, from 0.52 ms to 1.41 ms, demonstrating efficient proof checking. Commit size scales linearly from 0.313 KB to 1.563 KB, and proof size increases from 4.125 KB to 10.313 KB, both remaining manageable for network bandwidth. Compared to Breeze's Bulletproofs-based commitments (e.g., 8 KB proof size at 50 nodes), bAVSS-PQ incurs higher overhead but achieves post-quantum security, a critical advantage for long-term resilience.

6. Discussion

6.1. Comparison with related works

We compare Rubato with three recent DRB protocols—HashRand [7], AsyRand [9], and Rondo [5].

Termination: HashRand employs Monte-Carlo (MC) termination with Approximate Agreement, ensuring fixed-round completion but permitting disagreement with probability $p = 1 - \delta$. This contrasts with AsyRand, which adopts Las Vegas (LV) termination via Byzantine Reliable Broadcast, guaranteeing consistent outputs but at the cost of variable termination times, leading to higher coordination overhead. Building on the LV approach, Rubato resolves circular dependencies through an epoch-staggered design, using secrets from epoch $e - 1$ to generate beacons for epoch e, integrated with DAG-based consensus. This ensures deterministic consistency and liveness with probability $1 - \text{negl}(\lambda)$, avoiding Approximate Agreement's trade-offs. In contrast, Rondo, designed for partially synchronous networks, leverages Global Stabilization Time (GST) and Rondo-BFT (a dynamic Hotstuff [59]) for deterministic termination, sidestepping FLP impossibility but sacrificing robustness in fully asynchronous settings due to GST dependence.

Efficiency: HashRand achieves the state-of-the-art highest beacon rate, exceeding 10^4 beacons/min at 10 nodes and 78 at 136 nodes, driven by fast hash-based commitments and AnyTrust sampling, which reduces communication complexity to $O(\lambda cn^2 \log n)$ by selecting c dealers, though risking unpredictability with probability $\binom{f}{c}/\binom{n}{c}$ when $c < f$. Rubato matches HashRand's performance in small committees but sees faster rate declines in medium-sized committees, due to its $O(\lambda n^3 \log n)$ complexity from deterministic LV termination and batching of $\mathcal{O}(\lambda^2)$ secrets. Rondo's lower rates stem from coupling beacon reconstruction with blockchain consensus. AsyRand, despite the state-of-the-art lowest communication complexity of $O(\lambda n^2)$ in asynchronous settings, has lower beacon rate due to its non-batched PVSS design.

VSS Scheme: HashRand's bAwVSS uses weak commitment, producing either a valid value or a consistent $\perp$, which suffices for DRBs but falls short for applications like multi-party computation requiring strong guarantees. In contrast, Rondo's Breeze employs strong commitment but constructs Merkle trees over polynomial commitments, hindering single-share verification during reconstruction. Addressing this, Rubato's bAVSS-PQ ensures strong commitment by verifying all θ shares in the Reply Phase using polynomial commitment and Merkle proofs, and individual shares during reconstruction, enhancing security despite higher $O(\lambda n^3 \log n)$ complexity.

Post-quantum Security: While AsyRand and Rondo rely on quantum-vulnerable primitives, HashRand mitigates this with hash functions modeled as programmable Random Oracles, offering computational post-quantum security but lacking provable guarantees due to idealized assumptions. Rubato, however, leverages its bAVSS-PQ scheme, grounded in the Short Integer Solution (SIS) problem [53], to provide provably post-quantum security with transparent setups, eliminating trusted initialization and establishing a future-proof foundation compared to HashRand's idealized model or the quantum-susceptible designs of AsyRand and Rondo.

6.2. Limitations of Our Study

Rubato provides robust post-quantum security and high-throughput randomness generation in asynchronous networks but faces scalability limitations, making it best suited for small-to-medium committees.

The primary bottleneck is the computational complexity of the bAVSS-PQ polynomial commitment scheme. The commitment phase, executed once per batch of θ polynomials, has a fixed cost, but the opening phase scales linearly with committee size n. Lattice-based security requires $m = \mathcal{O}(\lambda)$, and knowledge soundness demands $\kappa = \mathcal{O}(\lambda)$. The folding parameters r (multiplicity) and ℓ (depth) determine the maximum polynomial degree $r^{\ell+1}-1$, which must exceed the fault tolerance threshold f. For a 50-node system with $f = 17$, $r = 5$ and $\ell = 1$ suffice, but larger committees require higher r or ℓ, increasing folding operations. Consequently, proof computation scales as $O(n^2)$ due to the linear increase in evaluation points and polynomial degree, with matrix multiplications dominating. This prolongs share generation, and the DAG's waiting mechanism exacerbates delays, reducing consensus throughput. Commitments over Cyclotomic Rings [52] could lower parameters (e.g., $\kappa = \mathcal{O}(1)$) and computational costs, but optimized implementations are not yet mature.

The communication complexity of bAVSS-PQ, at $O(\lambda n^3 \log n)$ due to Merkle tree-based share verification across n nodes, further limits scalability. Reed-Solomon (RS) codes could reduce this to $O(\lambda n^3)$, correcting up to $\lfloor (n-k)/2 \rfloor$ errors (e.g., $k = n - 3f$ for $n \geq 4f + 1$), requiring $n - f$ shares. However, this increases latency in asynchronous networks due to unbounded delays and reduces Byzantine fault tolerance to $n/4$, better suited for synchronous or partially synchronous settings. Alternatively, using Monte-Carlo termination with AnyTrust, as in HashRand, could lower complexity to $O(\lambda cn^2 \log n)$, but sacrifices perfect consistency.

Reconfiguration for dynamic node changes poses another challenge. While bAVSS-PQ can support larger committees by pre-selecting parameters for higher Byzantine fault tolerance, reserving "empty seats" for new nodes, the DAG-based consensus (e.g., Bullshark, Tusk) lacks robust reconfiguration mechanisms. Supporting node joins or departures requires new protocols to manage DAG updates, ensuring beacon generation and consensus remain synchronized. This increases complexity, limiting Rubato to static or slowly changing committees. Extending Rubato for dynamic, large-scale deployments remains an open challenge.

7. Conclusion

We presented Rubato, a post-quantum secure distributed randomness beacon for asynchronous networks. Rubato integrates a lattice-based batched Asynchronous Verifiable Secret Sharing (bAVSS-PQ) scheme, using the SIS problem, with DAG-based consensus protocols. It achieves provable post-quantum security, efficient secret batching, and robust liveness through an epoch-staggered design and waiting mechanisms that resolve circular dependencies and prevent deadlocks. Rubato efficiently supports applications for small-to-medium committees.

Acknowledgments

This work is supported in part by National Key R&D Program of China, under Grant 2023YFB2703800, in part by the Tianjin Natural Science Foundation 23JCYBJC01750, the National Natural Science Foundation of China under Grants U23B2049, U22B2027, U2436208 and 62172297, the China Guangxi Science and Technology Plan Project (Guangxi Science and Technology Base and Talent Special Project) under Grant AD23026096 (Application Number 2022AC20001), Hainan Province Science and Technology Special Fund (Grant No. ZDYF2024GXJS008), and the Project of Guangdong Key Laboratory of Industrial Control System Security (2024B1212020010)

References

[1] A. Bhat, N. Shrestha, Z. Luo, A. Kate, and K. Nayak, "Randpiper - reconfiguration-friendly random beacons with quadratic communication," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3502–3524. [Online]. Available: https://doi.org/10.1145/3460120.3484574

[2] A. Bhat, N. Shrestha, A. Kate, and K. Nayak, "OptRand: Optimistically responsive distributed random beacons," Cryptology ePrint Archive, Paper 2022/193, 2022. [Online]. Available: https://eprint.iacr.org/2022/193

[3] Y. Yang, B. Li, Q. Wu, B. Qin, Q. Wang, S. Xiong, and W. Susilo, "Randflash: Breaking the quadratic barrier in large-scale distributed randomness beacons," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4710–4725, 2025.

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

[5] X. Meng, X. Sui, Z. Yang, K. Rong, W. Xu, S. Chen, Y. Yan, and S. Duan, "Rondo: Scalable and reconfiguration-friendly randomness beacon," Cryptology ePrint Archive, Paper 2024/641, 2024. [Online]. Available: https://eprint.iacr.org/2024/641

[6] L. Freitas, P. Kuznetsov, and A. Tonkikh, "Distributed randomness from approximate agreement," 2022. [Online]. Available: https://arxiv.org/abs/2205.11878

[7] A. Bandarupalli, A. Bhat, S. Bagchi, A. Kate, and M. K. Reiter, "Random beacons in monte carlo: Efficient asynchronous random beacon without threshold cryptography," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2621–2635. [Online]. Available: https://doi.org/10.1145/3658644.3670326

[8] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 444–460.

[9] L. Zhang, T. Liu, Z. Ou, H. Kan, and J. Zhang, "AsyRand: fast asynchronous distributed randomness beacon with reconfiguration," Cryptology ePrint Archive, Paper 2025/406, 2025. [Online]. Available: https://eprint.iacr.org/2025/406

[10] S. Das, Z. Xiang, and L. Ren, "Asynchronous data dissemination and its applications," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2705–2721. [Online]. Available: https://doi.org/10.1145/3460120.3484808

[11] M. Haahr, "True random number service," https://www.random.org/, 1998, [Online; accessed 16 April 2025].

[12] I. Abellán Álvarez, V. Gramlich, and J. Sedlmeir, "Unsealing the secrets of blockchain consensus: A systematic comparison of the formal security of proof-of-work and proof-of-stake," in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 278–287. [Online]. Available: https://doi.org/10.1145/3605098.3635970

[13] D. Grandjean, L. Heimbach, and R. Wattenhofer, "Ethereum proof-of-stake consensus layer: Participation and decentralization," in *Financial Cryptography and Data Security. FC 2024 International Workshops*, J. Budurushi, O. Kulyk, S. Allen, T. Diamandis, A. Klages-Mundt, A. Bracciali, G. Goodell, and S. Matsuo, Eds. Cham: Springer Nature Switzerland, 2025, pp. 253–280.

[14] M. Kelkar, S. Deb, and S. Kannan, "Order-fair consensus in the permissionless setting," in *Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, ser. APKC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3–14. [Online]. Available: https://doi.org/10.1145/3494105.3526239

[15] M. Alown, M. Sabir Kiraz, and M. Ali Bingol, "Enhancing democratic processes: A survey of dre, internet, and blockchain in electronic voting systems," *IEEE Access*, vol. 13, pp. 20 512–20 545, 2025.

[16] I. A. Omar, H. R. Hasan, R. Jayaraman, K. Salah, and M. Omar, "Implementing decentralized auctions using blockchain smart contracts," *Technological Forecasting and Social Change*, vol. 168, p. 120786, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162521002183

[17] E. Kokoris Kogias, E. C. Alp, L. Gasser, P. S. Jovanovic, E. Syta, and B. A. Ford, "Calypso: Private data management for decentralized ledgers," vol. 14, no. 4, 2021, p. 586–599. [Online]. Available: https://infoscience.epfl.ch/handle/20.500.14299/180167

[18] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary version*, vol. 78, no. 110, pp. 1–108, 1998.

[19] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," 2022. [Online]. Available: https://arxiv.org/abs/2109.00984

[20] Y. Lu, Z. Lu, Q. Tang, and G. Wang, "Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited," in *Proceedings of the 39th Symposium on Principles of Distributed Computing*, ser. PODC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 129–138. [Online]. Available: https://doi.org/10.1145/3382734.3405707

[21] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 31–42. [Online]. Available: https://doi.org/10.1145/2976749.2978399

[22] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and tusk: a dag-based mempool and efficient bft consensus," in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 34–50. [Online]. Available: https://doi.org/10.1145/3492321.3519594

[23] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, "Bullshark: Dag bft protocols made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2705–2718. [Online]. Available: https://doi.org/10.1145/3548606.3559361

[24] N. Shrestha, R. Shrothrium, A. Kate, and K. Nayak, " Sailfish: Towards Improving the Latency of DAG-based BFT ," in *2025 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 21–21. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00021

[25] J. Zhang, J. Ou, D. Lan, B. Ma, and L. Luo, "Infinityrand: Blockchain non-interactive randomness beacon protocol based on trapdoor verifiable delay function," in *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2024, pp. 1621–1626.

[26] P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, and E. Weippl, "RandRunner: Distributed randomness from trapdoor VDFs with strong uniqueness," Cryptology ePrint Archive, Paper 2020/942, 2020. [Online]. Available: https://eprint.iacr.org/2020/942

[27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 514–532.

[28] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, Aug 2001. [Online]. Available: https://doi.org/10.1007/s102070100002

[29] "Drand - a distributed randomness beacon daemon," 2020. [Online]. Available: https://github.com/drand/drand

[30] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," 2018. [Online]. Available: https://arxiv.org/abs/1805.04548

[31] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, vol. 35, no. 2, p. 288–323, Apr. 1988. [Online]. Available: https://doi.org/10.1145/42282.42283

[32] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 295–310.

[33] K. Gurkan, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu, "Aggregatable distributed key generation," in *Advances in Cryptology – EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert, Eds. Cham: Springer International Publishing, 2021, pp. 147–176.

[34] M. Stadler, "Publicly verifiable secret sharing," in *Advances in Cryptology — EUROCRYPT '96*, U. Maurer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 190–199.

[35] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous verifiable secret sharing and proactive cryptosystems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 88–97. [Online]. Available: https://doi.org/10.1145/586110.586124

[36] R. Bacho, C. Lenzen, J. Loss, S. Ochsenreither, and D. Papachristoudis, "Grandline: Adaptively secure dkg and randomness beacon with (log-)quadratic communication complexity," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 941–955. [Online]. Available: https://doi.org/10.1145/3658644.3690287

[37] I. Cascudo and B. David, "Scrape: Scalable randomness attested by public entities," in *Applied Cryptography and Network Security*, D. Gollmann, A. Miyaji, and H. Kikuchi, Eds. Cham: Springer International Publishing, 2017, pp. 537–556.

[38] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, p. 374–382, Apr. 1985. [Online]. Available: https://doi.org/10.1145/3149.214121

[39] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, and G. Stern, "Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation," in *Advances in Cryptology – CRYPTO 2023*, H. Handschuh and A. Lysyanskaya, Eds. Cham: Springer Nature Switzerland, 2023, pp. 39–70.

[40] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu, "Reaching consensus for asynchronous distributed key generation," in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, ser. PODC'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 363–373. [Online]. Available: https://doi.org/10.1145/3465084.3467914

[41] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999. [Online]. Available: https://doi.org/10.1137/S0036144598347011

[42] J. Chow, O. Dial, and J. Gambetta, "Ibm quantum breaks the 100-qubit processor barrier," *IBM Research Blog*, vol. 2, 2021.

[43] D.-T. Dam, T.-H. Tran, V.-P. Hoang, C.-K. Pham, and T.-T. Hoang, "A survey of post-quantum cryptography: Start of a new race," *Cryptography*, vol. 7, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2410-387X/7/3/40

[44] G. Alagic, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the first round of the nist post-quantum cryptography standardization process," 2019.

[45] N. Kiamari, M. Hadian, and S. Mashhadi, "Non-interactive verifiable lwe-based multi secret sharing scheme," *Multimedia Tools and Applications*, vol. 82, no. 14, pp. 22 175–22 187, Jun 2023. [Online]. Available: https://doi.org/10.1007/s11042-022-13347-4

[46] C. Gentry, S. Halevi, and V. Lyubashevsky, "Practical non-interactive publicly verifiable secret sharing with thousands of parties," in *Advances in Cryptology – EUROCRYPT 2022*, O. Dunkelman and S. Dziembowski, Eds. Cham: Springer International Publishing, 2022, pp. 458–487.

[47] M. Hadian Dehkordi and R. Ghasemi, "A lightweight public verifiable multi secret sharing scheme using short integer solution," *Wireless Personal Communications*, vol. 91, no. 3, pp. 1459–1469, Dec 2016. [Online]. Available: https://doi.org/10.1007/s11277-016-3539-7

[48] V. Shoup and N. P. Smart, "Lightweight asynchronous verifiable secret sharing with optimal resilience," *Journal of Cryptology*, vol. 37, no. 3, p. 27, Jun 2024. [Online]. Available: https://doi.org/10.1007/s00145-024-09505-6

[49] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, "Random oracles in a quantum world," in *Advances in Cryptology – ASIACRYPT 2011*, D. H. Lee and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 41–69.

[50] A. R. Choudhuri, A. Jain, and Z. Jin, "Snargs for $\mathcal{P}$ from lwe," in *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 68–79.

[51] J. Buchmann, N. Büscher, F. Göpfert, S. Katzenbeisser, J. Krämer, D. Micciancio, S. Siim, C. van Vredendaal, and M. Walter, "Creating cryptographic challenges using multi-party computation: The lwe challenge," in *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography*, ser. AsiaPKC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 11–20. [Online]. Available: https://doi.org/10.1145/2898420.2898422

[52] V. Cini, G. Malavolta, N. K. Nguyen, and H. Wee, "Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup," in *Advances in Cryptology – CRYPTO 2024*, L. Reyzin and D. Stebila, Eds. Cham: Springer Nature Switzerland, 2024, pp. 207–242.

[53] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.

[54] I. Abraham, G. Ashsarov, A. Patra, and G. Stern, "Asynchronous agreement on a core set in constant expected time and more efficient asynchronous vss and mpc," in *Theory of Cryptography: 22nd International Conference, TCC 2024, Milan, Italy, December 2–6, 2024, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 451–482. [Online]. Available: https://doi.org/10.1007/978-3-031-78023-3_15

[55] A. R. Block, A. Garreta, J. Katz, J. Thaler, P. R. Tiwari, and M. Zajac, "Fiat-shamir security of FRI and related SNARKs," Cryptology ePrint Archive, Paper 2023/1071, 2023. [Online]. Available: https://eprint.iacr.org/2023/1071

[56] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 315–334.

[57] I. Keidar, E. Kokoris-Kogias, O. Naor, and A. Spiegelman, "All you need is dag," in *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, ser. PODC'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 165–175. [Online]. Available: https://doi.org/10.1145/3465084.3467905

[58] R. Cohen, A. Shelat, and D. Wichs, "Adaptively secure mpc with sublinear communication complexity," in *Advances in Cryptology – CRYPTO 2019*, A. Boldyreva and D. Micciancio, Eds. Cham: Springer International Publishing, 2019, pp. 30–60.

[59] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, ser. PODC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 347–356. [Online]. Available: https://doi.org/10.1145/3293611.3331591

[60] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, Nov. 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[61] D. Unruh, "Computationally binding quantum commitments," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 497–527.

Appendix

This appendix provides detailed proofs for the security theorems presented in Section 4 and the complexity analysis of the Rubato protocol.

1. Security Proofs

Proof of Theorem 1. Consider the bAVSS-PQ protocol with $n \geq 3f+1$, polynomial degree $t = f$, and an adaptive PPT adversary $\mathcal{A}$ corrupting up to f nodes.

Liveness: If the dealer d is honest, all honest nodes eventually complete the reply phase, as they receive and validate shares, enabling d to collect $2t+1$ signatures to form $\mathcal{C}$. If $t+1$ honest nodes initiate reconstruction, each honest node receives at least $f+1$ validated shares, allowing interpolation and output generation, satisfying liveness.

Strong Commitment: Upon formation of $\mathcal{C}$ with $2t+1$ signatures on commitment $\mathbf{t}$, at least $t+1$ honest nodes have validated shares consistent with $\mathbf{t}$. The polynomial commitment's binding property (Theorem 2) ensures $\mathbf{t}$ uniquely determines $\{s_{d,j}\}_{j\in[\theta]}$. In reconstruction, share consistency is enforced, and $t+1$ shares yield consistent secrets via interpolation, even against a malicious dealer.

Privacy: If d is honest and reconstruction is not initiated, $\mathcal{A}$, corrupting at most $f = t$ nodes (except the dealer), obtains f shares per secret. Since $t+1$ shares are required to reconstruct each $s_{d,j}$, Shamir's secret sharing [60] ensures $\mathcal{A}$ gains no information about $\{s_{d,j}\}_{j\in[\theta]}$. □

Proof of Theorem 2. The properties of completeness, binding, and knowledge soundness follow from similar arguments established in [52] (Theorems 2, 3, and 4). We focus on proving the hiding property.

The commitment is defined as:

$$\mathbf{t} = (\mathbf{I}_\kappa \otimes \mathbf{A}) \cdot \mathbf{s}_0 = (\mathbf{A} \cdot \mathbf{s}_0^{(1)}, \ldots, \mathbf{A} \cdot \mathbf{s}_0^{(\kappa)}) \in \mathbb{Z}_q^{\kappa m},$$

where $\mathbf{A} \in \mathbb{Z}_q^{m \times L \log q}$ is uniformly random and $\mathbf{s}_0$ is a short vector with $\|\mathbf{s}_0\|_\infty \leq 1$. Per [61], $\mathbf{t}$ is an SIS-hash of $\mathbf{s}_0$, and the uniformity of $\mathbf{A}$ ensures $\mathbf{t}$ is uniformly distributed over $\mathbb{Z}_q^{\kappa m}$, revealing no information about $\mathbf{s}_0$. Since $\mathbf{s}_0 = \mathbf{G}_{r\kappa m}^{-1}(\mathbf{f}_1)$, $\mathbf{f}_1 = (\mathbf{I}_{r\kappa} \otimes \mathbf{A}) \cdot \mathbf{s}_1$, and recursively to $\mathbf{f}$, the uniformity holds, so $\mathbf{t}$ reveals no information about $\mathbf{f}$.

The proof $\pi_i = \{(\mathbf{y}_i, \mathbf{v}_i)\}_{i\in[0,\ell]}$, for each round $i \in [0, \ell]$:

- $\mathbf{y}_0 = \mathbf{s}_0^{(0)}$, for $i \in [1, \ell]$, $\mathbf{y}_i = \mathbf{s}_i^{(0)} = (\mathbf{C}_i^\top \otimes \mathbf{I}_{rm \log q}) \cdot \mathbf{s}_{i-1}^{(1)}$, where $\mathbf{C}_i \in \{0,1\}^{r\kappa \times \kappa}$ is a uniformly random challenge, and $\mathbf{s}_{i-1}^{(1)} = \mathbf{G}^{-1}(\mathbf{f}_{i-1})$. Since $\mathbf{f}_{i-1}$ is uniform (due to recursive folding and $\mathbf{A}$'s uniformity), $\mathbf{s}_{i-1}^{(1)}$ is uniform. Combined with $\mathbf{C}_i$'s randomness, $\mathbf{y}_i$ is uniform.
- $\mathbf{v}_i = \prod_{k=1}^{\ell-i} (\mathbf{I}_{r^k \kappa m} \otimes \mathbf{x}_{\ell-i-k}^\top) \cdot \mathbf{f}_i$, and for $i \in [1, \ell]$ $\mathbf{f}_i = (\mathbf{C}^\top \otimes \mathbf{I}_{r^{\ell-i} m}) \cdot \mathbf{f}_{i-1}$ is uniform. Thus, $\mathbf{v}_i$ is uniform.

In the non-interactive setting, Fiat-Shamir generates $\mathbf{C}_i = H(\mathbf{t}, \mathbf{u}, \mathbf{x}_0, \ldots, \mathbf{x}_{\ell-i}, \mathbf{y}_i, \mathbf{v}_i)$. Since $\mathbf{t}, \mathbf{u}, \mathbf{v}$ are uniform, $\mathbf{C}_i$ remains random, preserving hiding. □

Proof of Theorem 3. The protocol organizes vertices into rounds and waves (4 rounds per wave in Bullshark, 2 in Tusk), with leader election driven by Rubato's global coin requests. Liveness requires honest nodes to output at least one set of $f+1$ certificates $\{\mathcal{C}_e\}$ per epoch e, resolving the circular dependency between beacon generation and consensus, and preventing deadlocks due to certificate availability.

The DAG's liveness depends on global coin requests, which rely on the previous epoch's $f+1$ certificates $\{\mathcal{C}_{e-1}\}$ (Algorithm 1, Lines 8–13). For epoch e, at least $f+1$ honest nodes invoke PREP$(e, \mathbf{s})$, generating $f+1$ certificates $\mathcal{C}_e$ via bAVSS-PQ (Theorem 1). Each wave selects a leader vertex via a global coin, with probability at least $2/3$ of choosing an honest node, since $n-f \geq 2f+1$ of $n \geq 3f+1$ nodes are honest [57]. An honest leader proposes a vertex containing

$\mathcal{C}_e$, which is ordered (Line 14). An epoch contains $B \geq \lambda$ waves, so the probability of no honest leader in an epoch is:

$$\Pr[\text{all malicious leaders}] \leq \left(\frac{1}{3}\right)^B \leq \left(\frac{1}{3}\right)^\lambda = \text{negl}(\lambda).$$

Thus, with probability $1-\text{negl}(\lambda)$, at least one honest leader commits $f+1$ certificates $\{\mathcal{C}_e\}$.

The circular dependency is resolved by the epoch-staggered design: epoch e's global coin requests use $\{\mathcal{C}_{e-1}\}$. For epoch 1, $\{\mathcal{C}_0\}$ is decided by an auxiliary consensus protocol, assumed to terminate with probability 1 in the asynchronous network. By induction, if epoch $e-1$ outputs $\{\mathcal{C}_{e-1}\}$ (base case: epoch 0 via auxiliary consensus), epoch e's global coins are available, enabling leader election and certificate ordering. Since bAVSS-PQ ensures certificate generation (Theorem 1), the process continues indefinitely.

Deadlock prevention is achieved by the waiting mechanism (Algorithm 4 Line 26). At the first round of epoch e, nodes check the maximum epoch of proposed certificates (ec). If $ec \leq e$, indicating insufficient certificates for epoch $e+1$'s beacons, nodes pause until a new $\mathcal{C}_e$ is enqueued. Since $f+1$ honest nodes generate $\mathcal{C}_e$ (Theorem 1), and reliable broadcast ensures delivery, the queue becomes non-empty with probability 1. This mechanism provides $B \geq \lambda$ waves to commit $\{\mathcal{C}_e\}$ before epoch $e+1$, with failure probability $(1/3)^B \leq \text{negl}(\lambda)$.

Agreement, total order, and fairness are inherited from [57]: reliable broadcast ensures consistent vertex reception, deterministic ordering based on leader causal history guarantees agreement and total order, and honest vertices are eventually included. Thus, liveness, including circular dependency resolution and deadlock prevention, holds with probability $1-\text{negl}(\lambda)$. □

Proof of Theorem 4. Let $\mathcal{A}$ be an adaptive PPT quantum adversary corrupting up to $f < n/3$ nodes in an asynchronous network with $n \geq 3f+1$. We prove that $\mathcal{A}$ cannot distinguish the Rubato beacon $b_{e,i}$ (Algorithm 1) from a random element in $\mathbb{Z}_q$ with advantage greater than $\text{negl}(\lambda)$, using the following game:

1) Challenger $\mathbf{C}$ generates public parameters and public-private key pairs, sending public keys $\{\text{pk}_i\}_{i\in[n]}$ to $\mathcal{A}$.
2) $\mathcal{A}$ may provide public keys for nodes it later corrupts.
3) For epoch e, $\mathbf{C}$ runs PREP$(e, \mathbf{s})$ for honest nodes, sending bAVSS-PQ messages. $\mathcal{A}$ observes and reorders messages, runs DAG consensus, and may corrupt up to f nodes, accessing their full history.
4) $\mathbf{C}$ waits until an honest node decides $f+1$ certificates $\{\mathcal{C}_e\}$.
5) $\mathbf{C}$ samples $b \in \{0,1\}$. If $b=0$, it sends $b_{e,i} = H(\hat{b}_{e,(i-1)g+1}, \ldots, \hat{b}_{e,ig})$, where $\hat{b}_{e,j} = \sum_{d\in D} s_{d,j}$ and D is the set of $f+1$ dealers in $\{\mathcal{C}_e\}$; if $b=1$, it sends a random $r \in \mathcal{D}$.
6) $\mathcal{A}$ guesses b'.

The advantage is $\text{Adv}_{\mathcal{A}}^{\text{indist}}(\lambda) = \left|\Pr[b=b'] - \frac{1}{2}\right|$.

Assume $\mathcal{A}$ corrupts f nodes F after $\{\mathcal{C}_e\}$ is decided, maximizing information. For each dealer $d \in D$, $\mathcal{A}$ knows secrets $\{s_{d,j}\}_{j\in[\theta]}$ if $d \in F$, shares $\{u_{i,j} = s_j(i)\}_{i\in D, j\in[\theta]}$, commitment $\mathbf{t}_d$, and proofs $\{\pi_i\}_{i\in F}$. Since $|D| = f+1$ and $|F| = f$, there exists an honest dealer $d^* \in D \setminus F$.

For $j \in \{(i-1)g+1, \ldots, ig\}$, the intermediate beacon is $\hat{b}_{e,j} = \sum_{d\in F} s_{d,j} + s_{d^*,j}$, where $d^* \in D \setminus F$. Since $s_{d^*,j} \sim \mathbb{Z}_q$ is uniform and independent, $\hat{b}_{e,j} \sim \mathbb{Z}_q$ by the bijection $x \mapsto x + a \mod q$ in $\mathbb{Z}_q$. Thus, $b_{e,i}$ is uniform over $\mathcal{D} = \{0,1\}^\lambda$ (Theorem 6).

By Theorem 1 (Privacy), f shares $\{u_{i,j}\}_{i\in F}$ reveal no information about $s_{d^*,j}$, as $t+1 = f+1$ shares are needed. By Theorem 2 (Hiding), the commitment $\mathbf{t}_{d^*}$ is uniformly random over $\mathbb{Z}_q^{\kappa m}$, and every $(\mathbf{y}, \mathbf{v})$ in the proofs $\{\pi_i\}_{i\in F}$ is uniformly random, leaking no information about $s_{d^*,j}$.

Thus, $\mathcal{A}$ gains no information about $b_{e,i}$, and its advantage in distinguishing $b_{e,i}$ from a random element in $\mathcal{D}$ is at most $\frac{1}{2^\lambda}$, which is negligible: $\text{Adv}_{\mathcal{A}}^{\text{indist}}(\lambda) \leq \frac{1}{2^\lambda} \leq \text{negl}(\lambda)$. □

Proof of Theorem 5. We prove each property, leveraging Theorems 1, 2, 3, and 4.

1) **Unpredictability**: $\mathcal{A}$ guesses $b'_{e'',i''}$ for a future beacon $b_{e'',i''}$ ($e'' > e'$ or $e'' = e', i'' > i'$). By Theorem 4, $b_{e'',i''}$ is indistinguishable from a random element in $\mathcal{D}$, with advantage $\text{Adv}_{\mathcal{A}}^{\text{indist}}(\lambda) \leq \text{negl}(\lambda)$. Since $|\mathcal{D}| = 2^\lambda$, the probability of guessing correctly is:
$$\Pr[b'_{e'',i''} = b_{e'',i''}] \leq \frac{1}{2^\lambda} \leq \text{negl}(\lambda).$$
bAVSS-PQ's privacy (Theorem 1) ensures no information about $s_{j,i''}$ leaks before OPEN(e'', i''), satisfying unpredictability.
2) **Bias-resistance**: By Theorem 4, $b_{e,i}$ is indistinguishable from a random element in $\mathcal{D} = \{0,1\}^\lambda$. Since at least one honest dealer's contribution of $b_{e,i}$ is uniformly random, each bit of $b_{e,i}$ is uniform with deviation at most $\text{negl}(\lambda)$.
3) **Agreement**: By Theorem 3, DAG consensus ensures all honest nodes agree on the same $f+1$ certificates $\{\mathcal{C}_e\}$. Each certificate $\mathcal{C}_e$ fixes a unique secret set $\{s_{d,j}\}_{j\in[\theta]}$ (Theorem 1), and bAVSS-PQ's correctness ensures honest nodes reconstruct identical secrets $s_{d,j}$. If $\mathcal{A}$ attempts to cause inconsistency by forging shares, Merkle verification detects invalid shares. Since H is deterministic, all honest nodes compute the same $b_{e,i}$, satisfying agreement.
4) **Liveness**: If honest nodes invoke PREP$(e, \mathbf{s})$ and OPEN(e,i), they output $\langle e, i, b_{e,i}\rangle$. By Theorem 1, $f+1$ honest dealers generate certificates $\{\mathcal{C}_e\}$. Theorem 3 ensures DAG consensus outputs $\{\mathcal{C}_{e-1}\}$ with probability $1-\text{negl}(\lambda)$, enabling OPEN(e,i) to reconstruct $f+1$ secrets. bAVSS-PQ's liveness guarantees reconstruction, and $b_{e,i}$ is computed, satisfying liveness.

□

Proof of Theorem 6. Each $\hat{b}_{e,j}$ has entropy:

$$\mathcal{H}(\hat{b}_{e,j}) = \log q = \lambda/g.$$

For g independent beacons, the spliced input $(\hat{b}_{e,(i-1)g+1}, \ldots, \hat{b}_{e,ig}) \in (\mathbb{Z}_q)^g$ has total entropy:

$$\mathcal{H}(\hat{b}_{e,(i-1)g+1}, \ldots, \hat{b}_{e,ig}) = g \cdot \lambda/g = \lambda.$$

Since $\mathbb{Z}_q \cong \{0,1\}^{\lambda/g}$, the input space $(\mathbb{Z}_q)^g \cong \{0,1\}^\lambda$ is equivalent to a λ-bit random number. The hash function $H : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is indistinguishable from a random function, so $b_{e,i}$ is uniformly random over $\mathcal{D} = \{0,1\}^\lambda$. □

2. Complexity Analysis

The communication complexity of generating a single beacon $b_{e,i}$ in Rubato (Algorithm 1) is dominated by the bAVSS-PQ reconstruction phase (Algorithm 2, Lines 21–29). To form a beacon, $f + 1 = O(n)$ secrets are reconstructed, each requiring $t + 1 = f + 1 = O(n)$ valid shares, as $t = f < n/3$. Each node broadcasts a share $u_{i,j} \in \mathbb{Z}_q$ of size $O(1)$ and a Merkle proof of size $O(\lambda \log n)$, totaling $O(\lambda \log n)$ bits per message. For one secret, approximately n nodes broadcast shares to n nodes to ensure $t + 1$ valid shares despite Byzantine faults, costing $O(n \cdot n \cdot \lambda \log n) = O(\lambda n^2 \log n)$ bits. With $f + 1 = O(n)$ secrets, the reconstruction phase incurs $O(n \cdot \lambda n^2 \log n) = O(\lambda n^3 \log n)$ bits, dominating the overall communication cost. Other bAVSS-PQ phases contribute less: the sharing phase, where $f + 1$ dealers send commitments, shares, proofs, and broadcast $\theta = O(\lambda^2)$ Merkle roots via RBC, costs $O(\lambda^3 n^2 \log n)$ bits, amortized to $O(\lambda n^2 \log n)$ per beacon. The total communication complexity is thus $O(\lambda n^3 \log n)$.

The computational complexity of the bAVSS-PQ protocol is primarily driven by the sharing phase, where a single dealer generates commitments and proofs for a polynomial commitment with $L = r^{\ell+1}\kappa m$, where $r^{\ell+1} = O(n)$, $\kappa = O(\lambda)$, and $m = O(\lambda)$. The dealer's cost per commitment and proof is $O(L \cdot \lambda) = O(n\lambda^3)$ [52]. For n nodes, the total sharing phase complexity for a single dealer is $O(n \cdot n\lambda^3) = O(n^2\lambda^3)$. With $\theta = O(\lambda^2)$ secrets, the amortized complexity per beacon is $O(n^2\lambda^3)/O(\lambda^2) = O(n^2\lambda)$. For a single verifier, the per-share validation complexity is $O(\log L \cdot \lambda^2) = O(\log(n\lambda^2) \cdot \lambda^2)$. Validating shares from n dealers, the total verification cost is $O(n \cdot \log(n\lambda^2) \cdot \lambda^2)$, amortizing to $O(n \cdot \log(n\lambda^2))$ per beacon. The Reply, Confirm, and Reconstruct phases are negligible, yielding an overall computational complexity of $O(n^2\lambda)$ for the dealer and $O(n \cdot \log(n\lambda^2))$ for the verifier per beacon, dominated by the sharing phase for the dealer.