

Silk: Computation-Efficient and Post-Quantum-Friendly Randomness Beacon

Binyang Hou, Jiamian Yan, Liang Zhang, Borui Chen, and Haibin Kan

Abstract—Distributed randomness beacons enable mutually distrustful parties to jointly generate public randomness. Several existing beacons built from secret sharing use costly pairings or group operations to verify shares. Their underlying discrete-logarithm assumptions are also vulnerable to quantum attacks. Moreover, running a separate Byzantine fault-tolerant (BFT) consensus instance for each output incurs repeated computation and communication costs, even when secrets are shared in batches. We present *Silk*, a computation-efficient, post-quantum-friendly and deterministic randomness beacon under partial synchrony. *Silk*’s sharing protocol, *Mulberry*, realizes batched asynchronous verifiable secret sharing with partial output (bAVSS-PO), using hash-based share verification to reduce computation cost. *Silk* further amortizes BFT consensus across a batch of outputs, requiring only one BFT instance per batch. We establish *Mulberry*’s bAVSS-PO properties and *Silk*’s beacon properties under static corruption and the stated security assumptions. We implement *Silk* and *Mulberry* in Rust and evaluate their computation and communication costs. In WAN experiments with 121 replicas across eight AWS regions, *Silk* achieves a throughput of 5.414 outputs/s, $1.53\times$ that of Rondo [NDSS’25] and $4.18\times$ that of Spurt [IEEE S&P’22].

Index Terms—Distributed randomness beacon, verifiable secret sharing, post-quantum cryptography, partial synchrony, Byzantine fault tolerance

I. INTRODUCTION

Randomness beacons [1] provide public random values for leader election [2], committee selection [3], [4], and other decisions in distributed systems. Applications such as blockchains [4], [3], electronic voting [5], and Byzantine agreement [6], [7] also rely on randomness. A beacon should provide Unpredictability and Bias-resistance: outputs must remain unknown before their release, and adversaries must not be able to bias the output distribution. A centralized beacon [8], [9] relies on a trusted party to generate and publish these values. This party could embed a backdoor in the generator to predict or bias its outputs. A distributed randomness beacon (DRB) [10], [9] distributes this task among multiple parties, reducing reliance on any single party. A DRB must preserve its guarantees even when a bounded number of Byzantine parties collude and deviate arbitrarily from the protocol. Besides

Unpredictability and Bias-resistance, the beacon should provide Availability and Public verifiability: correct parties should continue to obtain outputs, and external users should be able to check them.

The choice of cryptographic primitives strongly affects the cost of this service. Threshold-signature-based beacons, including drand [11] and DFINITY [12], produce compact outputs from shares of a common signing key. Establishing that key requires trusted setup or distributed key generation (DKG) [13], [14], and membership changes can require further key generation or resharing [11]. Beacons based on verifiable secret sharing (VSS), such as Spurt [15] and Rondo [16], avoid this costly setup. Parties distribute shares of fresh random contributions and use Byzantine fault-tolerant (BFT) consensus to select which contributions to reconstruct and combine.

a) Existing approaches: Synchronous beacons such as RandPiper [18], OptRand [19], and GRandLine [20] generate randomness with low communication cost. However, these synchronous beacons rely on a known bound on message delays [25] that network congestion may violate. Spurt [15] relaxes this assumption to partial synchrony and achieves quadratic amortized total communication per output through publicly verifiable secret sharing (PVSS). However, Spurt repeats the sharing and verification of random contributions in every beacon round. Rondo [16] reduces this cost by sharing secrets in batches through Breeze, its batched asynchronous verifiable secret sharing protocol with partial output (bAVSS-PO), which allows secrets in a batch to be opened individually. Breeze also uses Bulletproof-style polynomial commitments [26] to keep proofs compact. The pairings in Spurt and the group operations needed to generate and check Breeze’s proofs remain computationally expensive compared with hashes and finite-field arithmetic [22], [27]. Moreover, the pairing and discrete-logarithm assumptions underlying Spurt and Breeze are vulnerable to quantum attacks [28].

Existing hash-based beacons reduce this computation cost while supporting post-quantum security and progress in fully asynchronous networks. HashRand [22] is a Monte Carlo beacon: it admits small probabilities of selecting no honest contribution or producing inconsistent outputs. Rubato [23] provides Las Vegas guarantees, preserving agreement but relying on randomized consensus with no deterministic bound on the number of rounds before termination. For applications such as leader or committee selection, Monte Carlo consistency failures can therefore lead correct parties to conflicting choices. Las Vegas beacons avoid this risk, but leave applications without a deterministic bound on the protocol rounds needed to obtain each output.

Binyang Hou, Jiamian Yan, Borui Chen, and Haibin Kan are with the College of Computer Science and Artificial Intelligence, Fudan University, Shanghai, China. Haibin Kan is also with Shanghai Engineering Research Center of Blockchain, Shanghai 200433, China, and Shanghai Institute for Mathematics and Interdisciplinary Sciences, Shanghai 200433, China. (e-mail: byhou25@m.fudan.edu.cn; 25113050230@m.fudan.edu.cn; berrychen0w0@gmail.com; hbkan@fudan.edu.cn)

Liang Zhang is with the Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Hong Kong (e-mail: scottzhang@ust.hk).

Corresponding author: Haibin Kan.

TABLE I
DISTRIBUTED RANDOMNESS BEACONS: ASSUMPTIONS, SECURITY, AND AMORTIZED COST PER OUTPUT.

Protocol	Network	Termination	Unpredictability	PQ secure	Communication (total bits)	Computation (per node)	Setup	Crypto. assumptions
SCRAPE [17]	Sync.	D	Yes	No	$O(\kappa n^4)$	$O(n^2)$ DLE	CRS, PKI	DDH / DBS
HydRand [8]	Sync.	D	$t + 1$	No	$O(\kappa n^2 \log n)$	$O(n)$ DLE	CRS, PKI	DDH
BRandPiper [18]	Sync.	D	Yes	No	$O(\kappa n^3)$	$O(n^2)$ DLE	SRS	q -SDH
GRandPiper [18]	Sync.	D	$t + 1$	No	$O(\kappa n^2)$	$O(n^2)$ DLE	SRS	q -SDH
OptRand [19]	Sync.	D	Yes	No	$O(\kappa n^2)$	$O(n)$ DLE [†]	SRS, PKI	q -SDH, SXDH
GRandLine [20]	Sync.	D	Yes	No	$O(\kappa n^2)$	$O(n)$ DLE	CRS, DKG	AGM, co-OMDL
RandFlash [21]	Sync.	D	Yes	No	$O(\kappa n \log n)^*$	$O(n^2)$ DLE	CRS, PKI	DDH
Cachin et al. [6]	Async.	D	Yes	No	$O(\kappa n^2)$	$O(n)$ DLE	DKG	RO, CDH
RandShare [10]	Async.	LV	Yes	No	$O(\kappa n^4)$	$O(n^3)$ DLE	DKG	DLog
Das et al. [14]	Async.	LV	Yes	No	$O(\kappa n^3)$	$O(n^3)$ DLE	PKI	RO, DDH
HashRand [22]	Async.	MC	Yes	Yes	$O(\kappa n^2 \log n)$	$O(n \log n)$ H	Secure Channel [§]	RO [§]
Rubato [23]	Async.	LV	Yes	Yes	$O(\kappa n^2)$	$O(n)^{\ddagger}$	Auth Channel	RO
Dfinity [24]	Partial	D	Yes	No	$O(\kappa n^2)$	$O(n)$ DLE	DKG	CDH
Spurt [15]	Partial	D	Yes	No	$O(\kappa n^2)$	$O(n)$ DLE [†]	CRS, PKI	DBDH
Rondo [16]	Partial	D	Yes	No	$O(\kappa n^2 \log n)$	$O(n)$ DLE	CRS, PKI	DLog
Silk	Partial	D	Yes	Yes	$O(\kappa n^3)$	$O(n^3)$ H	Secure Channel, PKI	RO [◊]

Complexity: Communication counts the total bits sent by the n participating nodes per output. Computation is the amortized per-node count of dominant cryptographic operations: DLE denotes exponentiations and pairings, and H denotes hash computations. κ is the security parameter. c denotes HashRand’s AnyTrust sample size. [†] Amortized over leader rotation. **Termination Flavor:** D (deterministic) ensures consistent outputs and has a deterministic runtime under the stated network assumptions. MC (Monte Carlo) completes a fixed number of rounds but may produce inconsistent outputs with a small, adjustable probability. LV (Las Vegas) ensures consistency but has no deterministic termination bound. Its costs are reported in expectation. * RandFlash’s bound is optimistic. Its worst-case communication is $O(\kappa n^2)$. [‡] Rubato counts lightweight cryptographic operations, including hashing and authenticated symmetric-key encryption. [§] HashRand’s public-verification extension requires PKI and qEUF-CMA signatures. [◊] Silk additionally requires qEUF-CMA signatures. Here qEUF-CMA denotes existential unforgeability under chosen-message attacks against quantum polynomial-time adversaries.

Together, these limitations motivate the following research question: *Can we design a computation-efficient, post-quantum-friendly and deterministic randomness beacon in a partially synchronous network?*

b) Our approach: We develop *Silk* for a fixed committee of $n \geq 3t + 1$ parties with at most t Byzantine faults. *Silk* builds on Rondo’s bAVSS-PO abstraction [16].

Silk’s design combines two elements: computation-efficient and post-quantum-friendly sharing and amortized BFT consensus.

- *Computation-efficient and post-quantum-friendly sharing.* We present Mulberry, a primitive that implements the bAVSS-PO interface introduced by Rondo [16]. The weaker bAVSS-PO completion guarantee requires $t + 1$ valid shares at correct parties for later reconstruction, whereas asynchronous complete secret sharing (ACSS) [29], [30], [27] guarantees sharing completion for every correct party. The partial-output guarantee is sufficient for a beacon: each selected contribution must be fixed and reconstructible, but selection need not wait for all correct parties to complete sharing. Mulberry preserves this interface while replacing Breeze’s group-based verification with hashes and finite-field arithmetic, using lightweight degree checks from Shoup and Smart [27]. Mulberry’s share checks reduce verification work and remove Breeze’s discrete-logarithm assumptions. We use post-quantum signatures to make evidence of share acceptance and retention publicly verifiable. HashRand’s public-verification extension [22] likewise uses a public-key infrastructure (PKI) and digital signatures, requiring post-quantum signatures to retain post-quantum security.
- *Amortized BFT consensus and Quorum Release.* *Silk* amortizes consensus by fixing the contributing dealers

once for an entire batch. Rondo [16] already batches sharing but still runs BFT state-machine replication (BFT-SMR) for each beacon output. *Silk* reuses the dealer selection across outputs and uses a new mechanism called Quorum Release (QR) to control when reconstruction of each subsequent output begins. Without QR, Byzantine parties could selectively help one correct replica reconstruct successive outputs ahead of the others. QR requires a quorum of signed completion messages for the preceding output before a correct replica starts the next reconstruction. The quorum requirement ensures that enough correct replicas have completed the preceding output before shares for the next output are released by any correct replica. Replicas can therefore proceed with reconstruction without a new BFT decision for every output.

Silk’s lower computation cost comes with higher communication complexity: its amortized total communication per output is cubic in the committee size. Post-quantum signatures used to authenticate protocol evidence further increase the transmitted data volume. These signatures account for most of *Silk*’s communication in our evaluation (Section VII). This trade-off is intended to improve throughput in settings with sufficient bandwidth or moderately sized committees, where computation can be the limiting resource. HashRand [22] supports this focus by identifying cryptographic computation as a major scalability bottleneck in prior VSS-based asynchronous beacons. Rondo [16] further demonstrates that reducing computation and message counts can improve scalability despite a higher communication complexity. Table I summarizes *Silk*’s guarantees and costs alongside existing beacons. We give the detailed complexity analysis in Section VI-C.

In our WAN evaluation (Section VII), *Silk* achieves higher

throughput than Rondo [16] and Spurt [15], two elegant randomness beacon protocols under partial synchrony. Our CPU and bandwidth measurements indicate that computation, rather than communication bandwidth, is the bottleneck in our experimental setting.

c) Contributions and results: Our main contributions are as follows.

- **Mulberry.** We develop Mulberry, a computation-efficient and post-quantum-friendly bAVSS-PO protocol. Mulberry supports batched sharing and independent secret reconstruction, using hashes and finite-field arithmetic to verify shares. We prove that Mulberry satisfies the bAVSS-PO properties under the model of Section II-A.
- **Quorum Release.** We amortize BFT consensus by reusing one decision across a batch of beacon outputs. Our Quorum Release mechanism ensures that at least $t + 1$ correct replicas have completed the preceding output before any correct replica releases shares for the next reconstruction.
- **Silk.** We present Silk, a computation-efficient, post-quantum-friendly and deterministic randomness beacon under partial synchrony. Silk combines Mulberry with BFT-SMR and Quorum Release. We prove Unpredictability, Bias-resistance, Availability, and Public verifiability under the same model.
- **Implementation and evaluation.** We implement Silk and Mulberry in Rust and evaluate their computation and communication costs. In WAN experiments with 16–121 replicas across eight AWS regions, Silk achieves a throughput of 5.414 outputs/s at $n = 121$.

II. SYSTEM MODEL AND OVERVIEW

A. System Model

We consider n nodes $\mathcal{P} = \{P_1, \dots, P_n\}$ with $n \geq 3t + 1$. A quantum polynomial-time (QPT) adversary $\mathcal{A}$ chooses at most t nodes for corruption before execution. Corrupted nodes may deviate arbitrarily. Nodes communicate over pairwise authenticated and secure channels. The network is partially synchronous [25]. There is a finite global stabilization time (GST) that is unknown to the nodes. Message delays are arbitrary before GST. After GST, messages arrive within a fixed bound Δ .

A standard public-key infrastructure (PKI) gives each node a public/private key pair, binds its identity to its public key, and makes that key available to all nodes. This identity-to-key binding enables public verification: any verifier can authenticate signed protocol evidence without access to the pairwise secure channel. Silk uses post-quantum signatures under PKI for its publicly verifiable evidence. HashRand’s public-verification extension [22] likewise combines a PKI with post-quantum signatures.

Silk assumes a signature scheme Σ that is existentially unforgeable under chosen-message attacks by the QPT adversary (qEUF-CMA). Our implementation uses the Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [31] to instantiate Σ . We also assume a collision-resistant hash function. For the security analysis, we model its domain-separated uses

as independent random oracles [32], with quantum-query access as in the quantum random-oracle model [33], [34].

B. Security Goals

Definition 1 (Randomness beacon): A distributed randomness beacon produces a sequence $b_1, b_2, \dots$, where $b_r \in \{0, 1\}^\kappa$ is the output of beacon round r . We adapt the properties of Spurt [15] and Rondo [16] to our quantum-adversary model and require them against every QPT adversary corrupting at most t parties, except with negligible probability.

Unpredictability. Consider any target round r named before a correct party sends a reconstruction message for that round. Let V_r contain all information available to the adversary before the first such message. For every QPT predictor $\mathcal{A}$,

$$\Pr[\mathcal{A}(V_r) = b_r] \leq \text{negl}(\kappa).$$

Bias-resistance. Under the same choice of r and view V_r , the adversary cannot influence the distribution of b_r . In particular, for $U \leftarrow \{0, 1\}^\kappa$ independent of V_r and every QPT distinguisher $\mathcal{D}$,

$$|\Pr[\mathcal{D}(V_r, b_r) = 1] - \Pr[\mathcal{D}(V_r, U) = 1]| \leq \text{negl}(\kappa).$$

Availability. For every beacon round, each correct party eventually outputs a value.

Public verifiability. Every beacon output eventually has evidence of its correctness that anyone can verify.

Silk defines Unpredictability and Bias-resistance using the adversary’s view before the first correct party sends a reconstruction message for the target round. Spurt [15] additionally requires every honest node to learn an output within a small constant number of rounds after the adversary learns it. Rondo [16] states unpredictability for round $j > i + 1$ given the outputs through round i .

C. Protocol Overview

Silk follows the prepare–agree–reconstruct structure of VSS-based beacons [15], [16]. Figure 1 shows the overall flow of the three phases in one epoch.

a) Prepare: Each dealer runs one Mulberry instance to share B secrets. Each replica that accepts dealer P_j ’s share signs its approval. After collecting exactly $n - t$ signed approvals, dealer P_j forms $\text{cert}_{e,j}$, a validation certificate for its Mulberry instance. At least $n - 2t \geq t + 1$ signers are correct replicas that retained shares for every position, which is enough for later reconstruction. Dealer P_j then distributes $\text{cert}_{e,j}$ to every replica. Each replica verifies and caches the validation certificate. Section IV gives the complete protocol. Once a replica has cached validation certificates for at least $t + 1$ dealers in epoch e , it enters *Agree*.

b) Agree: Replica P_i selects $t + 1$ dealers with cached validation certificates and forms the epoch proposal $E_{e,i}$. BFT-SMR decides one proposal as the common set E_e . This set contains exactly $t + 1$ selected dealers, and BFT-SMR ensures that all correct replicas agree on E_e , which Silk reuses for all B beacon rounds in the epoch.

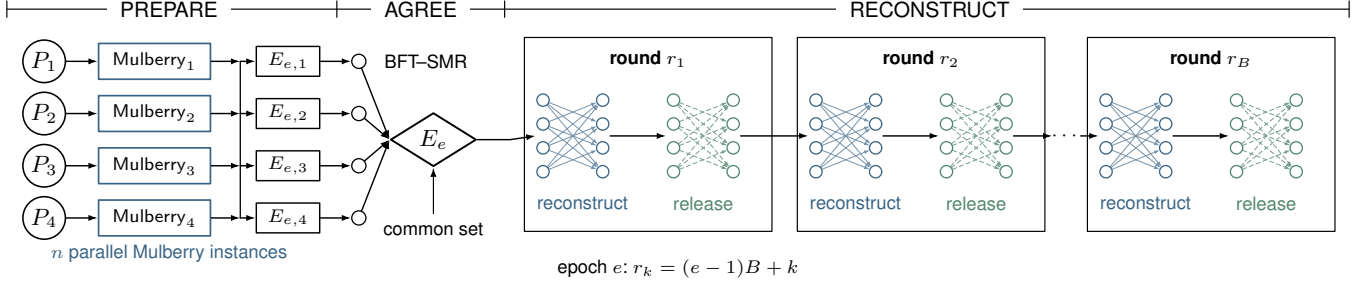


Fig. 1. Silk's prepare-agree-reconstruct workflow for one epoch with $n = 4$. BFT-SMR decides the common set E_e , reused for all B beacon outputs.

TABLE II
FREQUENTLY USED NOTATION.

Notation	Description
$\mathbb{F}$	Finite field used for Shamir sharing.
n, t	Committee size and fault bound, $n \geq 3t + 1$.
κ	Security parameter and bit length of hash outputs and leaf nonces.
B, R	Batch size and degree-check repetitions.
e, r, k	Epoch number, global beacon round, and batch index.
P_i	Party i , acting as a replica in BFT-SMR.
$s_{j,k}$	Secret at position k from dealer P_j .
$v_{j,k,i}$	Share $f_{j,k}(i)$ held by P_i .
$z_{j,k}$	Dealer P_j 's fixed value at batch index k .
$\text{nonce}_{j,k,i}$	Commitment nonce for leaf (j, k, i) .
$E_{e,i}$	Epoch proposal submitted by replica P_i .
E_e	Common set decided by BFT-SMR for epoch e .
τ_e	Digest binding E_e to beacon outputs and signed releases.
$\text{rel}_{i,r}$	Signed release from P_i for output b_r .
b_r	Beacon output for round r .

c) Reconstruct: Once replicas install E_e at the end of *Agree*, they produce the epoch's B beacon outputs one round at a time. Replicas exchange reconstruction messages, recover and combine the selected dealer contributions, and hash the result to obtain b_r . A correct replica then signs a release message that binds the round and b_r . Exactly $n - t$ matching signed releases provide public verification and allow reconstruction of the next output to begin. Section V gives the complete protocol.

III. PRELIMINARIES

A. Shamir Secret Sharing and Reed-Solomon Codes

Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| \geq n + 1$, and let $\alpha_1, \dots, \alpha_n \in \mathbb{F}$ be distinct nonzero evaluation points. We write $p(i)$ for $p(\alpha_i)$ and $i - u$ for $\alpha_i - \alpha_u$.

In Shamir's scheme [35], a dealer shares $s \in \mathbb{F}$ with reconstruction threshold $t + 1$ by sampling a random polynomial f of degree at most t subject to $f(0) = s$ and sending $f(i)$ to P_i . A set $I \subseteq [n]$ of $t + 1$ parties reconstructs

$$s = f(0) = \sum_{i \in I} \lambda_i^I f(i), \quad \lambda_i^I = \prod_{u \in I \setminus \{i\}} \frac{-u}{i - u}.$$

The shares held by any set of at most t parties reveal no information about s .

The share vector $(f(1), \dots, f(n))$ is a Reed-Solomon codeword [36]. For $P \subseteq [n]$, define the Reed-Solomon evaluation code on P by

$$\mathcal{C}_P = \{(p(i))_{i \in P} : p \in \mathbb{F}[X], \deg p \leq t\}.$$

For $|P| \geq t + 1$, this linear code has length $|P|$, dimension $t + 1$, and minimum distance $|P| - t$. A share vector $y_P = (y_i)_{i \in P}$ belongs to $\mathcal{C}_P$ exactly when there exists $p \in \mathbb{F}[X]$ with $\deg p \leq t$ and $y_i = p(i)$ for every $i \in P$. Any $t + 1$ shares uniquely determine p .

B. Lightweight Degree Check

A degree check determines whether the shares of one secret lie on a single degree- t polynomial. Some verifiable secret sharing protocols use polynomial commitments [37], [29] based on discrete logarithms or pairings to check that the shares lie on a polynomial of degree at most t . Shoup and Smart [27] instead use a probabilistic test based on random affine combinations and field arithmetic. GoSSamer [30] follows this lightweight approach with a bivariate-polynomial-based degree check. The test below applies to Shamir sharings and uses only finite-field arithmetic and hashes.

Consider shares $v_1, \dots, v_n$ of a secret s , where P_i holds v_i . In each repetition $\rho \in [R]$, the dealer samples a random polynomial g_ρ of degree at most t and sets $w_{\rho,i} = g_\rho(i)$. Let com be the Merkle root of $(v_i, (w_{\rho,i})_{\rho \in [R]})_{i \in [n]}$. The dealer publishes com before deriving the Fiat-Shamir challenge [38] θ_ρ :

$$\theta_\rho = \text{RO}_{\mathbb{F}}(\text{ShareChal}, \text{ctx}, \text{com}, \rho).$$

Here ShareChal is a domain-separation tag and ctx identifies the sharing instance. For an honest sharing polynomial f , the dealer publishes the response h_ρ of degree at most t :

$$h_\rho(X) = g_\rho(X) + \theta_\rho f(X).$$

Party P_i checks that h_ρ has degree at most t and verifies

$$h_\rho(i) \stackrel{?}{=} w_{\rho,i} + \theta_\rho v_i.$$

Appendix A-A gives the security proof.

C. Batched AVSS with Partial Output

Rondo [16] introduces batched asynchronous verifiable secret sharing with partial output (bAVSS-PO). The main difference between bAVSS-PO and asynchronous complete

secret sharing (ACSS) is the completion guarantee. ACSS [29], [30] additionally guarantees that every correct party can obtain its share, including through share recovery. When a bAVSS-PO sharing protocol completes, it guarantees that at least $t + 1$ correct parties hold valid partial outputs. Each partial output contains one share for each of the B secrets.

Definition 2 (bAVSS-PO security): Adapting Rondo’s interface [16] to our quantum-adversary model, we require the following properties against every static QPT adversary corrupting at most t parties, except with negligible probability.

Privacy. Before a correct party invokes reconstruction, a correct dealer’s secret vector is computationally hidden from the adversary.

Liveness. Every correct party completes the sharing of a correct dealer. If at least $t + 1$ correct parties validate their partial outputs and invoke reconstruction at batch index k , every correct party eventually reconstructs a value for that index.

Correctness. Once $t + 1$ correct parties complete sharing, each position k has a fixed value $z_{j,k}$. For a correct dealer, $z_{j,k} = s_{j,k}$, and every correct reconstruction of position k returns $z_{j,k}$.

Commitment. Once any party completes sharing, at least $t + 1$ correct parties can validate their partial outputs upon receiving the validation certificates. Every correct reconstruction of an index returns the same fixed value.

D. BFT State-Machine Replication

Byzantine fault-tolerant state-machine replication (BFT-SMR) [39] enables replicas to agree on a totally ordered sequence of decided values and execute them in the same order. We call the index of a value in this sequence its BFT-SMR height.

Silk requires the following properties against the QPT adversary of Section II-A.

Definition 3 (BFT-SMR): We model BFT-SMR as deciding one application value x_ℓ at each height ℓ . Silk relies on the following properties.

Safety. No two correct replicas decide different values at the same height (Agreement). If a correct replica executes value x before value y , every correct replica that executes y also executes x before y (Total order).

Liveness. If correct replicas propose valid values for a height, BFT-SMR eventually decides one and every correct replica executes it.

External validity. Every value decided by a correct replica satisfies the application’s validation checks.

Silk instantiates this interface with Simple-IT [40], a signature-free BFT protocol designed to reach agreement with a small number of message delays.

IV. MULBERRY: HASH-BASED BAVSS-PO

A. Construction Overview

Mulberry combines the lightweight degree check of Shoup and Smart [27] with Rondo’s bAVSS-PO interface [16]. In epoch e , dealer P_j shares $s_j = (s_{j,1}, \dots, s_{j,B}) \in \mathbb{F}^B$ using independent degree- t polynomials, allowing each secret to be reconstructed separately. Mulberry replaces Breeze’s

Bulletproof-style commitments with Merkle commitments and finite-field arithmetic, requiring neither discrete-logarithm hardness nor a common reference string. This construction reduces computation at the cost of higher communication complexity. Its validation approvals use post-quantum signatures, whose size further increases communication. Section VII evaluates this computation–communication trade-off.

B. Share Phase

For each position $k \in [B]$, dealer P_j shares the secret $s_{j,k}$ by sampling one degree- t polynomial $f_{j,k}$ with $f_{j,k}(0) = s_{j,k}$. Let R be the number of degree-check repetitions, indexed by $\rho \in [R]$. For each pair (k, ρ) , the dealer independently samples a degree- t mask polynomial $g_{j,k,\rho}$. The polynomial evaluations at party P_i are $v_{j,k,i} = f_{j,k}(i)$ and $w_{j,k,\rho,i} = g_{j,k,\rho}(i)$. Dealer P_j sends each replica P_i the private row

$$\text{row}_{j,i} = (v_{j,k,i}, (w_{j,k,\rho,i})_{\rho \in [R]}, \text{nonce}_{j,k,i})_{k \in [B]}. \quad (1)$$

Each $\text{nonce}_{j,k,i}$ is a fresh κ -bit commitment nonce.

The dealer commits to every private row through a two-level Merkle tree [41]. The entry

$$\text{LD}_j[i, k] = H(\text{nonce}_{j,k,i}, v_{j,k,i}, (w_{j,k,\rho,i})_{\rho \in [R]}) \quad (2)$$

is the public digest for position k in row i . All Bn digests form the table LD_j . The B entries of each row form an inner tree, and the n row roots form an outer tree with root R_j^{msg} . The values that open row i are sent only to P_i . Figure 2 shows the layout.

Only after fixing R_j^{msg} does the dealer derive the Fiat–Shamir challenge $\theta_{j,k,\rho}$. The dealer then publishes the degree- t response polynomial

$$h_{j,k,\rho}(X) = g_{j,k,\rho}(X) + \theta_{j,k,\rho} f_{j,k}(X). \quad (3)$$

The dealer forms the public object

$$\text{pub}_j = \left(\text{LD}_j, (h_{j,k,\rho}(X))_{k \in [B], \rho \in [R]} \right). \quad (4)$$

The message root prevents the dealer from changing shares after learning the challenge. Repeating the test R times lowers the probability that inconsistent shares pass. The fresh nonces prevent an observer from testing guesses for unopened shares. A nonce is revealed only with its position.

At the end of the Share phase, each replica P_i receives $(\text{row}_{j,i}, \text{pub}_j)$.

C. Verify and Confirm Phases

After receiving $\text{row}_{j,i}$ and pub_j , party P_i performs the following checks.

- 1) For every k , party P_i recomputes $\text{LD}_j[i, k]$ using Equation (2) with its private row, and compares the result with the corresponding entry in pub_j . Party P_i also derives R_j^{msg} and the challenges from the digest table.
- 2) Party P_i checks that every response polynomial $h_{j,k,\rho}$ has degree at most t .
- 3) For every (k, ρ) , party P_i checks

$$h_{j,k,\rho}(i) \stackrel{?}{=} w_{j,k,\rho,i} + \theta_{j,k,\rho} v_{j,k,i}. \quad (5)$$

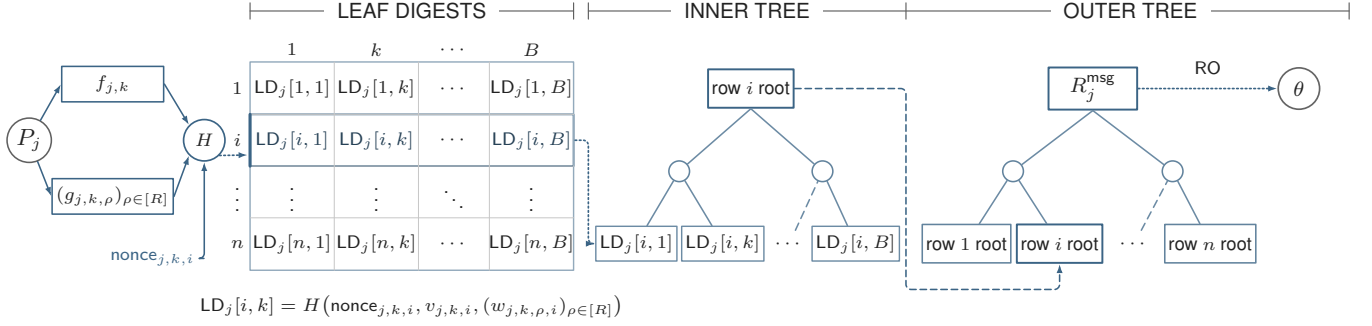


Fig. 2. Mulberry's two-level Merkle tree in the Share phase. Each public digest $LD_j[i, k]$ binds a share, R mask evaluations, and a commitment nonce in private row $row_{j,i}$. Each inner tree contains B digests. The outer tree binds the n row roots in R_j^{msg} , fixed before deriving Fiat–Shamir challenges.

This equation verifies that the private values held by P_i agree with the public response at coordinate i .

If all three checks pass, party P_i retains $row_{j,i}$ and returns approval $a_{e,i,j}$ to the dealer. This approval is its signature on (sid, e, j, pub_j) . A correct party signs at most one public object for each (e, j) . Dealer P_j verifies the approvals and places exactly $n - t$ approvals from distinct parties in the validation certificate $cert_{e,j}$. The dealer then broadcasts $cert_{e,j}$.

On receipt of $cert_{e,j}$, party P_i checks its $n - t$ approvals against the cached pub_j . If they are valid, P_i caches the certificate and, if it has retained $row_{j,i}$, outputs its local shares $(v_{j,1,i}, \dots, v_{j,B,i})$.

Each validation certificate includes at least $n - 2t \geq t + 1$ correct signers that verified and retained their rows. Since a correct replica signs at most one public object per instance, valid certificates for that instance bind the same object unless a signature is forged. Section VI-A proves these properties.

A replica that needs a certified public object but has not received it requests it from the replicas that retained it. Correct replicas serve these requests after accepting the object, independently of whether they have started reconstruction. The requester checks the recovered object against its expected digest and validation certificate, then caches it. Requests and responses use encoded fragments, so that multiple holders need not each send a full copy.

D. Reconstruction

Reconstruction opens one position at a time. To recover dealer P_j 's secret at position k , each replica P_i that retained $row_{j,i}$ sends its share $v_{j,k,i}$ and nonce $nonce_{j,k,i}$ for that position to every replica.

After receiving the share and nonce from P_i , each replica recovers the mask for each repetition ρ using

$$w_{j,k,\rho,i} = h_{j,k,\rho}(i) - \theta_{j,k,\rho}v_{j,k,i}. \quad (6)$$

The replica then recomputes the digest in Equation (2) from the received nonce, share, and recovered masks. The replica accepts the share if the result equals the table entry $LD_j[i, k]$ and discards it otherwise.

Once shares from exactly $t+1$ distinct replicas pass the check, the replica uses the Lagrange interpolation of Section III-A to reconstruct the secret at position k , denoted by $z_{j,k}$. For a correct dealer, $z_{j,k} = s_{j,k}$.

V. THE SILK RANDOMNESS BEACON

Silk divides the global beacon rounds into epochs of B consecutive rounds. For round r , write $e = \lceil r/B \rceil$ for its epoch and $k = r - (e - 1)B \in [B]$ for its position within that epoch.

While reconstructing the outputs of epoch e , replicas may run *Prepare* and *Agree* for epoch $e+1$. Reconstruction proceeds one round at a time, including across epoch boundaries. Quorum Release (QR) requires matching signed releases for the preceding output before the next round begins, as specified in Section V-C. Algorithm 1 summarizes the protocol.

A. Prepare

In epoch e , each party P_j samples $s_j = (s_{j,1}, \dots, s_{j,B}) \leftarrow \mathbb{F}^B$ and runs one Mulberry instance to share these secrets. The dealer forms and broadcasts the validation certificate as described in Section IV-C. Replicas verify and cache the certificate before including the dealer in an epoch proposal.

B. Agree

After caching validation certificates for at least $t+1$ dealers, replica P_i chooses distinct dealers $j_1 < \dots < j_{t+1}$. For each selected dealer P_{j_ℓ} , the proposal records the triple $(j_\ell, H(pub_{j_\ell}), cert_{e,j_\ell})$. The epoch proposal is

$$E_{e,i} = \left(e, \left((j_\ell, H(pub_{j_\ell}), cert_{e,j_\ell}) \right)_{\ell=1}^{t+1} \right). \quad (7)$$

Replica P_i submits $E_{e,i}$ at the BFT–SMR height for epoch e .

Upon receiving a proposal, a correct replica requests any missing public object referenced in it from all signers listed in the corresponding validation certificate, using the retrieval procedure of Section IV-C. Once all referenced public objects are cached, the replica performs the following application validation checks before voting for the proposal in BFT–SMR.

- 1) The proposal contains exactly $t+1$ triples with distinct dealer identifiers.
- 2) For every triple, the digest matches the cached pub_{j_ℓ} , and $cert_{e,j_\ell}$ contains exactly $n - t$ valid approvals from distinct replicas for that public object.

BFT–SMR decides one valid proposal as the common set E_e . Each correct replica installs the decided common set and derives $\tau_e = H(\text{epoch}, sid, E_e)$. The digest τ_e binds the selected

Algorithm 1 Silk at replica P_i .

Require: Session sid , committee $\mathcal{P}$, threshold t , batch size B , Mulberry, and BFT-SMR

Prepare

- 1: **upon** the start of epoch e at P_i :
- 2: sample $\mathbf{s}_i \leftarrow \mathbb{F}^B$, run Mulberry as dealer P_i , and participate in every other dealer's instance as a replica
- 3: **upon** receiving $\text{cert}_{e,j}$ at P_i :
- 4: **if** $\text{cert}_{e,j}$ contains exactly $n - t$ valid approvals from distinct replicas for the cached pub_j **then**
- 5: cache $\text{cert}_{e,j}$ with pub_j

Agree

- 6: **upon** first caching valid certificates for at least $t + 1$ distinct dealers in epoch e :
- 7: choose $j_1 < \dots < j_{t+1}$, form $E_{e,i}$ by (7), and submit it to BFT-SMR
- 8: **upon** receiving proposal $E_{e,\ell}$:
- 9: check that $E_{e,\ell}$ contains exactly $t + 1$ triples with distinct dealer identifiers
- 10: for every triple $(j, d_j, \text{cert}_{e,j})$, check $d_j = H(\text{pub}_j)$ and verify $\text{cert}_{e,j}$ once the public object is cached, retrieving any missing object as in Section V-B
- 11: **if** the proposal and every triple pass these checks **then**
- 12: accept the proposal for BFT-SMR voting
- 13: **upon** BFT-SMR deciding the common set E_e :
- 14: wait until the public object for every selected dealer is cached
- 15: **if** every digest and certificate in E_e is valid for its public object **then**
- 16: install E_e and set $\tau_e \leftarrow H(\text{epoch}, \text{sid}, E_e)$

Reconstruct

- 17: **upon** first having $E_{\lceil r/B \rceil}$ installed and either $r = 1$ or round $r - 1$ locally completed with $n - t$ matching releases:
 - 18: begin reconstruction of round r , with $e \leftarrow \lceil r/B \rceil$ and $k \leftarrow r - (e - 1)B$
 - 19: broadcast $\text{recon}_{i,r}$ from (8) using only retained rows of selected dealers
 - 20: **upon** receiving $\text{recon}_{\ell,r}$ after beginning reconstruction of round r at P_i :
 - 21: **for** each $(j, v_{j,k,\ell}, \text{nonce}_{j,k,\ell})$ in the message **do**
 - 22: **if** $j \in E_e$, no share from P_ℓ is retained for (j, k) , and Mulberry's digest check passes **then**
 - 23: retain the share from P_ℓ for dealer P_j
 - 24: **for** each $j \in E_e$ with exactly $t + 1$ valid shares and no $z_{j,k}$ **do**
 - 25: interpolate these shares at zero and assign the result to $z_{j,k}$
 - 26: **if** every $z_{j,k}$ is reconstructed and P_i has not released round r **then**
 - 27: compute b_r by (9) and broadcast $\text{rel}_{i,r}$ from (10)
 - 28: **upon** receiving $\text{rel}_{\ell,r}$ after completing round r at P_i :
 - 29: **if** it validly signs $(\text{sid}, e, r, \tau_e, b_r)$ and no release from P_ℓ is retained **then**
 - 30: retain $\text{rel}_{\ell,r}$
-

Mulberry instances to the beacon outputs and signed releases of epoch e .

C. Reconstruct

Replica P_i must install E_e before starting reconstruction in epoch e . For the first global round ($r = 1$), P_i begins reconstruction as soon as E_1 is installed. For each subsequent round $r > 1$, P_i must also complete round $r - 1$ locally and receive matching signed releases for $(r - 1, b_{r-1})$ from $n - t$ distinct replicas. This requirement applies both within an epoch and across epoch boundaries.

Once these conditions are met for round r , replica P_i broadcasts a single reconstruction message

$$\text{recon}_{i,r} = \left(r, \left((j, v_{j,k,i}, \text{nonce}_{j,k,i}) \right)_{j \in E_e} \right). \quad (8)$$

Here, j ranges only over selected dealers for which P_i retained $\text{row}_{j,i}$.

To reconstruct $z_{j,k}$ for $j \in E_e$, a replica uses the digest table and response polynomials in the cached pub_j to verify shares of $z_{j,k}$. The replica applies the digest check of Section IV-D and discards any share that fails. After shares from exactly $t + 1$ distinct replicas pass verification, the replica reconstructs $z_{j,k}$ by Lagrange interpolation. The valid senders may differ between secrets.

For each selected dealer, the validation certificate guarantees that at least $t + 1$ correct replicas retained valid shares. The retained shares suffice for reconstruction even if faulty replicas withhold their shares.

After reconstructing every selected dealer's secret, the replica computes

$$b_r = H \left(\text{beacon}, \text{sid}, e, r, \tau_e, \sum_{j \in E_e} z_{j,k} \right). \quad (9)$$

The $t + 1$ selected dealers include a correct dealer whose secret is sampled independently and uniformly. Mulberry hides this secret until a correct replica sends a reconstruction message for the round. The validation certificates already bind the selected dealers to their contributions. Section VI-B uses these properties to prove Unpredictability and Bias-resistance.

We design Quorum Release (QR) to require evidence that a quorum has completed the current round before reconstruction of the next round begins. After completing round r , replica P_i forms and broadcasts the signed release

$$\begin{aligned} \sigma_{i,r}^{\text{rel}} &= \text{Sign}_i(\text{release}, \text{sid}, e, r, \tau_e, b_r), \\ \text{rel}_{i,r} &= (\text{sid}, e, r, \tau_e, b_r, i, \sigma_{i,r}^{\text{rel}}). \end{aligned} \quad (10)$$

A correct replica signs at most one output per round and counts a received release only if the signature is valid and all signed

fields match its locally completed output. Once it has collected matching releases from $n - t$ distinct replicas, it may enter round $r + 1$, provided that the common set for the epoch containing that round is installed.

Without QR, advancing after local completion alone permits a message-scheduling attack before GST. In particular, if the adversary's t corrupted replicas and one correct replica hold valid shares for all selected dealers, the adversary can deliver reconstruction messages promptly within this group while delaying them to other correct replicas. The group then has the $t + 1$ shares per secret needed to reconstruct successive outputs, while the other correct replicas remain in earlier rounds.

QR prevents this group from advancing alone: the $n - t$ matching releases required for round r include at least $n - 2t \geq t + 1$ from correct replicas that have completed that round. Thus, before any correct replica sends shares for round $r + 1$, at least $n - 2t$ correct replicas have computed b_r .

Han and Yu [42] distinguish length-advantage from time-advantage in delivery fairness. QR addresses length-advantage relative to a quorum of correct replicas. Individual replicas can remain behind as the quorum advances, with their completion times determined by message delivery.

D. Optimizations

a) *Precomputing response evaluations:* Mask recovery in Equation (6) requires evaluating each public response polynomial at the senders' evaluation points. A replica uses standard fast multipoint evaluation [43] to compute these values together. The algorithm builds a *product tree* by multiplying the factors $X - i$ for these evaluation points. The algorithm then computes a *remainder tree* by taking polynomial remainders down the product tree. At a leaf, the remainder modulo $X - i$ is the response value at i . The product tree depends only on the evaluation points and can be reused across dealers and rounds. The response values are specific to each dealer and round, but can be computed from the cached public object before shares arrive. Each received share still requires mask recovery and the full digest check of Equation (2).

b) *Reusing interpolation weights:* After verifying shares of $z_{j,k}$ from $t + 1$ distinct replicas using the digest check in Equation (2), the replica computes their Lagrange weights at zero using standard fast polynomial algorithms [43]. Each weight specifies how much an accepted share contributes to the reconstructed secret. These weights depend only on the set of accepted senders, so the replica caches them for reuse by dealers and later rounds with the same sender set.

Within a round, the replica can also group dealers whose accepted sender sets coincide. For each sender, the replica adds the verified shares from the grouped dealers and interpolates once for the group. By linearity, this yields the sum of the group's secrets directly. Summing these results gives $\sum_{j \in E_e} z_{j,k}$ in Equation (9), without recovering each secret separately. Dealers with different sender sets are handled separately, allowing each group to be reconstructed as soon as its shares have been verified.

VI. SECURITY AND COMPLEXITY

We first establish the security of Mulberry and then use its guarantees to prove the beacon properties of Silk. The analysis follows the model of Section II-A, with parameters chosen as in Appendix A to make cryptographic losses negligible. Supporting lemmas are given in Appendices A and B.

A. Security of Mulberry

Lemma 1: Suppose every response polynomial $h_{j,k,\rho}$ in the public object has degree at most t . Fix a batch index k . Except with negligible probability, all shares accepted by the reconstruction check at index k lie on one polynomial of degree at most t .

Proof sketch: The reconstruction check recovers masks satisfying Equation (5) and checks the recomputed digest against the corresponding entry in the public digest table. Each accepted share therefore satisfies the degree check. If at most $t + 1$ shares from distinct replicas are accepted, the conclusion follows directly by interpolation. Otherwise, interpolate a polynomial p through any $t + 1$ accepted shares from distinct replicas. If another accepted share disagreed with p , these $t + 2$ shares could not all lie on such a polynomial: the first $t + 1$ already determine p uniquely. By Lemma 4, these inconsistent shares are accepted with probability at most ϵ_{qdeg} , which is negligible. ■

Theorem 1 (Mulberry): Mulberry satisfies Definition 2.

Proof sketch: **Privacy.** For a correct dealer, at most t evaluations of each randomly sampled sharing polynomial f have a distribution independent of the shared secret $f(0)$. Thus corrupted shares alone reveal no information about the secrets. The public objects must preserve this privacy. Lemma 5 gives one simulator for the execution with all correct dealers, without knowing their unopened secrets. The adversary can distinguish this simulation from the real execution only with negligible advantage. Each batch index uses independent sharing polynomials, masks, and nonces, so reconstructing other secrets preserves this guarantee.

Liveness. A correct dealer sends valid rows and public objects to all correct replicas. Under partial synchrony, the dealer eventually receives $n - t$ approvals and broadcasts the certificate. Every correct replica consequently completes sharing. If $t + 1$ correct replicas validate their partial outputs and invoke reconstruction, they broadcast their shares and nonces. Every correct replica receives the public object from the dealer or retrieves it from a correct signer, verifies the received shares, and reconstructs the secret by interpolation.

Correctness. Consider any two valid certificates for the same instance. Each contains $n - t$ approvals, so their signer sets intersect in at least $n - 2t > t$ replicas. A correct replica in this intersection signs at most one public object per instance. Both certificates therefore refer to that same object, although they may contain different sets of approvals. Their correct signers retain at least $t + 1$ accepted shares. By Lemma 1, the accepted shares determine a unique polynomial $p_{j,k}$, and every correct reconstruction returns $p_{j,k}(0)$. For a correct dealer, $p_{j,k}$ is the dealer's original sharing polynomial, so the reconstructed value is $s_{j,k}$.

Commitment. A party completes sharing only after obtaining a certificate with $n - t$ approvals. At least $n - 2t \geq t + 1$ signers are correct and have checked and retained their rows. With this certificate, they can validate their partial outputs without the dealer's assistance. As shown above, every valid certificate refers to the same object, whose accepted shares determine one polynomial per index. Each polynomial's constant term therefore fixes the value returned by every correct reconstruction. ■

B. Security of Silk

Lemma 2: A correct replica sends reconstruction shares for round $r > 1$ only after at least $n - 2t$ correct replicas have computed b_{r-1} . If all correct replicas obtain and verify the common set and its public objects for round r and compute the same b_{r-1} , they all eventually begin reconstruction in round r .

Proof sketch: A correct replica first verifies $n - t$ matching signed releases for its preceding output. At least $n - 2t$ signers are correct, and each signs only after computing that output. Conversely, when all correct replicas compute b_{r-1} , they broadcast matching releases. Every correct replica eventually receives at least $n - t$ matching releases. Once it has also verified the common set and public objects, it meets all conditions for reconstruction in Section V-C. ■

Theorem 2 (Silk): Suppose BFT-SMR satisfies Definition 3 with the application validation checks of Section V-B. Then Silk satisfies Definition 1.

Proof sketch: Availability. Mulberry Liveness supplies certificates for valid epoch proposals. Each certificate has at least $n - 2t \geq t + 1$ correct signers retaining shares and serving the public object, so correct replicas can complete the proposal checks. BFT Agreement, External validity, and Liveness then give them a common set E_e of $t + 1$ certified dealers, with enough retained shares to reconstruct every selected contribution.

We induct on r to show that all correct replicas compute the same b_r . For $r = 1$, they begin reconstruction after verifying E_1 and its public objects. Mulberry Liveness, Correctness, and Commitment ensure completion with the same output. Suppose they all compute b_{r-1} . By the liveness statement of Lemma 2, they eventually begin round r once they obtain and verify its common set and public objects. Mulberry again ensures completion with the same b_r . At epoch boundaries, the preceding argument for epoch proposals supplies the new common set and public objects, so the induction also covers the first round of each epoch.

Unpredictability and Bias-resistance. For a round r selected before its first correct reconstruction message, Lemma 6 bounds the advantage of distinguishing b_r from an independent uniform string, given the adversary's view V_r , by the negligible quantity ϵ_{ind} . This establishes Bias-resistance. A predictor succeeding with probability p gives a distinguisher by comparing its prediction with the challenge. Its advantage is $|p - 2^{-\kappa}|$, so $p \leq 2^{-\kappa} + \epsilon_{\text{ind}}$, establishing Unpredictability.

Public verifiability. Each correct replica computes the same output and signs Equation (10). The resulting $n - t$ or more signatures meet the public verification conditions of Section V-C. ■

TABLE III
SILK'S AMORTIZED COST PER BEACON OUTPUT BY PHASE.

Phase	Communication	Computation
Prepare	$O(\kappa R n^3)$	$O(R n^3)$
Agree	$O(\kappa n^3)$	$O(R n^3)$
Reconstruct	$O(\kappa n^3)$	$O(R n^{1+\log_2 3} \log n)$

C. Complexity

Table III reports amortized costs per beacon output. Communication counts bits sent by correct replicas across the committee. Computation counts field operations and hash input blocks per replica. κ is the security parameter. We use $t = \Theta(n)$, $R \geq 1$, and $B = \Omega(\log n)$. Our implementation uses one degree-check repetition ($R = 1$). Table I treats R as a constant.

a) Prepare: Prepare includes all n dealers' Mulberry instances. Each dealer's public object contains Bn digests and BR degree- t response polynomials, occupying $O(\kappa BR n)$ bits. Sending this object to all n replicas costs $O(\kappa BR n^2)$ bits per instance. Across n instances, dividing by the B outputs gives $O(\kappa R n^3)$ bits per output. Private rows, certificates, and public-data retrieval fit within this bound under the stated batch-size condition.

Each replica verifies n dealer certificates, each containing $n - t$ signatures. These $O(n^2)$ verifications each process a public object of $O(BR n)$ hash input blocks when message hashes are not reused. The resulting $O(BR n^3)$ work per epoch amortizes to $O(R n^3)$ per replica per output. The $O(R n^3)$ bound also covers the polynomial work for sharing, verification, and public-data recovery.

b) Agree: A proposal contains one certificate per selected dealer, using $O(\kappa n^2)$ bits. For Simple-IT [40], the BFT-SMR instantiation used in our evaluation, broadcasting a proposal of L_e bits costs $O(n L_e + \kappa n^2 \log n)$ bits. Substituting $L_e = O(\kappa n^2)$ gives $O(\kappa n^3)$ bits for an epoch decision and an amortized cost of $O(\kappa n^3 / B)$, for which the table reports the upper bound $O(\kappa n^3)$. Each replica checks $O(n^2)$ certificate signatures on public objects of $O(\kappa BR n)$ bits. Without reusing message hashes, this takes $O(BR n^3)$ hash input blocks per epoch, or $O(R n^3)$ per output. Proposal coding and the BFT control messages fit within this bound. One decision serves the epoch's B outputs. Rondo [16] instead uses a BFT decision for each output.

c) Reconstruct: Each replica sends a share and nonce for every selected dealer to all replicas, giving $O(\kappa n^3)$ bits per output. Local mask recovery makes this payload independent of the repetition count. Multipoint evaluation, fast interpolation, and leaf checks require $O(R n^{1+\log_2 3} \log n)$ computation per output, using the fast polynomial algorithms in Section V-D [43]. The release exchange adds $O(\kappa n^2)$ bits. Each replica signs its output once and performs $O(n)$ release-signature verifications. Public verification also requires $O(n)$ signature checks.

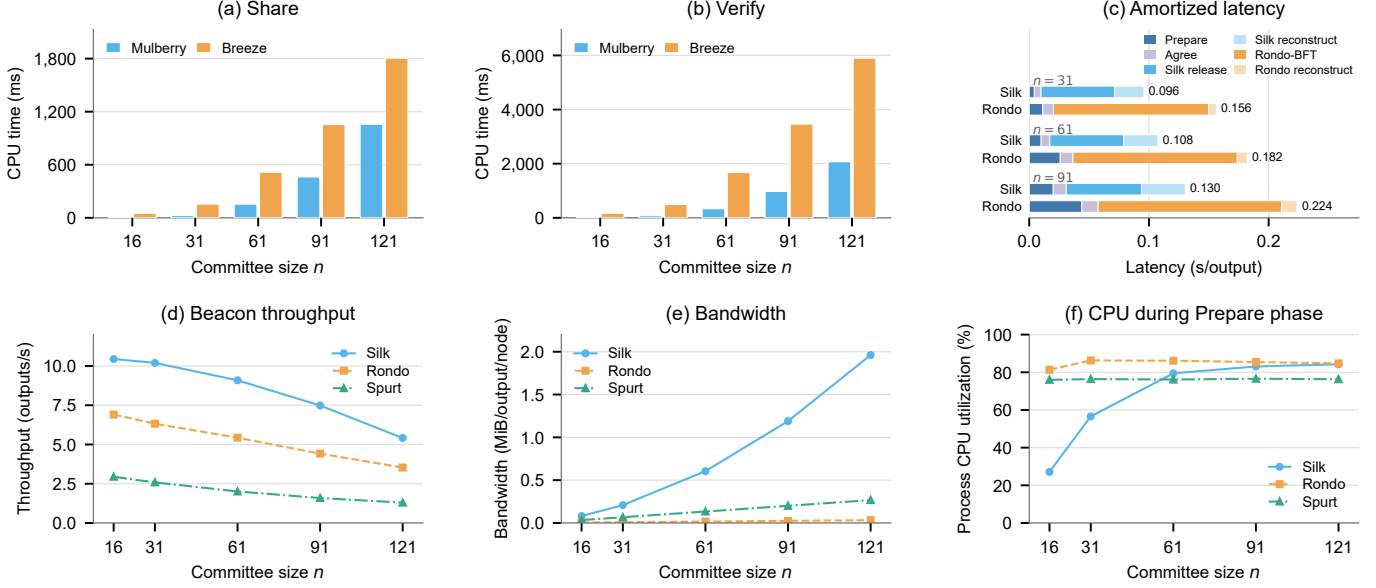


Fig. 3. bAVSS-PO computation costs and beacon performance on AWS. (a)–(b) Mulberry and Breeze CPU time per batch ($B = 2n$): (a) Share by one dealer, (b) Verify for all n receivers, including validation certificate generation. (c) Amortized latency breakdown for Silk and Rondo at $n \in \{31, 61, 91\}$. For each phase, latency is averaged over replicas and divided by the number of outputs. Rondo’s BFT-reconstruction overlap is counted once. (d) Beacon throughput of Silk, Rondo, and Spurt. (e) Bandwidth, measured as sent data in MiB/output per node. (f) Process CPU utilization during the Prepare phase, normalized to 2 vCPUs. All panels report arithmetic means of three runs.

VII. IMPLEMENTATION AND EVALUATION

We implement Mulberry and Silk in Rust.¹ Mulberry uses the scalar-field arithmetic provided by `curve25519-dalek` and `BLAKE3` from the `blake3` crate for hashing. We set $R = 1$ and use 256-bit hash outputs and 256-bit leaf nonces. For post-quantum signatures, we use ML-DSA-65 [31] from the `mldsa-native` v2.0.0 library.² Our implementation uses Simple-IT [40] for BFT-SMR. Our communication layer uses persistent TCP connections with `mio` for asynchronous network I/O and `postcard` for message serialization. We also implement Rondo [16] and Spurt [15], two elegant randomness beacon protocols under partial synchrony, within the same framework for comparison.

Remark. To ensure a fair comparison, our Spurt implementation uses data pre-aggregation and multi-exponentiation from Spurt [15]. Our Rondo implementation precomputes reconstruction coefficients as in Rondo [16]. It also overlaps BFT processing with reconstruction.

A. Experimental Setup

We use Amazon EC2 `t3a.medium` instances with 2 vCPUs and 4 GiB of memory. Each replica occupies one instance, with committees spread across eight AWS regions: Canada (Central), Europe (Ireland), US West (N. California), US East (N. Virginia), US West (Oregon), US East (Ohio), Asia Pacific (Singapore), and Asia Pacific (Tokyo). We evaluate committees of size $n \in \{16, 31, 61, 91, 121\}$ with $t = (n - 1)/3$. Replicas are assigned to the eight regions in round-robin order, so

regional replica counts differ by at most one. Each protocol runs three times at every committee size and all reported measurements are arithmetic means of the three runs. For beacon experiments, the cryptographic worker pool uses a cooperative CPU budget targeting 80% of the 2 vCPUs, leaving capacity for message handling and other protocol work. Without this budget, cryptographic tasks could delay message processing.

B. Evaluation Results

a) bAVSS-PO comparison: Figures 3(a)–(b) compare Mulberry and Breeze with $B = 2n$. On a single EC2 `t3a.medium` instance, we measure the CPU time used by one dealer and all n receivers. Share includes generating shares and public verification data for a batch. Verify includes checking all receivers’ shares and generating the validation certificate. For each batch, Share reports one dealer’s CPU time, and Verify sums the CPU time across all receivers.

Across the five committee sizes, Mulberry reduces Share CPU time by 41.4–92.2% and Verify CPU time by 64.9–87.6%. Mulberry’s speedup over Breeze decreases as n grows, consistent with its higher computational complexity. Nevertheless, even at $n = 121$, Mulberry remains $1.71\times$ faster in Share and $2.85\times$ faster in Verify.

At $n = 121$, Mulberry reduces Share CPU time from 1804.69 to 1058.00 ms and Verify CPU time from 5897.69 to 2068.21 ms compared with Breeze. Most of the savings come from Verify and help explain Silk’s shorter Prepare phase in the beacon evaluation below.

b) Beacon throughput and latency: Figures 3(d) and (c) compare beacon throughput and amortized latency, with $B =$

¹<https://github.com/hby-star/Silk-Implementation>

²<https://github.com/pq-code-package/mldsa-native>

50 for Silk and Rondo. Silk achieves higher throughput than Rondo and Spurt at all five sizes. At $n = 121$, Silk, Rondo, and Spurt produce 5.414, 3.532, and 1.296 outputs/s, respectively. Silk’s throughput is $1.53\times$ that of Rondo and $4.18\times$ that of Spurt.

Figure 3(c) breaks down the amortized latency for Silk and Rondo at $n \in \{31, 61, 91\}$. In Rondo’s pipeline, reconstruction partially overlaps with the preceding BFT work. The breakdown counts overlapping intervals once in the combined Rondo-BFT and reconstruction time. At $n = 91$, Silk’s amortized latency is 0.130 s/output, compared with 0.224 s/output for Rondo, a reduction of 41.7%.

The latency reduction comes mainly from Prepare and the work after the first BFT decision. At $n = 91$, Silk spends 20.0 ms/output in Prepare, compared with 43.9 ms/output for Rondo, consistent with the computation savings in Figures 3(a) and (b). Reusing one BFT decision across the batch yields a larger saving: subsequent outputs require 62.9 ms/output for Silk’s signed releases, compared with 153.3 ms/output for Rondo’s subsequent BFT decisions. Both differences persist at $n = 31$ and $n = 61$.

c) Computation–Communication Trade-off: Figure 3(e) shows that Silk’s throughput advantage comes with substantially higher communication costs. At $n = 121$, each Silk node sends an average of 1.961 MiB/output, compared with 0.267 MiB for Spurt and 0.033 MiB for Rondo. Under limited bandwidth, these communication costs may become the throughput bottleneck, reducing Silk’s measured advantage over Rondo and Spurt as the committee grows. We remark that the post-quantum signatures used to authenticate Silk’s public evidence account for most of its communication in this configuration. Each ML-DSA-65 signature occupies 3,309 bytes, about $34.5\times$ the 96-byte BLS aggregate signature used in our Rondo implementation. This size difference is compounded by the $n-t$ individual signatures in each Mulberry validation certificate, whereas Rondo combines its approvals into one aggregate signature. At $n = 121$ and $B = 50$, signature payloads in approvals, certificate broadcasts and retrievals, decision messages, and signed releases total 1.007 MiB/output per node, accounting for 51.37% of the measured communication.

In our experiments, Silk’s average outgoing bandwidth at $n = 121$ is 89.1 Mbps per node, well below the network burst bandwidth of up to 5 Gbps for `t3a.medium`.³ Figure 3(f) shows that Rondo and Spurt already exhibit CPU utilization near the 80% cryptographic CPU budget at $n = 16$ in the Prepare phase, while Silk approaches this level at $n \geq 61$. This suggests that Rondo and Spurt encounter computational limits at smaller committee sizes than Silk. Combined with the average bandwidth usage, these results support computation as the earlier bottleneck in our configuration. Silk is therefore well suited to applications with limited computational resources and either ample bandwidth or moderately sized committees, such as a consortium blockchain with a small validator committee running on commodity servers.

VIII. RELATED WORK

A. Verifiable Secret Sharing

Shamir secret sharing [35] and the VSS constructions of Feldman [44] and Pedersen [45], along with Schoenmakers’ PVSS [5], form the basis of many beacon designs. SCRAPE [17] builds on Schoenmakers’ PVSS, using a test based on Reed–Solomon dual codes to verify n shares with $O(n)$ exponentiations. Extending VSS to asynchronous networks has led to Cachin et al.’s AVSS [46], hbACSS [29], and Bingo [47]. Atapoor et al. [48] develop a unified framework for batched and packed VSS and PVSS in synchronous networks. Das et al. [49] simplify VSS constructions for synchronous and asynchronous networks while ensuring public verifiability.

Share verification in such protocols can use polynomial commitments, including Kate–Zaverucha–Goldberg (KZG) commitments [37] and Bulletproofs [26]. For hash-based verification, Shoup and Smart [27] develop lightweight AVSS using hashes and field-based degree checks, and GoSSamer [30] extends this approach to bivariate sharing.

B. Synchronous and Partially Synchronous Beacons

Threshold-signature-based beacons, including DFINITY [12], Internet Computer consensus [24], and drand [11], derive randomness from threshold signatures. Generating these signatures first requires establishing a distributed signing key through DKG [13], [14], which adds a costly setup phase.

Beacons based on distributed verifiable random functions (DVRFs) [50] combine partial function evaluations to obtain a pseudorandom output and a publicly verifiable proof. These constructions require a DKG phase [50], [9] to establish the key shares used for evaluation, adding setup overhead before beacon generation can begin.

VSS-based beacons generate randomness by sharing and reconstructing random contributions. In synchronous networks, SCRAPE [17] and HydRand [8] use PVSS for this purpose. Within this family, batching and reconfiguration address different deployment needs. ALBATROSS [51] uses PVSS to produce batches of random values with an authenticated public bulletin board. RandPiper [18] and OptRand [19] support committee reconfiguration under synchrony.

To operate under partial synchrony, Spurt [15] combines pairing-based PVSS with BFT–SMR, achieving $O(\kappa n^2)$ amortized system communication per output. Rondo [16] amortizes sharing across multiple outputs through batched asynchronous verifiable secret sharing with partial output (bAVSS-PO). Its Breeze construction uses Bulletproof-style commitments and allows each secret in a batch to be reconstructed independently. Rondo combines this batched sharing with dynamic BFT–SMR to support reconfiguration.

The security of PVSS-based beacons also depends on whether corruptions can occur during execution. Bacho and Loss [52] analyze aggregatable PVSS in the algebraic group model and establish adaptive security for the resulting Spurt and OptRand beacons. GRandomLine [20] addresses adaptive security together with the cost of DKG and beacon generation. It requires synchrony during preprocessing, but each subsequent output takes one asynchronous communication round.

³<https://aws.amazon.com/ec2/instance-types/t3/>

Timed primitives offer a different way to obtain randomness even when a party withholds information needed to generate an output. RandRunner [53] uses trapdoor verifiable delay functions (VDFs), while Bicorn [54] uses homomorphic timed commitments.

C. Asynchronous Beacons

Threshold cryptography also supports randomness generation in asynchronous networks. Cachin et al. [6] use it to construct a shared coin for asynchronous Byzantine agreement. Establishing keys in this setting is addressed by practical asynchronous DKG protocols [14], which can provide the setup for threshold-signature beacons.

For beacons that reconstruct random contributions, agreement determines which contributions can be safely revealed. RandShare [10] combines VSS with asynchronous Byzantine agreement to select recoverable contributions before reconstruction. Freitas et al. [55] use approximate agreement to generate randomness with a small probability of disagreement. HashRand [22] also uses approximate agreement, with hash-based batched asynchronous weak verifiable secret sharing (BAwVSS) and Gather. It improves throughput through pipelining and reduces communication through AnyTrust sampling. Sampling c dealers lowers its amortized system communication per output from $O(\kappa n^3 \log n)$ to $O(\kappa c n^2 \log n)$.

These savings come with Monte Carlo failure probabilities from sampling and approximate agreement. HashRand analyzes security against adaptive quantum adversaries, but its sampled variant requires secure erasures in this model. Without erasures, it must use the higher-cost unsampled variant [22]. Rubato [23] combines hash-based batched ACSS with asynchronous SMR to obtain Las Vegas guarantees: agreement is preserved, while termination depends on randomized consensus and has no deterministic round bound.

IX. CONCLUSION AND FUTURE DIRECTIONS

We presented Silk, a computation-efficient, post-quantum-friendly and deterministic randomness beacon under partial synchrony. We developed Mulberry, a new construction that implements the bAVSS-PO interface with hash-based share verification. Silk amortizes the cost of BFT consensus across multiple beacon outputs and uses a new mechanism called Quorum Release for round control. We prove Silk's beacon properties under the stated system model and BFT assumptions. Our evaluation shows lower computation costs and higher beacon throughput than prior works. Silk is therefore well suited to settings where computation is the main bottleneck, although its cubic communication complexity limits scalability.

Future work will focus on lowering communication complexity and reducing post-quantum signature overhead while retaining efficient share verification. Committee reconfiguration and security against adaptive adversaries are further directions for supporting long-running services. Both require extending the current protocol and proofs beyond a fixed committee and static corruption.

REFERENCES

- [1] M. O. Rabin, "Transaction protection by beacons," *Journal of Computer and System Sciences*, vol. 27, no. 2, pp. 256–267, 1983.
- [2] S. Azouvi, P. McCorry, and S. Meiklejohn, "Winning the caucus race: Continuous leader election via public randomness," 2018. [Online]. Available: <https://arxiv.org/abs/1801.07965>
- [3] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.
- [4] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.
- [5] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *Advances in Cryptology—CRYPTO '99*, ser. Lecture Notes in Computer Science, vol. 1666. Springer, 1999, pp. 148–164.
- [6] C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography," *Journal of Cryptology*, vol. 18, no. 3, pp. 219–246, 2005.
- [7] 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*. Association for Computing Machinery, 2016, pp. 31–42.
- [8] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "HydRand: Efficient continuous distributed randomness," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 73–89.
- [9] K. Choi, A. Manoj, and J. Bonneau, "SoK: Distributed randomness beacons," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 75–92.
- [10] E. Syta, P. Jovanovic, E. Kokoris Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 444–460.
- [11] drand Project, "drand: A distributed randomness beacon daemon," GitHub repository, 2020, accessed 2026-07-18. [Online]. Available: <https://github.com/drand/drand>
- [12] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series: Consensus system," 2018. [Online]. Available: <https://arxiv.org/abs/1805.04548>
- [13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology*, vol. 20, no. 1, pp. 51–83, 2007.
- [14] S. Das, T. Yurek, Z. Xiang, A. Miller, L. Kokoris-Kogias, and L. Ren, "Practical asynchronous distributed key generation," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2518–2534.
- [15] 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)*. IEEE, 2022, pp. 2502–2517.
- [16] X. Meng, X. Sui, Z. Yang, K. Rong, W. Xu, S. Chen, Y. Yan, and S. Duan, "Rondo: Scalable and reconfiguration-friendly randomness beacon," in *Proceedings 2025 Network and Distributed System Security Symposium*. Internet Society, 2025.
- [17] I. Cascudo and B. David, "SCRAPE: Scalable randomness attested by public entities," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science, vol. 10355. Springer, 2017, pp. 537–556.
- [18] 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*. Association for Computing Machinery, 2021, pp. 3502–3524.
- [19] A. Bhat, N. Shrestha, A. Kate, and K. Nayak, "OptRand: Optimistically responsive reconfigurable distributed randomness," in *Proceedings 2023 Network and Distributed System Security Symposium*. Internet Society, 2023.
- [20] R. Bacho, C. Lenzen, J. Loss, S. Ochseneither, 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*. Association for Computing Machinery, 2024, pp. 941–955.
- [21] 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.

- [22] A. Bandrupalli, 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*. Association for Computing Machinery, 2024, pp. 2621–2635.
- [23] L. Yang, T. Chong, J. Liu, J. Cui, G. Xu, Y. Bai, L. Zhang, and T. Luo, “Rubato: Efficient post-quantum asynchronous distributed randomness beacon with integrated consensus,” *IEEE Transactions on Dependable and Secure Computing*, vol. 23, no. 3, pp. 7012–7025, 2026.
- [24] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup, and D. Williams, “Internet computer consensus,” in *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*. Association for Computing Machinery, 2022, pp. 81–91.
- [25] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the presence of partial synchrony,” *Journal of the ACM*, vol. 35, no. 2, pp. 288–323, 1988.
- [26] 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)*. IEEE, 2018, pp. 315–334.
- [27] V. Shoup and N. P. Smart, “Lightweight asynchronous verifiable secret sharing with optimal resilience,” *Journal of Cryptology*, vol. 37, no. 3, p. 27, 2024, full version: <https://eprint.iacr.org/2023/536>.
- [28] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.
- [29] T. Yurek, L. Luo, J. Fairuze, A. Kate, and A. Miller, “hbACSS: How to robustly share many secrets,” in *Proceedings 2022 Network and Distributed System Security Symposium*. Internet Society, 2022.
- [30] X. Xing, Y. Liu, B. Liao, J. Liu, B. Hu, X. Lin, Y. Lu, and T. Zhang, “GoSSamer: Lightweight and linear-communication asynchronous (dynamic proactive) secret sharing and the applications,” in *2026 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2026, pp. 2482–2501.
- [31] National Institute of Standards and Technology, “Module-lattice-based digital signature standard,” U.S. Department of Commerce, Tech. Rep. FIPS 204, Aug. 2024.
- [32] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *Proceedings of the 1st ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 1993, pp. 62–73.
- [33] 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*, ser. Lecture Notes in Computer Science, vol. 7073. Springer, 2011, pp. 41–69.
- [34] D. Unruh, “Non-interactive zero-knowledge proofs in the quantum random oracle model,” in *Advances in Cryptology—EUROCRYPT 2015*, ser. Lecture Notes in Computer Science, vol. 9057. Springer, 2015, pp. 755–784.
- [35] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [36] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
- [37] A. Kate, G. M. Zaverucha, and I. Goldberg, “Constant-size commitments to polynomials and their applications,” in *Advances in Cryptology—ASIACRYPT 2010*, ser. Lecture Notes in Computer Science, vol. 6477. Springer, 2010, pp. 177–194.
- [38] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Advances in Cryptology—CRYPTO ’86*, ser. Lecture Notes in Computer Science, vol. 263. Springer, 1987, pp. 186–194.
- [39] 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*. Association for Computing Machinery, 2019, pp. 347–356.
- [40] Q. Yu, J. Villacis, G. Losa, Z. Xiang, and X. Wang, “Simple-IT: Practical low-latency signature-free BFT consensus,” 2026. [Online]. Available: <https://arxiv.org/abs/2606.14404>
- [41] R. C. Merkle, “A certified digital signature,” in *Advances in Cryptology—CRYPTO ’89*, ser. Lecture Notes in Computer Science, vol. 435. Springer, 1990, pp. 218–238.
- [42] R. Han and J. Yu, “Fair delivery of decentralised randomness beacon,” in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 13950. Springer, 2023, pp. 218–234.
- [43] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed. Cambridge University Press, 2013.
- [44] P. Feldman, “A practical scheme for non-interactive verifiable secret sharing,” in *28th Annual Symposium on Foundations of Computer Science*. IEEE, 1987, pp. 427–437.
- [45] T. P. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in *Advances in Cryptology—CRYPTO ’91*, ser. Lecture Notes in Computer Science, vol. 576. Springer, 1992, pp. 129–140.
- [46] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Stroh, “Asynchronous verifiable secret sharing and proactive cryptosystems,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2002, pp. 88–97.
- [47] 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*, ser. Lecture Notes in Computer Science, vol. 14081. Springer, 2023, pp. 39–70.
- [48] S. Atapoor, K. Bagheri, G. Nicolas, R. Pedersen, and J. Spiessens, “Batched and packed (publicly) verifiable secret sharing: A unified framework and applications,” *Cryptology ePrint Archive, Paper 2025/2018*, 2025. [Online]. Available: <https://eprint.iacr.org/2025/2018>
- [49] S. Das, Z. Xiang, A. Tomescu, A. Spiegelman, B. Pinkas, and L. Ren, “Verifiable secret sharing simplified,” in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 633–651.
- [50] D. Galindo, J. Liu, M. Ordean, and J.-M. Wong, “Fully distributed verifiable random functions and their application to decentralised random beacons,” in *2021 IEEE European Symposium on Security and Privacy*. IEEE, 2021, pp. 88–102.
- [51] I. Cascudo and B. David, “ALBATROSS: Publicly attestable batched randomness based on secret sharing,” in *Advances in Cryptology—ASIACRYPT 2020*, ser. Lecture Notes in Computer Science, vol. 12492. Springer, 2020, pp. 311–341.
- [52] R. Bacho and J. Loss, “Adaptively secure (aggregatable) PVSS and application to distributed randomness beacons,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2023, pp. 1791–1804.
- [53] P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, and E. R. Weippl, “RandRunner: Distributed randomness from trapdoor VDFs with strong uniqueness,” in *Proceedings 2021 Network and Distributed System Security Symposium*. Internet Society, 2021.
- [54] K. Choi, A. Arun, N. Tyagi, and J. Bonneau, “Bicom: An optimistically efficient distributed randomness beacon,” in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 13950. Springer, 2023, pp. 235–251.
- [55] L. Freitas, P. Kuznetsov, and A. Tonkikh, “Distributed randomness from approximate agreement,” in *36th International Symposium on Distributed Computing (DISC 2022)*, ser. Leibniz International Proceedings in Informatics, vol. 246. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2022, pp. 24:1–24:21.
- [56] M. Zhandry, “A note on the quantum collision and set equality problems,” *Quantum Information and Computation*, vol. 15, no. 7–8, pp. 557–567, 2015.
- [57] J. Don, S. Fehr, C. Majenz, and C. Schaffner, “Online-extractability in the quantum random-oracle model,” in *Advances in Cryptology—EUROCRYPT 2022*. Springer, 2022, pp. 677–706, full version: <https://arxiv.org/abs/2103.03085v2>.
- [58] —, “Security of the Fiat–Shamir transformation in the quantum random-oracle model,” in *Advances in Cryptology—CRYPTO 2019*. Springer, 2019, pp. 356–383, full version: <https://arxiv.org/abs/1902.07556v4>.
- [59] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, “Strengths and weaknesses of quantum computing,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1510–1523, 1997.

APPENDIX A PROOFS FOR MULBERRY

We use the model of Section II-A. We assume $|\mathbb{F}| \geq n + 1$, $\log_2 |\mathbb{F}| \geq \kappa$, and $R \log_2 |\mathbb{F}| \geq n + \kappa$. These sufficient conditions make the cryptographic bounds below negligible in κ .

A. Consistency of Accepted Shares

Lemma 3 (Schwartz–Zippel): Let $p \in \mathbb{F}[X]$ be nonzero with degree at most d , and let a be uniform in a nonempty set $S \subseteq \mathbb{F}$. Then

$$\Pr[p(a) = 0] \leq d/|S|.$$

Proof: A nonzero degree- d polynomial has at most d distinct roots. ■

To analyze the degree check, we must fix the committed shares before the challenge. Merkle binding prevents conflicting openings [56], while online extraction lets the proof determine these shares before the challenge [57].

Lemma 4 (Consistency of accepted shares): Suppose every response polynomial $h_{j,k,\rho}$ in the public object has degree at most t . The probability that there exists a batch index k at which $t + 2$ shares from distinct replicas are accepted but do not lie on a common degree- t polynomial is at most

$$\epsilon_{\text{qdeg}} = O(N_{\text{inst}}(Q_{\text{FS}} + 1)^2 (2^n |\mathbb{F}|^{-R} + \epsilon_{\text{ext}})).$$

In this bound, N_{inst} counts Mulberry instance–batch index pairs, and Q_{FS} counts challenge-oracle queries. The negligible term ϵ_{ext} accounts for changes to the adversary’s state caused by extraction and accepted openings that disagree with the extracted values. The Merkle binding error is $O((Q_H + 2)^3 2^{-\kappa})$, where Q_H counts commitment-oracle queries. All bounds assume polynomially many instances and queries, including verification and queries after the challenge.

Proof: Fix a set of replicas whose committed shares are inconsistent, together with their committed masks. At most one challenge θ can make $w + \theta v$ lie on a degree- t polynomial: two such challenges would, by subtraction, imply that v itself lies on one. The root bound in Lemma 3 therefore gives acceptance probability at most $|\mathbb{F}|^{-R}$ for R independent random challenges.

Mulberry must also handle a dealer that chooses the accepting set after seeing the challenges and searches for favorable Fiat–Shamir roots. A union bound over at most 2^n sets handles the first issue [27]. For the second, apply the measure-and-reprogram theorem to the vector of R challenges, using independent commitment and challenge oracles [58]. This gives a quadratic query loss, with the additive inverse-challenge-space term absorbed in the bound. Including extraction error and summing over instance–index pairs gives ϵ_{qdeg} . ■

B. Privacy

To prove privacy, we simulate the adversary’s view. The simulator is given the shares held by corrupted replicas and learns each secret just before a correct replica first reveals a share for its reconstruction. The adversary may choose which secrets to reconstruct and which unopened secret to distinguish

based on the messages seen so far. We compare (V_{real}, s) and (V_{sim}, s) , where the two views are from the real and simulated executions and s contains all correct dealers’ secrets.

Lemma 5: For every adversary in our model, there is a QPT simulator that generates its view of the correct dealers’ protocol executions using corrupted replicas’ shares and secrets released for reconstruction, without receiving unopened secrets. The distinguishing advantage between the two pairs defined above is at most

$$\epsilon_{\text{hide}} = \min\{1, 4N_{\text{leaf}}(Q_H + 1)2^{-\kappa/2}\},$$

where N_{leaf} counts leaves containing nonces.

Proof: At most t shares reveal no information about a secret. We use the masking simulation of Shoup and Smart [27]. To simulate the public objects, the simulator samples masks for corrupted replicas and random digests with fresh nonces for correct replicas. It then derives the challenges and samples each response h_ρ uniformly subject to the corrupted shares and masks. For any compatible sharing polynomial f , setting $g_\rho = h_\rho - \theta_\rho f$ yields the mask distribution specified by Mulberry. Thus the responses can be generated before the simulator learns the secret.

Just before the first correct reconstruction message, the simulator learns the secret and samples f consistent with it and the corrupted shares. It computes the masks as $g_\rho = h_\rho - \theta_\rho f$. Before revealing the shares and nonces, it programs the simulated random oracle to return the published digests on these leaf inputs. The published digests remain unchanged, and later openings reuse the same shares and nonces.

The difference from an unmodified oracle is confined to inputs containing the unrevealed nonces. Because these nonces are independently uniform, the adversary has limited ability to detect the change through prior queries. The quantum hybrid bound [59] limits the resulting distinguishing advantage to $4(Q_H + 1)2^{-\kappa/2}$ per leaf. Summing over all N_{leaf} leaves gives ϵ_{hide} . The simulator repeats this procedure for each requested secret. Since dealers and indices use independent polynomials, masks, and nonces, each opening leaves the other unopened secrets hidden, even when later requests depend on earlier openings. Comparing the views for two choices of unopened secrets through the same simulation gives distinguishing advantage at most $2\epsilon_{\text{hide}}$. ■

APPENDIX B PROOF OF BEACON INDISTINGUISHABILITY

The following game compares a beacon output with an independent uniform string.

Definition 4 (Beacon indistinguishability game): A challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ in the model of Section II-A play the following game.

- 1) $\mathcal{C}$ initializes the PKI and independent protocol oracles and publishes the public parameters. Before execution, $\mathcal{A}$ selects at most t replicas for corruption and receives their private keys and state.
- 2) $\mathcal{C}$ runs correct replicas with independent uniform dealer secrets. Adversary $\mathcal{A}$ controls corrupted replicas and message scheduling within the system model.

- 3) $\mathcal{A}$ names a target round r before any correct replica sends its reconstruction message for r . Pause just before that message, with adversarial view V_r consisting of the messages available to $\mathcal{A}$ and its quantum state.
- 4) Let $e = \lceil r/B \rceil$, $k = r - (e-1)B$, and $S_r = \sum_{j \in E_e} z_{j,k}$, as in Equation (9). Using correct signers' retained shares, $\mathcal{C}$ computes S_r and samples a uniform bit β . It sends the beacon output if $\beta = 0$, or an independent uniform κ -bit string otherwise.
- 5) $\mathcal{A}$ returns a guess β' . It may make further oracle queries, but receives no further protocol messages. It wins exactly when $\beta' = \beta$. If no target is named in time or reconstruction data are invalid or inconsistent, the game instead assigns an independent fair win/loss result.

Lemma 6 (Beacon indistinguishability): In the game of Definition 4, the distinguishing advantage satisfies

$$\begin{aligned} \text{Adv}_{\text{ind}}(\mathcal{A}) &:= 2 \left| \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| \\ &\leq \epsilon_{\text{ind}} := 2\delta_{\text{P}} + 2\epsilon_{\text{hide}} + \epsilon_{\text{out}}, \end{aligned} \quad (11)$$

where δ_{P} bounds the probability that any of the following guarantees fails: Merkle binding and accepted-share consistency, signature authenticity, and BFT-SMR Safety and External validity. The output-oracle term is $\epsilon_{\text{out}} = \min\{1, 4(Q_{\text{out}} + 1)|\mathbb{F}|^{-1/2}\}$, where Q_{out} counts output-oracle queries.

Proof: The selected $t+1$ dealers include a correct dealer. If its secret remains independent and uniform, adding the other contributions leaves S_r uniform. The main issue is that the adversary can influence dealer selection after seeing the public objects.

Apply Lemma 5 to the adversary's entire view before common-set selection, including public objects, corrupted shares, and earlier openings. The simulation covers all correct dealers without using their unopened secrets. Once the common set is selected, correct signers' retained shares fix the faulty dealers' contributions. The selected correct secrets can therefore be sampled independently of this view and these fixed contributions. One uniform correct secret makes S_r uniform. Earlier openings concern other indices with independent randomness.

We compare the hashed-output and random-output cases through an intermediate experiment. It returns an independent uniform challenge and sets the output oracle's value at the target input to that challenge.

To bound the effect of this change, consider the oracle without programming. Each eligible target sum is uniform and independent of the adversary's view, so a query's expected squared amplitude at its hidden input is at most $1/|\mathbb{F}|$. Choosing the target from that view preserves the bound because squared amplitudes over candidate contexts sum to at most one. Queries after the independent challenge satisfy the same bound. The quantum hybrid argument [59], applied before and after the challenge, bounds the total change by ϵ_{out} .

Excluding protocol failures and simulating the view costs at most $\delta_{\text{P}} + \epsilon_{\text{hide}}$ in each of the two cases. Adding these costs and ϵ_{out} gives Equation (11). ■