Balanced and Adaptively Secure Asynchronous Common Coin and Byzantine Agreement With Sub-Quadratic Communication*

Hanwen Feng[1]**, Tiancheng Mai[2], and Qiang Tang[2]

[1] University of Canterbury, Christchurch, New Zealand
hanwen.feng@canterbury.ac.nz
[2] The University of Sydney, Sydney, Australia
{tiancheng.mai,qiang.tang}@sydney.edu.au

Abstract. Distributed common randomness generation (i.e., the common coin problem) is a cornerstone of randomized distributed computing. While a long line of research has sought scalable solutions, the asynchronous setting remains a challenge. Specifically, while Blum et al. (TCC'21) achieved sub-quadratic communication complexity, their approach lacks "balance": certain nodes must still send $\Omega(n)$ messages, creating a scalability bottleneck. Furthermore, their solution only tolerates a $1/3-\epsilon$ fraction of corrupted nodes, whereas the classic construction by Cachin et al. (PODC'00) tolerates up to $1/2$ under the same setup assumptions.

In this work, we close these gaps by presenting the first balanced asynchronous common coin protocol with sub-quadratic communication complexity. In our construction, the communication cost of every honest node is bounded by $\widetilde{O}(\sqrt{n})$. Our protocol tolerates an adaptive adversary corrupting up to $(1/2-\epsilon)n$ nodes. Beyond these asymptotic improvements, our solution avoids the heavy cryptographic machinery (such as fully homomorphic encryption) required by Blum et al. and terminates in just two deterministic rounds, compared to the dozens of expected rounds in prior work.

At the heart of our construction are explicit and efficient sampler constructions. These samplers partition a population with an honest majority of at least $(1/2+\epsilon)n$ nodes into $O(\sqrt{n})$ communities, ensuring that a majority of these communities maintain a "forever-honest" majority. By leveraging how communities are allocated, we design mechanisms that allow each community to collectively emulate a single "virtual node" in Cachin et al.'s protocol. Reducing the number of participants from n physical nodes to $O(\sqrt{n})$ virtual nodes drives the total communication complexity to a sub-quadratic level.

Finally, we extend our methodology to Asynchronous Binary Byzantine Agreement (ABA), yielding the first balanced ABA protocol with sub-quadratic communication complexity that tolerates up to $(1/3-\epsilon)n$ adaptive corruptions.

Keywords: Byzantine agreement; common coin; sub-quadratic communication.

1 Introduction

Randomness lies at the heart of computation. In distributed systems, achieving public randomness, also known as a common coin [46,31], is often important and necessary. Such a coin must: (i) be accessible to all participants, (ii) remain unpredictable until revealed, and (iii) be resistant to manipulation even if some parties behave maliciously. This form of randomness is a key building block for randomized distributed algorithms, as it enables greater efficiency [51], helps bypass classical impossibility results [36], improves network security [1], and fosters fairness [6].

In this work, we focus on scalable and adaptive common coin solutions in the asynchronous setting. In this model, messages may be arbitrarily delayed and up to f out of n nodes may be corrupted. We seek "balanced" scalability, where each honest node sends only $o(n)$ bits

* A preliminary version of this work will appear in the proceedings of the ACM Symposium on Principles of Distributed Computing (PODC '26), Egham, England, July 2026.

** Part of this work was done while Hanwen Feng was at The University of Sydney.

to generate a shared, unpredictable random value. Despite extensive study, most existing asynchronous common coin protocols [23,35,7] require $\Omega(n^2)$ total communication complexity, precluding a per-node cost of $o(n)$.[3] The only known sub-quadratic solution, proposed by Blum et al. [16], lacks balance: it requires a small subset of nodes to communicate with all others, imposing an $O(n)$ overhead on those nodes. Furthermore, their solution only tolerates up to a $1/3 - \epsilon$ fraction of Byzantine nodes, whereas a $1/2$ corruption tolerance is achievable under the same setup, as demonstrated by Cachin et al. [23].

Asynchronous Binary Byzantine Agreement. A primary application of the common coin is Asynchronous Binary Byzantine Agreement (ABA). ABA is a fundamental primitive that allows a network to reach consensus on a binary value, ensuring that if all honest nodes share the same input, that value is chosen. Regarding scalability, ABA faces the same hurdles as the common coin. Specifically, the only sub-quadratic, adaptive solution [16] relies on a non-balanced communication structure where certain nodes must send $\Omega(n)$ messages. While balanced, sub-quadratic solutions exist for the synchronous setting [18], extending these results to the asynchronous setting remains an open and challenging problem.

This intriguing gap leads us to the core question studied in this paper:

Can we construct scalable and adaptive asynchronous common coin (and Byzantine agreement) with near-optimal resilience and balanced per-party communication of $o(n)$ bits?

Our Contributions. We answer the above question affirmatively by presenting scalable and adaptively secure constructions for both common coin and ABA. Our results are as follows.

Scalable and Adaptively Secure Asynchronous Common Coin Protocols. We present balanced asynchronous common coin protocols with a total communication complexity of $O(\lambda n^{1.5}(\gamma + \log n))$. In our construction, each honest node sends at most $O(\lambda\sqrt{n}(\gamma + \log n))$ bits, where λ is the computational security parameter (governing signature size) and γ is the statistical security parameter (ensuring a failure probability of at most $2^{-\gamma}$). Regarding round complexity, our protocols require exactly two deterministic rounds to generate common randomness. Our protocols are secure against weakly adaptive adversaries, which may corrupt up to $f < (1/2-\epsilon)n$ nodes (for any constant $0 < \epsilon < 1/2$) at any point during execution. Consistent with the weakly adaptive model, the adversary cannot retract messages sent by a node prior to its corruption. We provide a detailed comparison of our results with existing literature in Table 1 and a comprehensive overview of related work in Appendix A.

Efficient and Explicit Samplers. At the heart of our construction are density-tailored samplers [38]. These samplers allow us to organize a network of n nodes into $O(\sqrt{n})$ potentially overlapping communities, such that regardless of which f nodes the adversary chooses to corrupt after observing the allocation, a majority of these communities will consistently maintain an honest majority. While samplers have been employed, most notably in synchronous Byzantine agreement protocols such as [40,18], it is non-trivial to adapt these schemes to asynchronous or adaptive scenarios because their components or analyses are vulnerable to adaptive adversaries in asynchrony. We defer detailed discussions to Appendix A. More importantly, most existing works rely on the *existence* of such samplers without providing concrete, implementable constructions. In this work, we provide *explicit and efficient* sampler constructions. Specifically, we present a general construction that remains robust for any $f < (1/2 - \epsilon)n$, where $0 < \epsilon < 1/2$ is a constant. Additionally, we provide a more efficient sampler specifically for the $f < n/3$ case, which is valid for any $n \geq 9,312$.

[3] While $\Omega(n^2)$ communication is necessary for strongly adaptive security [4], sub-quadratic protocols target weakly adaptive security, where the adversary cannot retract messages sent by a node prior to its corruption.

Scalable and Adaptively Secure ABA. We extend our results to ABA. In particular, each party incurs $O(\lambda\sqrt{n}(\gamma+\log n))$ bits per ABA round and the protocol lasts for $O(1)$ ABA rounds in expectation. Therefore, the expected per-party communication is $O(\lambda\sqrt{n}(\gamma+\log n))$. Each ABA round lasts for at most $O(\gamma)$ actual asynchronous rounds; hence the expected total round complexity is $O(\gamma)$. We remark that our result answers open questions posed by [40] and [19]. King et al. ask whether their result, an $\widetilde{O}(\sqrt{n})$-balanced Byzantine agreement in the synchronous setting, can be adapted to the asynchronous setting, while Boyle et al. ask whether balanced sub-quadratic BA is possible even in the partially synchronous model.

Necessity of Trusted Setup. Both our common coin and ABA protocols require a setup phase in which a trusted dealer distributes private keys to each node in the network. As demonstrated by Blum et al. [16], some form of trusted setup is necessary to achieve sub-quadratic ABA with weakly adaptive security. We extend their lower-bound argument to show that this necessity also holds for asynchronous common coin protocols.

Table 1. Comparison of Near- and Sub-Quadratic Asynchronous Common Coin Schemes[1]

Scheme	Total Comm. Cost	Max Per-node Comm. Cost	Round	Balanced?	Adaptive?	Fault Tol.	Setup	Crypto.Assp.
CKS00 [23]	$O(\lambda n^2)$	$O(\lambda n)$	1	✓	Strongly	$f<n/2$ †	Private	RO, DDH, co-CDH
CKS20 $^{\$}$ [25]	$E[O(\lambda\gamma n)]$	$O(\lambda n)$	2	✓	Delayed-adaptive	$f<n/4.45$	Trusted-PKI	VRF
FT25 [35]	$E[O(\lambda n^2)]$	$E[O(\lambda n)]$	$E[O(1)]$	✓	Strongly	$f<n/3$	Bare-PKI, CRS	RO, GGM
ABLS25 [3]	$E[O(\lambda^2 n^2\log^c n)]$	$E[O(\lambda^2 n\log^c n)]$	$E[O(\log n)]$	✓	Strongly	$f<n/3$	Bare-PKI, Trusted-VRF	RO, AGM, OMDL, Erasures, VRF
BBBK24 [12]	amortized $O(\lambda\gamma n^2\log n)$	amortized $O(\lambda\gamma n\log n)$	$O(\gamma+\log n)$	✓	Weakly	$f<n/3$	None	RO, Erasures
BKLL20 [16]	$E[O(\lambda\gamma^3 n)]$	$E[O(\lambda n)]$	$E[O(1)]$	✗	Weakly	$f<(1-\epsilon)n/3$ ‡	Private	TFHE, UC-NIZK, Erasures
Ours	$O(\lambda n^{1.5}(\gamma+\log n))$	$O(\lambda\sqrt{n}(\gamma+\log n))$	2	✓	Weakly	$f<(1-\epsilon)n/2$	Private	RO, AGM, OMDL

†: [23] originally shows $f<n/3$ but the same result can be improved to $f<n/2$ by using a threshold setup. It is strongly adaptive secure if using an adaptively secure unique threshold signature scheme.
$^{\$}$: [25] constructs a WHP coin, where $\gamma=\Theta(\log n)$. Termination and agreement err with probability $1/n^c$ where c is a constant.
‡: [16] can achieve $f<n/3$ when instantiating only a fixed κ number of coins. However, when instantiating an unbounded number of coins, i.e., polynomially many coins, the fault tolerance reduces to $f<(1-\epsilon)n/3$ due to the sub-sampling in the BA subroutine.
Total Comm. Cost is the number of bits sent by all honest nodes, and **Max Per-node Comm. Cost** is the maximum number of bits sent among all individual honest nodes, where λ and γ are the computational security parameter and the statistical security parameter, respectively. **Round** measures the number of rounds within which all honest nodes can terminate. $E(\cdot)$ denotes expected value, otherwise the displayed costs are deterministic. **Balanced?** asks if the max per-node communication is only $1/n$ of the total communication. **Adaptive?** asks if the protocol is strongly adaptive secure [4]. **Fault Tol.** asks what is the maximum number of faults that the protocol can tolerate, denoted as f. **Setup:** *CRS* stands for Common Reference String. *Bare-PKI* stands for bare PKI where every party locally chooses its keys and corrupted parties can set their keys as a function of all verification keys (and any additional public information), and *Trusted-PKI* stands for honestly generated PKI where corrupted parties cannot change their verification keys. **Crypto.Assp.** means the cryptographic assumptions adopted. **Chan.Assp.** means the communication channel assumptions where *Auth* stands for authenticated channel, and *Sec* stands for secure channel in which messages are both private and authenticated.

2 Technical Overview

We first overview our common coin techniques step-by-step, followed by a discussion on how these techniques are adapted to obtain our scalable ABA scheme.

[1] This table only includes computational asynchronous coin protocols, there are also asynchronous coins against information-theoretic adversaries, which naturally have even higher complexity.

Table 2. Comparison of Near-Quadratic or Balanced Byzantine Agreement Schemes

Scheme	Total Comm. Cost	Max Per-node Comm. Cost	Network	Round	Balanced?	Adaptive?	Fault Tol.	Setup	Crypto.Assp.
CKS00 [23]	$E[O(\lambda n^2)]$	$E[O(\lambda n)]$	Async	$E[O(1)]$	✓	Strongly	1/3	Private	RO, CDH/DDH
KS11 [40]	$O(n^{1.5}\log^9 n)$	$O(\sqrt{n}\log^9 n)$	Sync	$O(\frac{\log^{10} n}{\log\log n})$	✓	Weakly	$1/3-\epsilon$	None	None
CKS20 [25]	$E[O(\lambda\gamma^2 n)]$	$E[O(\lambda\gamma n)]$	Async	$E[O(1)]$	✗	Delayed-adaptive	1/4.45	Trusted-PKI	VRF
BKLL20 [16]	$E[O(\lambda\gamma^3 n)]$	$E[O(\lambda n)]$	Async	$E[O(1)]$	✗	Weakly	$1/3-\epsilon$	Private	TFHE, UC-NIZK, Erasures
BCG21 [18]	$O(n\log^c n$ poly $\lambda)$	$O(\log^c n$ poly $\lambda)$	Sync	1	✓	Static	$1/3-\epsilon$	Trusted-PKI	OWF
								Bare-PKI, CRS	SNARKs, CRH
Ours	$E[O(\lambda n^{1.5}(\gamma+\log n))]$	$E[O(\lambda\sqrt{n}(\gamma+\log n))]$	Async	$E[O(\gamma)]$	✓	Weakly	$1/3-\epsilon$	Private	RO, AGM, OMDL

Recall: Common Coin from Non-interactive Unique Threshold Signatures. Cachin et al. [23] presented an asynchronous common coin protocol by leveraging a non-interactive unique threshold signature, such as the BLS threshold signature [29]. For example, in threshold BLS, each party P_i holds a secret share $sk_i = F(i)$, where $F \in \mathbb{Z}_p[X]$ is a univariate polynomial of degree f, while the threshold secret key is $sk = F(0)$. The partial signature for a message msg is $H(\mathsf{msg})^{sk_i}$, and the final signature is $H(\mathsf{msg})^{sk}$. To generate a common random value, the network jointly signs a common session identifier ID and uses the hash of the resulting signature as the common coin. Because the signature is unique, meaning there is exactly one valid signature for any given message, all nodes in the network are guaranteed to observe the same value. Furthermore, since the signature is unforgeable, no party can compute it prematurely. Thus, in the random oracle model, the resulting value is both unpredictable and unbiased.

However, because reconstructing the final signature requires at least $f+1$ partial signatures (corresponding to the $f+1$ shares needed to recover the secret key), existing BLS threshold signature protocols typically require every node to multicast its partial signature. To ensure that all participants can reconstruct the final value, this approach incurs an $\Omega(n)$-bit communication overhead for every party, creating a significant scalability barrier.

Support Two-layer Aggregation with Bivariate-polynomial-based Key Shares. As discussed, if every party must perform signature aggregation locally, an $\Omega(n)$-bit per-node communication cost is unavoidable, as each node must receive $f+1$ partial signatures. To reduce this cost, we must facilitate "gradual" aggregation. In this model, partial signatures are first combined into "intermediate" signatures, allowing each node to require far fewer signatures for the final reconstruction.

However, such gradual aggregation is challenging for univariate-based schemes. There, given a set of shares $\{sk_i = F(i)\}_{i\in\mathbb{I}}$, the secret $sk = F(0)$ is reconstructed by calculating $F(0) = \sum_{i\in\mathbb{I}} \lambda_{i,\mathbb{I}} F(i)$, where $\lambda_{i,\mathbb{I}}$ is the Lagrange coefficient. The final signature aggregation essentially performs this secret reconstruction homomorphically in the exponent. Crucially, the coefficients $\{\lambda_{i,\mathbb{I}}\}$ depend on the entire index set $\mathbb{I}$. This prevents meaningful *intermediate* aggregation for a subset of partial signatures, as the final set of participating indices $\mathbb{I}$ is unknown during the early stages of the protocol.

To overcome this limitation, we employ a bivariate-based scheme[4]. We define the secret as a bivariate polynomial $F(x, y)$. By partitioning the nodes into m communities, each with q nodes, the j-th node in the i-th community is assigned the share $F(i, j)$. This setup allows nodes within community i^* to first aggregate their shares into an intermediate signature for the "community secret" $F(i^*, 0)$, which we term a *syndicate signature*. Every node can then obtain the final signature by aggregating these syndicate signatures.

[4] We note that while prior works use bivariate polynomials for share verification [14], share recovery [5,50], or realizing various access structures [49], we explore a novel application of these polynomials specifically to improve efficiency.

In this two-layer structure, a node only needs to collect $O(q)$ partial signatures (q is the community size) to form the intermediate signature and $O(m)$ syndicate signatures (m is the number of communities) for the final reconstruction. This structure successfully bypasses the $\Omega(n)$ barrier, provided that $q + m = o(n)$.

Semi-interactive Threshold Signature Protocol: Realizing Efficient Two-layer Aggregation. We design a semi-interactive threshold signature protocol that implements the two-layer aggregation mechanism in a communication-efficient manner. Our protocol utilizes a community allocation, which organizes the network into m potentially overlapping communities. We ensure that a majority (at least $m/2 + 1$) of these communities possess an honest majority (at least $q/2 + 1$ honest members).

During the setup phase, each member $\widetilde{P}_{i,j}$ (the j-th node in the i-th community) receives a share $F(i, j)$, where F is sampled row-wise: first a community-level polynomial $A(x) := F(x, 0)$ of degree $m/2$ is sampled, and then, for each community i, an intra-community polynomial $F(i, y)$ of degree $q/2$ in y is sampled subject to $F(i, 0) = A(i)$. To sign a message msg, each $\widetilde{P}_{i,j}$ multicasts its partial signature to all other nodes within its community. This allows the syndicate signature, $H(\mathsf{msg})^{F(i,0)}$, to be reconstructed by all community members. Crucially, in any community with an honest majority, all honest members will eventually obtain the syndicate signature, even in a fully asynchronous environment.

The subsequent challenge is to disseminate these syndicate signatures to all honest nodes without triggering another $\Omega(n^2)$ communication explosion. Since all honest members of a particular community hold the same syndicate signature, simple multicasting would be redundant and inefficient. To resolve this, we employ a probabilistic dissemination strategy: each node in a community randomly selects $O((\gamma + \log n) \cdot n/q)$ nodes to which it sends its syndicate signature. This approach ensures that every node in the network eventually receives at least one valid syndicate signature from each honest-majority community with probability at least $1 - 2^{-\gamma}$.

We formally establish the adaptive security of the proposed scheme by adapting recent techniques for the conventional threshold BLS [8]. Please see Section 5 for more details.

Community Allocation with Explicit Sampler Design. In our semi-interactive threshold signature protocol, we require a community allocation that ensures a majority of communities possess an honest majority, even if the adversary adaptively corrupts up to f nodes after observing the allocation. We first observe that this is straightforward to fulfill when $f < n/4$, where we simply partition the network into $O(\sqrt{n})$ non-overlapping communities of equal size. By the Pigeonhole Principle, it is easy to see a majority of these communities will inherently maintain an honest majority.

Nonetheless, we aim to enable a community allocation that could work in the nearly optimal resilience setting, i.e., $f < (1/2 - \epsilon)n$. To this end, we construct a new community allocation strategy via *density-tailored samplers*. Essentially, we abstract the community allocation as a bipartite graph and establish a connection between such graphs and samplers. Samplers are functions that output sample points ("community members") of a given universe (all parties), with the property that the fraction of samples from any particular subset (corrupted parties) of the universe is roughly equal to the density of that subset $(1/2 - \epsilon)$. With a $(1/2 - \epsilon, \epsilon, 1/2)$-density-tailored sampler, we can construct a $(1/2 - \epsilon, 1/2, 1/2)$-robust community allocation strategy such that at most half of the communities are corrupted, where a corrupted community is one with a dishonest majority.

Specifically, we provide three classes of combinatorial designs, which are BIBDs [47,2], imbalanced bipartite Ramanujan graphs [22], and (k, q)-nets [15]. They offer a set of properties to control the community allocation and can be explicitly constructed without exponential-time exhaustive search. At a high level, these designs create more communities (up to a constant multiplicative factor), allow overlapping between them, precisely control the

intersections, and are regular and uniform, yet keep the community size and count small enough to save communication cost. See Section 4 for more details.

Extending the Techniques to ABA. The techniques developed for our asynchronous common coin can be viewed through the following lens: first, a sampler-based community allocation partitions the network into $O(\sqrt{n})$ communities of $O(\sqrt{n})$ members each, ensuring a majority of communities maintain an honest majority. We then enable each community to emulate a "virtual node" capable of (1) generating a partial signature and (2) disseminating that signature to the network, mirroring the tasks a physical node performs in the protocol of Cachin et al. [23]. Conceptually, this approach extends the "Dragon" method [33], which reduces communication complexity by deterministically dividing the network into multiple communities and having each emulate a virtual node.

Building on this intuition, we apply samplers with fine-tuned parameters to partition the network into m communities such that at least $2m/3 + 1$ communities maintain a "super-honest majority" (i.e., the fraction of corrupted nodes within these communities is less than $1/3$). We then design protocols that allow each community to emulate a virtual node in a conventional ABA protocol. Specifically, we adapt the protocol by Cachin et al. [23], where each node must (1) sign a value determined by its received messages and (2) disseminate that signed value to the network.

While task (2) leverages the same dissemination mechanism used for our common coin, task (1) presents a unique challenge: the value to be signed is derived from incoming asynchronous messages rather than a predetermined session ID. To enable a community to collectively perform task (1), the members run an intra-community Multi-valued Byzantine Agreement (MBA) instance to reach consensus on the value to be signed. Once agreed upon, they jointly generate a valid syndicate signature.

For communities with a super-honest majority, the intra-community MBA is highly efficient, incurring a communication cost of only $O(q^2)$, where q is the number of nodes in the community. This efficiency stems from the fact that the common coins required for the MBA can be realized via the unique threshold signature within the community. However, special care is required for communities where the fraction of corrupted nodes exceeds $1/3$. In such communities, MBA instances may fail to terminate, potentially forcing honest nodes to send an unbounded number of messages—a scenario that directly undermines our efficiency goals. To address this, we leverage the property that, in an MBA instance with a super-honest majority, the probability of non-termination decreases exponentially as the number of rounds grows. Consequently, we mandate that each honest node ceases transmitting messages after $O(\gamma)$ rounds. This round limit ensures that all super-honest-majority instances terminate except with probability $2^{-\gamma}$, while strictly bounding the communication complexity of every instance to $O(\gamma q^2)$. Further details are provided in Section 6.

3 Preliminaries

Notations. Throughout the paper, we let λ and γ represent the bit lengths of the computational and statistical security parameters, respectively. The notation $[i, n]$ represents the set $\{i, i + 1, \cdots, n\}$, where i and n are integers with $i < n$. We might abbreviate $[1, n]$ simply as $[n]$. For a set $\{x_1, x_2, \ldots, x_n\}$ and a sequence $(x_1, x_2, \ldots, x_n)$, we denote them by $\{x_i\}_{i\in[n]}$ and $(x_i)_{i\in[n]}$ for short, respectively. A t-subset of a set S is a subset T of S with cardinality t. A function $f(n)$ is deemed negligible in n, denoted by $f(n) \leq \mathsf{negl}(n)$, if for every positive integer c, there exists an n_0 such that for all $n > n_0$, $f(n) < n^{-c}$. Conversely, a non-negligible function is denoted as $f(n) > \mathsf{negl}(n)$. For a set $\mathbb{X}$, the notation $x \leftarrow\!\$\ \mathbb{X}$ signifies uniform sampling x from $\mathbb{X}$. Given a distribution X, $x \leftarrow\!\$\ X$ denotes sampling x from X. For a probabilistic algorithm A, $A(x_1, x_2, \cdots; r)$ represents the result of running A with inputs

$x_1, x_2, \cdots$ and random coins r. We use $y \leftarrow A(x_1, x_2, \cdots)$ to represent randomly choosing r and computing $A(x_1, x_2, \cdots; r)$ to obtain the result y.

Communication Model. The network consists of n nodes $\mathcal{P} = \{P_i\}_{i\in[n]}$ and the communication complexity is determined by the number of bits sent by the honest node. We assume a fully connected asynchronous network where each pair of nodes has an authenticated communication channel. Communication complexity is measured by the number of bits sent by honest nodes. The adversary can arbitrarily delay messages and learn information about their content but cannot tamper with them. Messages sent between honest parties must *eventually* be delivered. The communication steps in our protocol can proceed in "rounds", such that messages sent from an uncorrupted node can be delivered to an uncorrupted recipient, however without a delivery time bound.

Threat Model. We assume a private setup. In the initial setup phase, the trusted dealer generates the initial state for all n parties. This includes secret shares and public key shares. The community allocation information is publicly verifiable and may be generated by a centralized party. The adversary $\mathcal{A}$ is assumed to be probabilistic polynomial time (PPT) bounded, and it can corrupt up to t participants and arbitrarily control the behavior of corrupted nodes. We consider an *adaptive* adversary, following the formalization of [4], independent of network synchronicity. At any time in the protocol, an *adaptive* adversary can gradually choose the participating nodes to corrupt during the initial phase and protocol execution. Once a node gets corrupted, the adversary gets access to its local states and controls its subsequent behaviors. All nodes that have been corrupted are under the control of $\mathcal{A}$, i.e., the messages they receive are forwarded to $\mathcal{A}$, and $\mathcal{A}$ controls what messages they will send once they become corrupted. Participants that remain honest so far are referred to as *so-far-honest* and the ones remain honest till the end of the protocol are referred to as *forever-honest* nodes. We assume that when a *so-far-honest* node i sends a message m, it can immediately become corrupted and be made to send one or more messages. For a *strongly* adaptive adversary, if an honest node i sends a message m, a strongly adaptive adversary (e.g., who controls the egress routers of many nodes) can observe m and then decide to corrupt i and erase the message m that node i has just sent. This erasure is called an after-the-fact removal of m. However, for a *weakly* adaptive adversary, this message m that has already been sent before i became corrupted cannot be retracted. It will be received by all *so-far-honest* receiver nodes eventually. In other words, the adversary is adaptive but not strongly adaptive (i.e., incapable of after-the-fact removal).

Cryptographic Assumptions. The security of our threshold signature scheme relies on the random oracle model (ROM), the one-more discrete logarithm (OMDL) assumption and the algebraic group model (AGM).

The One-More Discrete Logarithm and Algebraic Group Model Assumptions. The security of our threshold signature scheme and common coin protocol is based on the one-more discrete logarithm (OMDL) assumption [13] and the algebraic group model (AGM) [37]. We recall the definitions in the following.

Definition 1 (One-More Discrete Logarithm Assumption). *Let $\mathbb{G}$ be a cyclic group of prime order p, and let g be a generator of $\mathbb{G}$. For a positive integer $k \in \mathbb{N}$, the k-OMDL assumption states that, for any PPT adversary $\mathcal{A}$,*

$$\Pr\begin{bmatrix}(z_1,\ldots,z_k) \leftarrow\$\ \mathbb{Z}_p^k, \xi_i \leftarrow g^{z_i} \in \mathbb{G}, \forall i \in [k] \\ (z'_1,\ldots,z'_k) \leftarrow \mathcal{A}^{\mathsf{DL}_g(\cdot)}(\mathbb{G}, p, g, (\xi_i)_{i\in[k]}) : \\ z'_i = z_i, \forall i \in [k]\end{bmatrix} \leq \mathsf{negl}(\lambda),$$

where the discrete logarithm oracle DL_g *on input a group element* X *in* $\mathbb{G}$ *returns* x *s.t.* $X = g^x$*, and* $\mathcal{A}$ *can access* DL_g *at most* $k-1$ *times. We say that the one-more discrete logarithm problem of degree* k *is* (ϵ, T)*-hard if for all algorithms* A *running in time at most* T*,* $\Pr[\mathsf{A}^{k-\mathsf{OMDL}_g(\cdot)}(\mathbb{G}, p, g, (\xi_i)_{i\in[k]}) = 1] \leq \epsilon$*. Conversely, we say that an algorithm* A (ϵ, T)*-solves the one-more discrete logarithm problem of degree* k *if it runs in time at most* T *and* $\Pr[\mathsf{A}^{k-\mathsf{OMDL}_g(\cdot)}(\mathbb{G}, p, g, (\xi_i)_{i\in[k]}) = 1] > \epsilon$*.*

Fuchsbauer, Kiltz, and Loss [37] introduced the algebraic group model (AGM) as a model in between the generic group model (GGM) and the standard model. In AGM, all algorithms are algebraic, meaning that an algorithm must output a representation of any outputted group element where the representation is relative to all of the inputs the algorithm has received up to outputting the group element.

Definition 2 (Algebraic Algorithm). *An algorithm* A *is called algebraic (over group* $\mathbb{G}$*) if for all group elements* $\zeta \in \mathbb{G}$ *that* A *outputs, it additionally outputs a vector* $\boldsymbol{z} = (z_1, \ldots, z_k)$ *of integers such that* $\zeta = \prod_{i\in[k]} g_i^{z_i}$*, where* $(g_1, \ldots, g_k)$ *is the list of group elements* A *has received so far (w.l.o.g.* $g_1 = g$*).*

Definition 3 (Common Coin). *Let* Π *be a protocol involving* n *participants in which each* P_i *does not have an input and will be instructed to output a value* $v_i \in V$*. We say* Π *is an* (n, f, φ) *common coin protocol for* $\varphi \in (0, 1]$*, if, in any instance of* Π*, and for any PPT adversary* $\mathcal{A}$ *which can corrupt up to* f *parties, the following properties are ensured:*

- ***Termination:*** *Except with negligible probability, every honest party* P_i *eventually outputs a value* $r_i \in V$*.*
- φ***-Fairness:*** *With a probability of at least* $\varphi - \mathsf{negl}(\lambda)$*, the protocol has a fair outcome, satisfying:*
  Agreement: There is a value $v \in V$ *such that* $v_i = v$ *for any* $i \in \mathcal{H}$*, where* $\{P_i\}_{i\in\mathcal{H}}$ *is the set of honest parties.*
  Unpredictability and bias-resistance: Let $\mathrm{State}_{\mathcal{A}}^{\mathrm{before}}$ *denote the internal state of* $\mathcal{A}$ *before* $n - f - |\mathcal{C}|$ *so-far-honest nodes participate in the protocol, where* $|\mathcal{C}|$ *is the number of corrupted parties. For any PPT distinguisher* $\mathcal{D}$ *and* $u \leftarrow\!\!\$\, V$*,*

$$\left|\Pr[\mathcal{D}(v, \mathrm{State}_{\mathcal{A}}^{\mathrm{before}}) = 1] - \Pr[\mathcal{D}(u, \mathrm{State}_{\mathcal{A}}^{\mathrm{before}}) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Remark 1. When $\varphi = 1$, the definition captures the multivalued version of the cryptographic strong common coin [23], which ensures termination, agreement, unpredictability, and bias-resistance except with negligible probability; in this case, we simply write (n, f)-common coin. When $\varphi < 1$, it can be seen as a multivalued and cryptographically secure version of the weak common coin by Feldman and Micali [32], and it can trivially imply a protocol satisfying their definition, as we show in [35, Lemma 13].

Definition 4 (Asynchronous Byzantine Agreement (ABA)). *In an* (n, t)*-Asynchronous Byzantine Agreement protocol, there are* n *parties* $\mathcal{D} = \{P_1, \ldots, P_n\}$*, where each* P_i *has a* 1*-bit input* v_i *and decides an output, denoted as* $\mathsf{ABA}\langle\mathcal{D}, P_i(v_i)\rangle$*. Against any weakly adaptive PPT adversary* $\mathcal{A}$ *that corrupts up to* t *parties, a protocol solves ABA with probability* p *if the protocol achieves the following properties with probability at least* p*:*

- ***Validity.*** *If all honest parties share the same input* v*, they all output* v*.*
- ***Agreement.*** *The output messages of all honest parties are the same.*
- ***Termination.*** *Every honest party eventually decides on an output message.*

In particular, such a protocol is also called an Asynchronous ***Binary*** *Byzantine Agreement protocol.*

In ABA, the input values are restricted to either 0 or 1. In contrast, in MBA, honest nodes provide input values $v_i \in \{0,1\}^\ell \cup \{\perp\}$, where the values are not limited to binary values $\{0,1\}$. Formally, MBA satisfies the following properties with probability at least p:

Definition 5 (Multi-valued Byzantine agreement (MBA)). *In an (n,t,ℓ)-Asynchronous Multi-valued Byzantine Agreement protocol, there are n parties $\mathcal{D} = \{P_1,\ldots,P_n\}$, where each P_i has an ℓ-bit input v_i and decides an output, denoted as $\mathsf{MBA}\langle\mathcal{D}, P_i(v_i)\rangle$. Against any weakly adaptive PPT adversary $\mathcal{A}$ that corrupts up to t parties, a protocol solves MBA with probability p if the protocol achieves the following properties with probability at least p:*

- ***Validity.*** *If all honest parties share the same input v, they all output v.*
- ***Agreement.*** *The output messages of all honest parties are the same.*
- ***Termination.*** *Every honest party eventually decides on an output message.*

Shamir Secret Sharing. The Shamir secret sharing [48] hides a secret s in the constant term $f(0)$ of a t-degree univariate polynomial f where non-constant coefficients are randomly sampled from a field $\mathbb{Z}_p$. The i-th share of the secret becomes $f(i)$ and $t+1$ distinct shares suffice to efficiently reconstruct the polynomial f and therefore the secret s using Lagrange interpolation.

Extend Shamir SS to Bivariate. A generalization of the Shamir secret sharing for bivariate polynomial is that the secret s is hidden in the constant term $F(0,0)$ of a randomly sampled bivariate polynomial, with α-degree in X and β-degree in Y. Suppose $A_i(X) := F(X,i)$ is the partial evaluation of F with degree α and $B_j(Y) := F(j,Y)$ is the partial evaluation of F with degree β. Note that they are univariate. The j-th share of the secret $s = F(0,0)$ becomes $B_j(0) = F(j,0)$, while the i-th share of $B_j(0)$ is $F(j,i)$. Given $\beta+1$ distinct shares of any $B_j(0)$, i.e., $\beta+1$ distinct evaluations of a univariate polynomial B_j with degree β, one can efficiently reconstruct B_j by interpolation and compute $B_j(0)$. Furthermore, with $\alpha+1$ distinct shares of s, the univariate polynomial $A_0(X) = F(X,0)$ can be reconstructed and $s = A_0(0) = F(0,0)$ can be recovered. It is easy to verify that s is information-theoretically hidden from an adversary that knows α or fewer shares of s. Thus, each share $B_j(0)$ of the secret remains IT hidden when the adversary knows β or fewer shares of $B_j(0)$.

Bivariate Polynomial. For any bivariate polynomial $F(X,Y)$ of degree-(α,β), F has degree α in X and degree β in Y. A partial evaluation of F is realized when one of the two indeterminates is fixed while leaving the other as is. When fixing the second indeterminate $Y = y$, the partial evaluation $A_y(X) := F(X,y)$ is a univariate polynomial of degree α. $(\alpha+1)$ points on A_y, $F(u_1,y),\ldots,F(u_{\alpha+1},y)$ suffice to interpolate A_y. Similarly, when fixing the first indeterminate $X = x$, the partial evaluation $B_x(Y) = F(x,Y)$ is a univariate polynomial of degree β, and $(\beta+1)$ points on $B_x(Y)$, $F(x,v_1),\ldots,F(x,v_{\beta+1})$ suffice to interpolate $B_x(Y)$.

4 Explicit constructions of density-tailored sampler

In this section, we first recall the definitions of sampler and our community allocation strategy, then show the equivalence between samplers and community allocation strategy. Observing that bipartite biregular graphs with sufficiently large spectral gap are density-tailored samplers, we provide three explicit constructions based on three different combinatorial designs: BIBD, Imbalanced Bipartite Ramanujan Graph and (k,q)-net. They become our candidates due to their desirable spectral properties.

The three designs trade off between construction simplicity, achievable fault tolerance, and admissible network size coverage. The BIBD instantiation is the simplest, yielding a

(1/3)-fault tolerance for any $n \geq 9,312$. The imbalanced bipartite Ramanujan graph attains the asymptotically optimal Ramanujan ratio giving $(1/2-\epsilon)$- and $(1/3-\epsilon)$-fault tolerance for arbitrarily small ϵ, at the cost of requiring the network size n to exceed an ϵ-dependent threshold $N(\epsilon)$ (heuristically on the order of ϵ^{-4}; see Corollary 3 and 4). The (k,q)-net matches the same spectral ratio and achieves the same $(1/2-\epsilon)$ asymptotic tolerance, and relaxes the integral admissible size constraint to q being any prime power instead of prime.

4.1 Definition

We define sampler and its density-tailored variant in bipartite graph formulation. Its original probabilistic formulations can be found in sampler literature, such as [38, Definition 16 and 20].

Definition 6 (Sampler). *Let $G = (L \cup R, E)$ be a bipartite graph where $L = \{u_1, \ldots, u_N\}$ represents the set of network nodes and $R = \{v_1, \ldots, v_m\}$ represents the set of random seeds. We define the neighbor subsets of a vertex $x \in L \cup R$ as $\mathcal{N}(x)$ and the sampling function derived from G as $\Gamma : R \to \mathcal{P}(L)$, where $\Gamma(v) = \mathcal{N}(v) = \{u \in L \mid (u,v) \in E\}$. The graph G is an (θ,γ)-**sampler** if, for any subset of nodes $S \subseteq L$ with density $\mu(S) = \frac{|S|}{N}$, the fraction of neighbors in S approximates $\mu(S)$ such that: $\Pr_{r \leftarrow \$ R}\left[\frac{|\Gamma(r) \cap S|}{|\Gamma(r)|} \geq \mu(S) + \theta\right] < \gamma$ where the probability is taken over the uniform choice of the seed $r \in R$.*

*Particularly, given a fixed density upper bound μ, we say that the graph G is a (μ,θ,γ)-**density-tailored sampler** if, for any subset $S \subseteq L$ with $\mu(S) \leq \mu$, it holds that*

$$\Pr_{r \leftarrow \$ R}\left[\frac{|\Gamma(r) \cap S|}{|\Gamma(r)|} \geq \mu + \theta\right] < \gamma,$$

i.e., the deviation threshold is fixed to $\mu+\theta$ across all such S. Note that since $\mu(S)+\theta \leq \mu+\theta$, every (θ,γ)-sampler is also a (μ,θ,γ)-density-tailored sampler. If G is right-regular with degree q, such that $|\Gamma(r)| = q$ for all $r \in R$, then we say that G is q-regular. If G is left-regular with degree k, such that $|\{v \in R \mid (u,v) \in E\}| = k$ for all $u \in L$, then we say that G is k-uniform.

An application of the sampler is community allocation. For $r \in R$, we say r allocates a community $\Gamma(r)$. Let $\mathcal{X} = \{\Gamma(r) \mid r \in R\}$ be the set of all communities, i.e., $\mathcal{X}$ allocates m communities.

Definition 7 ((μ,δ,γ)-Robustness of a Community Allocation Strategy). *Given a positive integer n, let $\mathcal{X} = \{\mathsf{CMNT}_j\}_{j \in [m]}$ be a community allocation strategy such that for all $j \in [m], \mathsf{CMNT}_j \subset [n]$. We say that the community allocation strategy $\mathcal{X}$ is (μ,δ,γ)-**robust** if and only if for any $S \subset [n]$ with $|S| \leq \mu n$, it holds that*

$$\frac{\left|\left\{j \in [m] \mid \frac{|\mathsf{CMNT}_j \cap S|}{|\mathsf{CMNT}_j|} \geq \delta\right\}\right|}{m} < \gamma.$$

*In particular, when $\delta = \gamma = 1/2$, we say that the community allocation strategy $\mathcal{X}$ is μ-**robust**. If all communities have the same size q, then we say that $\mathcal{X}$ is q-**regular**. If each integer $i \in [n]$ belongs to exactly k communities, then we say that $\mathcal{X}$ is k-**uniform**.*

It is easy to see the following relationship.

Proposition 1. *Let $G = (L,R,E)$ be a bipartite graph. G is a (μ,θ,γ)-density-tailored sampler if and only if the community allocation strategy $\mathcal{X} = \{\Gamma(r) \mid r \in R\}$ is $(\mu,\mu+\theta,\gamma)$-robust, where $\Gamma(r) = \{u \in L \mid (u,r) \in E\}$.*

Proof (Proposition 1). Let $G = (L, R, E)$ be a (μ, θ, γ)-density-tailored sampler. Then by definition, for any subset of nodes $S \subseteq L$ with density $\frac{|S|}{|L|} \leq \mu$, the fraction of neighbors in S approximates μ such that: $\Pr_{r \leftarrow \$ R}\left[\frac{|\Gamma(r) \cap S|}{|\Gamma(r)|} \geq \mu + \theta\right] < \gamma$ where the probability is taken over the uniform choice of the seed $r \in R$. Hence, we form $\mathcal{X}$ as the set of all $\Gamma(r)$ for $r \in R$. It holds that $\Pr_{x \leftarrow \$ \mathcal{X}}\left[\frac{|x \cap S|}{|x|} \geq \mu + \theta\right] < \gamma \iff \left|\left\{x \in \mathcal{X} \mid \frac{|x \cap S|}{|x|} \geq \mu + \theta\right\}\right| < \gamma|\mathcal{X}|$. Therefore, the community allocation strategy $\mathcal{X}$ is $(\mu, \mu + \theta, \gamma)$-robust. Conversely, suppose $\mathcal{X}$ is $(\mu, \mu + \theta, \gamma)$-robust. Then for any $S \subseteq L$ with $\mu(S) \leq \mu$, fewer than a γ-fraction of the communities $\Gamma(r)$ satisfy $\frac{|\Gamma(r) \cap S|}{|\Gamma(r)|} \geq \mu + \theta$; reading the displayed equivalence from right to left yields exactly the defining condition of a (μ, θ, γ)-density-tailored sampler. □

Next, in Theorem 1, we will show *bipartite biregular graphs with sufficiently large spectral gap are density-tailored samplers*. A sampler on its own aims to uniformly sample over a population with regard to any fixed-size subset S of the population. In this sense, the intersection between a specific S and each sample should be as uniform as possible and therefore attain a small variance. On the high level, we utilize a statistical argument in Theorem 1 to bridge the spectral ratio λ_2/λ_1 and that variance. The exact statistical technique to estimate the variance is employed by Cooper and Linial in [26].

For any subset $S \subset L$ with density μ, i.e., of size $\mu|L|$, we build a distribution of the normalized cardinality of intersections between samples and S. Let Y be a random variable representing the normalized in-sample density, i.e., the intersection cardinality normalized by the size of each sample. We aim to show that the probability of Y deviating from μ by θ is less than γ. We bound the variance of Y using the spectral expansion properties of the graph. Let M be the biadjacency binary matrix of the sampler. Intuitively, the second largest eigenvalue λ_2 of the matrix MM^T controls how much the graph behaves like a random graph; a smaller λ_2 implies tighter concentration around the mean. Let $\boldsymbol{u}$ be the characteristic vector of S, i.e., a binary vector with density μ, so it selects $\mu|L|$ elements from L by the ones. By projecting $\boldsymbol{u}$ onto the eigenspace of MM^T, we derive that the variance is bounded by the spectral ratio λ_2/λ_1 multiplied by $\mu(1-\mu)$. Applying Cantelli's inequality yields the required condition on λ_2/λ_1 to ensure the tail probability is less than γ.

Theorem 1. *Let $G = (L, R, E)$ be a (b, v)-bipartite (r, c)-biregular graph and M be the biadjacency matrix of G, a $b \times v$ 0-1 binary matrix where $M_{i,j} = 1$ if the i-th vertex in L is incident to the j-th vertex in R. Let λ_1, λ_2 be the first and second largest eigenvalues of MM^T. For given $\mu \in (0, 1/2)$, $\theta \in (0, 1-\mu]$, and $\gamma \in (0, 1)$, G is a (μ, θ, γ)-density-tailored sampler if the following inequality holds: $\frac{\lambda_2}{\lambda_1} < \frac{\theta^2}{\mu(1-\mu)(\gamma^{-1}-1)}$.*

Proof. Let $|L| = b, |R| = v$, and since M is the biadjacency matrix of G, it is a $b \times v$ 0-1 binary matrix where $M_{i,j} = 1$ if the i-th vertex in L is incident to the j-th vertex in R. Since G is (r, c)-biregular, M has fixed row and column sum: $M\mathbf{1}_{v\times 1} = r \cdot \mathbf{1}_{b\times 1}$, $\mathbf{1}_{1\times b}M = c \cdot \mathbf{1}_{1\times v}$, and $br = vc$. For any μb-subset S of L, the characteristic vector $\boldsymbol{u}$ is a binary vector of size b where $\boldsymbol{u}[i] = 1$ if the i-th vertex of L is in S, otherwise $\boldsymbol{u}[i] = 0$. Therefore, the cardinality of S equals to the sum of elements in $\boldsymbol{u}$, i.e., $\langle \boldsymbol{u}, \mathbf{1} \rangle = \mu|L| = \mu b$. Let $\oslash$ denote the component-wise division between two vectors. Let vector $\boldsymbol{v} := \boldsymbol{u}M$, $\boldsymbol{t} := \mathbf{1}M$ and $\boldsymbol{w} := \boldsymbol{v} \oslash \boldsymbol{t}$. Then $\boldsymbol{v}[j]$ denotes the cardinality of $C_j \cap S$ where C_j refers to $\Gamma(r_j)$, the set of neighbors of the j-th vertex r_j in R. Notice it is already known that $\boldsymbol{t} = c\mathbf{1}_{1\times v}$. Therefore, $w_i = \frac{v_i}{c}$ and $\boldsymbol{w} = \frac{\boldsymbol{v}}{c} = (w_1, \ldots, w_v)$. Consider a sample space Ω induced by the entries $\{w_1, \ldots, w_v\}$ of vector $\boldsymbol{w}$. We uniformly sample a random variable Y over Ω. It is easy to see the following

equivalence holds:

$$\Pr_{j \leftarrow \$ R}\left[\frac{|\Gamma(j) \cap S|}{|\Gamma(j)|} \geq \mu + \theta\right] < \gamma \iff \Pr_{j \leftarrow \$ R}\left[\frac{\boldsymbol{v}[j]}{c} \geq \mu + \theta\right] < \gamma \iff \Pr\left[Y \geq \mu + \theta\right] < \gamma. \tag{1}$$

Hence, by Definition 6, G is a (μ, θ, γ)-density-tailored sampler if and only if the last inequality holds for every subset S with $\mu(S) \leq \mu$. We first prove it for subsets of density exactly μ and handle the case $\mu(S) < \mu$ at the end of the proof.

To continue our proof, we estimate the ℓ_1 and ℓ_2 norm of $\boldsymbol{v}$. Then, we calculate the first and second moments of Y and apply Cantelli's inequality to bound the probability $\Pr[Y \geq \mu + \theta]$. We estimate $\boldsymbol{v}$ in both ℓ_1 and ℓ_2 norm. $\sum_{i=1}^{v} v_i = \langle \boldsymbol{v}, \mathbf{1}_{1\times v}\rangle = \langle \boldsymbol{u}M, \mathbf{1}_{1\times v}\rangle = \boldsymbol{u}M\mathbf{1}_{v\times 1} = \boldsymbol{u}(r\mathbf{1}_{b\times 1}) = \mu r b$. We now calculate the ℓ_2 norm. Note that $\sum_{i=1}^{v} v_i^2 = \langle \boldsymbol{v}, \boldsymbol{v}\rangle = \boldsymbol{u}MM^T\boldsymbol{u}^T$.

Let $\{\boldsymbol{\alpha_i}\}$ be an orthonormal basis of eigenvectors of MM^T corresponding to eigenvalues $\{\lambda_i\}$. Decomposing $\boldsymbol{u}$ under the basis, we have $\boldsymbol{u} = \sum c_i\boldsymbol{\alpha_i}$. Since $\boldsymbol{u}$ is a binary vector, $\langle \boldsymbol{u}, \mathbf{1}\rangle = \langle \boldsymbol{u}, \boldsymbol{u}\rangle$. Moreover, since the basis is orthonormal, observe that $\mu b = \langle \boldsymbol{u}, \mathbf{1}\rangle = \langle \boldsymbol{u}, \boldsymbol{u}\rangle = (\sum c_i\boldsymbol{\alpha_i})(\sum c_i\boldsymbol{\alpha_i}^T) = \sum c_i^2$. We know that M^TM and MM^T share the same nonzero eigenvalues, which are nonnegative. Since the corresponding bipartite graph is (r, c)-biregular, it is known that the largest singular value of M (M^T) is $\sqrt{rc}$ and the largest eigenvalue of M^TM is rc. Hence, $\lambda_1 = rc$. By definition, since $MM^T\boldsymbol{\alpha_1}^T = \lambda_1\boldsymbol{\alpha_1}^T = rc\boldsymbol{\alpha_1}^T$ and $||\boldsymbol{\alpha_1}|| = 1$, it is easy to verify that $\boldsymbol{\alpha_1} = \frac{1}{\sqrt{b}}\mathbf{1}$. This implies $c_1 = \mu\sqrt{b}$ since $\langle \boldsymbol{u}, \boldsymbol{\alpha_1}\rangle = (\sum c_i\boldsymbol{\alpha_i})\boldsymbol{\alpha_1}^T = c_1\boldsymbol{\alpha_1}\boldsymbol{\alpha_1}^T = c_1$ and $\langle \boldsymbol{u}, \boldsymbol{\alpha_1}\rangle = \boldsymbol{u} \cdot \frac{1}{\sqrt{b}}\mathbf{1}^T = \mu b/\sqrt{b} = \mu\sqrt{b}$. We are now ready to calculate $\sum v_i^2$:

$$\begin{aligned}
\sum_{i=1}^{v} v_i^2 = \langle \boldsymbol{v}, \boldsymbol{v}\rangle = \boldsymbol{u}MM^T\boldsymbol{u}^T &= \left(\sum c_i\boldsymbol{\alpha_i}\right) MM^T \left(\sum c_i\boldsymbol{\alpha_i}^T\right) = \left(\sum c_i\boldsymbol{\alpha_i}\right)\left(\sum c_i\lambda_i\boldsymbol{\alpha_i}^T\right) \\
&= \sum c_i^2\lambda_i = c_1^2\lambda_1 + \sum_{i=2} c_i^2\lambda_i = \mu^2 b\lambda_1 + \sum_{i=2} c_i^2\lambda_i \leq \mu^2 b\lambda_1 + \max_{i\geq 2}\{\lambda_i\}\sum_{i=2} c_i^2 \\
&= \mu b\left(\mu\lambda_1 + (1-\mu)\lambda_2\right).
\end{aligned}$$

With ℓ_1 and ℓ_2 estimation, we now compute the first and second moments of random variable $Y \leftarrow\$\ \boldsymbol{w} = (\frac{v_1}{c}, \ldots, \frac{v_{|R|}}{c})$ by definition: (1) $\mathbb{E}[Y] = \frac{1}{|R|}\sum_{i=1}^{|R|}\frac{v_i}{c} = \frac{\sum v_i}{c|R|} = \frac{\mu br}{vc} = \mu$. (2)

$$\begin{aligned}
\mathbb{E}[Y^2] &= \frac{1}{|R|}\sum_{i=1}^{|R|}\left(\frac{v_i}{c}\right)^2 = \frac{\sum v_i^2}{c^2|R|} \\
&\leq \frac{\mu b\left(\mu\lambda_1 + (1-\mu)\lambda_2\right)}{c^2 v} = \frac{\mu b\left(\mu\lambda_1 + (1-\mu)\lambda_2\right)}{brc} = \mu^2 + \frac{\mu\left(1-\mu\right)\lambda_2}{\lambda_1}.
\end{aligned}$$

Hence, the variance $\sigma^2 = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \leq \frac{\mu(1-\mu)\lambda_2}{\lambda_1}$. By Cantelli's inequality, for $\beta > 0$, we have $\Pr(Y \geq \mathbb{E}[Y] + \beta) \leq \frac{1}{1+(\beta/\sigma)^2}$. Therefore, given $\mu \in (0, 1/2)$ and $\theta \in (0, 1-\mu]$, we have $\Pr(Y \geq \mu + \theta) \leq \frac{1}{1+(\theta/\sigma)^2}$. In order for $\Pr(Y \geq \mu + \theta) < \gamma$ to hold, we require $\frac{\theta^2}{\sigma^2} > \gamma^{-1} - 1$. Therefore, we need $\sigma^2 < \frac{\theta^2}{\gamma^{-1}-1}$. We thereby require $\frac{\mu(1-\mu)}{\lambda_1} \cdot \lambda_2 < \frac{\theta^2}{\gamma^{-1}-1}$. This implies λ_2/λ_1 has this upper bound: $\frac{\lambda_2}{\lambda_1} < \frac{\theta^2}{\mu(1-\mu)(\gamma^{-1}-1)}$. Since the assumption constrains λ_2/λ_1 by the same bound, we conclude $\Pr(Y \geq \mu + \theta) < \gamma$ for every subset S of density exactly μ.

Lastly, consider a subset S with density $\mu(S) = \mu' < \mu$. Repeating the argument above with μ' in place of μ gives $\mathbb{E}[Y] = \mu'$ and $\sigma^2 \leq \frac{\mu'(1-\mu')\lambda_2}{\lambda_1} \leq \frac{\mu(1-\mu)\lambda_2}{\lambda_1}$, where the last step uses $\mu' < \mu < 1/2$. Applying Cantelli's inequality with the larger deviation $\beta = \mu + \theta - \mu' > \theta$ yields $\Pr(Y \geq \mu + \theta) = \Pr(Y \geq \mathbb{E}[Y] + \beta) \leq \frac{1}{1+(\beta/\sigma)^2} \leq \frac{1}{1+(\theta/\sigma)^2} < \gamma$, where the final inequality

follows from the same bound on λ_2/λ_1 as above. Since we have shown the equivalence in Equation 1, we conclude that G is a (μ, θ, γ)-density-tailored sampler if the inequality holds. This completes the proof. □

4.2 Balanced Incomplete Block Design

We define a BIBD, standing for balanced incomplete block design, $\mathcal{B}$ to be a family of k-element subsets of a point set $\mathcal{V}$, called blocks, such that any x in $\mathcal{V}$ is contained in r blocks, and any pair of distinct x and y in $\mathcal{V}$ is contained in λ blocks. BIBD is biregular and the spectral ratio λ_2/λ_1 of MM^T (where M is its biadjacency matrix) is $\frac{1}{k}(1 - \frac{k-1}{v-1})$. See Lemma 1.

Lemma 1. *Let $\mathcal{D}$ be a (v, b, r, k, λ)-balanced incomplete block design (BIBD) with $b \geq v \geq k + 1$, and let M be its $v \times b$ incidence matrix. Let G be a bipartite graph with biadjacency matrix M^T and λ_1, λ_2 be the first and second largest eigenvalues of MM^T. Then (1) G is a (b, v)-bipartite (k, r)-biregular graph. (2) $\lambda_1 = rk$, $\lambda_2 = r - \lambda$, and $\frac{\lambda_2}{\lambda_1} = \frac{1}{k} - \frac{\lambda}{rk} = \frac{1}{k}(1 - \frac{k-1}{v-1})$.*

Proof (Lemma 1). Recall that for a BIBD, the incidence matrix M automatically has constant row sum r and constant column sum k, and satisfies the relationship: $MM^T = (r - \lambda)I_v + \lambda J_v$ where I_v is the identity matrix of order v and J_v is the all-ones matrix of order v. The spectrum of J_v consists of the eigenvalue v with multiplicity 1 (corresponding to the eigenvector $\mathbf{1}$) and the eigenvalue 0 with multiplicity $v - 1$ (corresponding to eigenvectors orthogonal to $\mathbf{1}$). Since MM^T is a linear combination of I_v and J_v, its eigenvalues are given by:

1. For the eigenvector $\mathbf{1}$: $(r - \lambda)(1) + \lambda(v) = r + \lambda(v - 1)$. Using the fundamental identity of BIBDs, $\lambda(v - 1) = r(k - 1)$, we substitute to find: $r + r(k - 1) = r + rk - r = rk$.
2. For the eigenvectors orthogonal to $\mathbf{1}$: $(r - \lambda)(1) + \lambda(0) = r - \lambda$.

Thus, the eigenvalues of MM^T are rk (multiplicity 1) and $r - \lambda$ (multiplicity $v - 1$). Finally, since MM^T and M^TM share the same non-zero eigenvalues, M^TM shares the spectrum of MM^T. Since M^TM is a $b \times b$ matrix, the remaining $b - v$ eigenvalues must be 0. □

Next, we show the necessary conditions for the existence of a BIBD. Although these conditions are necessary to the existence of a BIBD, there is no guarantee that a BIBD with specified parameters will exist, even if those parameters satisfy these conditions.

Fact 2 *The necessary conditions for the existence of a (v, b, r, k, λ)-balanced incomplete block design (BIBD) $\mathcal{B}$ with $b \geq v \geq k + 1$ are: $\lambda(v - 1) \equiv 0 \mod (k - 1)$ and $\lambda v(v - 1) \equiv 0 \mod k(k - 1)$ Moreover, we have the following relationships: $bk = vr$, $\lambda(v - 1) = r(k - 1)$, $r = \frac{\lambda(v-1)}{k-1}$ and $b = \frac{\lambda v(v-1)}{k(k-1)}$.*

Fortunately, Abel et al. [2] shows that a large family of BIBDs with block size $k = 9$ and index $\lambda = 72$ can be explicitly constructed.

Lemma 2 (Theorem 7.6 in [2]). *The necessary condition for the existence of a $(v, b, r, k = 9, \lambda = 72)$-BIBD, namely that $v \geq 9$, is sufficient. Particularly, we can explicitly construct a $(v, b, r, k = 9, \lambda = 72)$-BIBD for any $v \geq 9$ and $b = v(v - 1) \geq 72$.*

Combining Theorem 1, Lemma 1 and Lemma 2, we have the Corollary 1.

Corollary 1. *One can explicitly construct a $(1/2 - \epsilon, \epsilon, 1/2)$-density-tailored sampler for any $\epsilon \in (\sqrt{\frac{v-8}{8(5v-4)}}, 1/2)$ where $|L| = v(v - 1)$ and $|R| = v$ for all $v \geq 9$. Particularly, we explicitly obtain a $(1/2 - \sqrt{\frac{v-7}{8(5v-4)}}, 1/2, 1/2)$-robust community allocation strategy.*

Proof (Corollary 1). Following from Theorem 1, Lemma 1 and Lemma 2, $\mathcal{D}$ allows us to construct a (μ, θ, γ)-density-tailored sampler such that $\frac{\theta^2}{\mu(1-\mu)(\gamma^{-1}-1)} > \frac{1}{9}(1 - \frac{8}{v-1})$. Plugging in $\mu = 1/2 - \epsilon$, $\theta = \epsilon$ and $\gamma = 1/2$, the constraint on ϵ reduces to $\epsilon \in (\sqrt{\frac{v-8}{8(5v-4)}}, 1/2)$.

By Proposition 1, since G realizes a $(1/2 - \sqrt{\frac{v-7}{8(5v-4)}}, \sqrt{\frac{v-7}{8(5v-4)}}, 1/2)$-density-tailored sampler, it gives a $(1/2 - \sqrt{\frac{v-7}{8(5v-4)}}, 1/2, 1/2)$-robust community allocation strategy. □

Notice the limitation is that not all integers are admissible as the N-size of the community allocation strategy, but only a subset $\{v(v-1) \mid v \geq 9, v \in \mathbb{N}\}$. This could be improved by sacrificing the fault tolerance and padding a small number of dummy silent nodes to L to form L' such that $|L'| = v(v-1)$ for some v. Thus, we obtain a desirable community allocation strategy for all sufficiently large integers with an exact fault tolerance $1/3$.

Corollary 2. *We can explicitly construct a $(1/3, 1/2, 1/2)$-robust community allocation strategy where $N \in [v(v-1), v(v+1)-1]$ and $m = v+1$ for all $v \geq 97$.*[5]

Proof (Corollary 2). Let $N = v(v-1)$ and $N' = v(v+1)$ be two consecutive admissible sizes. Suppose we want to construct a strategy with a size $T \in [N, N'-1]$, we add $N' - T$ silent nodes to the node set such that it reaches the desired size N'. Then we obtain a new strategy G' with size N' and check the corruption rate while treating the $N' - T$ silent nodes as corrupted. By Corollary 1 (applied with $v+1$ points, so that $|L| = (v+1)v = N'$), G' can tolerate up to $\left(1/2 - \sqrt{\frac{v-6}{8(5v+1)}}\right) N'$ corrupted nodes, and the actual corruption count is bounded by $T/3 + (N' - T)$. Solving the inequality $T/3 + (N' - T) \leq \left(1/2 - \sqrt{\frac{v-6}{8(5v+1)}}\right) N'$ for all $T \in [N, N'-1]$, we get $v \geq 97$. □

4.3 Imbalanced Bipartite Ramanujan Graph

An imbalanced bipartite Ramanujan graph is a bipartite and biregular graph that is Ramanujan, i.e., has an optimal spectral gap, and has an imbalanced number of vertices on each side. Burnwal et al. [22] presented an explicit construction of these graphs for which MM^T has spectral ratio $\lambda_2/\lambda_1 = \frac{1}{l}$. Their construction is based on the "array code" matrices from LDPC (low-density parity check) coding theory. In Lemma 3, we convert their result into sampler formulation. The same limitation seen in Lemma 1 applies here to Lemma 3: The admissible size does not cover all sufficiently large integers but just a sparse distribution. To enrich the admissible size coverage, we apply the same treatment to Lemma 3 as how we improve Lemma 1 in Corollary 2. Thus, we obtain an exact fault tolerance of $1/3 - \epsilon$ and $1/2 - \epsilon$ for all sufficiently large integers N in Corollary 3 and 4.

Lemma 3. *One can explicitly construct (a) a $(1/2 - \epsilon, \epsilon, 1/2)$-density-tailored sampler for any $\epsilon \in (1/(2\sqrt{l+1}), 1/2)$ and (b) a $(1/3 - \epsilon, \epsilon, 1/3)$-density-tailored sampler for any $\epsilon \in (\frac{\sqrt{4l+9}-1}{3(l+2)}, 1/3)$; where $|L| = q^2$ and $|R| = ql$ for all $2 \leq l \leq q$ and q is a prime number.*

Proof (Lemma 3). Theorem 3 in [22] states how the (q^2, ql)-bipartite (l, q)-biregular graph with biadjacency matrix M can be explicitly constructed such that MM^T has spectral ratio $\frac{\lambda_2}{\lambda_1} = \frac{1}{l}$, for all $2 \leq l \leq q$ where q is a prime number. Then by Theorem 1, M allows us to construct a (μ, θ, γ)-density-tailored sampler such that $\frac{\theta^2}{\mu(1-\mu)(\gamma^{-1}-1)} > \frac{1}{l}$. Plugging in $\mu = 1/2 - \epsilon$, $\theta = \epsilon$ and $\gamma = 1/2$, the inequality reduces to $\epsilon > 1/(2\sqrt{l+1})$. Alternatively, for $\mu = 1/3 - \epsilon$, $\theta = \epsilon$ and $\gamma = 1/3$, the constraint on ϵ reduces to $\epsilon \in (\frac{\sqrt{4l+9}-1}{3(l+2)}, 1/3)$. By Proposition 1, the lemma follows. □

[5] The proceedings version [34] states $m = v$. The padding argument applies the $(v+1)$-point design over the whole range $N \in [v(v-1), v(v+1)-1]$, so the resulting strategy has $v+1$ communities.

Corollary 3. *For every $\epsilon \in (0, 1/3)$, there exists $N(\epsilon) = O(\epsilon^{-80/19})$ such that for every integer $N \geq N(\epsilon)$ one can explicitly construct a $(1/3-\epsilon, 1/3, 1/3)$-robust community allocation strategy on N nodes with $O(\epsilon^{-2}\sqrt{N}) = O_\epsilon(\sqrt{N})$ communities, each of size $O(\sqrt{N})$.*[6]

Proof (Corollary 3). Let $\mu_0 := \frac{1}{3} - \frac{\epsilon}{2}, \delta := \frac{1}{3}, \theta_0 := \delta - \mu_0 = \frac{\epsilon}{2}$. Put $L_\epsilon := \left\lceil \frac{16}{9\epsilon^2} \right\rceil$. For every prime $q \geq L_\epsilon$, the imbalanced bipartite Ramanujan construction with parameter $l = L_\epsilon$ gives a (q^2, qL_ϵ)-bipartite biregular graph with $\frac{\lambda_2}{\lambda_1} = \frac{1}{L_\epsilon}$. Since

$$\frac{\theta_0^2}{\mu_0(1-\mu_0)(3-1)} = \frac{\epsilon^2}{8(\frac{1}{3} - \frac{\epsilon}{2})(\frac{2}{3} + \frac{\epsilon}{2})} > \frac{9\epsilon^2}{16} \geq \frac{1}{L_\epsilon},$$

the spectral sampler criterion gives a

$$\left(\frac{1}{3} - \frac{\epsilon}{2}, \frac{\epsilon}{2}, \frac{1}{3}\right)$$

density-tailored sampler. Equivalently, we obtain a

$$\left(\frac{1}{3} - \frac{\epsilon}{2}, \frac{1}{3}, \frac{1}{3}\right)$$

robust community allocation strategy on q^2 nodes with qL_ϵ communities.

Let q_i denote the i-th prime. We claim that there is $i_0 = i_0(\epsilon)$ such that for every $i \geq i_0$, $q_i \geq L_\epsilon$ and $\left(\frac{2}{3} + \frac{\epsilon}{2}\right)\left(\frac{q_{i+1}^2}{q_i^2} - 1\right) \leq \frac{\epsilon}{2}$. Indeed, choose $\alpha_\epsilon := \sqrt{1 + \frac{3\epsilon}{4+3\epsilon}} - 1$. By the Prime Number Theorem, every sufficiently large interval $(x, (1+\alpha_\epsilon)x]$ contains a prime. Quantitatively, by the short-interval theorem of Baker, Harman, and Pintz [11], the interval $[x - x^{21/40}, x]$ contains a prime for all sufficiently large x; since $\alpha_\epsilon x \geq x^{21/40}$ whenever $x \geq \alpha_\epsilon^{-40/19}$, and $\alpha_\epsilon = \Theta(\epsilon)$, there is an absolute constant C such that for all i with $q_i \geq C\epsilon^{-40/19}$, $q_{i+1} \leq (1+\alpha_\epsilon)q_i$. Because $40/19 > 2$, this threshold also dominates the spectral constraint $L_\epsilon = \Theta(\epsilon^{-2})$, so we may take $q_{i_0} = O(\epsilon^{-40/19})$ and hence $N(\epsilon) = q_{i_0}^2 = O(\epsilon^{-80/19})$, as stated. Therefore

$$\frac{q_{i+1}^2}{q_i^2} - 1 \leq (1+\alpha_\epsilon)^2 - 1 = \frac{3\epsilon}{4+3\epsilon},$$

which is equivalent to the desired inequality.

Set $N(\epsilon) := q_{i_0}^2$. Let $N \geq N(\epsilon)$, and choose $i \geq i_0$ such that $q_i^2 \leq N < q_{i+1}^2$. Put $M := q_{i+1}^2$. We use the robust strategy on M nodes and designate $M - N$ nodes as silent.

Let S be any corrupted set among the remaining N active nodes with $|S| \leq \left(\frac{1}{3} - \epsilon\right) N$. After adding all silent nodes as corrupted, the total corrupted set has size at most

$$\left(\frac{1}{3} - \epsilon\right) N + (M - N) = M - \left(\frac{2}{3} + \epsilon\right) N.$$

Since $N \geq q_i^2$, it is enough to show

$$M - \left(\frac{2}{3} + \epsilon\right) q_i^2 \leq \left(\frac{1}{3} - \frac{\epsilon}{2}\right) M.$$

[6] The proceedings version [34] states this corollary with a $(1/3 - \epsilon, 1/2, 1/2)$-robust guarantee and an ϵ-independent $O(\sqrt{N})$ community count. The present version proves the stronger $(1/3-\epsilon, 1/3, 1/3)$ guarantee, on which the Byzantine agreement analysis (Appendix 6.3) relies, at the cost of an $O(\epsilon^{-2}\sqrt{N})$ community count. The proceedings also state $N(\epsilon) = O(\epsilon^{-2}\log^2(1/\epsilon))$; the present version proves $N(\epsilon) = O(\epsilon^{-80/19})$ unconditionally, with $O(\epsilon^{-4})$ under a Legendre-type prime-gap hypothesis (see the end of the proof).

Equivalently,

$$\left(\frac{2}{3}+\frac{\epsilon}{2}\right)M \le \left(\frac{2}{3}+\epsilon\right)q_i^2.$$

Since $M = q_{i+1}^2$, this is precisely

$$\left(\frac{2}{3}+\frac{\epsilon}{2}\right)\left(\frac{q_{i+1}^2}{q_i^2}-1\right) \le \frac{\epsilon}{2}.$$

Thus the padded corrupted set is within the fault tolerance of the original $(1/3-\epsilon/2, 1/3, 1/3)$-robust strategy. The silent-node padding argument therefore yields a $(1/3-\epsilon, 1/3, 1/3)$-robust strategy on the N active nodes. The resulting strategy has $q_{i+1}L_\epsilon = O(\epsilon^{-2}\sqrt{N})$ communities, each of size at most $q_{i+1} = O(\sqrt{N})$.

We also record the natural scale of $N(\epsilon)$ predicted by the first-order prime number theorem. The padding condition above is implied by $\frac{q_{i+1}}{q_i} \le 1+\alpha_3(\epsilon)$, where $\alpha_3(\epsilon) := \sqrt{1+\frac{3\epsilon}{4+3\epsilon}} - 1$. As $\epsilon \to 0$, $\alpha_3(\epsilon) = \frac{3}{8}\epsilon + O(\epsilon^2)$. The prime number theorem gives the estimate $\pi((1+\alpha)x) - \pi(x) \sim \frac{\alpha x}{\log x}$ for fixed $\alpha > 0$. Thus the PNT scale at which every multiplicative interval of relative length $\alpha_3(\epsilon)$ should contain a prime is determined by $\frac{\alpha_3(\epsilon)x}{\log x} \asymp 1$, or equivalently $x \asymp \frac{1}{\alpha_3(\epsilon)} \log \frac{1}{\alpha_3(\epsilon)}$. Since $\alpha_3(\epsilon) \asymp \epsilon$, this gives $q_{i_0} \asymp_{\mathrm{PNT}} \epsilon^{-1} \log \frac{1}{\epsilon}$. This is the scale required only for the padding argument.

In the present $(1/3-\epsilon, 1/3, 1/3)$ case there is also the spectral constraint $q \ge L_\epsilon, L_\epsilon = \left\lceil \frac{16}{9\epsilon^2} \right\rceil$. Since $\epsilon^{-2} \gg \epsilon^{-1}\log\frac{1}{\epsilon}$ as $\epsilon \to 0$, the spectral constraint dominates the padding constraint. Therefore the natural PNT-scale estimate is $q_{i_0} \asymp_{\mathrm{PNT}} \epsilon^{-2}$, and consequently $N(\epsilon) = q_{i_0}^2 \asymp_{\mathrm{PNT}} \epsilon^{-4}$.

We stress that $\asymp_{\mathrm{PNT}}$ records the scale predicted by the first-order asymptotic and is not a proven bound on $N(\epsilon)$: the Prime Number Theorem supplies no effective threshold beyond which every interval of relative length $\alpha_3(\epsilon)$ is guaranteed to contain a prime. The proven rate $N(\epsilon) = O(\epsilon^{-80/19})$ instead follows from the short-interval theorem of Baker, Harman, and Pintz [11], under which the padding constraint $q = \Omega(\epsilon^{-40/19})$ dominates the spectral one. The heuristic ϵ^{-4} scale is conditionally provable: if every interval $[x, x+\sqrt{x}]$ contains a prime (a Legendre-type hypothesis, implied by Cramér's conjecture), the padding constraint relaxes to $q = O(\epsilon^{-2})$, matching the spectral constraint, and $N(\epsilon) = O(\epsilon^{-4})$; under the Riemann Hypothesis, prime gaps are $O(\sqrt{x}\log x)$ [27] and $N(\epsilon) = O(\epsilon^{-4}\log^4(1/\epsilon))$. □

Corollary 4. *For every $\epsilon \in (0, 1/2)$, there exists $N(\epsilon) = O(\epsilon^{-80/19})$ such that for every integer $N \ge N(\epsilon)$ one can explicitly construct a $(1/2-\epsilon, 1/2, 1/2)$-robust community allocation strategy on N nodes with $O(\epsilon^{-2}\sqrt{N}) = O_\epsilon(\sqrt{N})$[7] communities, each of size $O(\sqrt{N})$.*[8]

Proof (Corollary 4). Let $\mu_0 := \frac{1}{2} - \frac{\epsilon}{2}, \delta := \frac{1}{2}, \theta_0 := \delta - \mu_0 = \frac{\epsilon}{2}$, and put $L_\epsilon := \lceil \epsilon^{-2} \rceil$. For every prime $q \ge L_\epsilon$, the imbalanced bipartite Ramanujan construction with parameter $l = L_\epsilon$ gives a (q^2, qL_ϵ)-bipartite biregular graph with $\frac{\lambda_2}{\lambda_1} = \frac{1}{L_\epsilon}$. Since $\frac{\theta_0^2}{\mu_0(1-\mu_0)} = \frac{\epsilon^2}{1-\epsilon^2}$ and $\frac{1}{L_\epsilon} \le \epsilon^2 < \frac{\epsilon^2}{1-\epsilon^2}$, the spectral sampler criterion gives a $\left(\frac{1}{2}-\frac{\epsilon}{2}, \frac{\epsilon}{2}, \frac{1}{2}\right)$ density-tailored sampler. Equivalently, we obtain a $\left(\frac{1}{2}-\frac{\epsilon}{2}, \frac{1}{2}, \frac{1}{2}\right)$ robust community allocation strategy on q^2 nodes with qL_ϵ communities.

Let q_i denote the i-th prime. We claim that there is $i_0 = i_0(\epsilon)$ such that for every $i \ge i_0$, $q_i \ge L_\epsilon$ and $\left(\frac{1}{2}+\frac{\epsilon}{2}\right)\left(\frac{q_{i+1}^2}{q_i^2}-1\right) \le \frac{\epsilon}{2}$. Indeed, choose $\alpha_\epsilon := \sqrt{1+\frac{\epsilon}{1+\epsilon}} - 1$. By the

[7] For fixed ϵ treated as a constant, the number of communities is $O(\sqrt{N})$.

[8] The proceedings version [34] states $N(\epsilon) = O(\epsilon^{-4})$. The present version proves $N(\epsilon) = O(\epsilon^{-80/19})$ unconditionally via the short-interval theorem of Baker, Harman, and Pintz [11]; the ϵ^{-4} rate holds under a Legendre-type prime-gap hypothesis and remains the prediction of the PNT heuristic (see the end of the proof).

Prime Number Theorem, every sufficiently large interval $(x, (1+\alpha_\epsilon)x]$ contains a prime. Quantitatively, as in the proof of Corollary 3, the short-interval theorem of Baker, Harman, and Pintz [11] together with $\alpha_\epsilon = \Theta(\epsilon)$ gives an absolute constant C such that for all i with $q_i \geq C\epsilon^{-40/19}$, $q_{i+1} \leq (1+\alpha_\epsilon)q_i$. Because $40/19 > 2$, this threshold dominates the spectral constraint $L_\epsilon = \Theta(\epsilon^{-2})$, so we may take $q_{i_0} = O(\epsilon^{-40/19})$ and hence $N(\epsilon) = q_{i_0}^2 = O(\epsilon^{-80/19})$, as stated. Therefore $\frac{q_{i+1}^2}{q_i^2} - 1 \leq (1+\alpha_\epsilon)^2 - 1 = \frac{\epsilon}{1+\epsilon}$, which is equivalent to the desired inequality.

Set $N(\epsilon) := q_{i_0}^2$. Let $N \geq N(\epsilon)$, and choose $i \geq i_0$ such that $q_i^2 \leq N < q_{i+1}^2$. Put $M := q_{i+1}^2$. We use the robust strategy on M nodes and designate $M - N$ nodes as silent.

Let S be any corrupted set among the remaining N active nodes with $|S| \leq \left(\frac{1}{2} - \epsilon\right) N$. After adding all silent nodes as corrupted, the total corrupted set has size at most $\left(\frac{1}{2} - \epsilon\right) N + (M - N) = M - \left(\frac{1}{2} + \epsilon\right) N$. Since $N \geq q_i^2$, it is enough to show $M - \left(\frac{1}{2} + \epsilon\right) q_i^2 \leq \left(\frac{1}{2} - \frac{\epsilon}{2}\right) M$. Equivalently, $\left(\frac{1}{2} + \frac{\epsilon}{2}\right) M \leq \left(\frac{1}{2} + \epsilon\right) q_i^2$. Since $M = q_{i+1}^2$, this is precisely $\left(\frac{1}{2} + \frac{\epsilon}{2}\right)\left(\frac{q_{i+1}^2}{q_i^2} - 1\right) \leq \frac{\epsilon}{2}$. Thus the padded corrupted set is within the fault tolerance of the original $(1/2 - \epsilon/2, 1/2, 1/2)$-robust strategy. The silent-node padding argument therefore yields a $(1/2 - \epsilon, 1/2, 1/2)$-robust strategy on the N active nodes.

We also record the natural scale of $N(\epsilon)$ predicted by the first-order prime number theorem. The padding condition above is implied by $\frac{q_{i+1}}{q_i} \leq 1 + \alpha_2(\epsilon)$, where $\alpha_2(\epsilon) := \sqrt{1 + \frac{\epsilon}{1+\epsilon}} - 1$. As $\epsilon \to 0$, $\alpha_2(\epsilon) = \frac{1}{2}\epsilon + O(\epsilon^2)$. The prime number theorem gives the estimate $\pi((1+\alpha)x) - \pi(x) \sim \frac{\alpha x}{\log x}$ for fixed $\alpha > 0$. Thus the PNT scale at which every multiplicative interval of relative length $\alpha_2(\epsilon)$ should contain a prime is determined by $\frac{\alpha_2(\epsilon)x}{\log x} \asymp 1$, and hence $x \asymp \frac{1}{\alpha_2(\epsilon)} \log \frac{1}{\alpha_2(\epsilon)} \asymp \epsilon^{-1} \log \frac{1}{\epsilon}$. This is the scale required only for the padding argument.

In the present $(1/2 - \epsilon, 1/2, 1/2)$ case there is also the spectral constraint $q \geq L_\epsilon, L_\epsilon = \lceil \epsilon^{-2} \rceil$. Since $\epsilon^{-2} \gg \epsilon^{-1} \log \frac{1}{\epsilon}$ as $\epsilon \to 0$, the spectral constraint dominates the padding constraint. Therefore the natural PNT-scale estimate is $q_{i_0} \asymp_{\mathrm{PNT}} \epsilon^{-2}$, and consequently $N(\epsilon) = q_{i_0}^2 \asymp_{\mathrm{PNT}} \epsilon^{-4}$.

As above, $\asymp_{\mathrm{PNT}}$ records the scale predicted by the first-order asymptotic and is not a proven bound on $N(\epsilon)$: the Prime Number Theorem supplies no effective threshold beyond which every interval of relative length $\alpha_2(\epsilon)$ is guaranteed to contain a prime. The proven rate $N(\epsilon) = O(\epsilon^{-80/19})$ again follows from Baker, Harman, and Pintz [11]; a Legendre-type prime-gap hypothesis yields $N(\epsilon) = O(\epsilon^{-4})$, and the Riemann Hypothesis yields $N(\epsilon) = O(\epsilon^{-4} \log^4(1/\epsilon))$ [27]. □

4.4 (k, q)-Net (Finite Bruck Net)

We present one more sampler instantiation from (k, q)-net.

A (k, q)-net is an incidence structure often seen in the context of incidence geometry and combinatorics. A k-net of order q, or a finite *Bruck* net (see [20,21,15,43]), generalizes a finite affine plane of order q where $k \leq q + 1$ and a $(q + 1, q)$-net is equivalent to an affine plane of order q. The net comprises q^2 points and $q \cdot k$ lines. It satisfies the following properties: (1) Every line contains exactly q points; (2) There are k parallel classes of lines. Each point lies on exactly k lines, one from each parallel class. (3) Each parallel class has q lines.

We adopt the following definition:

Definition 8 ((k, q)-net, Def 3.15 in [24]). *A (k, q)-net is a pair $(X, \mathcal{C})$ where X is a set of q^2 elements, and $\mathcal{C}$ is a set cover of X containing kq subsets (blocks) of X each of size q with the property that two distinct blocks intersect in at most one element. Moreover, the blocks of $\mathcal{C}$ can be partitioned into k parallel classes, each containing q blocks. Different blocks*

in the same parallel class do not intersect, while any two blocks in distinct classes intersect exactly once.

We now characterize the biregularity and spectral properties of (k, q)-nets.

Lemma 4. *Let $(X, \mathcal{C})$ be a (k, q)-net with $k \leq q + 1$, and let $G = (X, \mathcal{C}, E)$ be the bipartite graph where $E = \{(x, c) \mid x \in c \in \mathcal{C}\}$. Let M be the biadjacency matrix of G and λ_1, λ_2 be the first and second largest eigenvalues of MM^T. Then (1) G is a (q^2, kq)-bipartite (k, q)-biregular graph. (2) $\lambda_1 = kq$, $\lambda_2 = q$, and $\frac{\lambda_2}{\lambda_1} = \frac{1}{k}$.*

Proof. Let $(X, \mathcal{C})$ be a (k, q)-net, and M be a $q^2 \times kq$ 0-1 binary matrix representing $(X, \mathcal{C})$ where $M_{i,j} = 1$ indicates that the i-th element in X is in the j-th block (subset) of cover $\mathcal{C}$. By Definition 8, the properties of (k, q)-net guarantee that (1) $|X| = q^2$, $|\mathcal{C}| = kq$ and $\forall c \in \mathcal{C}, |c| = q$, (2) each element resides in exactly k blocks, one from each parallel class, and (3) each block contains exactly q elements. Therefore, each row sums to k, i.e., $M\mathbf{1}_{kq\times 1} = k \cdot \mathbf{1}_{q^2\times 1}$, and each column sums to q, i.e., $\mathbf{1}_{1\times q^2} M = q \cdot \mathbf{1}_{1\times kq}$. This implies M and G are (k, q)-biregular.

Since MM^T and M^TM share exactly the same non-zero eigenvalues, we will consider matrix $A = M^TM$ and analyze A and its eigenvalues.

Note $A_{i,j}$ is the inner product of the i-th and j-th column vector in M. Since i and j correspond to the block index of the (k, q)-net, we discuss the result in three cases. (1) If $i = j$, we get $A_{i,i} = q$, since each binary column vector has exactly q elements being one. (2) If $i \neq j$ and the i-th block and the j-th block are in the same parallel class, then they do not intersect and $A_{i,j} = 0$. (3) If $i \neq j$ and the i-th block and the j-th block are in distinct parallel classes, then they intersect exactly once and $A_{i,j} = 1$. Summing up, we see that

$$A_{i,j} = \begin{cases} q, & \text{if } i = j, \\ 0, & \text{if } i \neq j;\ i, j \text{ are in the same parallel class,} \\ 1, & \text{otherwise.} \end{cases}$$

Without loss of generality, assume the columns of M are clustered and sorted by parallel classes, then A can be rephrased as $A = q \cdot I_{kq} + B$, where I_{kq} is the $kq \times kq$ identity matrix and B is a $kq \times kq$ block matrix consisting of $q \times q$ submatrices: $B = \begin{bmatrix} 0_{q\times q} & 1_{q\times q} & \cdots & 1_{q\times q} \\ 1_{q\times q} & 0_{q\times q} & \cdots & 1_{q\times q} \\ \vdots & \vdots & \ddots & \vdots \\ 1_{q\times q} & 1_{q\times q} & \cdots & 0_{q\times q} \end{bmatrix}$. Its structure alternates between $0_{q\times q}$, a block of zeros, and $1_{q\times q}$, a block of ones. The main diagonal of B consists of $0_{q\times q}$ blocks corresponding to subsets of $\mathcal{C}$ within the same parallel class, while the off-diagonal blocks are $1_{q\times q}$, linking to non-parallel classes.

We can further rewrite B as a Kronecker product: $B = J_k \otimes 1_{q\times q} - I_k \otimes 1_{q\times q}$, where J_k is the $k \times k$ matrix of all ones, and I_k is the $k \times k$ identity matrix.

To determine the eigenvalues of B, we focus on its structure and symmetry. The eigenvalues of $J_k - I_k$ are $k - 1$ (with multiplicity 1) and -1 (with multiplicity $k - 1$). Since $1_{q\times q}$ has eigenvalues q (with multiplicity 1) and 0 (with multiplicity $q - 1$), the eigenvalues of B are the products of eigenvalues of $J_k - I_k$ and $1_{q\times q}$. Thus the eigenvalues of B are: $(k - 1)q$ with multiplicity 1, $-q$ with multiplicity $k - 1$, 0 with multiplicity $k(q - 1)$. Since $q \cdot I_{kq}$ has eigenvalues q with multiplicity kq and $A = q \cdot I_{kq} + B$, the eigenvalues of A are: kq with multiplicity 1, 0 with multiplicity $k - 1$, q with multiplicity $k(q - 1)$. Therefore, we see that the eigenvalues $\{\lambda_i\}$ of MM^T are explicitly $\lambda_1 = kq$, $\lambda_2 = \ldots = \lambda_{k(q-1)+1} = q$ and $\lambda_{k(q-1)+2} = \ldots = \lambda_{q^2} = 0$. □

Efficient construction of (k, p)-net. A well-known and natural construction of (k, p)-net is summarized in Moorhouse's handout [44, 4.1 Example: Translation Nets]. We formalize the construction in the following algorithm and analyze its time and space complexity.

Algorithm 1 NetGeneration(k, p)

Require: $p = q^n$ is a prime power, number of parallel classes $k \leq p + 1$
Ensure: A collection of k pairwise disjoint n-dimensional subspaces of F_q^{2n}, forming a (k, p)-net

Let $F = GF(q)$ be the finite field of order q
Fix a monic irreducible polynomial $f \in F[x]$ of degree n; let $C \in F^{n\times n}$ be the companion matrix of f ▷ $F[C] \cong F[x]/(f) \cong GF(p)$
$\Sigma \leftarrow \emptyset$
function CreateSubspace(M)
    $S \leftarrow$ zero matrix of size $2n \times n$ over F
    Set upper $n \times n$ block of S to identity matrix
    Set lower $n \times n$ block of S to M
    return S
$S_\infty \leftarrow$ zero matrix of size $2n \times n$ over F
Set lower $n \times n$ block of S_∞ to identity matrix ▷ Vertical subspace (infinity slope)
$\Sigma \leftarrow \Sigma \cup \{S_\infty\}$
for $i \leftarrow 0$ to $k - 2$ **do**
    $m_1 \leftarrow$ the i-th vector of F^n under a fixed enumeration ▷ Coefficient vector of the i-th field element $\alpha_i \in F[x]/(f)$
    for $j \leftarrow 2$ to n **do**
        $m_j \leftarrow C \cdot m_{j-1}$
    $M_i \leftarrow [m_1 \mid m_2 \mid \cdots \mid m_n]$ ▷ Matrix of multiplication by α_i, i.e., $M_i = \sum_{j=0}^{n-1} a_j C^j$ where $\alpha_i = \sum_{j=0}^{n-1} a_j x^j$
    $\Sigma \leftarrow \Sigma \cup \{$CreateSubspace$(M_i)\}$
return Σ

Algorithm 2 Generation of Biadjacency Matrix

Require: Translation net Σ, field size q, dimension n
Ensure: Biadjacency matrix B

$P \leftarrow$ enumerate all points in F^{2n} ▷ q^{2n} points
$L \leftarrow \emptyset$ ▷ Initialize empty set of all lines
function GenerateParallelLines(S)
    $L' \leftarrow \emptyset$ ▷ Initialize empty set of lines
    for all $v \in F^{2n}$ **do** ▷ All possible translation vectors
        $\ell \leftarrow \emptyset$ ▷ Initialize empty line
        for all $c \in F^n$ **do** ▷ All possible coefficient vectors
            $u \leftarrow Sc + v$ ▷ Point on the line
            $\ell \leftarrow \ell \cup \{u\}$
        $L' \leftarrow L' \cup \{\ell\}$
    return L'
for all $S \in \Sigma$ **do**
    $L \leftarrow L \cup$ GenerateParallelLines(S)
Initialize B as zero matrix of size $|P| \times |L|$
for all lines $\ell \in L$ **do**
    for all points $u \in \ell$ **do**
        $B[\text{index}(u), \text{index}(\ell)] \leftarrow 1$
return B

Complexity Analysis. Algorithm 1 takes $O(kn^2)$ time to compute since each of the k subspaces is a matrix of shape $2n \times n$ where $n = \log_q p$: each slope matrix M_i is built column by column, and each column is obtained from the previous one by a product with the companion matrix C, which takes $O(n)$ time owing to the sparsity of C. $O(kn^2 \log p)$ bits are needed to encode k subspaces as the output. Algorithm 2 generates the biadjacency matrix of shape $q^{2n} \times kq^n = p^2 \times kp$. It takes $O(q^{2n} \cdot q^n) = O(p^3)$ time to generate all parallel lines for a subspace, hence the overall time complexity of Algorithm 2 becomes $O(kp^3)$. In terms of storing the output, since the output matrix B is a zero-one binary matrix, $O(kp^3)$ bits are

needed for Algorithm 2. In summary, the time and space complexity to generate a (k, p)-net are $O(kp^3)$ and $O(kp^3)$ bits respectively.

Correctness. The correctness of these algorithms follows from basic facts about F_q^{2n}. Observe that there are $p^2 = q^{2n}$ points in F_q^{2n}, whereas a (k, p)-net has the same number of points. Besides, any n-dimensional subspace S of F_q^{2n} has $p = q^n$ disjoint cosets. Hence, an n-dimensional subspace S can model a parallel class in a (k, p)-net, and the p cosets of S model the p lines in the class. As a result, k pairwise disjoint subspaces (any two intersecting only at the origin) model a (k, p)-net. The k subspaces returned by Algorithm 1 are indeed pairwise disjoint: the slope matrices M_i lie in the field $F[C] \cong GF(p)$, so for $i \neq i'$ the difference $M_i - M_{i'}$ is a nonzero field element and hence invertible, which means $(x, M_i x) = (x', M_{i'} x')$ only if $x = x' = 0$; likewise, each subspace $\{(x, M_i x) \mid x \in F^n\}$ intersects the vertical subspace $\{(0, y) \mid y \in F^n\}$ only at the origin.

5 Our Semi-interactive Threshold Signature Protocol

We first introduce the syntax (Def. 9) and security notions (Def. 10 and 11) for semi-interactive threshold signature schemes. Then, we present our construction (Section 5.2), security (Thm. 3, 5, 6) and complexity analysis (Thm. 7). Regarding security, we consider the unforgeability and robustness against weakly adaptive adversaries.

5.1 Syntax

Definition 9 (Semi-Interactive Threshold Signature). *A semi-interactive $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature scheme, denoted as Π, involves n parties $\mathcal{P} = \{P_i\}_{i\in[n]}$ and is parameterized by a community allocation strategy $\mathcal{X} = \{\mathsf{CMNT}_j\}_{j\in[m]}$ that allocates the n parties into m communities of size q. (We denote $\mathsf{CMNT}^{-1}[i] = \{j \mid i \in \mathsf{CMNT}_j\} \subset [m]$ as the subset of communities that P_i belongs to.) On input a message $pt \in \mathcal{M}$, Π outputs a full signature σ that satisfies the verification algorithm. It consists of a setup phase $\mathsf{PublicParamSetup}$, a key generation algorithm KGen, a signing protocol $\mathsf{SignProto}$, and a verification algorithm Ver:*

- $\mathsf{siTS.PublicParamSetup}(1^\lambda, \mathcal{X}) \to \mathsf{pp}$*: The setup phase takes as input a security parameter and the community allocation strategy $\mathcal{X}$, and outputs public parameters pp to all parties (which are given implicitly as input to all other algorithms). In particular, pp embeds $\mathcal{X} = \{\mathsf{CMNT}_j \mid j \in [m], \mathsf{CMNT}_j \subset [n]\}$.*
- $\mathsf{siTS.KGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$*: The key generation algorithm outputs a vector of public keys pk and a vector of secret keys $\mathsf{sk} = \{sk_{(j,i)} \mid i \in [n], j \in \mathsf{CMNT}^{-1}[i]\}$, where pk is published and $\mathsf{sk}_i = \{sk_{(j,i)} \mid j \in \mathsf{CMNT}^{-1}[i]\}$ is exclusively obtained by P_i.*
- $\mathsf{siTS.SignProto}\langle pt, \mathsf{pk}, \mathsf{sk}_i\rangle \to \sigma$*: The signing protocol that each party P_i invokes with a message $pt \in \mathcal{M}$ and its keys; collectively the participating parties interactively produce a full signature σ.*
- $\mathsf{siTS.Ver}(\mathsf{pk}, pt, \sigma) \to 0/1$*: The signature verification algorithm takes as input a public key pk, a message pt, and a full signature σ. It outputs 1 (accept) or 0 (reject).*

Π is semi-interactive *in the sense that within $\mathsf{SignProto}$ the partial signature shares are generated non-interactively, whereas the aggregation process may involve multiple rounds.*

Security. A semi-interactive threshold signature scheme must satisfy *robustness*, *unforgeability*, and *uniqueness*. More precisely, let $\mathcal{A}$ be a PPT adversary that can adaptively corrupt parties but cannot retract undelivered messages, subject to the constraint that at most α communities contain more than β corrupted members. Against any such adversary $\mathcal{A}$, the scheme guarantees the following properties except with a negligible probability:

– **Robustness.** If all honest parties participate in signing the same message msg, all honest parties will eventually obtain a valid signature σ for msg under the group public key pk.
– **Unforgeability.** The adversary cannot generate a valid signature σ for a message msg if no forever-honest party has participated in signing msg. This guarantee holds even if the adversary is allowed to request honest parties to sign arbitrarily many other messages and observe the resulting communication transcripts.
– **Uniqueness.** Under the group public key pk, every message has a unique valid signature. That is, for any message msg, if σ and σ' are valid signatures for msg under pk, then $\sigma = \sigma'$.

We formalize unforgeability and robustness as the game-based definitions in Definition 10 and 11 below, and establish uniqueness in Theorem 6. We consider all of them in the weakly adaptive corruption setting.

On the choice of security definition. We adopt *UF-CMA-Community* (Def. 10) rather than the classical (n, t) UF-CMA game as our primary security notion. The reason is that the threshold parameter t plays no intrinsic role in our construction: corruption budgets are naturally expressed per-community as (α, β), and t only enters indirectly through the (t/n)-robustness of the community allocation strategy $\mathcal{X}$. Stating security directly in community terms yields a cleaner definition that exposes exactly the quantities the proof relies on. Readers seeking the classical formulation are referred to Appendix B, where we recall the standard (n, t) UF-CMA game (Def. 12) and establish a tight reduction (Lemma 17)

$$\Pr\big[\text{UF-CMA}^{\mathcal{B}}_{\Sigma,t,\mathcal{X}} = 1\big] \;\le\; \Pr\big[\text{UF-CMA-Community}^{\mathcal{B}}_{\Sigma,\alpha,\beta,\mathcal{X}} = 1\big].$$

At a high level, the reduction simulates a classical adversary $\mathcal{B}$ that corrupts up to t individual parties by a community adversary $\mathcal{A}$ that, for each corrupted party P_i, queries the corruption oracle on every pair (j, i) with $j \in \mathsf{CMNT}^{-1}[i]$. Whenever $\mathcal{X}$ is (t/n)-robust, any such set of corruptions necessarily leaves an $(m - \alpha)$-subset of communities with at most β corrupted shares each, so $\mathcal{A}$'s forgery satisfies the UF-CMA-Community winning condition exactly when $\mathcal{B}$'s forgery satisfies the classical one. Consequently, our community-level guarantee implies standard (n, t) UF-CMA security "for free" under any (t/n)-robust allocation, justifying the removal of t from the main definition.

Definition 10 (Unforgeability Under Chosen Message Attack). *Let siTS be a semi-interactive $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature scheme (Def. 9), with internal partial-signing function $\mathsf{siTS.PSign}$; we write $\Sigma := \mathsf{siTS}$ for brevity. For an algorithm $\mathcal{A}$, define experiment $\text{UF-CMA-Community}^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}}$ in Fig 1 and as follows:*

– ***Setup Phase.*** *The community allocation strategy $\mathcal{X} = \{\mathsf{CMNT}_j\}_{j\in[m]}$ is an input to the setup phase. A trusted party runs $\mathsf{siTS.PublicParamSetup}(1^\lambda)$ to generate public parameters pp and $\mathsf{siTS.KGen}(\mathsf{pp})$ to generate public and secret key shares $\{pk_j \mid j \in [m]\}$, $\{pk_{(j,i)} \mid j \in [m], i \in \mathsf{CMNT}_j\}$ and $\{sk_{(j,i)} | i \in [n], j \in \mathsf{CMNT}^{-1}[i]\}$. The public parameters pp and public key shares are published and the secret key shares $\{sk_{(j,i)} | j \in \mathsf{CMNT}^{-1}[i]\}$ are privately distributed to P_i. In particular, the input $\mathcal{X}$ is included in the public parameters pp as the community allocation strategy.*
Initialize sets $\mathcal{H} := \{(j, i) \mid j \in [m], i \in \mathsf{CMNT}_j\}$, $\mathcal{C} := \emptyset$, $\mathcal{Q}_{\mathsf{sign}} := \emptyset$. Run $\mathcal{A}$ on input par.
– ***Corruption Queries.*** *At any point of the experiment, $\mathcal{A}$ may corrupt a partial secret key share $sk_{(j,i)}$ by submitting an index pair (j, i). In this case, return $sk_{(j,i)}$ and set $\mathcal{H} = \mathcal{H} \setminus \{(j, i)\}$, $\mathcal{C} = \mathcal{C} \cup \{(j, i)\}$.*
– ***Online Phase.*** *Here, $\mathcal{A}$ gets additional access to oracles that answer queries of the following types:*

Fig. 1. The unforgeability security game UF-CMA-Community$^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}}$

Game UF-CMA-Community$^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}}$

$\mathcal{X} \to \{\mathsf{CMNT}_j\}_{j\in[m]}$;
$\mathsf{PublicParamSetup}() \to pp$;
$\mathsf{KGen}(pp) \to pk, \{pk_j\}_{j\in[m]}$,
  $\{pk_{(j,i)}\}_{j\in[m],\, i\in\mathsf{CMNT}_j}$,
  $\{sk_{(j,i)}\}_{j\in[m],\, i\in\mathsf{CMNT}_j}$;
$\mathcal{H} := \{(j,i) \mid j \in [m],\, i \in \mathsf{CMNT}_j\}$;
$\mathcal{C} := \emptyset$; $\mathcal{Q}_{\mathsf{sign}} := \emptyset$;
$\mathrm{inp} := (pp,\, pk,\, \{pk_j\}_j,\, \{pk_{(j,i)}\}_{j,i})$;
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{CORR},\,\mathrm{PSIG},\,\mathrm{RO}}(\mathrm{inp})$;
// shares known to $\mathcal{A}$ in community
$\forall j \in [m] : \mathcal{B}_j := \{i \mid (j,i) \in \mathcal{C}\} \cup$
  $\{i \mid (j,i,m^*) \in \mathcal{Q}_{\mathsf{sign}}\}$;
if $\big|\{\, j \in [m] : |\mathcal{B}_j| \le \beta \,\}\big| \ge m - \alpha \wedge$
  $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$: **return** 1;
return 0

Oracle CORR(j,i)

if $(j,i) \notin \mathcal{H}$: **return** $\perp$;
$\mathcal{C} := \mathcal{C} \cup \{(j,i)\}$;
$\mathcal{H} := \mathcal{H} \setminus \{(j,i)\}$;
return $sk_{(j,i)}$

Oracle PSIG(j,i,m)

if $(j,i) \notin \mathcal{H}$: **return** $\perp$;
$\mathcal{Q}_{\mathsf{sign}} := \mathcal{Q}_{\mathsf{sign}} \cup \{(j,i,m)\}$;
$\sigma_{(j,i)} \leftarrow \mathsf{PSign}(sk_{(j,i)}, m)$;
return $\sigma_{(j,i)}$

Oracle RO(m)

if $H[m] = \perp$: $H[m] \leftarrow\!\$\ \mathbb{G}$;
return $H[m]$

- ***Signing Queries.*** *When $\mathcal{A}$ submits a tuple (j,i,m) for $(j,i) \in \mathcal{H}$, return $\sigma_{(j,i)} \leftarrow \mathit{siTS.PSign}(sk_{(j,i)}, m)$ where $sk_{(j,i)}$ denotes the partial secret key of party P_i in the j-th community. Record the tuple (j,i,m) in $\mathcal{Q}_{\mathit{sign}} = \mathcal{Q}_{\mathit{sign}} \cup \{(j,i,m)\}$.*
- ***Random Oracle Queries.*** *When $\mathcal{A}$ submits a query m, check if $H[m] = \perp$ and if so, set $H[m] \leftarrow\!\$\ \mathbb{G}$. Return $H[m]$.*

– ***Output Determination.*** *Let $\mathcal{S}$ denote the subset of pairs (j,i) that are corrupted or for which $\mathcal{A}$ has ever made a signing query of the form (j,i,m^*), i.e., $\mathcal{S} := \{(j,i) \mid (j,i,m^*) \in \mathcal{Q}_{\mathit{sign}}\} \cup \mathcal{C}$. When $\mathcal{A}$ outputs a message m^* and a signature σ^*, output 1 if*
  - *there exists an $(m-\alpha)$-subset $\mathcal{H}_{\mathsf{CMNT}} \subset [m]$ of communities, such that for any $j^* \in \mathcal{H}_{\mathsf{CMNT}}$, $|\mathcal{S} \cap \{(j^*, i) \mid i \in \mathsf{CMNT}_{j^*}\}| \le \beta$; and*
  - *$\mathit{Ver}(pk, m^*, \sigma^*) = 1$.*

  Otherwise, output 0.

*We say that Σ is (ϵ, T, q_h, q_s)-*unforgeable under chosen message attacks in communities (UF-CMA-Community) *if for all algorithms $\mathcal{A}$ running in time at most T, making at most q_h random oracle queries, and making at most q_s signing queries, $\Pr[\mathit{UF\text{-}CMA\text{-}Community}^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}} = 1] \le \epsilon$. Conversely, we say that $\mathcal{A}$ (ϵ, T, q_h, q_s)-breaks unforgeability of Σ under chosen message attacks if it runs in time at most T, makes at most q_h queries to the random oracle, makes at most q_s queries to the signing oracle, and $\Pr[\mathit{UF\text{-}CMA\text{-}Community}^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}} = 1] > \epsilon$.*

Definition 11 (Robustness). *Let siTS be a semi-interactive $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature scheme. Given a community allocation strategy $\mathcal{X} = \{\mathsf{CMNT}_j\}_{j\in[m]}$ and any input message pt, siTS is robust if, whenever there are at least $\alpha+1$ communities in each of which at most β parties are adaptively compromised, every uncompromised party P_i eventually outputs the same σ such that $\mathit{siTS.Ver}(\mathsf{pk}, pt, \sigma) = 1$.*

5.2 Construction

We describe our semi-interactive $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature construction in this section.

$\underline{\mathsf{PublicParamSetup}(1^\lambda, \mathcal{X})} \to \mathsf{pp}$: Let $\mathsf{GGen}(1^\lambda) \to (\mathbb{G}, \mathbb{G}_T, p, g)$ be pairing groups of prime order p, generator $g \in \mathbb{G}$, and bilinear pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Let $\mathsf{H} : \mathcal{M} \to \mathbb{G}$ be a hash function modeled as a random oracle.

The public parameters of our scheme are then $\mathsf{pp} = (\mathbb{G}, g, \mathsf{H},$
$\mathcal{X} = \{\mathsf{CMNT}_j\}_{j\in[m]})$. We assume that all the algorithms below implicitly take the public parameters as input. We note that the community allocation strategy $\mathcal{X}$ is externally generated.

Remark 2 (Pairing setting (erratum)). The proceedings version [34] presents the construction in the notation of Type-3 asymmetric pairing groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ with $\mathsf{H} : \mathcal{M} \to \hat{\mathbb{G}}$. This notation is inaccurate: the scheme and its security analysis, which follow the template of Bacho and Loss [8], are in fact carried out over a symmetric pairing group, and the present version corrects the statement accordingly, with $\mathbb{G} = \hat{\mathbb{G}}$ throughout. The unforgeability proof in Appendix 5.4 relies on this symmetric setting: algorithm $\mathcal{A}_1$ answers signing queries using the challenge public keys directly, which is not possible when hash outputs and public keys live in different source groups. We do not claim that the analysis carries over to an asymmetric (Type-3) instantiation, and we leave its security in that setting open. For threshold BLS over Type-3 groups, Bacho and Loss [9] discuss this issue and introduce the co one-more discrete logarithm (Co-OMDL) assumption; see Appendix D of the full version [10].

$\underline{\mathsf{KGen}(\mathsf{pp})}$: To instantiate the key generation, we define a two-level polynomial sharing structure, denoted by F on the points used by the scheme: $F(\cdot, 0)$ is a univariate polynomial of degree α, and $F(j, \cdot)$ is a univariate polynomial of degree β for every community $j \in [m]$. Concretely, sample $A(X) := F(X, 0)$ by choosing $\alpha + 1$ coefficients independently and uniformly from $\mathbb{Z}_p$. For each community $j \in [m]$, sample $B_j(Y) := F(j, Y)$ by fixing $B_j(0) = A(j)$ and choosing its β non-constant coefficients independently and uniformly from $\mathbb{Z}_p$. Thus $F(0, 0) := A(0)$, $F(j, 0) := A(j) = B_j(0)$, and $F(j, i) := B_j(i)$ for all $j \in [m]$ and $i \in \mathsf{CMNT}_j$. The signing key share of the i-th signer (P_i) in the j-th community (CMNT_j) is $sk_{(j,i)} := F(j, i) = B_j(i)$ and it is exclusively obtained by P_i. We set $pk = g^{F(0,0)} = g^{A(0)}$ as the group public key, $pk_j = g^{F(j,0)} = g^{A(j)} = g^{B_j(0)}$ as the public key share of the j-th signing community, and $pk_{(j,i)} = g^{F(j,i)} = g^{B_j(i)}$ as the public key share of the i-th signer in the j-th signing community. In concrete steps, the key generation algorithm proceeds as follows:

1. **Sample the community polynomial.** Sample $a_0, \ldots, a_\alpha \leftarrow\!\$\ \mathbb{Z}_p$ independently and define
$$A(X) := \sum_{\ell=0}^{\alpha} a_\ell X^\ell.$$
2. **Sample the community-local polynomials.** For every community $j \in [m]$, sample $b_{j,1}, \ldots, b_{j,\beta} \leftarrow\!\$\ \mathbb{Z}_p$ independently and define
$$B_j(Y) := A(j) + \sum_{\ell=1}^{\beta} b_{j,\ell} Y^\ell.$$
3. **Define the share notation.** For the points used by the scheme, define
$$F(0, 0) := A(0), \quad F(j, 0) := A(j) = B_j(0), \quad F(j, i) := B_j(i).$$
4. **Generate signing key shares.** For every community $j \in [m]$ and every signer $P_i \in \mathsf{CMNT}_j$, define the signing key share
$$sk_{(j,i)} := F(j, i) = B_j(i).$$
The key $sk_{(j,i)}$ is given only to signer P_i.

5. **Generate the global public verification key.** Define $pk := g^{F(0,0)} = g^{A(0)}$.
6. **Generate community public keys.** For every community $j \in [m]$, define its public key share as $pk_j := g^{F(j,0)} = g^{A(j)} = g^{B_j(0)}$.
7. **Generate signer public keys.** For every community $j \in [m]$ and every signer $P_i \in \mathsf{CMNT}_j$, define the public key share of P_i in community j as $pk_{(j,i)} := g^{F(j,i)} = g^{B_j(i)}$.
8. **Publish the public keys.** Publish

$$\mathsf{pk} := \left(pk, \{pk_j\}_{j\in[m]}, \{pk_{(j,i)}\}_{\{(j,i)|j\in[m],\, P_i\in\mathsf{CMNT}_j\}}\right).$$

9. **Distribute the secret keys.** For every signer P_i, privately send P_i the set of secret key shares

$$\{sk_{(j,i)} \mid j \in [m],\, P_i \in \mathsf{CMNT}_j\}.$$

The algorithm returns the public key

$$\mathsf{pk} := \left(pk, \{pk_j\}_{j\in[m]}, \{pk_{(j,i)}\}_{\{(j,i)|j\in[m],\, P_i\in\mathsf{CMNT}_j\}}\right)$$

and the distributed secret key shares

$$\mathsf{sk} := \{sk_{(j,i)}\}_{\{(j,i)|j\in[m],\, P_i\in\mathsf{CMNT}_j\}}.$$

$\underline{\mathsf{Ver}(\mathsf{pk}, pt, \sigma)}$: On input a public key (share) pk, a (partial) signature σ and a message pt, the (partial) signature verification function returns 1 if $e(g,\sigma) = e(pk, \mathsf{H}(pt))$ and 0 otherwise.
$\underline{\mathsf{SignProto}\langle pt, \mathsf{pk}, \mathsf{sk}_i\rangle \to \sigma}$: To instantiate the semi-interactive signing protocol $\mathsf{SignProto}$, we provide the code P_i executes in Figure 2 and elaborate on the internal functions $\mathsf{SignProto}$ invokes:

Fig. 2. The Signing Protocol $\mathsf{SignProto}$ for each party P_i

Initialization: each party P_i with $\mathcal{X}$ **do**:

The initialization is the same
as in Game 1 Line 1-8.
$\forall j \in \mathsf{CMNT}^{-1}[i], \mathsf{Sig}[j] \leftarrow \{\}$,
$\mathsf{Idx}[j] \leftarrow \{\}; T \leftarrow \{\}, T_{idx} \leftarrow \{\}$;
$\forall j \in [m], abort[j] \leftarrow \textbf{false}; done \leftarrow \textbf{false}$

Upon inputting $pt : P_i$ **do**:

// intra-community level
for each $j \in \mathsf{CMNT}^{-1}[i]$ **do**
  $\sigma_{(j,i)} \leftarrow \mathsf{PSign}(sk_{(j,i)}, pt)$
  for each $i' \in \mathsf{CMNT}_j$ **do**
    Send $(\sigma_{(j,i)}, (j,i))$ to $P_{i'}$

Upon receiving $(\sigma_{(j,i')}, (j,i'))$ from $P_{i'} : P_i$ **do**:

// intra-community aggregation
if $i' \in \mathsf{CMNT}_j \wedge \mathsf{PVerIntra}(pk_{(j,i')}, pt, \sigma_{(j,i')}) = 1$ **then**
  $\mathsf{Idx}[j] \leftarrow \mathsf{Idx}[j] \cup \{i'\}$;
  $\mathsf{Sig}[j] \leftarrow \mathsf{Sig}[j] \cup \{(pk_{(j,i')}, \sigma_{(j,i')})\}$
if $|\mathsf{Sig}[j]| = \beta + 1$ **then** $\sigma_j \leftarrow \mathsf{PCombine}(\mathsf{Idx}[j], \beta, pt, \mathsf{Sig}[j])$
  if $\sigma_j \neq \perp$ **then** // for the first time
    $\mathbb{T}_i \leftarrow\!\!\$$ sample a subset of size $\chi = \Omega\left(\sqrt{n}(\gamma + \log n)\right)$;
    Send (σ_j, j) to P_ℓ for all $\ell \in \mathbb{T}_i$;
    $abort[j] \leftarrow \textbf{true}; T_{idx} \leftarrow T_{idx} \cup \{j\}; T \leftarrow T \cup \{(pk_j, \sigma_j)\}$
    if $done = \textbf{false} \wedge |T| \geq \alpha + 1$ **then**
      $done \leftarrow \textbf{true}$; Output $\sigma = \mathsf{Combine}(T_{idx}, \alpha, pt, T)$

Upon receiving (σ_j, j) from $P_{i'} : P_i$ **do**:

// inter-community level
if $abort[j] = \textbf{false} \wedge i' \in \mathsf{CMNT}_j \wedge \mathsf{PVer}(pk_j, pt, \sigma_j) = 1$ **then**
  $abort[j] \leftarrow \textbf{true}; T_{idx} \leftarrow T_{idx} \cup \{j\}; T \leftarrow T \cup \{(pk_j, \sigma_j)\}$
if $done = \textbf{false} \wedge |T| \geq \alpha + 1$ **then**
  $done \leftarrow \textbf{true}$; Output $\sigma = \mathsf{Combine}(T_{idx}, \alpha, pt, T)$

$\underline{\mathsf{PSign}(sk_{(j,i)}, pt)} \to \sigma$: On input $sk_{(j,i)}$ and $pt \in \mathcal{M}$, the partial signing function returns the signature share $\sigma_{(j,i)} := \mathsf{H}(pt)^{sk_{(j,i)}}$.
$\underline{\mathsf{PVerIntra/PVer}(\mathsf{pk}, pt, \sigma)}$: It has an identical instantiation of the global verification algorithm $\mathsf{Ver}(\mathsf{pk}, pt, \sigma)$.
$\underline{\mathsf{PCombine/Combine}(S, thld, pt, \{(\mathrm{pk}_{i'}, \rho_{i'})\}_{i' \in S})} \to \sigma/\perp$: On input a set of signer indices S with $|S| \geq thld+1$, a message $pt \in \mathcal{M}$, a set of tuples $(\mathrm{pk}_{i'}, \rho_{i'})$ consisting of public key shares and signature shares of each signer in S, the combine algorithm runs $\mathsf{PVerIntra}(pk_{i'}, pt, \rho_{i'})$ (resp. PVer) for all $i' \in S$. If any of these calls returns 0, return $\perp$. Otherwise, return $\sigma = \prod_{i' \in S} \rho_{i'}^{L_{i',S}}$, where $L_{i',S} = \prod_{j \in S \setminus \{i'\}} j/(j-i')$ denotes the i'-th Lagrange coefficient for the set S.

We note that our theorems assume there exists an externally generated "robust" community allocation strategy $\mathcal{X}$. For a network with $n \geq 3t+1$ where t is the number of corrupted parties, the allocation strategy $\mathcal{X}$ can be instantiated by Corollary 2 and Proposition 1. For $f < (1/2 - \epsilon)n$, $\mathcal{X}$ can be instantiated by Corollary 4.

5.3 Robustness

We prove the robustness of our semi-interactive threshold signature scheme under the adaptive corruption setting and the existence of a $(1/2-\epsilon)$-robust community allocation strategy.

Theorem 3 (Robustness of the Threshold Signature Scheme). *Let $\mathcal{X}$ be a community allocation strategy partitioning n parties into m communities, each of size q, with $m, q \in \Theta(\sqrt{n})$, $m \geq 2\alpha+1$, and $q \geq 2\beta+1$. Conditioned on at most α communities containing more than β corrupted members, our $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature scheme in Section 5.2 is robust against any weakly adaptive, polynomial-time adversary with overwhelming probability at least $1 - 2^{-\gamma}$.*

Proof (Theorem 3). By the theorem assumption, for any corrupted subset S with at most α communities containing more than β corrupted members, it holds that $|\{j \mid |\mathsf{CMNT}_j \cap S| > \beta\}| \leq \alpha < m/2$. Therefore, it follows that there always exists a majority of forever-honest communities of $m - \alpha$ communities, such that each of these communities contains at most β corrupted parties from S, i.e., the majority of communities will never be corrupted and be forever honest. Let $\mathcal{H}$ hold the community indices of the forever-honest communities where $|\mathcal{H}| = m - \alpha \geq \alpha + 1$. (In the following, we abbreviate the "forever-honest" prefix as "honest" for brevity.)

In each of the honest communities in $\mathcal{H}$, the honest parties follow the protocol in Sect. 5.2 to generate the signature share $\sigma_{(j,i)}$ for each community $j \in \mathcal{H}$ and each honest party $P_i \in \mathsf{CMNT}_j$. Since each honest community contains at least $\beta+1$ uncorrupted parties, these parties output valid signature shares $\sigma_{(j,i)}$ such that $\mathsf{PVerIntra}(\mathsf{pk}_{(j,i)}, pt, \sigma_{(j,i)}) = 1$.

Therefore, for any community j, $j \in \mathcal{H}$, at least $\beta+1$ honest parties would eventually accumulate at least $\beta+1$ valid signature shares. This means for honest parties in an honest community, the condition on line 5 of the intra-community aggregation procedure in Figure 2 would be satisfied. Honest parties in the honest community j would compute the same valid syndicate signature σ_j such that $\mathsf{PVer}(\mathsf{pk}_j, pt, \sigma_j) = 1$.

Next we show that any individual party P_i would receive at least $\alpha+1$ valid syndicate signatures from the honest communities in $\mathcal{H}$ with high probability. For any community j, $j \in \mathcal{H}$, it contains at least $\beta+1$ honest parties who have obtained a valid syndicate signature σ_j. Each honest party i in community j would send σ_j to a random set $\mathbb{T}_i$, a χ-sized subset of n parties. This is the **S**ample and **S**end process in our $\mathsf{SignProto}$. Since each party in n parties is selected with probability χ/n, the probability that P_i never receives a valid syndicate signature σ_j from an honest community j is at most $(1-\chi/n)^{\beta+1}$. Therefore, P_i

receives at least one valid σ_j with probability at least $1-(1-\chi/n)^{\beta+1}$. Hence, P_i receives at least $(\alpha+1)$ valid syndicate signatures from the different honest communities in $\mathcal{H}$ with probability at least $(1-(1-\chi/n)^{\beta+1})^{\alpha+1}$. By Fact 4, we have $\chi=\Omega(n\,(\gamma+\log\alpha)\,/\beta)$. With $\alpha=\Theta(m),\beta=\Theta(q),mq=kn$, χ reduces to $\Omega(m\,(\gamma+\log m)\,/k)$. Taking a union bound over all n parties adds a $\log n$ term to the exponent, which is absorbed asymptotically. Thus, with $k=O(1)$, we lower bound this probability by $1-2^{-\gamma}$ when setting $\chi=\Omega(\sqrt{n}\,(\gamma+\log n))$. Therefore, with probability $1-2^{-\gamma}$, every honest party P_i eventually receives at least $(\alpha+1)$ valid syndicate signatures, combines them into a unique threshold signature σ, and outputs the same σ such that $\mathsf{siTS.Ver}(\mathsf{pk},pt,\sigma)=1$. □

Fact 4 *For any $n\gg 1$, $\alpha\beta=\Theta(n)$, it holds that $\left(1-(1-\chi/n)^{\beta+1}\right)^{\alpha+1}>1-2^{-\gamma}$ for some $\chi=\Omega(n\,(\gamma+\log\alpha)\,/\beta)$. In particular, we may take $\chi\geq n\,(\gamma\log 2+\log(\alpha+1)+\log 2)\,/(\beta+1)$. With $\alpha=\Theta(m),\beta=\Theta(q),mq=kn$, χ reduces to $\Omega(m\,(\gamma+\log m)\,/k)$.*

Proof (Fact 4). Let $p=(1-\chi/n)^{\beta+1}$ and $t=(\beta+1)\chi/n$. Since $\log(1-x)\leq -x$, we have $p\leq e^{-t}$. Also, $\log(1-x)\geq -x-x^2$ for $0\leq x<1$, so $(1-p)^{\alpha+1}\geq\exp\big(-(\alpha+1)(p+p^2)\big)\geq\exp\big(-(\alpha+1)(e^{-t}+e^{-2t})\big)$. As $e^{-u}>1-u$ for $u>0$, $\exp\big(-(\alpha+1)(e^{-t}+e^{-2t})\big)>1-(\alpha+1)(e^{-t}+e^{-2t})\geq 1-2(\alpha+1)e^{-t}$, where the last step uses $e^{-2t}\leq e^{-t}$. Thus, it suffices to have $2(\alpha+1)e^{-t}<2^{-\gamma}$, or $t>\log\big(2(\alpha+1)\big)+\gamma\log 2=\gamma\log 2+\log(\alpha+1)+\log 2$. The assumption on χ implies exactly this inequality on t, so the result follows. □

5.4 Unforgeability

We prove the unforgeability of our threshold signature scheme under the random oracle model (ROM), the one-more discrete logarithm (OMDL) assumption (Def. 1) and the algebraic group model (AGM) (Def. 2).

We brief the high-level proof ideas of the main Theorem 5. Under the assumption that a given community allocation strategy $\mathcal{X}$ is "robust", we directly prove the unforgeability game $\text{UF-CMA-Community}^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}}$ (Def. 10). In this game, the adversary $\mathcal{A}$ specifies corrupt queries in a different way, compared with a classical unforgeability game $\text{UF-CMA}^{\mathcal{B}}_{\Sigma,t,\mathcal{X}}$ where $\mathcal{B}$ can corrupt t individual parties. We argue the adversary $\mathcal{A}$ in game UF-CMA-Community is at least as strong as $\mathcal{B}$ in classic UF-CMA. Roughly speaking, $\mathcal{A}$ no longer corrupts individual P_i by an index $i\in[n]$ as in the classic game; instead, it specifies a (j,i) pair only to learn the partial secret key share $sk_{(j,i)}$. To make $\mathcal{A}$ as strong as the classic $\mathcal{B}$, $\mathcal{A}$ is allowed to query all mq pairs except the restriction enforced by the community allocation strategy that says there always exists an $(m-\alpha)$-subset $\mathcal{H}_{\mathsf{CMNT}}\subset[m]$ of communities, such that the adversary can corrupt at most β partial secret key shares in each community in $\mathcal{H}_{\mathsf{CMNT}}$. This restriction is necessary to ensure the robustness of the community structure can be maintained in the face of up to t corruptions. We show this reduction is valid in Appendix B Lemma 17.

In Theorem 5, we prove the unforgeability of the reduced game $\text{UF-CMA-Community}^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}}$ by following the key ideas in Bacho and Loss [8]. The main challenge is to construct a simulator that embeds $k-1$ hardness challenges either in secret key shares or inside the random oracle queries where $k=\beta m+\alpha+1$. Note that how the simulator embeds the challenge is hidden from the adversary. For convenience, we follow the trusted dealer assumption to prepare the secret key shares and do not adopt the oracle-aided algebraic simulatability assumption, which is utilized in multiple DKG protocols. We assume the secret key shares are dealt by a trusted party where [28] adopts the same approach. We take note that [8, Theorem 4.7] provides an instance of a $(t,t+1)$-oracle-aided algebraic secure DKG, formalized as $\mathsf{TD\text{-}DKG}$, where a traditional trusted dealer deals with the secret key shares as the trusted party.

Theorem 5 (Unforgeability of the Threshold Signature Scheme Under Chosen Message Attacks). *Let $\mathcal{X}$ be a community allocation strategy partitioning n parties into m communities, each of size q, with $m, q \in \Theta(\sqrt{n})$, $m \geq 2\alpha + 1$, and $q \geq 2\beta + 1$. Conditioned on at most α communities containing more than β corrupted members, if k-OMDL is (ϵ, T)-hard in the AGM, then siTS is $(\epsilon', T', q_h, q_s)$-unforgeable under chosen message attacks in communities (UF-CMA-Community) in the AGM + ROM, where $k = \beta m + \alpha + 1$, $\epsilon \geq \frac{\epsilon'}{4} - \frac{q_h^2}{4p}, T \leq T' + 2k^2 + 3q_h + q_s$.*

Proof (Theorem 5).

We prove via a sequence of games. Let $\mathcal{A}$ be an algebraic algorithm that $(\epsilon', T', q_h, q_s)$-breaks unforgeability of siTS under chosen message attacks in communities. W.l.o.g. we assume that $\mathcal{A}$ always queries the random oracle H before any signing query for the same message m. (Note that the challenger can always enforce this by making appropriate random oracle queries for itself.) Similarly, we may assume that queries to the random oracle are distinct and that $\mathcal{A}$ queries the random oracle H on m^* (the forgery message) before producing its forgery.

Game $\mathbf{G_0}$: This is the real game. The challenger runs the setup phase. Whenever $\mathcal{A}$ decides to corrupt a pair (j, i) for $j \in [m]$ and $i \in \mathsf{CMNT}_j$, the challenger faithfully returns the internal state of that party (which is merely the partial secret key share $sk_{(j,i)}$) and sets $\mathcal{C} = \mathcal{C} \cup \{(j, i)\}, \mathcal{H} = \mathcal{H} \setminus \{(j, i)\}$. In addition, $\mathcal{A}$ gets full control over the partial secret key share $sk_{(j,i)}$. For all $j \in [m], i \in \mathsf{CMNT}_j$, let $y_{(j,i)} \in \mathbb{G}$ denote the public key share assigned to the pair (j, i) and $x_{(j,i)}$ denote the secret key share of the pair (j, i), let y_j be the public key share of community j. Moreover, let $x \in \mathbb{Z}_p$ and $y = g^x$ denote the secret key and public key, respectively. Random oracle queries are answered by sampling $r_i \leftarrow \mathbb{Z}_p^*$ and returning $h_i = g^{r_i} \in \mathbb{G}$. Partial signing queries (j, i, m) are answered by returning $h_{(j,i)} = \mathsf{H}[m]^{x_{(j,i)}}$. At the end of the game, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*).

Game $\mathbf{G_1}$: This game is identical to $\mathbf{G_0}$, except that the game aborts and the adversary loses when there is a collision $H[m_1] = H[m_2]$ among distinct random oracle queries $m_1 \neq m_2$ from $\mathcal{A}$. By a standard argument, $\Pr[G_0^{\mathcal{A}} = 1] \leq \Pr[G_1^{\mathcal{A}} = 1] + q_h^2/p$.

Let $\mathbb{T} = \{(j, i) \mid i \in [n], j \in \mathsf{CMNT}^{-1}[i]\} = \{(j, i) \mid j \in [m], i \in \mathsf{CMNT}_j\}$, $V = \mathcal{C} \cup S$, where $S \subset \mathbb{T}$ is the subset of pairs for which $\mathcal{A}$ made a signing query of the form (j, i, m^*). As $\mathcal{A}$ is an algebraic adversary, at the end of G_1 it returns a forgery σ^* on a message m^* together with a representation

$$\begin{aligned}
&\mathbf{a} = (\hat{a}, a', b_1, \ldots, b_m,\\
&\ldots, c_{(j,i)}, \ldots \qquad \text{// for all } (j,i) \in \mathbb{T}\\
&d_1, \ldots, d_{q_h},\\
&\ldots, s_{(k,j,i)}, \ldots) \qquad \text{// for all } (j,i) \in \mathbb{T} \text{ and } k \in [q_s]
\end{aligned} \tag{2}$$

of elements in $\mathbb{Z}_p$. Since $\mathcal{A}$ wins $\mathbf{G_1}$, $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$, i.e., $e(g, \sigma^*) = e(y, \mathsf{H}[m^*]) = e(g, (\mathsf{H}[m^*])^x)$, which by non-degeneracy of the pairing implies $\sigma^* = (\mathsf{H}[m^*])^x$. Hence

$$\begin{aligned}
\sigma^* = (H[m^*])^x = g^{\hat{a}} \cdot y^{a'} \cdot \prod_{j=1}^{m} y_j^{b_j} \cdot \prod_{(j,i)\in\mathbb{T}} y_{(j,i)}^{c_{(j,i)}} \cdot\\
\prod_{k=1}^{q_h} h_k^{d_k} \cdot \prod_{k=1}^{q_s} \prod_{(j,i)\in\mathbb{T}} \sigma_{(k,j,i)}^{s_{(k,j,i)}}
\end{aligned} \tag{3}$$

Here, the representation is split (from left to right) into powers of the generator g, the public key $y = g^x$, the public key shares of the j-th community $y_j = g^{x_j}, j \in [m]$, the public

key shares of the pair (j,i) for all $(j,i) \in \mathbb{T}$, answers to hash queries h_k for all $k \in [q_h]$ returned by the random oracle, answers to partial signatures $s_{(k,j,i)}$ for all $(j,i) \in \mathbb{T}$ and $k \in [q_s]$ returned by the signing oracle.

Note that we have tacitly combined the other elements that are publicly communicated during the key generation phase into the term $g^{\hat{a}}$. We will clarify later why this is possible.

In the following, let m_k denote the k-th query to H and let $k^* \in [q_h]$ denote the index corresponding to the forgery message m^*. Recall that we write r^* and r_k for $k \in [q_h]$ to denote the values such that $\mathsf{H}[m^*] = g^{r^*}$ and $\mathsf{H}[m_k] = g^{r_k}$. Having said that, the above equation is equivalent to

$$\begin{aligned} r^*x &= \hat{a} + xa' + \sum_{j=1}^{m} x_j b_j + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} c_{(j,i)} + \sum_{k=1}^{q_h} r_k d_k + \\ &\quad \sum_{k=1}^{q_s} r_k \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s_{(k,j,i)} \\ &= \hat{a} + xa' + \sum_{j=1}^{m} x_j b_j + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} c_{(j,i)} + \sum_{k\in Q_h} r_k d_k + \\ &\quad \sum_{k\in Q_s} r_k \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s_{(k,j,i)} + \boxed{r^*d^* + r^* \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s^*_{(j,i)}} \end{aligned} \tag{♠}$$

where in the last equation we split the answers to the (hash and partial signing) queries for m^* from those of the other messages, with appropriate sets Q_h and Q_s (to be precise, $Q_h = [q_h] \setminus \{k^*\}$ and $Q_s = [q_s] \setminus \{k^*\}$) and the notation $s^*_{(j,i)} = s_{(k^*,j,i)}$ for all $(j,i) \in \mathbb{T}$. Since $\mathcal{A}$ wins $\mathbf{G_1}$, we remark that neither $\sum_{k\in Q_h} r_k d_k$ nor $\sum_{k\in Q_s} r_k \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s_{(k,j,i)}$ may include the terms r^*d^* or $r^* \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s^*_{(j,i)}$ respectively. [9]

We define event E as the event that $x \neq d^* + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s^*_{(j,i)}$. We have the following Lemma 5.

Lemma 5. *Let game* $\mathbf{G_1}$ *and event* E *be as defined above. Then there exist (algebraic) algorithms* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *playing in game k-OMDL that run in time at most* T *such that:*

$$\begin{aligned} \Pr[k\text{-}OMDL^{\mathcal{A}_1} = 1] &= \Pr[G_1^{\mathcal{A}} = 1 \wedge \neg E], \\ \Pr[k\text{-}OMDL^{\mathcal{A}_2} = 1] &\geq \left(1 - \frac{1}{p}\right) \cdot \Pr[G_1^{\mathcal{A}} = 1 \wedge E]. \end{aligned}$$

Moreover, $k = \beta m + \alpha + 1$, $T \leq T' + 2k^2 + 3q_h + q_s$.

Proof. Let $\xi \in \mathbb{G}^k$ be the OMDL instance. $\mathcal{A}_1$ and $\mathcal{A}_2$ both have access to a discrete logarithm oracle $DL_g(\cdot)$ which they can query at most $k-1$ times. Both simulate G_1, as we now describe.

Algorithm $\mathcal{A}_1(\xi, par)$: Algorithm $\mathcal{A}_1$ works as follows.

$\mathcal{A}_1$ directly executes key generation phase on input ξ. For convenience, we index the $k = \beta m + \alpha + 1$ challenges as $(\xi_0, \xi_1, \ldots, \xi_\alpha, \xi_{(1,1)}, \ldots, \xi_{(1,\beta)}, \ldots, \xi_{(m,1)}, \ldots, \xi_{(m,\beta)})$. Let $\xi_j = g^{z_j}, \forall j \in [0,\alpha]$ and $\xi_{(j,i)} = g^{z_{(j,i)}}, \forall (j,i) \in [m] \times [\beta]$. $\mathcal{A}_1$ lets $F(X,0) := A(X)$ be $\sum_{\ell=0}^{\alpha} a_\ell X^\ell \in \mathbb{Z}_p[X]$ such that $g^{a_j} = \xi_j$ for all $j \in [0,\alpha]$. This implicitly sets $a_j = z_j, \forall j \in [0,\alpha]$. Then for

[9] Here, we consider r^* and $r_k, k \in [q_h]$ as formal variables over $\mathbb{Z}_p$ rather than the concrete value that they may take. Note that it is indeed possible that for some k, $r_k d_k = r^*d^*$ or $r^*(\ldots) = r_k(\ldots)$ when considering concrete values in $\mathbb{Z}_p$ for r^*, r_i.

all $j \in [m]$, $\mathcal{A}_1$ computes $g^{F(j,0)} = g^{A(j)}$ as

$$g^{A(j)} = g^{\sum_{l=0}^{\alpha} a_l j^l} = \prod_{l=0}^{\alpha} (g^{a_l})^{j^l} = \prod_{l=0}^{\alpha} (\xi_l)^{j^l} .$$

Moreover, for all $j \in [m]$, $\mathcal{A}_1$ sets $g^{a_{(j,0)}} = g^{A(j)} = \prod_{l=0}^{\alpha} (\xi_l)^{j^l}$ and lets each of m univariate polynomials $F(j, Y) := B_j(Y)$ be $\sum_{\ell=0}^{\beta} a_{(j,\ell)} Y^{\ell} \in \mathbb{Z}_p[Y]$ such that $g^{a_{(j,i)}} = \xi_{(j,i)}$ for all $j \in [1, m], i \in [1, \beta]$. This implicitly sets $a_{(j,i)} = z_{(j,i)}$, for all $j \in [1, m], i \in [1, \beta]$. Then for all $j \in [m]$ and $i \in \mathsf{CMNT}_j$, $\mathcal{A}_1$ computes $g^{F(j,i)} = g^{B_j(i)}$ as

$$g^{B_j(i)} = g^{a_{(j,0)} + \sum_{l=1}^{\beta} a_{(j,l)} i^l} = g^{a_{(j,0)}} \prod_{l=1}^{\beta} (g^{a_{(j,l)}})^{i^l} = \prod_{l=0}^{\alpha} (\xi_l)^{j^l} \prod_{l=1}^{\beta} \left(\xi_{(j,l)}\right)^{i^l} .$$

Then $\mathcal{A}_1$ publishes the public key shares $(y_1, \ldots, y_m) = \left(g^{F(1,0)}, \ldots, g^{F(m,0)}\right)$ and $\left(y_{(j,i)}\right)_{(j,i)\in\mathbb{T}} = \left(g^{F(j,i)}\right)_{(j,i)\in\mathbb{T}}$ along with the public key $y = g^{F(0,0)} = \xi_0$.

Random oracle queries are answered by sampling $r_k \leftarrow \mathbb{Z}_p^*$ and returning $h_k = g^{r_k} \in \mathbb{G}$. $\mathcal{A}_1$ aborts when there is a collision $H[m_1] = H[m_2]$ among different random oracle queries $m_1 \neq m_2$ from $\mathcal{A}$. Signing queries (j, i, m_k) are answered by returning $h_k^{x_{(j,i)}}$ via the algebraic identity $H[m_k]^{x_{j,i}} = (g^{r_k})^{x_{(j,i)}} = (g^{x_{j,i}})^{r_k} = y_{(j,i)}^{r_k}$.

Corruption queries are answered by accessing the discrete log oracle accordingly. Whenever adversary $\mathcal{A}$ decides to corrupt a pair (j, i), $\mathcal{A}_1$ queries $DL_g\left(g^{F(j,i)}\right)$ and returns $sk_{(j,i)} = F(j, i) = B_j(i)$, only for the first $\beta + 1$ corruption queries in the same community j. Notice that $sk_{(j,\cdot)}$ in the same community j is fully determined once the univariate polynomial $F(j, \cdot) = B_j$ is known. Since B_j has $\beta + 1$ coefficients, $\beta + 1$ evaluations at distinct points of B_j suffice to solve all $\beta + 1$ coefficients of B_j, i.e., $(a_{(j,0)}, \ldots, a_{(j,\beta)})$. In other words, once $\mathcal{A}$ has made $\beta + 1$ corruption queries in the same community j, $\mathcal{A}_1$ learns the polynomial B_j and any further corruption queries from $\mathcal{A}$ in community j no longer require the DL_g oracle.

Since $\mathcal{A}$ can corrupt every pair except $q - \beta$ pairs in $m - \alpha$ communities, $\mathcal{A}_1$ needs to learn (at most) univariate polynomials B_j for α communities, and in addition, at most β pairs in each of the remaining $m - \alpha$ communities. Therefore, $\mathcal{A}_1$ needs to query DL_g at most $\alpha(\beta + 1) + \beta(m - \alpha) = \beta m + \alpha = k - 1$ times. Hence $\mathcal{A}_1$ can successfully simulate G_1 and answer $\mathcal{A}$'s queries including corruption and signing queries.

Suppose that $\mathcal{A}$ wins G_1 and that event $\neg E$ happens, i.e. $x = d^* + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s^*_{(j,i)}$. We note that the partial signing queries of the form (j, i, m^*) do not reveal any more information than if the challenger were simply handing over the corresponding private key share $x_{(j,i)}$. We thus treat these partial signing queries for m^* as corruption queries. We may also assume w.l.o.g. that $|\mathcal{C}| = mq - (m - \alpha)(q - \beta) = \alpha q + \beta m - \alpha\beta$. Otherwise, $\mathcal{A}_1$ simulates, for itself, $\alpha q + \beta m - \alpha\beta - |\mathcal{C}|$ corruption queries for random parties from the set of uncorrupted parties $\mathcal{H}$ after receiving the forgery, as if these were regular queries from $\mathcal{A}$. In other words, w.l.o.g., $\mathcal{A}_1$ must have known exactly α univariate polynomials B_j for all $j \in [\alpha]$ and β evaluations for each of B_j for $j \in [\alpha + 1, m]$. Since $\mathcal{A}_1$ knows all values $\{x_{(j,i)}\}_{(j,i)\in V=\mathcal{C}\cup S}$, it can compute the secret key x efficiently via the identity $x = d^* + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s^*_{(j,i)}$.

Finally, we look at the system of linear equations that $\mathcal{A}_1$ has gathered from the discrete logarithm oracle queries. W.l.o.g., $\mathcal{A}_1$ must have known exactly α univariate polynomials B_j for all $j \in [\alpha]$. This means $\mathcal{A}_1$ solves the system of linear equations for $z_{(j,1)}, \ldots, z_{(j,\beta)}$ for all $j \in [\alpha]$ and the values of $A(j) = B_j(0)$ for all $j \in [\alpha]$. Since $x = z_0$ can be solved from the forgery message, with $\alpha + 1$ distinct evaluations of polynomial A at distinct points, $\mathcal{A}_1$ can solve this degree α polynomial A and hence the values of $z_0, z_1, \ldots, z_\alpha$. W.l.o.g., $\mathcal{A}_1$ must have known exactly β evaluations for each of B_j for $j \in [\alpha + 1, m]$. This means $\mathcal{A}_1$ solves the system of linear equations for $z_{(j,1)}, \ldots, z_{(j,\beta)}$ for all $j \in [\alpha + 1, m]$ and the

values of $A(j) = B_j(0)$ for all $j \in [\alpha+1, m]$. At this point, $\mathcal{A}_1$ solves all k variables, i.e., $\{z_j\}, \forall j \in [0,\alpha]$ and $z_{(j,i)}, \forall (j,i) \in [m] \times [\beta]$ and hence the OMDL instance.

Overall, we obtain $\Pr[k\text{-OMDL}^{\mathcal{A}_1} = 1] = \Pr[G_1^{\mathcal{A}} = 1 \wedge \neg E]$.

The bound on the running time of $\mathcal{A}_1$ (number of group operations and exponentiations) comes from

- generating all public key shares and joint public key: $(\alpha+1)m + m(\beta+1)q < mq + mq^2$ exponentiations,
- one exponentiation for each random oracle query and one exponentiation for each signing query.

Algorithm $\mathcal{A}_2(\xi, par)$: Algorithm $\mathcal{A}_2$ works as follows. For convenience, we index the $k = \beta m + \alpha + 1$ challenges as $(\xi_1, \ldots, \xi_k)$. A trusted party runs the KGen honestly and $\mathcal{A}_2$ knows all the secret key shares $x_{(j,i)}, \forall (j,i) \in \mathbb{T}$. Whenever the adversary $\mathcal{A}$ decides to corrupt a pair (j,i), $\mathcal{A}_2$ faithfully reveals $x_{(j,i)}$, which is the only internal state of the pair. Random oracle queries are answered by sampling $\tilde{b}_\ell, \tilde{d}_\ell \leftarrow \mathbb{Z}_p^*$ and returning $h_\ell = g^{r_\ell} = \xi_1^{\tilde{b}_\ell} g^{\tilde{d}_\ell}$, which implicitly sets $r_\ell = z_1 \tilde{b}_\ell + \tilde{d}_\ell$. $\mathcal{A}_2$ aborts in case it detects a collision among answers in the list H. Partial signing queries (j, i, m_ℓ) are answered by returning $h_\ell^{x_{(j,i)}}$. Again, it is not hard to see that $\mathcal{A}_2$'s simulation of G_1 is perfect.

In case $\mathcal{A}_2$ does not abort, let $\tilde{A}_k := \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} s_{(k,j,i)}$ for all $k \in [q_s]$, where we write $\tilde{A}^*$ for $\tilde{A}_{k^*}$. Suppose that $\mathcal{A}$ wins G_1 and that event E happens, i.e. $x \neq d^* + \tilde{A}^*$. With our notation, equation (♠) is equivalent to

$$
\begin{aligned}
& z_1 \tilde{b}^* x + \tilde{d}^* x \\
&= \hat{a} + x a' + \sum_{j=1}^{m} x_j b_j + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} c_{(j,i)} + \sum_{k\in Q_h} r_k d_k + \\
&\quad \sum_{k\in Q_s} r_k \tilde{A}_k + r^* d^* + r^* \tilde{A}^* \\
&= \hat{a} + x a' + \sum_{j=1}^{m} x_j b_j + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} c_{(j,i)} + \sum_{k\in Q_h} (z_1 \tilde{b}_k + \tilde{d}_k) d_k + \\
&\quad \sum_{k\in Q_s} (z_1 \tilde{b}_k + \tilde{d}_k) \tilde{A}_k + (z_1 \tilde{b}^* + \tilde{d}^*) d^* + (z_1 \tilde{b}^* + \tilde{d}^*) \tilde{A}^* \\
&= \hat{a} + x a' + \sum_{j=1}^{m} x_j b_j + \sum_{(j,i)\in\mathbb{T}} x_{(j,i)} c_{(j,i)} + \\
&\quad z_1 \left(\sum_{k\in Q_h} \tilde{b}_k d_k + \sum_{k\in Q_s} \tilde{b}_k \tilde{A}_k + \tilde{b}^* d^* + \tilde{b}^* \tilde{A}^* \right) + \\
&\quad \sum_{k\in Q_h} \tilde{d}_k d_k + \sum_{k\in Q_s} \tilde{d}_k \tilde{A}_k + \tilde{d}^* d^* + \tilde{d}^* \tilde{A}^*.
\end{aligned}
$$

With the further notations

$$B := \tilde{b}^* x - \left(\sum_{k \in Q_h} \tilde{b}_k d_k + \sum_{k \in Q_s} \tilde{b}_k \tilde{A}_k + \tilde{b}^* d^* + \tilde{b}^* \tilde{A}^* \right),$$

$$D := \hat{a} + xa' + \sum_{j=1}^{m} x_j b_j + \sum_{(j,i) \in \mathbb{T}} x_{(j,i)} c_{(j,i)} + \sum_{k \in Q_h} \tilde{d}_k d_k + \sum_{k \in Q_s} \tilde{d}_k \tilde{A}_k + \tilde{d}^* d^* + \tilde{d}^* \tilde{A}^* - \tilde{d}^* x,$$

this reduces to $Bz_1 = D$. Recall that we have tacitly combined the other group elements that were publicly communicated during the key generation phase into the term $g^{\hat{a}}$. This is possible because $\mathcal{A}_2$ faithfully runs key generation on behalf of the honest parties and therefore has knowledge of the exponents of those elements relative to the base g and can combine them into the value $\hat{a}$. Let us now consider the case where $B = 0$. With the definition of event E we get

$$\begin{aligned} & 0 = \tilde{b}^* x - \left(\sum_{k \in Q_h} \tilde{b}_k d_k + \sum_{k \in Q_s} \tilde{b}_k \tilde{A}_k + \tilde{b}^* d^* + \tilde{b}^* \tilde{A}^* \right) \\ \iff & \sum_{k \in Q_h} \tilde{b}_k d_k + \sum_{k \in Q_s} \tilde{b}_k \tilde{A}_k = \tilde{b}^* \left(x - d^* - \tilde{A}^* \right) \\ \iff & \tilde{b}^* = \left(\sum_{k \in Q_h} \tilde{b}_k d_k + \sum_{k \in Q_s} \tilde{b}_k \tilde{A}_k \right) \left(x - d^* - \tilde{A}^* \right)^{-1}. \end{aligned} \tag{♣}$$

As already noted, $\sum_{k \in Q_h} r_k d_k$ and $\sum_{k \in Q_s} r_k \tilde{A}_k$ do not include the terms $r^* d^*$ or $r^* \tilde{A}^*$ respectively. Hence, $\sum_{k \in Q_h} \tilde{b}_k d_k$ and $\sum_{k \in Q_s} \tilde{b}_k \tilde{A}_k$ do not include the terms $\tilde{b}^* d^*$ or $\tilde{b}^* \tilde{A}^*$ respectively.[10] As defined by the identity $H[m^*] = g^{z_1 \tilde{b}^* + \tilde{d}^*}$, $\tilde{b}^*$ remains information-theoretically hidden from $\mathcal{A}$. This implies that the right-hand side of the (♣) equation is statistically independent of the uniform value $\tilde{b}^*$. Therefore, with probability $1 - 1/p$ we have $B \neq 0$, and thus $\mathcal{A}_2$ can compute z_1 efficiently as $z_1 := B^{-1} D$. Finally, $\mathcal{A}_2$ recovers the remaining values $z_2, \ldots, z_k$ by querying DL_g on $\xi_2, \ldots, \xi_k$ (its $k - 1$ allowed oracle queries), thereby solving the k-OMDL instance. Overall, we obtain

$$\Pr[k\text{-OMDL}^{\mathcal{A}_2} = 1] \geq \left(1 - \frac{1}{p}\right) \cdot \Pr[G_1^{\mathcal{A}} = 1 \wedge E].$$

The bound on the running time of $\mathcal{A}_2$ comes from

- generating all public key shares and joint public key: $1 + m + mq$ exponentiations,
- two exponentiations and one multiplication for each random oracle query and one exponentiation for each signing query.

□

Consider Algorithm $\mathcal{B}$ playing in k-OMDL as follows: $\mathcal{B}$ samples $i^* \leftarrow\!\!\$\ \{1, 2\}$ and then internally emulates $\mathcal{A}_{i^*}$. Clearly, $\mathcal{B}$ is an algebraic algorithm running in time at most T (the

[10] Here, we consider $\tilde{b}^*$ as a formal variable over $\mathbb{Z}_p$ rather than its concrete value.

running time of $\mathcal{A}_1$ and $\mathcal{A}_2$). An application of the law of total probability yields for $p \geq 2$,

$$\begin{aligned}
&\Pr[k\text{-OMDL}^{\mathcal{B}} = 1] \\
&= \sum_{i=1}^{2} \Pr[k\text{-OMDL}^{\mathcal{B}} = 1 \mid i^* = i] \cdot \Pr[i^* = i] \\
&= \frac{1}{2} \sum_{i=1}^{2} \Pr[k\text{-OMDL}^{\mathcal{A}_i} = 1] \\
&\geq \frac{1}{2} \left(1 - \frac{1}{p}\right) \left(\Pr[G_1^{\mathcal{A}} = 1 \wedge E] + \Pr[G_1^{\mathcal{A}} = 1 \wedge \neg E]\right) \\
&= \frac{1}{2} \left(1 - \frac{1}{p}\right) \Pr[G_1^{\mathcal{A}} = 1] \geq \frac{1}{4} \Pr[G_1^{\mathcal{A}} = 1] \\
&\geq \frac{1}{4} \Pr[G_0^{\mathcal{A}} = 1] - \frac{q_h^2}{4p}.
\end{aligned}$$

□

5.5 Uniqueness

We finally show that our scheme satisfies uniqueness, i.e., under the group public key pk, every message has at most one valid signature.

Theorem 6 (Uniqueness of the Threshold Signature Scheme). *Our $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature scheme in Section 5.2 satisfies uniqueness: for any message msg, if σ and σ' are valid signatures for msg under the group public key pk, then $\sigma = \sigma'$.*

Proof. Let $\mathsf{msg} \in \{0,1\}^*$ be any message, and let σ and σ' be two valid signatures for msg under the group public key pk. By our construction, it follows that $e(g, \sigma) = e(pk, \mathsf{H}(\mathsf{msg})) = e(g, \sigma')$. Assume $\sigma = g^a$ and $\sigma' = g^{a'}$ for some $a, a' \in \mathbb{Z}_p$. The pairing equation above implies that $e(g,g)^a = e(g,g)^{a'}$. Since the groups have prime order p and the bilinear pairing is non-degenerate, $e(g,g)$ is a generator of the target group. Therefore, $a \equiv a' \pmod{p}$, which implies that $\sigma = \sigma'$. □

5.6 Complexity Analysis

Theorem 7 (Complexity of the Threshold Signature Scheme). *Let λ and γ be the computational and statistical security parameter, respectively, and let $\mathcal{X}$ be a community allocation strategy partitioning n parties into m communities of size q with $m, q \in \Theta(\sqrt{n})$ (so each party belongs to $k = O(1)$ communities). Our $(n, m, \alpha, q, \beta, \mathcal{X})$-threshold signature scheme in Section 5.2 incurs two rounds and has the per-node communication complexity $O(\lambda\sqrt{n}(\gamma + \log n))$ and computational complexity $O(t)$.*

Proof. It is easy to see that our threshold signature scheme in Section 5.2 incurs two rounds. The ideal community allocation strategy $\mathcal{X}$, which follows from Corollary 2, allocates each party to $k = O(1)$ communities, where each community has size $q = O(\sqrt{n})$ and there are $m = O(\sqrt{n})$ communities in total. With $m, q \in \Theta(\sqrt{n})$, we have $k = mq/n = O(1)$.

We first look at computational complexity. Each party P_i belongs to exactly $\left|\mathsf{CMNT}^{-1}[i]\right| = k$ communities. It takes $O(k)$ time to generate all intra-community partial signatures $\sigma_{j,i}$ for $j \in \mathsf{CMNT}^{-1}[i]$. To verify intra-community partial signatures generated by members with whom it shares a community, there are $\sum_{j \in \mathsf{CMNT}^{-1}[i]} |\mathsf{CMNT}_j| = kq$ partial signatures, thus it takes $O(kq)$ time. For aggregation in each of the k communities, P_i takes $O(q \log q)$ time

to aggregate shares, since the aggregation complexity per party per community is equivalent to that of a classic $(\lfloor q/2 \rfloor + 1)$-out-of-$q$ BLS threshold signature scheme, i.e., $O(q \log q)$.

On the inter-community level, we first consider how many syndicate signatures σ_j a party P_i can receive. In expectation, each honest party receives $E\left[O\left(\frac{(n-t)\chi}{n}\right) \cdot k\right] \overset{\text{Fact 4}}{=\!=} E\left[O\left(\frac{(n-t)\cdot m(\gamma+\log m)}{n}\right)\right]$ syndicate signatures from all honest parties. But for each specific community, it does not need to verify all received syndicate signatures because it immediately accepts the first correct one. Similarly, note that any dishonest party $P_{i'}$ can flood all honest parties by sending at most k incorrect syndicate signatures, one for each community $P_{i'}$ resides in. On the other hand, since honest parties never send incorrect syndicate signatures, honest receiving parties can adopt an early-abortion policy to avoid handling all k incorrect syndicate signatures but only the first one. This means t adversaries can force each honest receiver to handle at most t incorrect syndicate signatures, costing $O(t)$ verification time. Summing it up, each honest party needs to verify $m + t = O(t)$ syndicate signatures before aggregating the valid ones in $O(m \log m)$ time. Then, in the worst case, each party incurs $O(kq) + O(q \log q) + O(t) + O(m \log m) = O(t)$ computational time.

We proceed to analyze communication complexity. On the intra-community level, each party P_i is allocated to k communities and in each community P_i sends q signatures. This incurs $O(\lambda kq)$-bit messages per party. On the inter-community level, each party P_i randomly selects a χ-subset to send syndicate signatures to for the k communities it belongs to. This incurs $O(\lambda k\chi)$-bit messages per party. Therefore, the per-node communication complexity is $O(\lambda kq) + O(\lambda k\chi) \overset{\text{Fact 4}}{=\!=} O(\lambda(kq + m(\gamma + \log m))) \overset{q,m=O(\sqrt{n}),k=O(1)}{=\!=\!=\!=} O(\lambda\sqrt{n}(\gamma + \log n))$.

Note that whenever the constraints $kn = mq$ and $k = O(1)$ hold, the per-node communication complexity is lower bounded by the geometric mean $O\left(\lambda\sqrt{kqm(\gamma + \log m)}\right) \overset{k=O(1),mq=O(n)}{=\!=\!=\!=} O\left(\lambda\sqrt{n(\gamma + \log n)}\right)$. □

5.7 Recursion Ideas and Limitations of Our Explicit Sampler Constructions

We find limitations if we want to directly apply the recursion technique in our construction. Our $(1/3 - \epsilon)$-robust community allocation strategies cannot support a k-level recursion due to their current formulation.

Our current solution can be seen as a recursion up to $k = 2$ levels. In a k-level recursion, level i contains $n_i = n^{1-i/k}$ communities, each of size $q_i = n^{i/k}$. Level 0 captures just n network participants, containing $n_0 = n$ communities and each individual node is a community of size $1 = n^{0/k}$. A sampler $Sampler_i$ is needed between any consecutive levels and it allocates n_i virtual nodes (communities at level i) into n_{i+1} communities at level $i + 1$. On level i, community members invoke threshold signature aggregation to form syndicate signatures of its community on level i, which are threshold signature shares of that community on level $i + 1$.

Limitations apply to our explicit constructions. Suppose we recurse to $k = 3$ levels. We have $n^{2/3}$ communities on level 1, each of size $n^{1/3}$ and n nodes (communities) on level 0. With our bipartite Ramanujan construction, even if we ignore prime and integral constraints, we have $|L| = n = p^2$ and $|R| = n^{2/3} = pl$ where $2 \leq l \leq p$. This implies $l = n^{1/6}$. The sampler is (l, p)-biregular, meaning each node in L has $l = n^{1/6}$ neighbors in R and each node in R has $\sqrt{n}$ neighbors in L. However, since we are always bounded by the $kn = mq$ bipartite graph constraint, we must have $m = n^{2/3}$, $q = p = n^{1/2}$ and $k = l = n^{1/6}$. Since each party needs to pay for $O(\lambda kq)$, the communication complexity worsens compared to our current recursion level $k = 2$.

The barrier may be removed via instantiating suitable samplers with probabilistic methods, such as establishing a good biregular bipartite random graph via the configuration model

[17] with rejection. However, this deviates from our sampler constructions and we treat this as a limitation.

5.8 From Balanced Sub-quadratic Threshold Signature to Asynchronous Common Coin

Cachin et al. [23] mentioned how to instantiate an efficient asynchronous common coin protocol by leveraging a non-interactive unique threshold signature such as BLS threshold signature [29]. Specifically, to obtain a common random value, network participants need to jointly sign a common session number ID, and take the cryptographic hash value of the final unique threshold signature as the common random value. Since the threshold signature is unique, i.e., there is only one valid threshold signature for a particular message, all participants can see at most one value corresponding to ID. In addition, as the signature is unforgeable (Theorem 5), no one can obtain the signature in advance, so the random value is unpredictable and unbiased in the random oracle model.

Careful readers may have noticed that the "non-interactiveness" of the threshold signature scheme is not necessary, while uniqueness and availability are sufficient for this reduction. Indeed, in our semi-interactive threshold signature, uniqueness follows from the bivariate secret sharing scheme (Theorem 6) and the availability follows from robustness (Theorem 3). Hence, our threshold signature scheme and a cryptographic hash trivially realize a common coin in ROM.

The last puzzle to instantiate this coin is to explicitly construct a "robust" community allocation strategy $\mathcal{X}$ assumed in both Theorems 3 and 5. By Corollary 4, we can explicitly construct the desirable strategy $\mathcal{X}$. This closes the gap and gives Theorem 8.

Theorem 8 (Balanced and Sub-quadratic Asynchronous Common Coin). *There is an asynchronous (n, t)-common coin protocol with per-node communication complexity $O(\lambda\sqrt{n}(\gamma + \log n))$ where $n \geq N(\epsilon) = O(\epsilon^{-80/19})$, the threshold of Corollary 4, and $t \leq (1/2 - \epsilon)n$.*[11]

6 Balanced Sub-quadratic Asynchronous Byzantine Agreement

In this section, we first introduce a modification to conventional Asynchronous Multi-valued Byzantine Agreement (MBA) protocols. This modification ensures that the expected communication complexity remains quadratic even when the super-honest majority assumption fails. Building on this modified MBA, we then present our balanced, sub-quadratic ABA protocol.

6.1 Building Block: MBA with Quadratic Communication in Dishonest Majority Cases

As discussed in Section 2, a conventional MBA protocol can be modified to strictly bound its communication cost to quadratic, even in the absence of a super-honest majority. We formalize this result with the following lemma.

Lemma 6. *Let $\gamma \in \mathbb{N}$ be a statistical security parameter. There exists an MBA protocol such that: (1) when more than $2/3$ of the participants are honest, the instance satisfies the standard properties of termination, agreement, and validity with probability at least $1 - 2^{-\gamma}$, while incurring an expected communication cost of $O(n^2)$ and taking $O(1)$ expected rounds; and (2) even if more than $1/3$ of the nodes are corrupted, each honest node sends at most $O(\gamma n)$ bits and terminates in $O(\gamma)$ rounds.*

[11] The proceedings version [34] states $n = \Omega(\epsilon^{-4})$; the ϵ^{-4} scale holds under a Legendre-type prime-gap hypothesis (see Corollary 4).

Proof. Most randomized consensus protocols operate in a "Las Vegas" style. For example, an execution of the MBA protocol in [45] consists of a sequence of iterations, terminating after the first successful iteration. Under a super-honest majority, each iteration requires a constant number of rounds and succeeds with an independent probability of $1/2$. Consequently, the probability that an MBA instance terminates within T' iterations is $1 - 2^{-T'}$.

Leveraging this property, we modify the MBA protocol by instructing each honest node to simply quit the instance after γ iterations. This cap bounds the per-node communication cost to $O(\gamma n)$. With this modification, instances equipped with a super-honest majority terminate with probability at least $1 - 2^{-\gamma}$, while agreement and validity are preserved as normal. □

6.2 Construction

We will present our balanced and sub-quadratic Asynchronous Binary Byzantine Agreement (ABA) protocol in Fig. 3 and we proceed by preparing the readers with some notations and shorthands which will appear.

Notation and ingredients used in Fig. 3. For readability, Fig. 3 uses a handful of shorthand terms (*valid*, *proper*, *properly justified*, "*the corresponding justification*"); we fix their precise meaning here. Throughout, each per-community vote message that a party samples and broadcasts has the canonical shape

$$\big(m,\ J,\ \sigma_k\big), \qquad m \in \big\{\,(ID, \text{pre-process}, W_k),\ (ID, \text{pre-vote}, r, b),\ (ID, \text{main-vote}, r, v)\,\big\},$$

where σ_k is the community-k *syndicate signature* on the payload m, and J is the *justification* certifying that b (resp. v) is a legitimate choice in this position of the protocol. Pre-processing messages carry no J field.

- **Valid message.** A vote message is *valid* if its embedded syndicate signature σ_k verifies under community k's public key against the payload m.
- **Proper pre-processing message.** A pre-processing message $(ID, \text{pre-process}, W_k, \sigma_k)$ is *proper* if it is valid in the above sense. In step 0 of Fig. 3, P_i counts only proper pre-processing messages, and only from $2\alpha + 1$ *distinct* communities, before computing its first pre-vote.
- **Properly justified pre-vote / main-vote.** A round-r pre-vote (resp. main-vote) message is *properly justified* if (i) it is valid, and (ii) the attached justification J verifies against the rules laid out in the Pre-Vote Justification (resp. Main-Vote Justification) paragraphs below.

The two auxiliary procedures invoked in Fig. 3 have the following interfaces; their behaviour is exactly the construction described in the Pre-Vote / Main-Vote Justification paragraphs below, to which we refer for the full rules.

GenPreVoteJustification$(r,\ b,\ \mathsf{votes}) \to J$: produces a justification J for the round-r pre-vote b from a set votes of valid vote messages from the previous step.

GenMainVoteJustification$(r,\ v,\ \mathsf{votes}) \to J$: produces a justification J for the round-r main-vote v from a set votes of $m - \alpha$ valid, properly justified round-r pre-vote messages from distinct communities.

With this vocabulary, the phrases "*the first proper pre-processing message*", "*properly justified main-votes*", "*the first valid and properly justified round-r pre-vote messages*" and "*the first valid and properly justified round-r main-vote messages*" that appear in Fig. 3 are to be read in the senses fixed above.

Fig. 3. Balanced Sub-quadratic Asynchronous Byzantine Agreement.

Protocol ABA for party P_i with initial value V_i and community allocation strategy $\mathcal{X}$

0. **PRE-PROCESSING.** Parse $\mathcal{X} \to \{\mathsf{CMNT}_j\}_{j\in[m]}$. For each community CMNT_k that P_i belongs to, i.e., $\forall k \in \mathsf{CMNT}^{-1}[i]$,
   (a) $\forall i \in \mathsf{CMNT}_k$, P_i participates in $\mathsf{MBA}\langle \mathsf{CMNT}_k, P_i(V_i)\rangle$. The MBA instance decides W_k for CMNT_k.
   (b) CMNT_k produces a syndicate signature σ_k on the message $(ID, \text{pre-process}, W_k)$
   (c) Send the message $(ID, \text{pre-process}, W_k, \sigma_k)$ to $\chi = \Omega\left(\sqrt{n}(\gamma + \log n)\right)$ randomly sampled parties $\subset [n]$.

   P_i collects the first proper pre-processing messages from $2\alpha + 1$ different communities.

Repeat the following steps 1–4 for rounds $r = 1, 2, \ldots$.

1. **PRE-VOTE.** If $r = 1$, let $b_{r,i}$ be the simple majority of the received pre-processing votes. Otherwise, if $r > 1$, select $m - \alpha$ properly justified main-votes from round $r - 1$. Let $b_{r,i} = 0$ if there is a main-vote for 0; let $b_{r,i} = 1$ if there is a main-vote for 1; let $b_{r,i} = \mathsf{CommonCoin}(ID\|r-1)$ if all main-votes are abstain. For each community CMNT_k that P_i belongs to, i.e., $\forall k \in \mathsf{CMNT}^{-1}[i]$,
   (a) $\forall i \in \mathsf{CMNT}_k$, P_i participates in $\mathsf{MBA}\langle \mathsf{CMNT}_k, P_i(b_{r,i})\rangle$. The MBA instance decides b_k for CMNT_k.
   (b) CMNT_k produces a syndicate signature σ_k on the message $(ID, \text{pre-vote}, r, b_k)$.
   (c) If possible, compute $J \leftarrow \mathsf{GenPreVoteJustification}(r, b_k, \mathsf{votes})$, where votes is the set of $2\alpha + 1$ proper pre-processing messages from step 0 (if $r = 1$) or the $m - \alpha$ properly justified round-$(r-1)$ main-vote messages (if $r > 1$), and send to all parties in CMNT_k the message $(ID, \text{pre-vote}, r, b_k, J)$.
   (d) Send the message $(ID, \text{pre-vote}, r, b_k, J, \sigma_k)$ to χ randomly sampled parties.
2. **MAIN-VOTE.** P_i collects the first valid and properly justified round-r pre-vote messages from $m - \alpha$ different communities. Let $v_{r,i} = 0$ if there are $m - \alpha$ pre-votes for 0; let $v_{r,i} = 1$ if there are $m - \alpha$ pre-votes for 1; let $v_{r,i} =$ abstain if there are pre-votes for both 0 and 1. For each community CMNT_k that P_i belongs to, i.e., $\forall k \in \mathsf{CMNT}^{-1}[i]$,
   (a) $\forall i \in \mathsf{CMNT}_k$, P_i participates in $\mathsf{MBA}\langle \mathsf{CMNT}_k, P_i(v_{r,i})\rangle$. The MBA instance decides v_k for CMNT_k.
   (b) CMNT_k produces a syndicate signature σ_k on the message $(ID, \text{main-vote}, r, v_k)$.
   (c) If possible, compute $J \leftarrow \mathsf{GenMainVoteJustification}(r, v_k, \mathsf{votes})$, where votes is the set of $m - \alpha$ valid and properly justified round-r pre-vote messages collected at the start of this step, and send to all parties in CMNT_k the message $(ID, \text{main-vote}, r, v_k, J)$.
   (d) Send the message $(ID, \text{main-vote}, r, v_k, J, \sigma_k)$ to χ randomly sampled parties.
3. **CHECK FOR DECISION.** P_i collects the first valid and properly justified round-r main-vote messages from $m - \alpha$ different communities. If these are all main-votes for $b \in \{0, 1\}$, then *decide* the value b for ID, and continue for one more round (up to step 2). Otherwise, proceed.
4. **COMMON COIN.** All parties participate in the coin tossing protocol with name $ID\|r$.

Overview. For a given transaction identifier ID, each party P_i has an initial value $V_i \in \{0, 1\}$, and the protocol proceeds in rounds $r = 1, 2, \ldots$. The first round starts with a special pre-processing step:

0. For each community k, all parties in CMNT_k first run the classic MBA protocol with input V_i and decide a pre-processing value W_k for CMNT_k. Then P_i produces an $\mathcal{S}_0$-signature share $\sigma_{k,i} = \mathsf{H}((ID, \text{pre-process}, W_k))^{\mathcal{S}_0.sk_{(k,i)}}$, and sends it to all $q = 3\beta + 1$ parties in CMNT_k. Within community CMNT_k, on receiving $q - \beta$ such signature shares of W_k, each party combines them into a syndicate signature σ_k, samples $O\big(\sqrt{n}(\gamma + \log n)\big)$ parties randomly to send the pre-processing message with the syndicate signature σ_k. On receiving pre-processing messages with valid syndicate signatures from $2\alpha + 1$ different communities, each party combines the signature shares of the value with the simple majority (i.e., at least $\alpha + 1$ votes) into a threshold signature of $\mathcal{S}_0$. This value will be the value used in the first pre-vote.

   After that each round contains three steps:

1. Each party casts a *pre-vote* for a value $b \in \{0,1\}$. These pre-votes must be *justified* by an appropriate $\mathcal{S}$-threshold signature (or $\mathcal{S}_0$-threshold signature for the first round), and must be accompanied by a valid $\mathcal{S}$- (or $\mathcal{S}_0$-) signature share on an appropriate message.
2. After collecting valid pre-votes from $m - \alpha$ different communities in round r, each party P_i casts a *main-vote* $v_{r,i} \in \{0, 1, \text{abstain}\}$. As with the pre-processing value, for each community k that P_i belongs to, P_i runs the MBA protocol with input $v_{r,i}$ and decides v_k for CMNT_k. Then P_i produces an $\mathcal{S}$-signature share $\mathsf{H}((ID, \text{main-vote}, r, v_k))^{\mathcal{S}.sk_{(k,i)}}$, and sends it to all $q = 3\beta + 1$ parties in CMNT_k. Within community CMNT_k, on receiving $q - \beta$ such signature shares of v_k, each party combines them into a syndicate signature, samples $O\big(\sqrt{n}(\gamma + \log n)\big)$ parties randomly to send the main-vote message with the syndicate signature. As with pre-votes, these main-votes must be justified by an appropriate $\mathcal{S}$-threshold signature, and must be accompanied by a valid $\mathcal{S}$-signature share on an appropriate message.
3. After collecting valid main-votes from $m - \alpha$ different communities in round r, each party examines these votes. If all votes are for a value $b \in \{0,1\}$, then the party *decides* b for ID, but continues to participate in the protocol for one more round. Otherwise, the party proceeds.
4. The value of coin $ID\|r$, i.e., ID concatenated with a binary representation of r, is revealed, which may be used in the next round.

We now proceed with the informal details of the protocol given in Fig. 3. The pre-vote and main-vote messages have to contain a proper *justification*, which consists of threshold signatures on collected votes as follows.

Pre-Vote Justification. In round $r = 1$, party P_i's pre-vote is the majority of the pre-processing votes from step 0. There must be at least $\alpha + 1$ votes, voted by at least $\alpha + 1$ different communities, for the same value $b \in \{0,1\}$ (although this b might not be unique). For the justification, a party selects $\alpha + 1$ such votes, and combines the accompanying $\mathcal{S}_0$-syndicate signatures to obtain an $\mathcal{S}_0$-threshold signature on the message $(ID, \text{pre-process}, b)$. In rounds $r > 1$, a pre-vote for b may be justified in two ways:

- either with an $\mathcal{S}$-threshold signature on the message $(ID, \text{pre-vote}, r-1, b)$. We call this a *hard* pre-vote for b;
- or with an $\mathcal{S}$-threshold signature on the message $(ID, \text{main-vote}, r-1, \text{abstain})$ for the pre-vote $b = \mathsf{CommonCoin}(ID\|r-1)$. We call this a *soft* pre-vote for b.

Intuitively, a hard pre-vote expresses P_i's preference for b based on evidence for preference b in round $r-1$, whereas a soft pre-vote is just a vote for the value of the coin, based on evidence of conflicting votes in round $r-1$. The threshold signatures are obtained from the computations in previous rounds (see below). We assume that the justification indicates whether the pre-vote is hard or soft.

$\mathsf{GenPreVoteJustification}(r, b, \mathsf{votes}) \to J$: produces a justification J for the round-r pre-vote b from a set votes of valid vote messages from the previous step.

- If $r = 1$: votes is a set of $2\alpha + 1$ valid pre-processing messages from distinct communities. The procedure selects $\alpha + 1$ of them whose payloads agree on the simple-majority value b and combines their $\mathcal{S}_0$-syndicate signatures into an $\mathcal{S}_0$-threshold signature on $(ID, \text{pre-process}, b)$, which it outputs as J.
- If $r > 1$: votes is a set of $m - \alpha$ valid round-$(r-1)$ main-vote messages from distinct communities. Either (i) at least one main-vote in votes is for b, in which case J is the $\mathcal{S}$-threshold signature on $(ID, \text{pre-vote}, r-1, b)$ aggregated from the $m - \alpha$ pre-vote syndicate signatures of the previous round (*hard* pre-vote); or (ii) all main-votes in votes

are abstain, in which case J is the $\mathcal{S}$-threshold signature on $(ID, \text{main-vote}, r-1, \text{abstain})$ aggregated from these abstain syndicate signatures and $b = \mathsf{CommonCoin}(ID \| r-1)$ (*soft* pre-vote).

Main-Vote Justification. A main-vote v in round r is one of the values $\{0, 1, \text{abstain}\}$ and, like pre-votes, accompanied by a *justification* as follows:

- If among the $m - \alpha$ justified round-r pre-votes collected by P_i there is a pre-vote for 0 and a pre-vote for 1, then P_i's main-vote for round r is abstain. The justification for this main-vote consists of the two conflicting pre-votes, including their justifications and syndicate signatures.
- Otherwise, P_i has collected $m - \alpha$ justified pre-votes for some $b \in \{0, 1\}$ in round r, and since each of these comes with a valid $\mathcal{S}$- or $\mathcal{S}_0$-syndicate signature on the message $(ID, \text{pre-vote}, r, b)$, party P_i can combine these syndicate signatures to obtain a valid $\mathcal{S}$- or $\mathcal{S}_0$-threshold signature on this message. Party P_i's main-vote in this case is b, and its justification is this threshold signature.

Each pre-vote and main-vote message also includes an $\mathcal{S}$- or $\mathcal{S}_0$-syndicate signature on its content, issued by members of the community jointly. We say that a vote message is *valid* if the embedded syndicate signature is valid. By the non-interactiveness of aggregation of syndicate signatures, this allows the receiver to obtain the threshold signature that justifies its next vote message, which is aggregated from $m - \alpha$ valid vote messages from different communities.

$\mathsf{GenMainVoteJustification}(r, v, \mathsf{votes}) \to J$: produces a justification J for the round-r main-vote v from a set votes of $m - \alpha$ valid, properly justified round-r pre-vote messages from distinct communities.

- If $v \in \{0, 1\}$: all $m - \alpha$ pre-votes in votes are for v, and J is the $\mathcal{S}$-/$\mathcal{S}_0$-threshold signature on $(ID, \text{pre-vote}, r, v)$ obtained by combining their syndicate signatures.
- If $v =$ abstain: votes contains two conflicting properly-justified pre-votes, one for 0 and one for 1, with justifications J_0 and J_1 respectively; J is the pair (J_0, J_1).

6.3 Analysis

Assuming the corruption count $t = (1/3 - \epsilon)n$ for some $\epsilon > 0$, we can allocate n parties into m communities of q-size each and each party P_i belongs to k communities where $m, q = O(\sqrt{n})$ and $k = O(1)$. Let $m = 3\alpha + 1$ and $q = 3\beta + 1$. By Corollary 3, we are guaranteed that there is an honest super-majority of at least $2\alpha + 1$ honest communities where an honest community has an honest super-majority of $2\beta + 1$ forever-honest members.

With this community structure, we simulate virtual nodes by communities so there are at most α dishonest virtual nodes. Our goal is then to efficiently simulate virtual nodes to run a modified ABA protocol of [23] (referred to as the classic ABA later) such that each virtual node (a community) acts like one single node compared with how network participants run the ABA protocol in the classic ABA protocol. Our threshold signature scheme allows a virtual node (its members) to generate syndicate signatures on messages the members agree on. From the point of view of other virtual nodes, a syndicate signature simulates a signature share of a non-interactive unique threshold signature (NIUTS). When needed, syndicate signatures are combined into a threshold signature to justify votes. To simulate two NIUTS in the classic ABA, we use an $(n, m = 3\alpha + 1, 2\alpha, q = 3\beta + 1, \beta, \mathcal{X})$ threshold signature scheme $\mathcal{S}$ and an $(n, m = 3\alpha + 1, \alpha, q = 3\beta + 1, \beta, \mathcal{X})$ threshold signature scheme $\mathcal{S}_0$ (See Section 5), as well as a threshold common coin scheme (see Section 5.8), assuming $t = (1/3 - \epsilon)n$. Let $CommonCoin(str)$ denote the value of the coin with name str.

Security Analysis

Theorem 9 (Security of the ABA Protocol). *Assuming threshold signature schemes $\mathcal{S}$ and $\mathcal{S}_0$ are secure, a threshold coin-tossing scheme is secure, and an authenticated channel, the* ABA *protocol in Algorithm 3 solves asynchronous Byzantine agreement (Definition 4) for $t < (1/3 - \epsilon)n$ with probability at least $1 - \mathsf{negl}(\lambda)$.*

We sketch the proof ideas first. When virtual nodes can be treated as plain individual network participants, the overall proof idea is identical to the classic ABA protocol in [23], and the key difference is that single parties in the classic ABA are now simulated by communities (virtual nodes) in our protocol. The classic protocol requires a super-majority of honest parties, e.g. $n \geq 3t+1$, and we ensure the same for our virtual nodes. The threshold signature and coin-tossing scheme in the classic ABA are replaced by our semi-interactive threshold signature scheme and sub-quadratic asynchronous threshold coin-tossing scheme, respectively. Consequently, the individual partial signature shares and threshold signatures as to the classic ABA protocol are syndicate signatures and threshold signatures as to our virtual nodes (communities).

To make each honest virtual node act and look like one single node, virtual node members need to run the classic MBA to agree on the same view. This further requires a super-majority of honest parties within each honest virtual node, which is ensured by the community allocation strategy.

The next important thing to realize is that each forever-honest virtual node can always cast a non-equivocating vote: By the agreement of the classic MBA, its honest members must agree on one unique value to vote for and $\mathcal{A}$ cannot make a forever-honest virtual node deviate.

Lastly, it is important to see that honest virtual nodes must always able to receive votes cast by other honest virtual nodes against weakly adaptive adversaries (except for negligible probability), even when each honest party only sends a vote to $O\big(\sqrt{n}(\gamma + \log n)\big)$ other actual nodes. This follows Fact 4 and each vote message is endorsed by a syndicate signature, which shows consensus on the vote over the community members.

We proceed to a proof of Theorem 9. We have to show *validity*, *agreement*, and *termination* where termination consists of two conditions: *liveness* and *efficiency*. *Liveness.* If all honest parties have been activated on ID and all messages relating to this ID generated by honest parties have been delivered, then all honest parties have decided for ID, except with negligible probability. *Efficiency.* For any tag ID introduced by adversary A, let $X_{ID,A}$ be the total number of messages generated by all honest parties that relate to ID; then there exists a fixed polynomial $T(\kappa)$ and a fixed negligible function $\delta(l)$ (in l) such that for all adversaries A, there exists a negligible function $\epsilon_A(\kappa)$ such that for all $l \geq 0$, $\kappa \geq 0$, and ID,

$$\Pr[X_{ID,A}(\kappa) > lT(\kappa)] \leq \delta(l) + \epsilon_A(\kappa).$$

Lemma 7. *Assuming threshold signature schemes $\mathcal{S}$ and $\mathcal{S}_0$ are secure, a threshold coin-tossing scheme is secure, and an authenticated channel, the* ABA *protocol in Algorithm 3 satisfies validity with probability at least $1 - \mathsf{negl}(\lambda)$.*

Proof. We first find that the $(1/3 - \epsilon, 1/3, 1/3)$-robust community allocation strategy ensures $2/3$ of the virtual nodes are forever-honest with $2/3$ of their members being honest. Hence, when all honest parties share the same input b, members of all honest virtual nodes can execute a classic MBA (step (0a)) with input b. Since there are $O(m) = O(\sqrt{n})$ virtual nodes, $O(\sqrt{n})$ instances of classic MBA are run in parallel. Since the probability that an instance advances more than $2r+1$ rounds is bounded by $2^{-r} + \epsilon$, where ϵ is negligible, the probability that the last of $O(\sqrt{n})$ concurrent instances has not terminated within $2r$ rounds is roughly

bounded by $1 - \exp(-2^{-r+(\log n+1)/2})$, and the expected number of actual asynchronous rounds for the last instance to terminate is $O(\log n)$. The termination of concurrent classic MBA instances ensures honest parties proceed to step (0b) with $W_k = b$ for all honest virtual nodes k. This further ensures the majority vote among any set of $2\alpha + 1$ valid pre-processing votes is the unique value b proposed by the honest parties. At step (1a), every honest party P_i casts $b_{1,i} = b$. Consequently, the MBA at step (1a) terminates and outputs b and any properly justified pre-vote is also for b, as is any properly justified main-vote, and the protocol decides b in the first round. □

We now prove the remaining properties assuming the adversary corrupts exactly $f = t$ parties and $f_\alpha = \alpha$ virtual nodes; we then discuss the modifications necessary for the case that $f < t$ and $f_\alpha < \alpha$.

Fix a given ID and consider the pre-votes cast by honest virtual nodes in round $r \geq 1$. Because $m > 3\alpha$, there will be at most one value $b \in \{0, 1\}$ that garners at least $m - 2\alpha$ such pre-votes, and we define ρ_r to be this value (if it exists), and otherwise we say that ρ_r is undefined. We say that ρ_r is defined at the point in the game at which time $m - 2\alpha$ round-r pre-votes for ρ_r are cast.

Lemma 8. *For $r \geq 1$, the following hold, except with negligible probability:*

(a) if an honest virtual node casts or accepts a main-vote of $b \in \{0, 1\}$ in round r, then ρ_r is defined and $\rho_r = b$;
(b) if an honest virtual node casts or accepts a hard pre-vote for $b \in \{0, 1\}$ in round $r + 1$, then ρ_r is defined and $\rho_r = b$;
(c) if an honest virtual node casts or accepts a main-vote of abstain in round $r + 1$, then ρ_r is defined and $\rho_r = CommonCoin(ID\|r) \oplus 1$;
(d) if r is the first round in which any honest virtual node decides, then all honest virtual nodes that ever decide, decide on the same value in either round r or $r + 1$.

Proof. To prove (a), suppose an honest virtual node accepts a main-vote of $b \in \{0, 1\}$ in round r. To be justified, this main-vote must be accompanied by a valid threshold signature on the message

$$(ID, \text{pre-vote}, r, b).$$

By the non-forgeability property of the signature scheme and because there is a unique instance of the protocol with tag ID, this implies there are at least $m - \alpha$ virtual nodes who cast pre-votes for b, among which at least $(m - \alpha) - \alpha = m - 2\alpha$ honest virtual nodes cast pre-votes for b. Thus, ρ_r has been defined and is equal to b. That proves (a).

Part (b) now simply follows from the fact that a hard pre-vote for $b \in \{0, 1\}$ in round $r + 1$ is justified by the same threshold signature as the main-vote from round r in part (a).

Now for (c). A main-vote of abstain in round $r + 1$ must be accompanied by a justification for two conflicting pre-votes, both 0 and 1, in round $r + 1$. These pre-votes cannot both be soft pre-votes because of the agreement property of the common coin protocol. They cannot both be hard pre-votes either. Suppose, towards a contradiction, that both were hard. By the definition of a hard pre-vote, the round-$(r + 1)$ pre-vote for b carries, for each $b \in \{0, 1\}$, a valid threshold signature on the message

$$(ID, \text{pre-vote}, r, b).$$

Arguing exactly as in (a), non-forgeability implies that for each $b \in \{0, 1\}$ at least $m - \alpha$ virtual nodes cast a round-r pre-vote for b, among which at least $(m - \alpha) - \alpha = m - 2\alpha$ are honest. Let H_b denote the set of honest virtual nodes that cast a round-r pre-vote for b, so that $|H_0| \geq m - 2\alpha$ and $|H_1| \geq m - 2\alpha$. As noted above, a forever-honest virtual node casts

exactly one round-r pre-vote and cannot equivocate, so H_0 and H_1 are disjoint; moreover, under the assumption $f_\alpha = \alpha$ made above there are exactly $m - \alpha$ honest virtual nodes, and $H_0 \cup H_1$ is contained in that set. Therefore

$$2(m - 2\alpha) \;\le\; |H_0| + |H_1| \;=\; |H_0 \cup H_1| \;\le\; m - \alpha,$$

which rearranges to $m \le 3\alpha$ and contradicts $m > 3\alpha$. This count leaves no slack: at $m = 3\alpha+1$ the two sides of the display read $2\alpha + 2$ and $2\alpha + 1$, so the contradiction is obtained by a margin of a single virtual node; bounding $|H_0 \cup H_1|$ by the total number m of virtual nodes rather than by the number $m - \alpha$ of honest ones would only yield the weaker $m \le 4\alpha$, which $m = 3\alpha + 1$ satisfies. Equivalently and more briefly, by (b) the two hard pre-votes would force ρ_r to be defined and equal to both 0 and 1, which is impossible. So one of these two pre-votes must be hard and the other must be soft. It follows from (b) that this hard pre-vote must be for ρ_r, and hence the other pre-vote must be a soft pre-vote for $1 - \rho_r$, and hence $CommonCoin(ID\|r) = \rho_r \oplus 1$. Part (c) now follows.

Now for (d). Suppose some virtual node CMNT_i decides $b \in \{0, 1\}$ in some round r, and no virtual node has decided in a previous round. Then in this round, CMNT_i accepted $m - \alpha$ main-votes for b. By part (a), we must have $b = \rho_r$. So any other honest virtual node who decides in round r must also decide ρ_r.

Of the $m - \alpha$ main-votes for b that CMNT_i accepted, at least $m - 2\alpha$ came from honest virtual nodes who main-voted b. Since $m > 3\alpha$, fewer than $m - 2\alpha$ syndicate signatures on the message

$$(ID, \text{main-vote}, r, \text{abstain})$$

have been or ever will be generated by honest virtual nodes, otherwise too many conflicting votes are generated by honest virtual nodes, which violates $m > 3\alpha$. This in turn implies that a soft pre-vote in round $r + 1$ cannot be justified. Thus, the only justifiable pre-votes in round $r + 1$ are hard pre-votes, and, by (b), these must be hard pre-votes of b. Finally, this implies that the only justifiable main-votes in round $r + 1$ are main-votes for b, and so all main-votes accepted by honest virtual nodes in round $r + 1$ will be main-votes for b. Hence, any honest virtual node that receives sufficiently many main-votes in round $r + 1$ also decides for b. □

Agreement follows from part (d). It remains to show *liveness* and *efficiency.*

Liveness is fairly straightforward. Here we rely on the robustness of the semi-interactive threshold signatures in an essential way: whenever a virtual node must compute the threshold signature needed to justify its next pre-vote (or main-vote for 0 or 1, respectively) according to the protocol, it can do so based on the syndicate signatures received previously in $m - \alpha$ valid main-votes (or pre-votes, respectively). It is then clear that honest virtual nodes will proceed from one round to the next, provided the adversary delivers enough messages between the honest virtual nodes. The *liveness* property follows from this observation, along with part (d) of Lemma 8, and the fact that virtual nodes who decide play along for one more round and then halt.

All that remains is *efficiency.* Lemma 8 says that in a given round $r + 1$, for $r \ge 1$, the set of $m - \alpha$ main-votes accepted by an honest virtual node in step (3) contains votes for either 0 or 1, but not both. Also, such an honest virtual node will decide in this round unless it accepts at least one main-vote of abstain. However, if it does accept an abstain, then $\rho_r = CommonCoin(ID\|r) \oplus 1$. The key to showing fast termination will be to show that *the value of ρ_r is determined before the coin $ID\|r$ is revealed.*

By "ρ_r being determined at a particular point in time," we mean the following: There is an efficient procedure W that takes as input a transcript describing the adversary's interaction with the system up to the given point in time, along with ID and $r \ge 1$, and outputs $w \in \{0, 1, ?\}$. Furthermore, if the output is $w \neq ?$, then if ρ_r ever becomes defined, it must be

equal to w (or at least, it should be computationally infeasible for an adversary to cause this not to happen).

When we say "the coin $(ID\|r)$ is revealed at a particular point in time," we mean the point in time when an honest virtual node generates the $(m - 2\alpha)$th share of the coin $(ID\|r)$.

Lemma 9. *There is a function W that determines ρ_r, as described above, such that for all $r \geq 1$, either ρ_r is determined before coin $ID\|r$ is revealed, or ρ_{r+1} is determined before $ID\|r + 1$ is revealed.*

Proof. Suppose an honest virtual node CMNT_i is just about to generate the $(m - 2\alpha)$-th share (all by honest virtual nodes) of coin $ID\|r$ in step 4 of round r. Hence, there is a set A of at least $m - 2\alpha$ honest virtual nodes who have also reached step 4 of round r. This set includes CMNT_i, who is the last member of A to release its share. Almost all round $r + 1$ pre-votes for the virtual nodes in A, as well as their justifications, are completely determined at this point, even if these votes have not actually been cast. The only exceptions are soft pre-votes, whose actual values are equal to $CommonCoin(ID\|r)$, which is not yet known.

If any virtual node in A is going to cast a hard pre-vote for $b \in \{0, 1\}$, then by part (b) of Lemma 8, b is the only possible value for ρ_r. Thus, ρ_r is already determined—in fact, it is already defined.

Otherwise, all virtual nodes in A are going to cast soft pre-votes, choosing the value $CommonCoin(ID\|r)$ as the value of their round $r + 1$ pre-vote. It follows that the only possible value for ρ_{r+1} is $CommonCoin(ID\|r)$. Therefore, immediately *after* CMNT_i reveals its share of coin $ID\|r$, ρ_{r+1} is determined. Moreover, the coin $ID\|r + 1$ has not yet been revealed at this point, since fewer than $m - 2\alpha$ honest virtual nodes have gone beyond step 2 of round $r + 1$. Thus, ρ_{r+1} is determined before coin $(ID\|r + 1)$ is revealed. □

This lemma, together with the unpredictability property of sequences of coins, implies that the probability that any honest virtual node advances more than $2r + 1$ rounds is bounded by $2^{-r} + \epsilon$, where ϵ is negligible. *Efficiency* now follows since every honest virtual node generates only a fixed polynomial number of messages in every round. Note that to make this argument rigorous, we need to be able to "predict" explicitly the desired value of the coin $\rho_r \oplus 1$ that would delay termination, which is why we defined the notion of "determining" ρ_r as we did.

We remark that if the first honest virtual node to decide makes its decision in round r, there may be others who make their decision in round $r + 1$. The "early deciders" play along for round $r + 1$, which allows the "late deciders" to decide. However, the "late deciders" do not "know" they are "late," so they attempt to play along for round $r + 2$. What happens is that in round $r + 2$, the protocol will "fizzle out": the "late deciders" will simply end up waiting in step 2 for $m - \alpha$ messages that never arrive. This "fizzling out" does indeed satisfy our technical definition of termination, and is perhaps adequate for some settings; however, a more "decisive" termination can be achieved with a minor modification of the protocol (see Section 5.3.1 of [23]).

That completes the proof of *agreement, liveness,* and *efficiency* for the case $f = t$ and $f_\alpha = \alpha$. We now sketch the differences for the case $f < t$ and $f_\alpha < \alpha$. There are some annoying technical problems that arise in this case because there is a gap between the number $(m - 2\alpha)$ of shares for a signature (or coin) that need to be revealed before the signature (or coin) *may* be reconstructible, and the number $(m - \alpha - f_\alpha)$ of shares that need to be revealed before it *can* be reconstructed. We could have defined security for threshold signatures (coins) so that this gap did not exist; however, such a definition would be stronger than necessary.

Consider an adversary that chooses to corrupt a set C of $f_\alpha < \alpha$ virtual nodes. Let H denote the set of $m - f_\alpha$ honest virtual nodes. We choose an arbitrary subset $Q \subset H$ of $\alpha - f_\alpha$ "quasi-corrupted" virtual nodes. The idea is that for the purposes of the protocol properties,

virtual nodes in Q are considered to be honest, but for the purposes of the threshold signature and coin-tossing schemes, virtual nodes in Q are considered corrupted.

What this means concretely is that for virtual nodes in Q, their secret shares for the threshold schemes are revealed to the adversary, but they otherwise behave as honest players with which the adversary interacts in the usual way. The main implication of this is that a particular signature or coin can be reconstructed if and only if at least $m - 2\alpha$ virtual nodes in $H \setminus Q$ contribute shares. We also modify the proof as follows:

- In formulating the definition of ρ_r, we only count votes cast by members of $H \setminus Q$.
- In formulating the notion of precisely when a coin is revealed, we only count shares generated by virtual nodes in $H \setminus Q$.

With these modifications, Lemmas 8 and 9 can easily be proved, exactly as stated, and Theorem 9 follows.

Efficiency Analysis Let Num_R denote the number of iterations of steps 1–4 required before all honest nodes terminate in an instance of Algorithm 3. Let Comm_{Pre} represent the communication cost of the pre-processing phase, and let Comm_i denote the communication cost for executing steps 1–4 during the i-th iteration. The overall communication cost of the instance is therefore:

$$\mathsf{Comm} = \mathsf{Comm}_{Pre} + \sum_{i=1}^{\mathsf{Num}_R} \mathsf{Comm}_i.$$

Similarly, let Rnd_{Pre} and Rnd_i denote the number of rounds incurred during the pre-processing phase and the i-th iteration, respectively. The total round complexity is:

$$\mathsf{Rnd} = \mathsf{Rnd}_{Pre} + \sum_{i=1}^{\mathsf{Num}_R} \mathsf{Rnd}_i.$$

During the pre-processing phase, each community CMNT_k performs the following tasks: (1) Executes an intra-community MBA instance, incurring $O(q^2\lambda\gamma)$ communication cost and $O(\gamma)$ rounds; (2) Generates a syndicate signature, costing $O(q^2\lambda)$ communication and 1 round; and (3) Has each member send an $O(\lambda)$-sized message to $\chi = \Omega\big(\sqrt{n}(\gamma + \log n)\big)$ parties, resulting in $O(q\chi\lambda)$ bits of communication and 1 round. Since m communities perform these tasks in parallel, the total pre-processing communication cost is $\mathsf{Comm}_{Pre} = m \cdot O(q^2\gamma\lambda + q^2\lambda + q\chi\lambda)$, and the round complexity is $\mathsf{Rnd}_{Pre} = O(\gamma)$. Given that $m, q = O(\sqrt{n})$, substituting χ yields $\mathsf{Comm}_{Pre} = O(\lambda n^{1.5}(\gamma + \log n))$.

During the i-th iteration, each community CMNT_k performs the following: (1) executes two intra-community MBA instances; (2) generates two syndicate signatures; (3) has each member send two constant-sized messages to χ parties; and (4) executes one common coin generation, which alone incurs $O(\lambda n^{1.5}(\gamma + \log n))$ communication and two rounds. (Although the final iteration omits the common coin generation, its cost is strictly bounded by the preceding iterations). Thus, we bound the per-iteration costs as $\mathsf{Comm}_i = O(\lambda n^{1.5}(\gamma + \log n))$ and $\mathsf{Rnd}_i = O(\gamma)$.

Consequently, the total communication complexity is $\mathsf{Comm} = \mathsf{Num}_R \cdot O(\lambda n^{1.5}(\gamma + \log n))$, and the total round complexity is $\mathsf{Rnd} = \mathsf{Num}_R \cdot O(\gamma)$. Regardless of the adversarial strategy, the expected value of Num_R is a small constant (i.e., $\mathbb{E}[\mathsf{Num}_R] = O(1)$). This implies that the expected communication complexity of our ABA protocol is $O(\lambda n^{1.5}(\gamma + \log n))$, and the expected round complexity is $O(\gamma)$.

7 Lower bounds on Asynchronous Setup-Free Common Coin Communication Complexity Against Weakly Adaptive Adversaries

We show that some form of setup is necessary for weakly adaptive secure asynchronous strong common coin with (non-amortized) subquadratic communication complexity. We note that [16] shows an impossibility result in the same setting for Byzantine agreement, and our lower bound adapts their ideas to common coin protocols where parties do not have explicit inputs but honest parties still agree on an unpredictable common output value. Like theirs, our bound holds even if we allow secure erasure, and secret channels between all the parties.

We provide an outline of our proof that omits several technical details, but conveys the main ideas. Let Π be a setup-free protocol for asynchronous common coin with subquadratic communication complexity. We show an efficient attacker $\mathcal{A}$ who succeeds in violating the security of Π. The attacker exploits the fact that with high probability, a uniformly sampled (honest) party P will communicate with only $o(n)$ other parties during an execution of Π. The adversary $\mathcal{A}$ can use this fact to "isolate" P from the remaining honest parties in the network and cause a violation of φ-fairness. In more detail, $\mathcal{A}$ delays communication from the other honest parties to P in the real execution, sends messages to P from corrupted parties to trick P into thinking that it is running in an independent simulated execution of Π in which all parties are honest, while fooling other honest parties into believing they are running an independent execution and at most f (corrupted) parties are silent. By an indistinguishability argument, P will output some value V while other honest parties may output V' with too high probability. This contradicts agreement.

To "isolate" P as described, $\mathcal{A}$ runs one simulated execution of Π alongside the real execution of the protocol. We note that it is crucial that Π is setup-free, so $\mathcal{A}$ can run the simulated execution on behalf of all parties. $\mathcal{A}$ delays messages sent by honest parties to P in the real execution until P terminates. This may seem easy to do in the asynchronous setting, but an issue remains in whether P can *eventually* terminate, since $\mathcal{A}$ should not delay messages "indefinitely", i.e., beyond polynomial time. When a party $Q \in S'$ sends a message to P in the simulated execution, $\mathcal{A}$ corrupts Q in the real execution and then sends that message on Q's behalf. Analogously, when P sends a message to some honest party $Q \in S'$ in the real execution, $\mathcal{A}$ "intercepts" that message and forwards it to the corresponding party in the simulation. A subtlety here is that messages sent between two honest parties cannot be observed via eavesdropping, because we allow secret channels, and can not necessarily be observed by adaptively corrupting the recipient Q after it receives the message, since we allow erasure. Instead, $\mathcal{A}$ must adaptively corrupt Q before it receives the message sent by P. It only remains to argue that, in carrying out this strategy, $\mathcal{A}$ does not exceed the corruption bound.

We first present two auxiliary lemmas before moving on to the main theorem.

Lemma 10. *Let U be a uniform random variable over the space $\mathcal{V}$, and let X be any random variable over the same space yet independent of U. It follows that* $\Pr[U = X] = \frac{1}{|\mathcal{V}|}$.

Proof. Basic probability analysis. □

Lemma 11. *Let* aux *be an arbitrary auxiliary string. Let V be a random variable over $\mathcal{V}$ which is computationally indistinguishable with a uniform random variable U even conditioned on* aux*, i.e., for any PPT adversary $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}(V, \mathsf{aux}) = 1] - \Pr[\mathcal{A}(U, \mathsf{aux}) = 1]| \leq \mathsf{negl}(\lambda). \tag{4}$$

Then, for any PPT algorithm C*, it holds that*

$$\Pr[V = \mathsf{C}(\mathsf{aux})] \leq \frac{1}{|\mathcal{V}|} + \mathsf{negl}(\lambda). \tag{5}$$

Proof. Assume there exists a PPT algorithm C such that the inequality in Eq.5 is violated. That is, for some non-negligible function $\epsilon(\lambda)$:

$$\Pr[V = \mathsf{C}(\text{aux})] > \frac{1}{|\mathcal{V}|} + \epsilon(\lambda)$$

We now construct a PPT adversary $\mathcal{A}$ that aims to distinguish whether a given input $Z \in \mathcal{V}$ is sampled from the distribution V or the uniform distribution U. Specifically, when $\mathcal{A}$ receives a challenge value $Z \in \mathcal{V}$ and the auxiliary string aux, it runs the algorithm $\mathsf{C}(\text{aux})$ to obtain a prediction Z^*. If $Z = Z^*$, $\mathcal{A}$ outputs 1 (guessing the input is V). If $Z \neq Z^*$, $\mathcal{A}$ outputs 0 (guessing the input is U).

Now we analyze the success probability in the following cases:

- **Case 1:** Z is sampled from V. By our initial assumption, the probability that $\mathcal{A}$ outputs 1 is:
$$\Pr[\mathcal{A}(V, \text{aux}) = 1] = \Pr[V = \mathsf{C}(\text{aux})] > \frac{1}{|\mathcal{V}|} + \epsilon(\lambda).$$
- **Case 2:** Z is sampled from U. U is a uniform random variable independent of aux (and thus independent of the output of C). According to Lemma 10, the probability that a uniform variable matches any independent variable is exactly $1/|\mathcal{V}|$, namely,
$$\Pr[\mathcal{A}(U, \text{aux}) = 1] = \Pr[U = \mathsf{C}(\text{aux})] = \frac{1}{|\mathcal{V}|}.$$

Thus, the advantage of $\mathcal{A}$ is larger than $\epsilon(\lambda)$, which is non-negligible. By contradiction, Eq.5 must hold. □

Theorem 10. *Let Π be a setup-free (n, f, φ)-common coin protocol operated by n nodes against any weakly adaptive PPT adversary that corrupts up to f nodes in an asynchronous network, where $\varphi = \frac{1}{2} + 4c + \delta(\lambda) + \frac{1}{2|\mathcal{V}|} \leq 1$ and $c \in [0, 1/9)$ is a constant, $\delta(\lambda)$ is a small yet non-negligible function, and $|\mathcal{V}|$ is the size of the output space. Then, in expectation, the honest nodes in Π collectively send at least $c^2 f^2$ messages.*

Proof. We assume $f < n/3$ and show that in this case, the expected number of messages is at least $c^2 f^2$. Our analysis still holds for $f \geq n/3$ by the same argument.

Fix an (n, f, φ)-common coin protocol Π whose expected number of messages is less than $c^2 f^2$. Fix a subset $S \subset [n]$ with $|S| = \lfloor f/2 \rfloor$. Let S' denote the remaining parties with $|S'| = n - \lfloor f/2 \rfloor$.

Consider an execution (Ext1) of Π that proceeds as follows: At the start of the execution, an adversary corrupts all parties in S and they immediately abort. The parties in S' remain honest and run Π honestly. By (n, f, φ)-security of Π we have:

Lemma 12. *In Ext1, regardless of the strategy of $\mathcal{A}$, there is a good event T_1 that happens except with negligible probability such that all parties in S' terminate. There is also a good event E_1 that happens with probability at least $\varphi - \mathsf{negl}(\lambda)$, such that (1) all parties in S' output the same value $V_1 \in \mathcal{V}$, and (2) V_1 satisfies the unpredictability and bias-resistance, i.e., for any PPT algorithm $\mathcal{D}$,*

$$\begin{aligned} &|\Pr[\mathcal{D}(V_1, \mathsf{State}_{\mathcal{A}}^{\mathit{Ext1,before}}) = 1] - \\ &\Pr[\mathcal{D}(U_1, \mathsf{State}_{\mathcal{A}}^{\mathit{Ext1,before}}) = 1]| \leq \mathsf{negl}(\lambda), \end{aligned}$$

where $\mathsf{State}_{\mathcal{A}}^{\mathit{Ext1,before}}$ is the state of $\mathcal{A}$ before $n - f - |\mathcal{C}|$ so-far-honest nodes participate in the protocol, and U_1 is the uniform random variable over the output space.

Now consider an execution (Ext2) of Π involving an adversary $\mathcal{A}$. As explained in the proof intuition, $\mathcal{A}$'s goal is to make P believe it is running in a simulated execution in which all parties are honest, to make the honest parties in S' believe they are running in Ext1. The key idea is to separate the honest party P from all the honest parties in S'. At the start of the execution, $\mathcal{A}$ chooses a uniform $P \in S$ and corrupts all parties in S except for P. All parties in S' are initially honest. $\mathcal{A}$ maintains one simulated execution that we label blue. In the blue execution, $\mathcal{A}$ plays the role of all parties except P; all these virtual parties run Π honestly, except that P in the simulation replicates its behaviors in the real execution. $\mathcal{A}$ uses the blue simulation to determine how to interact with the honest parties in the real execution. Now, we specify how nodes and $\mathcal{A}$ behave in real and simulated (blue) execution.

In blue execution:

- P: P does not execute Π. $\mathcal{A}$ can "intercept" what P sends to every other node in the real execution (with high probability, as we will show later), and it orders P in the blue execution to send exactly the same outgoing messages. When P receives an incoming message m from Q in the blue execution, $\mathcal{A}$ adaptively corrupts Q in the real execution (if it has not already), and lets Q send m to P via its blue thread.
- Other nodes: they honestly execute Π. When Q intends to send m to P, $\mathcal{A}$ corrupts Q in the real execution and orders Q to send m to P in the real execution.

In real execution:

- P and uncorrupted parties in S': P is never corrupted and always runs Π honestly; likewise, uncorrupted parties in S' execute Π honestly.
- $\mathcal{A}$: $\mathcal{A}$ initially corrupts all $\lfloor f/2 \rfloor - 1$ nodes in $S \setminus \{P\}$. $\mathcal{A}$ adaptively corrupts $Q \in S'$ in the real execution upon two events: (a) In real execution, when P sends m to Q. (b) In blue execution, when Q sends m to P. When event (a) occurs, $\mathcal{A}$ delays the message, corrupts Q (if Q has not already been corrupted), and delivers the message (so that $\mathcal{A}$ can learn m).
  For message scheduling, $\mathcal{A}$ makes sure messages sent by honest parties in S' to P are delayed, and delivered only after P terminates. To show feasibility, we provide a formal justification in Lemma 14.
- Corrupted $S \setminus \{P\}$: They only handle incoming messages sent by P and forward them to the blue execution. They deliver messages only to P if the blue execution indicates so.
- Corrupted parties in S': A corrupted party $Q \in S'$ maintains three isolated threads: the sink thread, the real thread and the blue thread. The sink thread handles incoming messages from P without letting other threads know. The real thread handles all other incoming messages and executes Π honestly. The blue thread does not handle any incoming messages but only sends m to P in the real execution as the blue execution indicates. (Note that the real thread does not know the fact that the blue thread has made Q send messages to P due to the blue execution or that P has sent any messages to Q.)

If $\mathcal{A}$ would ever need to corrupt more than f parties in total, then it simply aborts. However, the real execution continues without any further interference from $\mathcal{A}$.

Finally, consider an execution (Ext3) in which $P \in S$ is chosen uniformly and then Π is run honestly. By Definition 3, we establish the following.

Lemma 13. *Let T_3 be the event that all parties terminate. Let E_3 be the event that all parties output the same value $V_3 \in \mathcal{V}$ and V_3 satisfies the unpredictability and bias-resistance, i.e., for any PPT algorithm $\mathcal{D}$,*

$$|\Pr[\mathcal{D}(V_3, \mathsf{State}_{\mathcal{A}}^{\mathit{Ext3,before}}) = 1] - \Pr[\mathcal{D}(U_3, \mathsf{State}_{\mathcal{A}}^{\mathit{Ext3,before}}) = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{State}_{\mathcal{A}}^{Ext3,before}$ *is the state of* $\mathcal{A}$ *before* $n - f - |\mathcal{C}|$ *so-far-honest nodes participate in the protocol, and* U_3 *is the uniform random variable over the output space.*

In particular, in Ext3, regardless of the strategy of $\mathcal{A}$*, we have* $\Pr[T_3] \geq 1 - \mathsf{negl}(\lambda)$ *and* $\Pr[E_3] \geq \varphi - \mathsf{negl}(\lambda)$*.*

Suppose a good event G_1 happens in Ext2, so $\mathcal{A}$ does not abort. Then conditioned on G_1, we show P terminates with high probability. Finally, we show both events simultaneously happen with positive probability.

Lemma 14. *Let* G_1 *be the event that* $\mathcal{A}$ *does not exceed its corruption bound in Ext2. In Ext2, conditioned on the event* G_1*, the view of* P *is distributed identically to the view of* P *in Ext3. In particular,* P *terminates with* $1 - \mathsf{negl}(\lambda)$ *probability.*

Proof. In Ext2, when event G_1 occurs, P receives no messages from the real execution. A remaining subtlety is that some messages sent by $Q \in S'$ to P are still being delayed by $\mathcal{A}$ but $\mathcal{A}$ has to deliver them *eventually*. It is easy to see that without the delivery of these messages, the view of P in Ext2 is determined by the blue execution in which all parties, including a P replica, run Π honestly. Therefore, the view of P in Ext2 is distributed identically to the view of P in Ext3. Thus, P terminates with $1 - \mathsf{negl}(\lambda)$ probability by Lemma 13. This means $\mathcal{A}$ can eventually deliver the delayed messages. □

Lemma 15. *In Ext2, let* G_2 *be the event that* P *terminates and* G *be* $G_1 \wedge G_2$*. Then* $\Pr[G] \geq 1 - 8c - \mathsf{negl}(\lambda)$*.*

Proof. We first show that $\Pr[G_1] \geq 1 - 4c$. $\mathcal{A}$ aborts if it would exceed the corruption bound. Initially, only the $\lfloor f/2 \rfloor - 1$ parties in S are corrupted. Let M denote the total number of messages sent either by the parties in S' to the parties in S or by parties in S to parties in S' in the blue execution. By assumption, $Exp[M] < c^2 f^2$. Let X be the event that $M \leq cf^2/2$. By Markov's bound, we have

$$\Pr[X] \geq Pr\left[M \leq \frac{Exp[M]}{2c}\right] \geq 1 - 2c.$$

Among the first $cf^2/2$ messages exchanged between parties in S' and S, let Y be the event that at most $\lceil f/2 \rceil$ messages are sent or received by a uniformly chosen $P \in S$.

By the pigeonhole principle, at most $(cf^2/2)/\lceil f/2 \rceil \leq cf$ parties in S can receive and/or send $\lceil f/2 \rceil$ or more of those messages, and so $\Pr[Y] \geq 1 - (cf^2/2)/(|S|\lceil f/2 \rceil) \geq 1 - cf/|S| = 1 - 2c$. [12] Thus, $\Pr[G_1] \geq \Pr[X \wedge Y] = \Pr[X] + \Pr[Y] - \Pr[X \cup Y] \geq (1-2c) + (1-2c) - 1 = 1 - 4c$. The first inequality is the inclusion $X \wedge Y \subseteq G_1$: when X and Y occur, every message exchanged between S and S' is among the first $cf^2/2$ such messages, and at most $\lceil f/2 \rceil$ of them are sent or received by P, so $\mathcal{A}$ adaptively corrupts at most $\lceil f/2 \rceil$ parties in S'; together with the $\lfloor f/2 \rfloor - 1$ parties of $S \setminus \{P\}$ corrupted upfront, this amounts to at most $f - 1$ corruptions, and $\mathcal{A}$ does not abort. The converse inclusion fails, so this relation cannot be strengthened to an equality: $\mathcal{A}$ likewise does not abort when P exchanges more than $\lceil f/2 \rceil$ messages with at most $\lceil f/2 \rceil$ distinct parties of S', as each such party is corrupted only once, or when M exceeds $cf^2/2$ owing to the parties in $S \setminus \{P\}$ while P itself stays quiet.

Lemma 14 states $\Pr[G_2|G_1] \geq 1 - \mathsf{negl}(\lambda)$. Therefore, we have $\Pr[G] = \Pr[G_1 \wedge G_2] = \Pr[G_1] + \Pr[G_2] - \Pr[G_1 \vee G_2] \geq \Pr[G_1] + \Pr[G_2] - 1 > \Pr[G_1] + \Pr[G_2|G_1]\Pr[G_1] - 1 \geq 1 - 4c + (1 - \mathsf{negl}(\lambda))(1 - 4c) - 1 \geq 1 - 8c - \mathsf{negl}(\lambda)$. □

[12] It is convenient to view the communication between S and S' as an undirected, bipartite multi-graph in which each node represents a party and an edge (U, V) represents a message sent between parties $U \in S$ and $V \in S'$. As the number of edges in this graph is at most $cf^2/2$, there can not be more than cf nodes in S whose total degree is at least $\lceil f/2 \rceil$.

Moreover, we show that honest parties in S' cannot distinguish Ext1 and Ext2.

Lemma 16. *In Ext2, the distribution of the joint view of all parties in S' who remain uncorrupted is identical to the distribution of their joint view in Ext1.*

Let T_2 be the event that all parties in S' who remain uncorrupted terminate. Let E_2 be the event that all parties in S' who remain uncorrupted output the same value $V_2 \in \mathcal{V}$ and V_2 satisfies the unpredictability and bias-resistance, i.e., for any PPT algorithm $\mathcal{D}$,

$$\begin{aligned}&|\Pr[\mathcal{D}(V_2, \mathsf{State}_{\mathcal{A}}^{\mathit{Ext2,before}}) = 1] - \\ &\Pr[\mathcal{D}(U_2, \mathsf{State}_{\mathcal{A}}^{\mathit{Ext2,before}}) = 1]| \leq \mathsf{negl}(\lambda),\end{aligned}$$

where $\mathsf{State}_{\mathcal{A}}^{\mathit{Ext2,before}}$ is the state of $\mathcal{A}$ before $n - f - |\mathcal{C}|$ so-far-honest nodes participate in the protocol, and U_2 is the uniform random variable over the output space.

In particular, in Ext2, regardless of the strategy of $\mathcal{A}$, we have $\Pr[T_2] \geq 1 - \mathsf{negl}(\lambda)$ and $\Pr[E_2] \geq \varphi - \mathsf{negl}(\lambda)$.

Proof. The only messages received by the parties in S' in either Ext1 or Ext2 are those that arise from an honest execution of Π among the parties in S'. Moreover, in Ext2 the decision as to whether or not a party in S' is corrupted is independent of the joint view of all uncorrupted parties in S'. Note that in Ext2, honest parties in S' still hear from corrupted parties' real threads, which believe they have never heard from P and otherwise receive all incoming messages from other parties in S' and behave honestly. The final statement follows from Lemma 12. □

Finally, we combine the above intermediate lemmas and show Π errs with too high probability. Observe that the honest node P in Ext2 must follow a specific (possibly randomized) algorithmon input its view view, including its initial state init_P and any received messages msg_P of the protocol execution, to decide its final output. Let V be the random variable of P's output in Ext2. Conditioned on event G, $\mathcal{A}$ knows all msg_P messages received by P in Ext2 since $\mathcal{A}$ solely controls the generation of them in the blue execution, depending only on its initial state $\mathsf{init}_{\mathcal{A}}$. Obviously, $\mathsf{init}_{\mathcal{A}} \in \mathsf{State}_{\mathcal{A}}^{\mathsf{Ext2,before}}$, whereas $\mathsf{init}_P \notin \mathsf{State}_{\mathcal{A}}^{\mathsf{Ext2,before}}$.

Thus, we can write $V = \mathsf{C}(\mathsf{view}')$ for some algorithm C where $\mathsf{view}' = \mathsf{State}_{\mathcal{A}}^{\mathsf{Ext2,before}} \cup \{\mathsf{init}_P\}$. Let V' be the random variable of the output of another honest node $Q \in S'$ in Ext2.

Next, we proceed to show that, when both good events G and E_2 happen, it holds that, for any PPT algorithm $\mathcal{D}$,

$$\begin{aligned}&|\Pr[\mathcal{D}(V', \mathsf{view}') = 1] - \\ &\Pr[\mathcal{D}(U, \mathsf{view}') = 1]| \leq \mathsf{negl}(\lambda),\end{aligned} \tag{6}$$

The challenge is due to $\mathsf{view}' \notin \mathsf{State}_{\mathcal{A}}^{\mathsf{Ext2,before}}$, as init_P is the state of an honest party P. Therefore, $\mathcal{A}$ only needs to know the initial state of P to generate view'. Recall that $\mathcal{A}$ only corrupts $|S| - 1 + \lceil f/2 \rceil = f - 1$ parties in Ext2. This allows us to consider an extended adversary $\widetilde{\mathcal{A}}$. $\widetilde{\mathcal{A}}$ is identical to $\mathcal{A}$, except it corrupts P upfront but always honestly executes the code of P. Let $\widetilde{V}$ and $\widetilde{V'}$ be the output of P and another honest Q in the execution with $\widetilde{\mathcal{A}}$, respectively. In this execution, when the good events G and E_2 happen, it holds that

$$\begin{aligned}&|\Pr[\mathcal{D}(\widetilde{V'}, \mathsf{State}_{\widetilde{\mathcal{A}}}^{\mathsf{Ext2,before}}) = 1] - \\ &\Pr[\mathcal{D}(U, \mathsf{State}_{\widetilde{\mathcal{A}}}^{\mathsf{Ext2,before}}) = 1]| \leq \mathsf{negl}(\lambda).\end{aligned}$$

Since the initial view init_P of P is in $\mathsf{State}_{\widetilde{\mathcal{A}}}^{\mathsf{Ext2,before}}$, we have view' fully captured by $\mathsf{State}_{\widetilde{\mathcal{A}}}^{\mathsf{Ext2,before}}$. Therefore, we have

$$\left|\Pr[\mathcal{D}(\widetilde{V'}, \mathsf{view}') = 1] - \Pr[\mathcal{D}(U, \mathsf{view}') = 1]\right| \leq \mathsf{negl}(\lambda).$$

On the other hand, it is easy to see the execution under $\mathcal{A}$'s influence is identical to the execution under $\widetilde{\mathcal{A}}$'s influence, so we have $\widetilde{V} = V$ and $\widetilde{V'} = V'$. Eq. 6 follows.

Finally, we proceed to analyze the probability $V \neq V'$. In particular, we have that:

$$
\begin{aligned}
& \Pr[V \neq V'] \\
\geq\ & \Pr[V \neq V' | E_2 \wedge G] \Pr[E_2 \wedge G] \\
=\ & \Pr[\mathsf{C}(\mathsf{view}') \neq V' | E_2 \wedge G] \Pr[E_2 \wedge G] \\
>\ & (1 - \frac{1}{|\mathcal{V}|} - \mathsf{negl}(\lambda)) \Pr[E_2 \wedge G] \\
=\ & (1 - \frac{1}{|\mathcal{V}|} - \mathsf{negl}(\lambda))(\Pr[E_2] + \Pr[G] - \Pr[E_2 \vee G]) \\
>\ & (1 - \frac{1}{|\mathcal{V}|} - \mathsf{negl}(\lambda))(\varphi - \mathsf{negl}(\lambda) + 1 - 8c - \mathsf{negl}(\lambda) - 1) \\
=\ & (1 - \frac{1}{|\mathcal{V}|} - \mathsf{negl}(\lambda))(\varphi - 8c - \mathsf{negl}(\lambda)) \\
=\ & (1 - \frac{1}{|\mathcal{V}|} - \mathsf{negl}(\lambda))(\frac{1}{2} + 4c + \delta(\lambda) + \frac{1}{2|\mathcal{V}|} - 8c - \mathsf{negl}(\lambda)) \\
=\ & (1 - \frac{1}{|\mathcal{V}|} - \mathsf{negl}(\lambda))(\frac{1}{2} - 4c + \delta(\lambda) + \frac{1}{2|\mathcal{V}|} - \mathsf{negl}(\lambda)) \\
>\ & \frac{1}{2} - 4c + \delta(\lambda) - \frac{1}{2|\mathcal{V}|}(-8c + 2\delta(\lambda) + \frac{1}{|\mathcal{V}|}) - \mathsf{negl}(\lambda) \\
>\ & \frac{1}{2} - 4c + \delta(\lambda) - \frac{1}{2|\mathcal{V}|} - \mathsf{negl}(\lambda) \\
>\ & \frac{1}{2} - 4c - \frac{1}{2|\mathcal{V}|} \\
=\ & 1 - \varphi + \delta(\lambda)
\end{aligned}
\tag{7}
$$

where the inequality at Line 4 follows from Lemma 11 (seeing view' as the auxiliary string aux). The equation above suggests that the agreement is violated with probability at least $1 - \varphi + \delta(\lambda)$, for some non-negligible $\delta(\lambda)$, which contradicts our assumption that the protocol satisfies φ-fairness. □

8 Conclusion and Open Problems

In this work, we have addressed the long-standing challenge of achieving balanced scalability in asynchronous distributed randomness and consensus. By introducing the first balanced asynchronous common coin and Binary Byzantine Agreement (ABA) protocols with sub-quadratic communication complexity, we eliminate the scalability bottleneck where a small subset of participants bears a disproportionate $\Omega(n)$ communication burden. At the heart of our design are explicit sampler constructions that partition n nodes, tolerating up to $f < (1/2 - \epsilon)n$ corruptions, into $O(\sqrt{n})$ communities. This allows each community to emulate a "virtual node" within a conventional protocol framework, while the sublinear number of virtual nodes drives the overall communication complexity down.

While our current protocols incur an $\widetilde{O}(\sqrt{n})$-bit communication cost per node, several promising directions for future research remain. It may be possible to further reduce this per-node cost to $O(\mathrm{poly}(\log n))$ through the application of a recursive partitioning strategy. However, limitations apply to our construction when the recursion technique is applied directly. Additionally, while our designs currently rely on a private trusted setup, investigating whether such a setup is strictly necessary to maintain our efficiency and security goals remains an open and interesting question. We leave this for future study.

Acknowledgment

We thank anonymous reviewers of PODC'26 and Abigail Gentle for valuable discussions. This work is supported in part by ARC Discovery Project (DP250101739), and research awards from the Stellar Development Foundation, and Input Output (iog.io) as part of the Cardano Vision 2026 research initiative.

References

1. Tor shared random subsystem specification. https://spec.torproject.org/srv-spec/index.html
2. Abel, R.J.R., Bluskov, I., Greig, M.: Balanced incomplete block designs with block size˜9: Part III. Australas. J Comb. **30**, 57–74 (2004), http://ajc.maths.uq.edu.au/pdf/30/ajc_v30_p057.pdf
3. Abraham, I., Bacho, R., Loss, J., Stern, G.: Nearly quadratic asynchronous distributed key generation. IACR Cryptol. ePrint Arch. p. 6 (2025), https://eprint.iacr.org/2025/006
4. Abraham, I., Chan, T.H., Dolev, D., Nayak, K., Pass, R., Ren, L., Shi, E.: Communication complexity of byzantine agreement, revisited. Distributed Comput. **36**(1), 3–28 (2023)
5. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G.: Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 14081, pp. 39–70. Springer (2023)
6. Adida, B.: Helios: Web-based open-audit voting. In: USENIX Security Symposium. pp. 335–348. USENIX Association (2008)
7. Bacho, R., Kavousi, A.: Sok: Dlog-based distributed key generation. In: Blanton, M., Enck, W., Nita-Rotaru, C. (eds.) IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025. pp. 614–632. IEEE (2025). https://doi.org/10.1109/SP61157.2025.00127, https://doi.org/10.1109/SP61157.2025.00127
8. Bacho, R., Loss, J.: On the adaptive security of the threshold BLS signature scheme. In: CCS. pp. 193–207. ACM (2022)
9. Bacho, R., Loss, J.: Adaptively secure (aggregatable) PVSS and application to distributed randomness beacons. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023. pp. 1791–1804. ACM (2023). https://doi.org/10.1145/3576915.3623106, https://doi.org/10.1145/3576915.3623106
10. Bacho, R., Loss, J.: Adaptively secure (aggregatable) PVSS and application to distributed randomness beacons. IACR Cryptol. ePrint Arch. **2023**, 1348 (2023), https://eprint.iacr.org/2023/1348
11. Baker, R.C., Harman, G., Pintz, J.: The difference between consecutive primes, ii. Proceedings of the London Mathematical Society **83**(3), 532–562 (2001). https://doi.org/https://doi.org/10.1112/plms/83.3.532, https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/83.3.532
12. Bandarupalli, A., Bhat, A., Bagchi, S., Kate, A., Reiter, M.K.: Random beacons in monte carlo: Efficient asynchronous random beacon *without* threshold cryptography. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024. pp. 2621–2635. ACM (2024). https://doi.org/10.1145/3658644.3670326, https://doi.org/10.1145/3658644.3670326
13. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. J. Cryptol. **16**(3), 185–215 (2003). https://doi.org/10.1007/S00145-002-0120-1, https://doi.org/10.1007/s00145-002-0120-1
14. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10. ACM (1988)
15. Beth, T., Jungnickel, D., Lenz, H.: Design theory. Cambridge University Press, USA (1986)
16. Blum, E., Katz, J., Liu-Zhang, C., Loss, J.: Asynchronous byzantine agreement with subquadratic communication. In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12550, pp. 353–380. Springer (2020). https://doi.org/10.1007/978-3-030-64375-1_13, https://doi.org/10.1007/978-3-030-64375-1_13
17. Bollobás, B.: A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. Eur. J. Comb. **1**(4), 311–316 (1980). https://doi.org/10.1016/S0195-6698(80)80030-8, https://doi.org/10.1016/S0195-6698(80)80030-8
18. Boyle, E., Cohen, R., Goel, A.: Breaking the o($\sqrt{}$ n)-bit barrier: Byzantine agreement with polylog bits per party. In: Miller, A., Censor-Hillel, K., Korhonen, J.H. (eds.) PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021. pp. 319–330. ACM (2021). https://doi.org/10.1145/3465084.3467897, https://doi.org/10.1145/3465084.3467897

19. Boyle, E., Cohen, R., Goel, A.: Breaking the $o(\sqrt{n})$-bit barrier: Byzantine agreement with polylog bits per party. J. Cryptol. **37**(1), 2 (2024). https://doi.org/10.1007/S00145-023-09484-0, https://doi.org/10.1007/s00145-023-09484-0
20. Bruck, R.H.: Finite nets, i. numerical invariants. Canadian Journal of Mathematics **3**, 94–107 (1951). https://doi.org/10.4153/cjm-1951-012-7, http://dx.doi.org/10.4153/CJM-1951-012-7
21. Bruck, R.H.: Finite nets. ii. uniqueness and imbedding. Pacific Journal of Mathematics **13**(2), 421–457 (Jun 1963). https://doi.org/10.2140/pjm.1963.13.421, http://dx.doi.org/10.2140/pjm.1963.13.421
22. Burnwal, S.P., Sinha, K., Vidyasagar, M.: New and explicit constructions of unbalanced ramanujan bipartite graphs. The Ramanujan Journal **57**(3), 1043–1069 (Mar 2022). https://doi.org/10.1007/s11139-021-00384-0, https://doi.org/10.1007/s11139-021-00384-0
23. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography (extended abstract). In: Neiger, G. (ed.) Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, July 16-19, 2000, Portland, Oregon, USA. pp. 123–132. ACM (2000). https://doi.org/10.1145/343477.343531, https://doi.org/10.1145/343477.343531
24. Charles J. Colbourn, J.H.D.: Handbook of Combinatorial Designs. Chapman and Hall/CRC (Nov 2006). https://doi.org/10.1201/9781420010541, http://dx.doi.org/10.1201/9781420010541
25. Cohen, S., Keidar, I., Spiegelman, A.: Not a coincidence: Sub-quadratic asynchronous byzantine agreement WHP. In: Attiya, H. (ed.) 34th International Symposium on Distributed Computing, DISC 2020, Virtual Conference, October 12-16, 2020. LIPIcs, vol. 179, pp. 25:1–25:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPICS.DISC.2020.25, https://doi.org/10.4230/LIPIcs.DISC.2020.25
26. Cooper, J., Linial, N.: Fast perfect-information leader-election protocols with linear immunity. Comb. **15**(3), 319–332 (1995). https://doi.org/10.1007/BF01299739, https://doi.org/10.1007/BF01299739
27. Cramér, H.: On the order of magnitude of the difference between consecutive prime numbers. Acta Arithmetica **2**, 23–46 (1936)
28. Crites, E.C., Komlo, C., Maller, M.: Fully adaptive schnorr threshold signatures. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14081, pp. 678–709. Springer (2023). https://doi.org/10.1007/978-3-031-38557-5_22, https://doi.org/10.1007/978-3-031-38557-5_22
29. Das, S., Ren, L.: Adaptively secure BLS threshold signatures from DDH and co-cdh. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 14926, pp. 251–284. Springer (2024). https://doi.org/10.1007/978-3-031-68394-7_9, https://doi.org/10.1007/978-3-031-68394-7_9
30. Dwork, C., Peleg, D., Pippenger, N., Upfal, E.: Fault tolerance in networks of bounded degree. SIAM J. Comput. **17**(5), 975–988 (1988). https://doi.org/10.1137/0217061, https://doi.org/10.1137/0217061
31. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: STOC. pp. 148–161. ACM (1988)
32. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997). https://doi.org/10.1137/S0097539790187084, https://doi.org/10.1137/S0097539790187084
33. Feng, H., Lu, Z., Tang, Q.: Dragon: Decentralization at the cost of representation after arbitrary grouping and its applications to sub-cubic DKG and interactive consistency. In: PODC. pp. 469–479. ACM (2024)
34. Feng, H., Mai, T., Tang, Q.: Balanced and adaptively secure asynchronous common coin and byzantine agreement with sub-quadratic communication. In: Olivetti, D., Ruppert, E., Ovens, S. (eds.) Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2026, Egham, United Kingdom, July 6-10, 2026. pp. 271–281. ACM (2026). https://doi.org/10.1145/3796701.3815954, https://doi.org/10.1145/3796701.3815954
35. Feng, H., Tang, Q.: Asymptotically optimal adaptive asynchronous common coin and DKG with silent setup. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part III. Lecture Notes in Computer Science, vol. 16002, pp. 642–675. Springer (2025). https://doi.org/10.1007/978-3-032-01881-6_20, https://doi.org/10.1007/978-3-032-01881-6_20
36. Fischer, M.J., Lynch, N.A.: A lower bound for the time to assure interactive consistency. Inf. Process. Lett. **14**(4), 183–186 (1982)
37. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 33–62. Springer (2018). https://doi.org/10.1007/978-3-319-96881-0_2, https://doi.org/10.1007/978-3-319-96881-0_2
38. Gradwohl, R., Vadhan, S.P., Zuckerman, D.: Random selection with an adversarial majority. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference,

Santa Barbara, California, USA, August 20-24, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4117, pp. 409–426. Springer (2006). https://doi.org/10.1007/11818175_25, https://doi.org/10.1007/11818175_25
39. King, V., Lonargan, S., Saia, J., Trehan, A.: Load balanced scalable byzantine agreement through quorum building, with full information. In: Aguilera, M.K., Yu, H., Vaidya, N.H., Srinivasan, V., Choudhury, R.R. (eds.) Distributed Computing and Networking - 12th International Conference, ICDCN 2011, Bangalore, India, January 2-5, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6522, pp. 203–214. Springer (2011). https://doi.org/10.1007/978-3-642-17679-1_18, https://doi.org/10.1007/978-3-642-17679-1_18
40. King, V., Saia, J.: Breaking the $O(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. J. ACM **58**(4), 18:1–18:24 (2011). https://doi.org/10.1145/1989727.1989732, https://doi.org/10.1145/1989727.1989732
41. King, V., Saia, J., Sanwalani, V., Vee, E.: Scalable leader election. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006. pp. 990–999. ACM Press (2006), http://dl.acm.org/citation.cfm?id=1109557.1109667
42. Maram, S.K.D., Zhang, F., Wang, L., Low, A., Zhang, Y., Juels, A., Song, D.: CHURP: dynamic-committee proactive secret sharing. In: CCS. pp. 2369–2386. ACM (2019)
43. Moorhouse, G.E.: Bruck nets, codes, and characters of loops. Des. Codes Cryptogr. **1**(1), 7–29 (1991). https://doi.org/10.1007/BF00123956, https://doi.org/10.1007/BF00123956
44. Moorhouse, G.E.: Incidence geometry. University of Wyoming pp. 16–17 (2007)
45. Mostéfaoui, A., Raynal, M.: Signature-free asynchronous byzantine systems: From multivalued to binary consensus with t < n/3, o(n2) messages, and constant time. In: SIROCCO. pp. 194–208. Lecture Notes in Computer Science, Springer (2015)
46. Rabin, M.O.: Randomized byzantine generals. In: FOCS. pp. 403–409. IEEE Computer Society (1983)
47. Ryser, H.J.: Combinatorial Mathematics. Carus Mathematical Monographs, Mathematical Association of America (1963)
48. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979). https://doi.org/10.1145/359168.359176, https://doi.org/10.1145/359168.359176
49. Tassa, T., Dyn, N.: Multipartite secret sharing by bivariate interpolation. J. Cryptol. **22**(2), 227–258 (2009)
50. Yurek, T., Xiang, Z., Xia, Y., Miller, A.: Long live the honey badger: Robust asynchronous DPSS and its applications. In: USENIX Security Symposium. pp. 5413–5430. USENIX Association (2023)
51. Zamani, M., Movahedi, M., Raykova, M.: Rapidchain: Scaling blockchain via full sharding. In: CCS. pp. 931–948. ACM (2018)

A Related Work

Table 1 and 2 provide comparisons with the most relevant prior asynchronous common coin and Byzantine agreement schemes with (sub-)quadratic communication cost and optimal or near-optimal resilience. We detail the existing protocols in the following.

Existing Balanced and Efficient BA Protocols. We first look at a few existing balanced sub-quadratic BA protocols in the synchronous setting. King and Saia [40] show a synchronous BA with $\widetilde{O}(\sqrt{n})$ per-party communication complexity against weakly adaptive adversaries, assuming secure erasure and secure channels. However, their technique in the a.e.BA-to-BA reduction is not applicable to the asynchronous setting. Specifically, Step (4) of their Algorithm 3 requires bounded-round delivery in the synchronous setting to support a counting argument. This clearly relies on the synchronous assumption that ensures honest nodes can deliver messages to all other honest nodes within one round. Therefore, it is not clear if their technique can be adapted to the asynchronous setting.

Boyle et al. [18] show a synchronous BA with $\widetilde{O}(\text{polylog } n)$ per-party communication complexity against static adversaries. Their reduction from the succinctly reconstructed distributed signatures (SRDS) to BA requires the almost-everywhere Byzantine agreement (a.e. BA) protocol of [41]. Introduced by Dwork et al. [30], the notion of a.e. BA requires not all but almost all correct processors to come to agreement. However, the a.e. BA in [41] has a vulnerable component against adaptive adversaries: the leader election required by the a.e. BA protocol. As [40] points out, the adaptive adversary can simply wait until a small

set is elected and then can take over all processors in that set. This also motivates [40] to develop new techniques to ensure adaptive security in the synchronous setting, although not applicable to the asynchronous setting.

On the other hand, in the regime of the full-information model where the adversary can observe all messages sent by honest nodes, King et al. [39] show a synchronous BA with $\widetilde{O}(\sqrt{n})$ per-party communication complexity against static adversaries, assuming authenticated channels. However, their technique is also not applicable to the asynchronous setting. Specifically, their technique uses an averaging sampler to form n quorums of size $O(\log n)$ to ensure with high probability all n quorums are good where a good quorum has an honest majority. The same attack above applies here, allowing the adaptive adversary to simply wait until a small quorum is known and then can take over the majority of it.

Weakly Adaptive Secure Asynchronous Byzantine Agreement and Common Coin Protocols with (Sub-)quadratic Total Communication Complexity. Next, we turn to existing asynchronous Byzantine agreement (ABA) and common coin protocols with (sub-)quadratic communication cost. Note that any sub-quadratic protocol that is also balanced will automatically achieve $o(n)$-balancedness (per-party communication complexity) due to a simple pigeonhole argument. Cachin et al. [23] show that under a private threshold setup, both ABA and common coin are possible with $O(\lambda n^2)$ communication against weakly adaptive adversaries and achieve optimal resilience of $f < n/3$. Bandarupalli et al. [12] show that under secure channels, an asynchronous random beacon can be achieved with amortized $O(\lambda\gamma n^2 \log n)$ communication cost against weakly adaptive adversaries with $f < n/3$. Their construction is hash-based but they assume a secure communication channel. If we consider a one-shot coin usage, the concrete cost will see an $O(n)$ multiplier because the cost is amortized over a batch of $\Theta(n)$ beacons. Blum et al. [16] show that **unbounded** numbers of sub-quadratic ABA and common coin can be achieved with a fixed-size trusted setup against weakly adaptive adversaries. To justify the necessity of trusted setups for weakly adaptive security, they also show an $\Omega(n^2)$ lower bound result if the protocol is setup-free. However, both their (unbounded) ABA and common coin can tolerate up to $f = (1/3 - \epsilon)n$ faults, presenting a gap of ϵn faults from $n/3$. One may conjecture that with a trusted setup, there exists sub-quadratic ABA and asynchronous common coin that tolerate exactly $f = \lfloor n/3 \rfloor$ faults. Therefore, if the above conjecture holds, $1/3 - \epsilon$ is only near-optimal. We remark that this ϵ gap is hard to remove since [16] achieves the sub-quadratic communication complexity at the cost of sub-sampling the groups with tight intra-group resilience constraints $f/n = 1/3$ and sampling inevitably incurs tolerance reduction. Moreover, to generate an **unbounded** number of common coins, one needs to refresh the trusted setup by a computationally heavy MPC that requires threshold (compact) fully homomorphic encryption (TFHE) and UC-NIZK. Concurrently, Cohen et al. [25] show an ABA protocol with $o(n^2)$ communication against weakly adaptive adversaries that is resilient to up to f faults where $n \gtrsim 4.45f$. Moreover, they consider a non-standard asynchronous model in which the adversary cannot arbitrarily schedule delivery of messages. In particular, the adversary in their model cannot reorder messages sent by honest parties in the same protocol step.

Strongly Adaptive Secure Asynchronous Common Coin and Distributed Key Generation Protocols with Near-optimal Total Communication Complexity. More recently, asynchronous common coin and distributed key generation (DKG) protocols with *strongly* adaptive security have received more attention, where the strongly adaptive adversary can retract messages sent by a newly corrupted node before they are delivered, a so-called "after-the-fact-removal" attack. Feng and Tang [35] show an asynchronous common coin and DKG protocol with $O(\lambda n^2)$ communication complexity and $O(1)$ rounds, requiring a registered PKI setup with a key curator and a public silent setup. Abraham et al. [3] show an asynchronous DKG

protocol with $O(\lambda^2 n^2 \log^c n)$ communication complexity and $O(\log n)$ rounds, secure against strongly adaptive adversaries and requiring a trusted setup for verifiable random functions (VRFs). From the point of view of strongly adaptive secure asynchronous common coin, these protocols achieve the near-optimal total communication complexity.

Bivariate Polynomial-Based Secret Sharing Schemes. Prior to our work, bivariate polynomials have served as an essential tool for designing secret sharing schemes with advanced properties, particularly verifiability and recoverability. In the seminal work by Ben-Or, Goldwasser, and Wigderson [14], bivariate polynomials are employed to realize verifiable secret sharing (VSS) in the information-theoretic setting. At a high level, a dealer shares a secret s using a symmetric bivariate polynomial $F(X, Y)$, such that $F(0, 0) = s$ and $F(i, j) = F(j, i)$, with each party P_i receiving the univariate polynomial $F(X, i)$. Receivers can then verify share consistency by cross-checking their respective evaluations $F(j, i)$ and $F(i, j)$.

Another important application is asynchronous complete secret sharing (ACSS) [5], which guarantees that all honest receivers will obtain correct shares if any one of them successfully terminates. In these schemes, bivariate polynomials assist in share recovery: a share $F(i, 0)$ can be reconstructed by P_i once a sufficient number of parties P_j send $F(i, j)$ to P_i. Similarly, proactive secret sharing (PSS) schemes, such as CHURP [42] and Long-Lived HoneyBadger [50], employ this strategy to reshare a secret to a new committee, ensuring all honest members reliably obtain valid shares. Furthermore, bivariate polynomials have been utilized to realize various access structures, as demonstrated by Tassa and Dyn [49].

In contrast, our scheme employs bivariate polynomials for a fundamentally different purpose beyond verifiability and recoverability. Specifically, we leverage the two-phase recovery process naturally inherent in bivariate polynomial secret sharing and combine it with a grouping strategy to significantly reduce communication complexity. This novel application of bivariate polynomials may be of independent technical interest.

B Reduction from classic UF-CMA to UF-CMA-Community

Definition 12 (Unforgeability Under Chosen Message Attack).
Let $\mathsf{siTS} = (\mathsf{siTS.PublicParamSetup}, \mathsf{siTS.KGen}, \mathsf{siTS.PSign}, \mathsf{siTS.PVer}, \mathsf{siTS.Combine}, \mathsf{siTS.Ver})$ *be a semi-interactive* (n, t)*-threshold signature scheme; we write* $\Sigma := \mathsf{siTS}$ *for brevity. For an algorithm* $\mathcal{A}$*, define the game UF-CMA*$^{\mathcal{A}}_{\Sigma,t,\mathcal{X}}$ *in Fig. 4.*

Fig. 4. The unforgeability security game UF-CMA$^{\mathcal{A}}_{\Sigma,t,\mathcal{X}}$

Game UF-CMA$^{\mathcal{A}}_{\Sigma,t,\mathcal{X}}$:

1 : $\mathcal{X} \to \{\mathsf{CMNT}_j\}_{j\in[m]}; \mathsf{PublicParamSetup}() \to pp$
2 : $\mathsf{KGen}(pp) \to pk, \{pk_j\}_{j\in[m]}$,
3 : $\{pk_{(j,i')}\}_{j\in[m],i'\in\mathsf{CMNT}_j}$,
4 : $\{sk_{(j,i)}\}_{j\in[m],\, i\in\mathsf{CMNT}_j}; \mathcal{C} := \emptyset, \mathcal{H} := [n]$;
5 : $\mathrm{inp} := pp, pk, \{pk_j\}_{j\in[m]}, \{pk_{(j,i')}\}_{j\in[m],i'\in\mathsf{CMNT}_j}$
6 : // $Q[m]$, initially {}, denotes the set of signers
7 : $\mathcal{A}$ queries for partial signatures on m
8 : $(m, \sigma) \leftarrow \mathcal{A}^{\mathrm{CORR},\ \mathrm{PSIG},\ \mathrm{RO}}(\mathrm{inp})$
9 : **if** $|Q[m] \cup \mathcal{C}| \le t \wedge \mathsf{Ver}(pk, m, \sigma) = 1$: **return** 1
10 : **return** 0

Oracle CORR(i) :

1 : **if** $|\mathcal{C}| \ge t$: **return** $\perp$
2 : $\mathcal{C} := \mathcal{C} \cup \{i\}; \quad \mathcal{H} := \mathcal{H} \setminus \{i\}$
3 : **return** $\{sk_{(j,i)}\}_{j\in\mathsf{CMNT}^{-1}[i]}$

Oracle PSIG(j, i, m) :

1 : **if** $i \notin \mathcal{H} \vee i \notin \mathsf{CMNT}_j$: **return** $\perp$
2 : $Q[m] := Q[m] \cup \{i\}$
3 : $\sigma_{(j,i)} \leftarrow \mathsf{PSign}(sk_{(j,i)}, m)$
4 : **return** $\sigma_{(j,i)}$

Oracle RO(m) :

1 : **if** $H[m] = \perp$: $H[m] \leftarrow\!\!\$\ \mathbb{G}$
2 : **return** $H[m]$

- ***Setup Phase.*** *The community allocation strategy* $\mathcal{X} = \{\mathsf{CMNT}_j\}_{j\in[m]}$ *is an input to the setup phase. A trusted party runs* $\mathsf{siTS.PublicParamSetup}(1^\lambda)$ *to generate public parameters* pp *and* $\mathsf{siTS.KGen}(\mathsf{pp})$ *to generate public and secret key shares* $\{pk_j \mid j \in [m]\}$, $\{pk_{(j,i)} \mid j \in [m], i \in \mathsf{CMNT}_j\}$ *and* $\{sk_{(j,i)} | i \in [n], j \in \mathsf{CMNT}^{-1}[i]\}$. *The public parameters* pp *and public key shares are published and the secret key shares* $\{sk_{(j,i)} | j \in \mathsf{CMNT}^{-1}[i]\}$ *are privately distributed to* P_i. *The input* $\mathcal{X}$ *is included in the public parameters* pp *as the community allocation strategy. Initialize sets* $\mathcal{H} := \{1, \ldots, n\}$, $\mathcal{C} := \emptyset$, $\mathcal{Q}_{\mathsf{sign}} := \emptyset$. *Run* $\mathcal{A}$ *on input par.*
- ***Corruption Queries.*** *At any point of the experiment,* $\mathcal{A}$ *may corrupt a party* P_i *by submitting an index* i. *In this case, return the internal state of* P_i *and set* $\mathcal{H} = \mathcal{H} \setminus \{i\}$, $\mathcal{C} = \mathcal{C} \cup \{i\}$. *Henceforth,* $\mathcal{A}$ *has full control over* P_i.
- ***Online Phase.*** *Here,* $\mathcal{A}$ *gets additional access to oracles that answer queries of the following types:*
  - ***Signing Queries.*** *When* $\mathcal{A}$ *submits a tuple* (j, i, m) *for* $i \in \mathcal{H}$ *and* $j \in \mathsf{CMNT}^{-1}[i]$, *return* $\sigma_{(j,i)} \leftarrow \mathsf{siTS.PSign}(sk_{(j,i)}, m)$ *where* $sk_{(j,i)}$ *denotes the partial secret key of party* P_i *in the* j*-th community. Record the tuple* (j, i, m) *in a set* $\mathcal{Q}_{\mathsf{sign}} = \mathcal{Q}_{\mathsf{sign}} \cup \{(j, i, m)\}$.
  - ***Random Oracle Queries.*** *When* $\mathcal{A}$ *submits a query* m, *check if* $H[m] = \perp$ *and if so, set* $H[m] \leftarrow\!\!\$\ \mathbb{G}$. *Return* $H[m]$.
- ***Output Determination.*** *Let* $\mathcal{S}$ *denote the subset of parties for which* $\mathcal{A}$ *has ever made a signing query of the form* (j, i, m^*) *for some* j, *i.e.,* $\mathcal{S} := \{i \mid \exists j \in \mathsf{CMNT}^{-1}[i], (j, i, m^*) \in \mathcal{Q}_{\mathsf{sign}}\}$. *When* $\mathcal{A}$ *outputs a message* m^* *and a signature* σ^*, *output 1 if* $|\mathcal{C} \cup \mathcal{S}| \leq t$ *and* $Ver(pk, m^*, \sigma^*) = 1$. *Otherwise, output 0.*

We say that Σ *is* (ϵ, T, q_h, q_s)-unforgeable under chosen message attacks (UF-CMA) *if for all algorithms* $\mathcal{A}$ *running in time at most* T, *making at most* q_h *random oracle queries, and making at most* q_s *signing queries,* $\Pr[UF\text{-}CMA^{\mathcal{A}}_{\Sigma,t,\mathcal{X}} = 1] \leq \epsilon$. *Conversely, we say that* $\mathcal{A}$ (ϵ, T, q_h, q_s)*-breaks unforgeability of* Σ *under chosen message attacks if it runs in time at most* T, *makes at most* q_h *queries to the random oracle, makes at most* q_s *queries to the signing oracle, and* $\Pr[UF\text{-}CMA^{\mathcal{A}}_{\Sigma,t,\mathcal{X}} = 1] > \epsilon$.

Definition 13 (Robustness). *Let* siTS *be a semi-interactive* (n, t)*-threshold signature scheme. Given any input message* pt, siTS *is robust if, whenever up to* t *parties are adaptively compromised, every remaining uncompromised party* P_i *eventually outputs the same* σ *such that* $\mathsf{siTS.Ver}(\mathsf{pk}, pt, \sigma) = 1$.

Under a specifically designed community allocation strategy $\mathcal{X}$, we can reduce the classic unforgeability game $\text{UF-CMA}^{\mathcal{B}}_{\Sigma,t,\mathcal{X}}$ to our unforgeability game $\text{UF-CMA-Community}^{\mathcal{A}}_{\Sigma,\alpha,\beta,\mathcal{X}}$ (Def. 10) where our adversary $\mathcal{A}$ specifies corrupt queries differently, while it is at least as strong as $\mathcal{B}$ that can corrupt t individual parties. $\mathcal{A}$ no longer corrupts individual P_i by an index $i \in [n]$; instead, it specifies a (j, i) pair only to learn the partial secret key share $sk_{(j,i)}$. To make $\mathcal{A}$ as strong as $\mathcal{B}$, $\mathcal{A}$ is allowed to query all mq pairs except the restriction enforced by the community allocation strategy that says there always exists an $(m-\alpha)$-subset $\mathcal{H}_{\mathsf{CMNT}} \subset [m]$ of communities, such that the adversary can corrupt at most β partial secret key shares in each community in $\mathcal{H}_{\mathsf{CMNT}}$. This restriction is necessary to ensure the robustness of the community structure can be maintained in the face of up to t corruptions. We show in Lemma 17 that this reduction is valid.

Lemma 17 (Reduction from UF-CMA to UF-CMA-Community). *Let* $\mathcal{X}$ *be a community allocation strategy partitioning* n *parties into* m *communities, each of size* q, *with* $m, q \in \Theta(\sqrt{n})$, $m \geq 2\alpha + 1$, *and* $q \geq 2\beta + 1$. *For any* $\mathcal{X}$ *that has at most* α *communities containing more than* β *corrupted members, let* Σ *be a semi-interactive* $(n, m, \alpha, q, \beta, \mathcal{X})$*-threshold*

signature scheme. We have $\Pr[\textit{UF-CMA}^{\mathcal{B}}_{\Sigma,t,\mathcal{X}} = 1] \leq \Pr[\textit{UF-CMA-Community}^{\mathcal{B}}_{\Sigma,\alpha,\beta,\mathcal{X}} = 1]$. *Moreover, if* Σ *is* (ϵ, T, q_h, q_s)*-unforgeable under chosen message attacks (UF-CMA, Def. 12) and* $(\widetilde{\epsilon}, \widetilde{T}, \widetilde{q_h}, \widetilde{q_s})$*-unforgeable under chosen message attacks in communities (UF-CMA-Community, Def. 10), then we have* $\epsilon \leq \widetilde{\epsilon}, T \geq \widetilde{T}/n, q_h \geq \widetilde{q_h}, q_s \geq \widetilde{q_s}$.

Proof (Lemma 17). We construct a reduction $\mathcal{R}$ from the unforgeability game $\text{UF-CMA}^{\mathcal{B}_{\text{normal}}}_{\Sigma,t,\mathcal{X}}$ to the unforgeability game $\text{UF-CMA-Community}^{\mathcal{R}}_{\Sigma,\alpha,\beta,\mathcal{X}}$. $\mathcal{R}$ works as follows:

- **Setup Phase.** $\mathcal{R}$ runs the setup phase of Game $\text{UF-CMA-Community}_{\Sigma,\alpha,\beta,\mathcal{X}}$. $\mathcal{R}$ prepares the same setup for simulation of Game $\text{UF-CMA}_{\Sigma,t,\mathcal{X}}$.
- **Corruption Queries.** $\mathcal{R}$ handles the corruption queries of $\mathcal{B}_{\text{normal}}$ in the simulation as follows: When $\mathcal{B}_{\text{normal}}$ makes a corruption query for P_i in the simulation, $\mathcal{R}$ corrupts (j, i) for all $j \in \mathsf{CMNT}^{-1}[i]$ in Game UF-CMA-Community and returns the internal state of P_i to $\mathcal{B}_{\text{normal}}$ in the simulation.
- **Online Phase.** $\mathcal{R}$ simulates the online phase of $\mathcal{B}_{\text{normal}}$ in the simulation as follows: When $\mathcal{B}_{\text{normal}}$ makes a signing query (j, i, m) in the simulation, $\mathcal{R}$ makes the same query to its own signing oracle and passes the signature $\sigma_{(j,i)}$ to $\mathcal{B}_{\text{normal}}$ in the simulation. When $\mathcal{B}_{\text{normal}}$ makes a random oracle query m, $\mathcal{R}$ passes the query to its own random oracle and returns the result to $\mathcal{B}_{\text{normal}}$ in the simulation.
- **Output Determination.** $\mathcal{R}$ returns the same forgery as $\mathcal{B}_{\text{normal}}$ in the simulation.

We argue that $\mathcal{R}$ wins Game UF-CMA-Community if $\mathcal{B}_{\text{normal}}$ wins Game UF-CMA in simulation. If $\mathcal{B}_{\text{normal}}$ wins Game UF-CMA, then by definition $|\mathcal{C} \cup \mathcal{S}| \leq t$ and $\text{Ver}(pk, m^*, \sigma^*) = 1$. Since $\mathcal{S} := \{i \mid \exists j, (j, i, m^*) \in \mathcal{Q}_{\mathsf{sign}}\}$, this means $\mathcal{B}_{\text{normal}}$ makes corruption or signing queries for at most t individual parties. By the lemma assumption, there always exist $m - \alpha \geq \alpha + 1$ communities, such that each of these communities contains at most β parties from S. Therefore, $\mathcal{R}$ can successfully simulate all corruption and signing queries of $\mathcal{B}_{\text{normal}}$ while satisfying the corruption restriction that allows $\mathcal{R}$ to make corruption and signing queries for every pair $(j, i) \in \{(j, i) \mid j \in [m], i \in \mathsf{CMNT}_j\}$ except that $\mathcal{R}$ can corrupt at most β partial secret key shares in each of these $m - \alpha$ communities. Since, $\text{Ver}(pk, m^*, \sigma^*) = 1$, by the definition of UF-CMA-Community, $\mathcal{R}$ wins Game UF-CMA-Community.

It is not hard to see $\mathcal{R}$ simulates the game UF-CMA perfectly, and $\mathcal{R}$ can correctly answer $\mathcal{B}_{\text{normal}}$'s queries including corruption and signing queries for at most t parties, under the assumption. We argue that $\mathcal{B}_{\text{normal}}$'s view is indistinguishable from its view in a real game. Therefore, the lemma is proved. □