Consensus through Herding

T-H. Hubert Chan* Rafael Pass† Elaine Shi‡

Abstract

State Machine Replication (SMR) is an important abstraction for a set of nodes to agree on an ever-growing, linearly-ordered log of transactions. In decentralized cryptocurrency applications, we would like to design SMR protocols that 1) resist adaptive corruptions; and 2) achieve small bandwidth and small confirmation time. All past approaches towards constructing SMR fail to achieve either small confirmation time or small bandwidth under adaptive corruptions (without resorting to strong assumptions such as the erasure model or proof-of-work).

We propose a novel paradigm for reaching consensus that departs significantly from classical approaches. Our protocol is inspired by a social phenomenon called herding, where people tend to make choices considered as the social norm. In our consensus protocol, leader election and voting are coalesced into a single (randomized) process: in every round, every node tries to cast a vote for what it views as the *most popular* item so far: such a voting attempt is not always successful, but rather, successful with a certain probability. Importantly, the probability that the node is elected to vote for v is independent from the probability it is elected to vote for $v' \neq v$. We will show how to realize such a distributed, randomized election process using appropriate, adaptively secure cryptographic building blocks.

We show that amazingly, not only can this new paradigm achieve consensus (e.g., on a batch of unconfirmed transactions in a cryptocurrency system), but it also allows us to derive the first SMR protocol which, even under adaptive corruptions, requires only polylogarithmically many rounds and polylogarithmically many honest messages to be multicast to confirm each batch of transactions; and importantly, we attain these guarantees under standard cryptographic assumptions.

1 Introduction

State Machine Replication (SMR), also called consensus, is a core abstraction in distributed systems [5, 21, 25]: a set of nodes would like to agree on a linearly ordered log of transactions (e.g., in a public ledger or decentralized smart contract application), such that two important security properties, *consistency* and *liveness*, are satisfied. Loosely speaking, consistency requires that all honest nodes' logs are prefixes of one another and that no node's log will ever shrink; and liveness requires that if a client submits a transaction, the transaction will appear in every honest node's log in a bounded amount of time.

The classical literature on distributed systems typically considers deployment of consensus in a single organization (e.g., Google or Facebook), and on a small scale (e.g., a dozen nodes). Typically these nodes are connected through fast, local-area network where bandwidth is abundant. Thus the classical consensus literature typically focuses on optimizating the protocol's *round complexity*

*University of Hong Kong. This research was partially done in a consultancy agreement with Thunder Research. hubert@cs.hku.hk

†CornellTech and Thunder Research, rafael@cs.cornell.edu

‡Cornell and Thunder Research, elaine@cs.cornell.edu

which is directly related to the confirmation time — it is well-known that we can design consensus protocols where confirmation happens in expected constant rounds (i.e., independent of the number of players) even in the presence of adaptive corruptions [10, 14].

In the past decade, due to new blockchain systems such as Bitcoin and Ethereum, SMR protocols have been deployed in a decentralized setting in an open network. In such a blockchain setting, we typically have a *large* number n of nodes who communicate over a diffusion network (where nodes *multicast* messages to the whole network); and it is simply not practical to have protocols where the number of messages to be multicast grows linearly with the number of nodes. [Elaine: NOTE: changed defn, check consistency later.] In this paper, we care about achieving SMR in a *communication-efficient* way: we want both the confirmation time and the number of bits multicast (to confirm each batch of transactions) to be polylogarithmic (or even just sublinear) in the number of nodes. More precisely, we refer to an n-party protocol as being communication efficient if the total number the bits multicast is $o(n) \cdot |\mathsf{TXs}| \cdot \kappa$, and the confirmation time is $o(n) \cdot \kappa$ where κ is a security parameter such that the protocol's security must be respected except with negligible in κ probability. Achieving communication efficiency under *static* security is easy: one could randomly elect a small committee of $\mathsf{poly} \log \kappa$ size and next run any SMR protocol that may have polynomial bandwidth overhead to confirm a batch of transactions. If the committee election is random and independent of the choice of corrupt nodes, then except with negligible in κ probability, the committee's corrupt fraction approximates the overall corrupt fraction due to the Chernoff bound. Moreover, under honest majority assumptions, non-committee members can always be convinced of a decision as long as it is vouched for by the majority of the committee.

However, in typical blockchain applications (such as cryptocurrencies where the participating nodes are on an open network), the static corruption model is insufficient for security. Rather, we need to protect the protocol against *adaptive* corruptions, where an attacker may, based on the protocol execution so far, select which parties to attack. The above-mentioned "naïve" committee election approach miserably fails in the presence of adaptive corruptions: the adversary can always corrupt all committee members after having observed who they are and completely violate the security of the protocol. Indeed, obtaining a communication-efficient SMR protocol which withstands adaptive corruptions has been a long-standing open problem:

Does there exists a communication-efficient SMR protocol that withstands adaptive corruptions?

Nakamoto's beautiful blockchain protocol (through its analysis in [11, 19, 20]) was the first protocol to achieve communication-efficient SMR with adaptive security. This protocol, however, requires using proofs of work [9], and in particular requires honest players to "waste" as much computation as the total computational power of adversarial players. Consequently, in recent years, the research community has focused on removing the use of proofs-of-work and instead rely on standard bounds on the fraction of adversarial players (e.g., honest majority). In particular, the recent work by Chen and Micali [6] (see also David et al. [8], and Pass and Shi [23]) demonstrates communication-efficient SMR protocols with adaptive security without the use of proof-of-work in the so-called *erasures model*: in the erasure model, we assume that honest players have the ability to *completely* erase/dispose of some some parts of their local state (such that if some player later on gets corrupted, the erased state cannot be recovered by the attacker). However, as discussed in Canetti et al [4] (and the references therein) such erasures are hard to perform in software, without resorting to physical intervention, and the security of the heuristics employed in practice are not well understood. As such, solutions in the erasure model may not provide an adequate level of security and thus ideally, we would like to avoid the use of strong erasure assumptions.

In this work, we focus on the design of communication-efficient SMR protocols without proof-of-work and without assuming the possibility of erasures. As far as we know, the design of such

protocols is open even in the PKI model and even if assuming, say, 99% of the nodes are honest. We remark that very recently, a communication-efficient "single-shot" version of consensus, referred to as "multi-value agreement" (MVA), was achieved by Abraham et al. [2]; as we discuss in detail in Section 1.2, the validity conditions for MVA is much weaker and thus it is not clear how to extend these protocols to SMR.

1.1 Our Results

We propose the first communication-efficient SMR protocol with adaptive security (without assuming erasures or proof-of-work), solving the above-mentioned open problem. Our protocol works in a public-key-infrastructure (PKI) model, assuming a synchronous network, and standard cryptographic assumptions. The protocol tolerates $\frac{1}{3} - \epsilon$ fraction of adaptive corruptions where ϵ is an arbitrarily small constant, and moreover to achieve a failure probability that is negligible in the security parameter κ, every transaction tx gets confirmed in $\mathsf{poly}\log\kappa \cdot \Delta$ time where Δ is the maximum network delay, and requiring at most $|\mathsf{tx}| \cdot \mathsf{poly}\log\kappa$ bits of honest messages to be multicast (assuming that $|\mathsf{tx}|$ is at least as large as a suitable computational security parameter).

Theorem 1.1 (Adaptively secure, communication efficient synchronous state machine replication)**.** *Under standard cryptographic hardness assumptions (more precisely, assuming standard bilinear group assumptions), there exists a synchronous state machine replication protocol which, except with negligible in κ probability, satisfies consistency and confirms transactions in $\mathsf{poly}\log\kappa \cdot \Delta$ time where Δ is the maximum network delay — as long as the adversary corrupts no more than $\frac{1}{3} - \epsilon$ fraction of nodes. Moreover, (except with negligible in κ, χ probability) honest nodes only need to multicast $\mathsf{poly}\log\kappa \cdot (\chi + |\mathsf{TXs}|)$ bits of messages to confirm every batch of transactions denoted TXs where χ is a computational security parameter related to the strength of the cryptographic building blocks involved.*[1]

We remark that our communication complexity bound is asymtotically the same as that achieved by earlier protocols using either proofs of work or in the erasure model.

1.2 Technical Highlights

Why the problem is more challenging in SMR than in single-shot consensus. We stress that achieving communication efficiency under adaptive corruptions is more difficult in SMR than in single-shot consensus. In the latter, a designated sender aims to "broadcast", for once only, a (possibly multi-bit) value to everyone, retaining consistency even when the sender is corrupt, and achieving validity should the sender be honest. Henceforth we refer to the single-shot version as Multi-Valued Agreement (MVA). In MVA, if the sender is adaptively corrupt, typically no validity is necessary (or even attainable depending on the concrete definition). Thus it is not clear how to compose adaptively secure MVA to achieve adaptively secure SMR while preserving communication efficiency: if the "leader" who is supposed to broadcast the next block (of transactions) becomes corrupt, the adversary can cause a "bad block" to be confirmed (e.g., a block that censors some to all outstanding transactions). If only a small number of such "leaders" speak at any point of time, the adversary can continuously corrupt all leaders that speak until it has exhausted its corruption budget — such an attack will cause confirmation time to be large.

[1] If assuming subexponential security of the underlying cryptographic building blocks, χ can be set to $\mathsf{poly}\log\kappa$.

For this reason, we stress that *adaptively secure, communication-efficient MVA does NOT lead to adaptively secure, communication-efficient SMR* in any straightforward manner[2]. Also note that even for single-shot consensus, the only known solution that is communication efficient and does not assume erasures is the recent work by Abraham et al. [2].

Defining "batch agreement" with a quality metric. As mentioned, the validity definition in the classical notion of MVA is too weak if we wish to construct communication-efficient state machine replication. One contribution of our paper is to propose a new abstraction which we call "batch agreement" — on the surface it looks very much like MVA since nodes seek to agree on the next batch of transactions. However, batch agreement is defined with a *quality* metric that is lacking in the standard definition of MVA. We say that a block has good quality iff it contains all transactions that have been outstanding for a while; and our batch agreement notion requires that *a batch with good quality be chosen* even when the "leader" is adaptively corrupt (and upon corruption it can inject many blocks).

Constructing batch agreement. To understand the novelty of our approach, we first briefly review existing work. In classical approaches, if only a few number of leaders are elected to speak, all of them can be adaptively corrupt and made to propose multiple blocks in the same round. Even if the adversary is not fast enough to erase the good block that leader already proposed while still honest (i.e., just before it was adaptively corrupted), it can succeed in diverging honest nodes' voting efforts. For example, newly corrupt node may propose many good blocks and delivering them in different order to different honest nodes. At this moment, using classical techniques, it does not seem easy for the honest nodes to coordinate and vote on the same block. As a result, some classical approaches adopt an approach [18] where nodes jointly discover that no block has gained popular votes (e.g., by computing a grade), and then they initiate a binary agreement process to jointly decide to fall back to outputting a default value. Obviously the default value is pre-determined and cannot contain the set of outstanding transactions and thus does not have good quality.

Our approach departs from all known classical approaches: at the core we describe a new randomized process through which the network can jointly make a selection and converge to a good choice, when presented with polynomially many choices. During a batch agreement, a small set of nodes get elected to propose a block. All of these nodes may be adaptively corrupt and then made to propose more good or bad blocks. Thus honest nodes are faced with these polynomially many blocks to choose from, and moreover at any snapshot of time, the set of blocks observed by different honest nodes may differ, since the blocks do not necessarily arrive at the honest nodes at the same time.

To converge to a good choice, we start with an *initial* scoring function that is used to evaluate the quality of each block itself — basically a block that contains all sufficiently long outstanding transactions scores high, and a block that censors some of them will score very low. Since nodes may receive outstanding transactions at slightly different times, honest nodes may end up calculating different initial scores even for the same block — we carefully craft a initial score function to make sure that this difference is not too large as long as transactions are propagated to honest nodes around the same time (indeed, we show that this can be accomplished with small communication too).

Nodes then are randomly elected to vote on the blocks over time; the votes can serve to strengthen a block's score in an additive fashion. At any point of time, an honest node will always

[2]If communication efficiency is not a concern, we could have n broadcast instances (composed either sequentially or in parallel) where everyone is given the chance to act as the leader and suggest the next batch of transactions to confirm; we can then concatenate the outputs of these n broadcasts and treat it as the next block.

try to vote on the *most popular* block in its view (i.e., the one with the highest score), but the voting attempt only succeeds with somewhat small probability such that not too many people need to send votes. If a node is randomly elected to vote for some block B in some time step, it does not mean that it is eligible to vote for other blocks; thus adaptively corrupting a node that is elected to vote does not help the adversary. Since all honest nodes always choose the most popular item so far, honest nodes' voting efforts must be somewhat concentrated. After roughly polylogarithmically many steps, polylogarithmically many honest votes will have been cast. Although at any snapshot of time, nodes may never agree on the precise score or set of votes for each block, (except with negligible probability) it must be the case that everyone sees that the same highest-scoring block at the end, because its final score is significantly larger than the second-best choice. Finally, a block with an initial score that is too low will also not be selected (except with negligible probability) because it is too unlikely for it to ever collect enough votes (even when counting corrupt nodes' votes) to compete with the blocks with good quality.

With our approach, the ability to corrupt a leader on the fly and making it propose many additional blocks (on top of the good block it already proposed) does not help the adversary; nor does adaptively corrupting a voter help as mentioned.

2 Technical Roadmap

In this section, we begin by explaining our construction and proofs (of the primary building block) informally. We then give a more detailed comparison with related work.

2.1 Informal Description of our Protocol

At the core of our SMR construction is a new abstraction called "batch agreement". Every time a batch agreement instance is invoked, the nodes reach agreement on a set of transactions such that transactions that are sufficiently long-pending are guaranteed to be included (except with negligible probability). The entire SMR protocol simply runs multiple sequential instances of the batch agreement protocol.

As mentioned earlier, although on the surface it seems similar to the classical notion of Multi-Valued Agreement (MVA), our notion has a much stronger validity property that is lacking in classical MVA — specifically, we require that the confirmed block have good quality even when everyone who is randomly elected to speak is adaptively corrupt.

A herding-inspired protocol. Our protocol is inspired a social phenomenon called *herding* where people follow the popular social choice. We show how herding can be leveraged for reaching consensus. Recall that the adversary controls $\frac{1}{3} - \epsilon$ fraction. At a high level, nodes cast votes for batches of transactions over time. Imagine that in any round t, a node has a certain probability $p = \frac{1}{\lambda \Delta n}$ of being elected to vote for a particular batch TXs where λ is sufficiently large such that $\lambda = \omega(\frac{\log \kappa}{\epsilon^2})$ — henceforth if a node is elected to vote for TXs in round t, we say that the node "mines" a vote for TXs in round t. Importantly, the probability that a node mines a vote for TXs and round r is independent of its success probability for TXs' and round r' as long as $(\mathsf{TXs}, r) \neq (\mathsf{TXs}', r')$ — this is important for achieving adaptive security: if the adversary adaptively corrupts a node that has just cast a vote for TXs, corrupting this node does not make it more or less likely for the adversary to mine a vote for $\mathsf{TXs}' \neq \mathsf{TXs}$ in the same round than corrupting any other node.

In every round, an honest node would always pick the most popular batch (where popularity will be defined later) in its view, and it will only attempt to vote for this most popular batch — it will not cast a vote for any other batch even if it might be eligible. If a voting attempt for TXs

is successful in some round, the node multicasts the new vote as well as all existing votes it has seen observed for TXs to all other nodes. After some time, every node outputs a batch that has collected "ample" number of votes (where "ample" will be defined later in Section 2.2); if no such batch is found, output nothing.

Realizing "mining" with cryptography. So far in the above protocol, we did not fully specify how to realize the random eligibility election. As we explain later in the paper, this can be instantiated assuming a Verifiable Random Function (VRF) with appropriate *adaptive security* properties.

Assume that every player i has a VRF public key denoted pk_i that is common knowledge, and the corresponding VRF secret key sk_i is known only to player i. For i to determine its eligibility to vote for TXs in round r, it evaluates $(\mu, \pi) := \mathsf{VRF}(\mathsf{sk}_i, \mathsf{TXs}, r)$ where μ is the VRF evaluation outcome and π is a proof attesting to the evaluation outcome. If $\mu < D_p$ where D_p is an appropriate difficulty parameter, node i is deemed eligible to vote for TXs in round r. While only the secret-key owner can evaluate the VRF, anyone can verify the evaluation outcome. More specifically, any node that receives the tuple $(\mathsf{TXs}, r, \mu, \pi)$ can verify with pk_i that indeed μ is the correct VRF evaluation outcome and verify i's eligibility to vote for TXs in round r. Importantly, a vote received is only considered valid if its purported round is no greater than the current round number (this prevents corrupt nodes from mining into the future).

Later in Section 7, we will describe how to instantiate such an adaptively-secure VRF that satisfies our needs, using techniques from Abhraham et al. [2].

Popularity and initial score. It remains to specify how nodes determine the popularity of a batch TXs of transactions. The popularity is the sum of an *initial score* and the *number of valid votes collected* so far for TXs. To make sure that the protocol will preferentially select an all-inclusive batch TXs that omits no long-pending transaction, we design an initial score function that relies on a *discounting* mechanism to punish batches that omit long-pending transactions.

Specifically, we say that node i perceives the *age* of a transaction tx to be α, if at the start of the batch agreement protocol, exactly α rounds have elapsed since node i first observed tx. We assume that the underlying network medium satisfies the following "transaction diffusion" assumption: if any forever honest node observes a transaction tx in round r, then by round $r + \Delta$, all so-far honest nodes must have observed tx too[3]. In this way, we are guaranteed that the perceived age of a transaction tx must be somewhat consistent among all honest nodes. Now, imagine that the maximum initial score a batch can gain is $S_{\max}$ (to be parametrized later). We will discount the initial score of a batch TXs exponentially fast w.r.t. the oldest transaction that TXs omits. Specifically, imagine that node i computes the score of TXs as follows:

$$\mathsf{score}_i(\mathsf{TXs}) := S_{\max} \cdot \left(1 - \frac{1}{\lambda S_{\max}}\right)^{\frac{\alpha^*}{3\Delta}} \tag{1}$$

where α^* is the age (as perceived by node i) of the oldest transaction that is omitted from TXs. Given the transaction diffusion assumption, it is not difficult to see that every two so-far honest nodes' initial score difference for any batch TXs must be less than $\frac{1}{\lambda}$, i.e., honest nodes score every batch somewhat consistently.

[3]As discussed in the Supplemental Materials this assumption can be removed in a synchronous network while preserving communication efficiency.

2.2 Intuitive Analysis

We can now intuitively argue why such a herding-based protocol satisfies consistency and liveness under appropriate parameters. Imagine that the protocol is parametrized in the following way where λ is chosen such that $\epsilon^2\lambda = \omega(\log \kappa)$ — for example if ϵ is a(n arbitrarily small) constant, then λ may be any super-logarithmic function:

- in an all-honest execution, in expectation, every $\lambda\Delta$ rounds, some node mines a new vote. This means that each individual mining attempt is successful with probability $\frac{1}{\lambda\Delta n}$;
- the protocol is executed for $T_{\text{end}} := \lambda^2\Delta$ rounds, i.e., in an all-honest execution, in expectation a total of λ votes are mined; and
- at the end of the protocol, a node would only output a batch that has gained $\frac{2\lambda}{3}$ or more valid votes, i.e., a threshold of $\frac{2\lambda}{3}$ is considered ample.

Consistency. To argue consistency, it suffices to argue that any two different batches TXs and TXs' cannot both gain ample votes by T_{end}. This follows from the following observation: forever honest nodes make only a single mining attempt per round; while eventually corrupt nodes can make a mining attempt for TXs and one for TXs' corresponding for each round r (note that once corrupt, a node can retroactively make mining attempts for past rounds). Thus the total number of mining attempts made for either TXs or TXs' must be upper bounded by $\frac{4}{3} \cdot n \cdot T_{\text{end}}$. As mentioned earlier, adaptively corrupting a node that has just mined a vote for TXs does not increase the adversary's chance of mining a vote for $\mathsf{TXs}' \neq \mathsf{TXs}$ (for any round). Thus by Chernoff bound, we have that except with $\exp(-\Omega(\epsilon^2\lambda))$ probability (which is negligible in κ), the total number of successfully mined votes (including honest and adversarial) for TXs or TXs' must be strictly less than $\frac{4\lambda}{3}$ — this means that the two different batches cannot both have ample votes. To complete the argument, we need to take a union bound over all pairs of batches. If the adversary and all nodes are polynomially bounded, then the only batches we care about are those that appear in some honest node's view at some point in the execution. Since there are at most polynomially many such batches, the union bound has only polynomial loss.

Liveness. Liveness crucially relies on the fact that the mining difficulty is large enough, such that the average time till some node finds the next vote (set to be $\lambda\Delta$) is much larger than the maximum network delay Δ. Intuitively, this condition is necessary for honest nodes to "concentrate" their voting efforts on the same batch. Recall that honest nodes would score each batch somewhat consistently. This means that if a so-far honest node mines a vote for what he thinks is the most popular batch TXs — if the network delay is small, very soon all so-far honest nodes would find TXs the most popular batch too, and would mine votes only for TXs. As long as all forever honest nodes concentrate their mining efforts, by Chernoff bound some batch would attract ample votes and thus liveness ensues. On the other hand, if the network delay is large w.r.t. to the time it takes to mine a vote, honest nodes will be mining on different batches and likely no batch will gain enough votes at the end. We defer a formal argument to the later technical sections.

Validity. For validity, we would like to argue that any batch that honest nodes agree on cannot omit "long-pending" transactions. To see this, note that except with negligible in κ probability, the total number of valid votes any batch TXs can gain is at most 1.1λ. Now, if we let $S_{\max} := 3\lambda$, then any batch TXs that omits transactions of age $c\lambda^2\Delta$ or higher for an appropriate constant c must have an initial score less than 1.5λ as perceived by any honest node (recall that honest nodes

would always assign somewhat consistent scores to every batch). This means that no honest node should ever attempt to mine a vote for such a batch TXs; and thus TXs cannot gain ample votes.

2.3 Additional Related Work

In the past, the only known protocol that achieves both small bandwidth and small confirmation time under adaptive corruptions is the celebrated Nakamoto consensus protocol [11, 19, 20, 22], however, at the price of making very strong, idealized, proof-of-work assumptions. Constrained to making standard cryptographic assumptions, it is known how to construct adaptively-secure SMR that achieves either small confirmation time or small bandwidth, but not both.

First, if we allow many nodes to speak at any point of time (i.e., if we did not care about bandwidth consumption), we can easily construct protocols that achieve small round complexity. Specifically, it is easy to compose multiple instances of small-round MVA protocols [2, 10, 14] to attain SMR with small confirmation time (while retaining adaptive security). Basically, in every round, we can fork n instances of MVA where each node i acts as the designated sender in the i-th instance, and the log of the SMR is derived by concatenating all instances of all rounds, ordered first by the round and then by the instance within the round. However, even if the underlying MVA achieved small bandwidth [2], the derived SMR protocol would be expensive in bandwidth.

In a second class of approaches, we would like to have only a small number of players speak at any given point of time [1, 6–8, 15, 16, 23, 26] — this is in fact necessary to achieve our notion of communication efficiency. Past work has suggested multiple ways to construct such protocols:

- One possible approach [1, 7, 8, 15, 16, 23, 26], is inspired by Nakamoto's longest-chain protocol but removing the proof-of-work in a permissioned setting assuming a public-key infrastructure (PKI). Specifically, in such protocols, in every time slot, a node has a chance of being elected leader. When it is elected leader, it signs the next block extending the current longest chain. For such protocols to retain consistency and liveness [1, 7, 8, 15, 16, 23, 26], some additional constraints have to be imposed on the validity of timestamps contained in a blockchain. Among these works, some use a randomized leader election strategy [7, 16, 16, 23]; and some use a deterministic leader election process [1, 15, 26].

- Another approach, represented by Algorand [6] and improved in subsequent works [2, 18], is to rely on a classical-style consensus protocol, but in every round, randomly subsample a small, polylogarithmically size committee to cast votes (e.g., by employing a verifiable random function).

No matter which approach is taken, an adaptive adversary can continuously corrupt the small number of players selected to speak until it exhausts its corruption budget. Once corrupt, these players can cast ambiguous votes or propose equivocating blocks (e.g., those that censor certain transactions). In all of the above approaches (without assuming erasure), when such an adaptive-corruption attack is taking place, all blocks confirmed may have bad quality (e.g., censoring certain transactions), causing confirmation time to be at least $\Theta(n/s)$ where s denotes an upper bound on the number of players who speak in every round.

Communication-efficient single-shot consensus. The recent work by Abraham et al. [2] achieves adaptive security and communication efficiency without erasures or PoW, but their approach works only for MVA and does not extend, in any non-trivial fashion, to SMR. As mentioned sequential or parallel repetition of MVA fails to work for this purpose due to the much weaker validity requirement of MVA (see the Supplemental Materials for additional explanations). As

will be obvious soon, although our paper adopts the vote-specific committee election technique from Abraham et al. [2], we require vastly new techniques to simultaneously achieve both adaptive security and communication efficiency for SMR.

3 Protocol Execution Model

A protocol refers to an algorithm for a set of interactive Turing Machines (also called nodes) to interact with each other. The execution of a protocol Π that is directed by an environment $\mathcal{Z}(1^\kappa)$ (where κ is a security parameter), which activates a number of nodes as either *honest* or *corrupt* nodes. Honest nodes faithfully follow the protocol's prescription, whereas corrupt nodes are controlled by an adversary $\mathcal{A}(1^\kappa)$ which reads all their inputs/message and sets their outputs/messages to be sent.

A protocol's execution proceeds in *rounds* that model atomic time steps. At the beginning of every round, honest nodes receive inputs from an environment $\mathcal{Z}$; at the end of every round, honest nodes may send outputs to the environment $\mathcal{Z}$.

Corruption model. $\mathcal{Z}$ spawns n number of nodes upfront, a subset of which may be corrupt upfront, and the remaining are honest upfront. During the execution, $\mathcal{Z}$ may *adaptively* corrupt any honest node. When a node becomes corrupt, $\mathcal{A}$ gets access to its local state, and subsequently, $\mathcal{A}$ controls the corrupt node. Henceforth, at any time in the protocol, nodes that remain honest so far are referred to as *so-far honest* nodes; and nodes that remain honest till the end of the protocol are referred to as *forever honest* nodes[4].

Communication model. We assume that there is a function $\Delta(\kappa, n)$ that is polynomial in κ and n, such that every message sent by a so-far honest node in round r is guaranteed to be received by a so-far honest recipient at the beginning of round $r + \Delta$ (if not earlier). The adversary can delay honest message arbitrarily but up to Δ rounds at the maximum.

All of our protocols will work in the *multicast* model: honest nodes always send the same message M to everyone. We assume that when a so-far honest node i multicasts a message M in some round r, it can immediately become corrupt in the same round and made to send one or more messages in the same round. However, the message M that was already multicast before i became corrupt cannot be retracted, and all nodes that are still honest in round $r+\Delta$ will have received the message M. In our paper we will also account for a protocol's communication efficiency by upper bounding how many bits of honest messages must be multicast during the protocol. Any message that is sent by a so-far honest node is an honest message — but if the node becomes corrupt in the same round and sends another message in the same round, the latter message is treated as a corrupt message. Since corrupt nodes can send any polynomially many messages, we do not seek to bound corrupt messages.

In this paper we consider *synchronous* protocols where the protocol is parametrized with Δ, i.e., Δ is hard-wired in the protocol's description.

Notational convention. Protocol execution is assumed to be probabilistic in nature. We would like to ensure that certain security properties such as consistency and liveness hold for almost all execution traces, assuming that both $\mathcal{A}$ and $\mathcal{Z}$ are polynomially bounded.

Henceforth in the paper, we use the notation $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ to denote a sample of the randomized execution of the protocol Π with $\mathcal{A}$ and $\mathcal{Z}$, and security parameter $\kappa \in \mathbb{N}$. The randomness in

[4]Note that "forever honest" is in fact defined w.r.t. the protocol we are concerned with.

the experiment comes from honest nodes' randomness, $\mathcal{A}$, and $\mathcal{Z}$, each sampling of $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$ produces an *execution trace*. We would like that the fraction of execution traces that fail to satisfy relevant security properties be negligibly small in the security parameter κ. A function $\mathsf{negl}(\cdot)$ is said to be negligible if for every polynomial $p(\cdot)$, there exists some κ_0 such that $\mathsf{negl}(\kappa) \leq 1/p(\kappa)$ for every $\kappa \geq \kappa_0$.

Throughout the paper, we assume that n is a polynomial function in κ and Δ is a polynomial function in κ and n — we note in the most general setting, Δ may be dependent on n if, for example, the network layer builds some kind of diffusion tree or graph to propagate messages.

Definition 3.1 ((ρ, Δ)-respecting)**.** We say that $(\mathcal{A}, \mathcal{Z})$ is (ρ, Δ)-respecting w.r.t. protocol Π iff for every $\kappa \in \mathbb{N}$, with probability 1 in $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, every honest message is delivered within Δ rounds and moreover $(\mathcal{A}, \mathcal{Z})$ adaptively corrupts at most ρ fraction of nodes.

When the context is clear, we often say that $(\mathcal{A}, \mathcal{Z})$ is (ρ, Δ)-respecting omitting saying which protocol Π is of interest.

4 Scoring Agreement

We define an abstraction called scoring agreement — this is at the of our batch agreement construction. We rely on a herding-based protocol to achieve it. In a scoring agreement protocol, each node starts with an element from some known universe $\mathcal{U}$. Each node can evaluate an initial score for each element from $\mathcal{U}$. The scoring agreement protocol seeks to reach agreement on some element with from $\mathcal{U}$ that is scored relatively highly by (almost) all forever honest nodes.

[Elaine: we never said U is polynomial in size]

4.1 Definition of Scoring Agreement

Syntax. A scoring agreement protocol, henceforth denoted Π_{score} is parametrized with a universe $\mathcal{U}$ that defines valid values. Moreover, suppose that there is a publicly known, polynomial-time computable function (also denoted $\mathcal{U}$ for convenience) for verifying whether a value v belongs to $\mathcal{U}$.

The environment $\mathcal{Z}$ instructs all nodes to start the protocol at the same time (treated as round 0 for the current protocol instance). When a node is instructed to start by $\mathcal{Z}$, it additionally receives the following as input from $\mathcal{Z}$:

1. a value $v_i \in \mathcal{U}$;
2. an efficiently computable function $\mathsf{score}_i : \mathcal{U} \to \mathbb{R}$ that can assign an initial, real-valued score for any value $v \in \mathcal{U}$; note that different nodes can receive different scoring functions.

Later when employed in our batch agreement, the value will be blocks of transactions.

Constraints on $\mathcal{Z}$. We require that the following conditions hold with probability 1:

- *ϑ-somewhat-consistent initial scoring:* for every $v \in \mathcal{U}$, for any initially honest i and j, it holds that $|\mathsf{score}_i(v) - \mathsf{score}_j(v)| < \vartheta$.
- *High initial scores:* for every forever honest i, let v_i be i's input — it must be that there is no $v' \in \mathcal{U}$ such that $\mathsf{score}_i(v') > \mathsf{score}_i(v_i)$.

The first condition above requires that initially honest nodes receive relatively consistent scoring functions from $\mathcal{Z}$, i.e., they assign somewhat consistent initial scores for every element in the

universe. The second condition requires that every forever honest node's input must be the highest scoring element in the universe (as perceived by the node itself).

Security properties. We want a protocol where nodes reach agreement on a value in $\mathcal{U}$. We say that a protocol Π_{score} (parametrized with $\mathcal{U}$ and Δ) satisfies a certain *property* w.r.t. $(\mathcal{A}, \mathcal{Z})$, iff there exists some negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, for all but $\mathsf{negl}(\kappa)$ fraction of the execution traces sampled from $\mathsf{EXEC}^{\Pi_{\text{score}}}(\mathcal{A}, \mathcal{Z}, \kappa)$, that property holds. In particular, we care about the the following properties.

- *Consistency.* If a so-far honest node i outputs $v \in \mathcal{U}$ and a so-far honest node j outputs $v' \in \mathcal{U}$, it must hold that $v = v'$.
- *d-Validity.* Suppose that for some $B \in \mathbb{R}$, there exists subset S of forever honest nodes of size at least $(\frac{1}{3} + 0.5\epsilon)n$, such that for every $i \in S$, i received an input value v_i satisfying $\mathsf{score}_i(v_i) \geq B$. Then, if any so-far honest node outputs $v^* \in \mathcal{U}$, then there must exist an initially honest node i^* such that $\mathsf{score}_{i^*}(v^*) \geq B - d$.

  In other words, if sufficiently many forever honest nodes receive a high-scoring input value and some honest node outputs v, then it cannot be that all honest nodes assign v a relatively low initial score.
- *T_{end}-Liveness.* Every forever honest node terminates and outputs a value in round T_{end}.

[Elaine: i removed the mention of non-trivial as reviewer pointed out]

4.2 Message-Specific Random Eligibility Election

To achieve small communication bandwidth, we use a technique proposed in Abraham et al. [2] for vote-specific, random eligibility election. A node with the identifier i should only send a message m if it is determined to be eligible for sending m — otherwise the the message m will be discarded by so-far honest nodes.

Random eligibility election with cryptography. Imprecisely speaking such random eligibility election is performed with the help of a Verifiable Random Function (VRF) [17]: assume that every player i has a VRF public key denoted pk_i that is common knowledge, and the corresponding VRF secret key sk_i is known only to player i. For i to determine its eligibility for sending m, it evaluates $(\mu, \pi) := \mathsf{VRF}(\mathsf{sk}_i, \mathsf{m})$ where μ is the VRF evaluation outcome and π is a proof attesting to the evaluation outcome. If $\mu < D_p$ where D_p is an appropriate difficulty parameter, node i is deemed eligible for sending the message m. While only the secret-key owner can evaluate the VRF, anyone can verify the evaluation outcome. More specifically, suppose that node i additionally attaches the pair (μ, π) when sending the message m; then, any node that receives the tuple (m, μ, π) can verify with pk_i that indeed μ is the correct VRF evaluation outcome and verify i's eligibility for m.

Now for technical reasons we will, for the time being, assume that such a VRF exists and moreover can resist adaptive attacks: specifically, even when the adversary can selectively open the secret keys of a subset of the honest nodes, the remaining honest nodes' VRFs will still give pseudo-random evaluation outcomes. Later in Section 7, we will describe how to instantiate such an adaptively-secure VRF that satisfies our needs, using techniques from Abraham et al. [2].

Remark 4.1 (Subtleties regarding the use of VRF)**.** Although earlier works such as Algorand [6] and others [8, 13, 23] also rely on a VRF; they do not use the *vote-specific* election technique; and this is why these earlier works must rely on erasures to achieve adaptive security. Abraham

et al. [2] relies on vote-specificity to remove the erasure assumption, but their technique works only for agreement on *a single bit* as explained in Section 2.3 and the Supplemental Materials. Finally, although not explicitly noted, Algorand [6] and other prior works [8] also require that the VRF be adaptively secure (i.e., honest VRF evaluations must remain pseudorandom even when the adversary can selectively open honest nodes' keys) — these earlier works rely on a random oracle to achieve such adaptive security. In our work, we instantiate such an adaptively secure VRF without relying on random oracles.

Random eligibility election in an idealized model. Henceforth for simplicity, in our protocol description we will abstract away the cryptographic details and instead assume that an idealized oracle $\mathcal{F}_{\text{mine}}$ exists that takes care of eligibility election — but later in Section 7, we will explain how to instantiate $\mathcal{F}_{\text{mine}}$ with adaptively secure cryptographic primitives. Specifically, $\mathcal{F}_{\text{mine}}$ is a trusted (i.e., incorruptible) party that performs the following — we assume that $\mathcal{F}_{\text{mine}}$ has been parametrized with an appropriate probability p:

1. Upon receiving $\texttt{mine}(\mathsf{m})$ from node i, if the coin $\mathsf{Coin}[\mathsf{m}, i]$ has not been recorded, flip a random coin b that is 1 with probability p and is 0 with probability $1 - p$. Record $\mathsf{Coin}[\mathsf{m}, i] := b$ and return $\mathsf{Coin}[\mathsf{m}, i]$
2. Upon receiving $\texttt{verify}(\mathsf{m}, i)$ from any node, if the coin $\mathsf{Coin}[\mathsf{m}, i]$ has been recorded, return its value; else return 0.

Basically, for node i to check its eligibility for the message m, it calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{m})$ — henceforth for simplicity we also call this act "*mining a vote for* m".

4.3 Herding-Based Scoring Agreement Protocol

The protocol Π_{score} is parametrized with some universe $\mathcal{U}$. We describe the protocol in the $\mathcal{F}_{\text{mine}}$-hybrid world, and later in Section 7 we show how to remove the $\mathcal{F}_{\text{mine}}$ idealized assumption.

1. *Parameters.* Recall that the adversary controls $\frac{1}{3} - \epsilon$ fraction. Let λ be large enough such that $\epsilon^2\lambda = \omega(\log \kappa)$; e.g., if ϵ is a(n arbitrarily small) constant then λ can be any super-logarithmic function.

   The mining difficulty parameter is set such that if all nodes were honest, on average exactly 1 vote (among all nodes) would be successfully mined every $\lambda\Delta$ number of rounds. In other words, each mining attempt is successful with probability $\frac{1}{\lambda\Delta n}$ where n is the total number of nodes.

2. *Mining.* In each round t, for every value $v \in \mathcal{U}$ node i has observed so-far, node i computes its *popularity* by adding v's initial score and the number of valid votes seen so far for v. Next, node i picks the most popular value $v \in \mathcal{U}$ that has been observed (breaking ties arbitrarily). The node i then contacts $\mathcal{F}_{\text{mine}}.\texttt{mine}(v, t)$ to mine a vote for the message (v, t) — if successful, node i multicasts (v, t, i) as well as all valid votes that it has observed so far for the value v.

3. *Vote validity.* A node can verify the validity of a received vote (v, t, i), by calling $\mathcal{F}_{\text{mine}}.\texttt{verify}(v, t, i)$. If a vote (v, t, i) is received where t is greater than the node's round number, discard the vote.

4. *Terminate.* Every node runs the protocol for $T_{\text{end}} = \lambda^2\Delta$ number of rounds, at the end of which the node attempts to output a value based on the following rule: the node has observed at least $\frac{2\lambda}{3}$ valid votes for any value $v \in \mathcal{U}$, then output v; else output nothing.

4.4 Theorem Statements for Scoring Agreement

We summarize this section with the following theorem statements, the proofs of which are deferred to Section 8.

Theorem 4.2 (Security of scoring agreement)**.** *Assume that* $\epsilon^2\lambda = \omega(\log\kappa)$*. The above* $\mathcal{F}_{\text{mine}}$*-hybrid scoring agreement protocol satisfies consistency,* $\frac{2\lambda}{3}$*-validity, and* $\lambda^2\Delta$*-liveness against any* $(\frac{1}{3}-\epsilon, \Delta)$*-respecting, non-uniform p.p.t.* $(\mathcal{A}, \mathcal{Z})$ *that satisfies the constraints*[5] *specified in Section 4.1.*

When the choice of λ is polylogarithmic in κ, the protocol achieves polylogarithmic multicast communication complexity, i.e., only polylogarithmically many honest messages are multicast regardless of how $(\mathcal{A}, \mathcal{Z})$ behaves.

Theorem 4.3 (Communication efficiency of scoring agreement)**.** *Suppose that* $\log^{1.1}\kappa \le \lambda \le \log^2\kappa$ *and that* n *is polynomial in* κ*. Then, for any* $(\mathcal{A}, \mathcal{Z})$*, there is a negligible function* $\mathsf{negl}(\cdot)$ *such that except with* $\mathsf{negl}(\kappa)$ *probability over the choice of* $\mathsf{EXEC}^{\Pi_{score}}(\mathcal{A}, \mathcal{Z}, \kappa)$ *where* Π_{score} *denotes the above* $\mathcal{F}_{\text{mine}}$*-hybrid scoring agreement protocol, honest nodes multicast no more than* $\log^3\kappa \cdot \Theta(\ell + \log\kappa)$ *bits of messages where* ℓ *is the number of bits for encoding each element in* $\mathcal{U}$*.*

5 Batch Agreement

In this section, we first define a new abstraction called *batch agreement*, a primitive that allows nodes to agree on a batch of transactions, such that long-pending transactions (for some notion of long-pending) must be included in the output batch. Our state machine replication protocol will simply sequentially compose multiple instances of batch agreement to agree on batches of transactions over time (see Section 6).

We show that one can construct batch agreement from scoring agreement by choosing an appropriate scoring function that severely discounts batches that omit sufficiently old transactions.

5.1 Formal Definition of Batch Agreement

Syntax. Suppose that nodes receive transactions as input from the environment $\mathcal{Z}$ over time. We assume that $\mathcal{Z}$ respects the following transaction diffusion assumption with probability 1 — later in the Supplemental Materials we shall describe how to remove this assumption while preserving communication efficiency:

> *Transaction diffusion assumption:* If some forever honest node receives a transaction tx as input in some round t, then all so-far honest nodes must have received tx as input by the end of round $t + \Delta$.

Remark 5.1 (About the transaction diffusion assumption)**.** One way to remove this assumption is to have every node echo the tx upon first seeing it — in real-world peer-to-peer networks such as those adopted by Bitcoin or Ethereum, everyone echoing the same message should charge only once to the communication cost. Later in the Supplemental Materials we discuss how to remove this assumption for synchronous networks requiring only a small number of so-far honest nodes to echo each tx.

[5] See the "Syntax" and "Constraints on $\mathcal{Z}$" paragraphs.

The environment $\mathcal{Z}$ starts all nodes in the same round denoted r_{start}. When starting an initially honest node i, $\mathcal{Z}$ informs node i a set of transactions that are already confirmed — the same set of confirmed transactions, henceforth denoted $\mathsf{TXs}_{\text{confirmed}}$ must be provided to all honest nodes.

At the end of the batch agreement protocol, every forever honest node must have output a batch of transactions.

Security properties. We require the following security properties. Specifically, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all but $\mathsf{negl}(\kappa)$ fraction of execution traces, the following properties must hold:

- *Consistency.* If a so-far honest node outputs TXs and another so-far honest node outputs TXs' in the batch agreement protocol, it must be that $\mathsf{TXs} = \mathsf{TXs}'$.
- T_{end}-*Liveness.* Let $T_{\text{end}} = \mathsf{poly}(\kappa, n, \Delta)$ be a polynomial function in κ, n, and Δ. Every node that remains honest in round $r_{\text{start}} + T_{\text{end}}$ must have output a batch by round $r_{\text{start}} + T_{\text{end}}$.
- D-*Validity.* If $D \leq r_{\text{start}}$ and some forever honest node has observed a transaction $\mathsf{tx} \notin \mathsf{TXs}_{\text{confirmed}}$ by round $r_{\text{start}} - D$ where r_{start} is the start of the batch agreement protocol, then tx must appear in any forever honest node's output batch.

5.2 Batch Agreement from Scoring Agreement

Intuition. It is easy to construct a batch agreement protocol from scoring agreement in the synchronous setting. The idea is to rely on a scoring function such that a batch would receive a significant penalty if long-pending transactions were excluded. To obtain liveness, we also need that initially honest nodes assign somewhat consistent initial scores to every batch. This is guaranteed by leveraging transaction diffusion assumption: a fresh transaction is propagated to all nodes at most Δ apart. This means that so-far honest nodes have a somewhat consistent view of any transaction's age. We design our scoring function to make sure that if the transaction diffusion assumption holds, then all honest nodes would assign somewhat consistent initial scores to every batch. Finally, we also need that initially honest nodes receive high-scoring inputs — this is also guaranteed because an honest node always tries to include all pending transactions observed so far in its input batch.

Detailed protocol. We now describe the batch agreement protocol which is build from a scoring agreement instance denoted Π_{score}.

- *Input.* Start a scoring agreement instance denoted Π_{score}, and choose the input to Π_{score} as follows: let TXs be the set of outstanding transactions in the node's view so far. Input $\mathsf{TXs} \backslash \mathsf{TXs}_{\text{confirmed}}$ to Π_{score}.
- *Initial scoring function.* Given a set of transactions TXs, its initial score is computed as the following by node i. Let $\mathsf{tx} \notin \mathsf{TXs} \cup \mathsf{TXs}_{\text{confirmed}}$ be the earliest transaction (not in $\mathsf{TXs} \cup \mathsf{TXs}_{\text{confirmed}}$) which node i has observed so far, and suppose that node i observed tx in round t — if there is no such tx, we simply let $t := r_{\text{start}}$. Then, the initial score of TXs is computed as:

$$\mathsf{score}_i(\mathsf{TXs}) := 3\lambda \cdot \left(1 - \frac{1}{3\lambda^2}\right)^{\left\lfloor \frac{r_{\text{start}} - t}{3\Delta} \right\rfloor}$$

- *Output.* Now execute the scoring agreement protocol Π_{score} for T_{end} number of rounds, and output whatever it outputs.

Theorem 5.2 (Synchronous batch agreement)**.** *Suppose that $\lambda(\kappa) > 0.5$ for sufficiently large κ. For any $0 < \rho < 1$, any Δ, suppose that $(\mathcal{A}, \mathcal{Z})$ is non-uniform p.p.t. and (ρ, Δ)-respecting and also respects the assumptions stated in Section 5.1. Assume that the scoring agreement protocol employed in the above batch agreement construction satisfies consistency, λ-validity, and T_{end}-liveness against $(\mathcal{A}, \mathcal{Z})$. Then the above batch agreement protocol in the $\mathcal{F}_{\text{mine}}$-hybrid world achieves consistency, T_{end}-liveness, and $\Theta(\lambda\Delta)$-validity against $(\mathcal{A}, \mathcal{Z})$.*

Proof. Consistency follows directly from the consistency of the scoring agreement. T_{end}-liveness of the batch agreement would follow if the scoring agreement also satisfies T_{end}-liveness — to show the latter, observe that

1. Due to the transaction diffusion assumption, for any valid batch TXs any two initially honest nodes' scores are at most $\frac{1}{\lambda}$ apart; and
2. All initially honest nodes score their own input 3λ.

It remains to prove $\Theta(\lambda\Delta)$-validity. If some forever honest node has observed a transaction $\mathsf{tx} \in \mathsf{TXs}_{\text{confirmed}}$ by round $r_{\text{start}} - c\lambda\Delta \geq 0$ [Elaine: note: changed param based on reviewer comment] for some appropriate constant c, then for any initially honest node i, it must have observed tx by round $t = r_{\text{start}} - c\lambda\Delta + \Delta$, by the transaction diffusion assumption; [Elaine: note: changed param based on reviewer comment] for any batch TXs that does not contain tx, we have that

$$\mathsf{score}_i(\mathsf{TXs}) = 3\lambda \cdot \left(1 - \frac{1}{3\lambda}\right)^{\left\lfloor \frac{r_{\text{start}} - t}{3\Delta} \right\rfloor} \leq 3\lambda \cdot \left(1 - \frac{1}{3\lambda}\right)^{\frac{c\lambda\Delta - \Delta}{3\Delta}}$$

For an appropriate constant $c = 20$ we have that $\left(1 - \frac{1}{3\lambda}\right)^{(c\lambda-1)/3} \leq 0.5$ for any $\lambda > 0.5$; [Elaine: i changed this to be a little more precise] therefore $\mathsf{score}_i(\mathsf{TXs}) \leq 1.5\lambda$. Recall that every initially honest node will score its own input value 3λ. Thus $c\lambda\Delta$-validity of the batch agreement follows from the λ-validity of the scoring agreement instance. □

Communication efficiency. Suppose that the above synchronous batch agreement adopts the scoring agreement protocol devised in Section 4; and further, assume that $\log^{1.1} \kappa \leq \lambda \leq \log^2 \kappa$. Then, due to Theorem 4.3, it is not difficult to see that regardless of $(\mathcal{A}, \mathcal{Z})$'s behavior, except with negligible in κ probability, forever honest nodes multicast no more than $\log^3 \kappa \cdot \Theta(|\mathsf{TX}_{\text{active}}| + \log \kappa)$ bits of messages in the synchronous batch agreement protocol where $\mathsf{TX}_{\text{active}} := \mathsf{TXs}_{\text{all}} \backslash \mathsf{TXs}_{\text{confirmed}}$ denotes the set of all transactions each of which observed by at least one so-far honest node by the end of the batch agreement protocol (denoted $\mathsf{TXs}_{\text{all}}$), subtracting those that were already confirmed prior to the start of the batch agreement instance (denoted $\mathsf{TXs}_{\text{confirmed}}$). Recall that the environment $\mathcal{Z}$ informs nodes of the $\mathsf{TXs}_{\text{confirmed}}$ set prior to starting a batch agreement instance.

6 SMR from Batch Agreement

6.1 Definition of State Machine Replication

State machine replication has been a central abstraction in the 30 years of distributed systems literature. In a state machine replication protocol, a set of nodes seek to agree on an ever-growing log over time. We require two critical security properties: 1) *consistency*, i.e., all forever honest nodes' logs agree with each other although some nodes may progress faster than others; 2) *liveness*, i.e., transactions received by initially honest nodes as input get confirmed in all forever honest

nodes' logs quickly. We now define what it formally means for a protocol to realize a "state machine replication" abstraction.

Syntax. In a state machine replication protocol, in every round, a node receives as input a set of transactions txs from $\mathcal{Z}$ at the beginning of the round, and outputs a LOG collected thus far to $\mathcal{Z}$ at the end of the round. As before, we assume that $\mathcal{Z}$ respects the *transaction diffusion assumption* with probability 1. In other words, two so-far honest nodes observe any transaction tx within Δ rounds apart.

Security. Let $T_{\text{confirm}}(\kappa, n, \Delta)$ be a polynomial function in the security parameter κ, the number of nodes n, and the maximum network delay Δ.

Definition 6.1. We say that a state machine replication protocol Π satisfies consistency (or T_{confirm}-liveness resp.) w.r.t. some $(\mathcal{A}, \mathcal{Z})$, iff there exists a negligible function $\mathsf{negl}(\cdot)$, such that for any $\kappa \in \mathbb{N}$, except with $\mathsf{negl}(\kappa)$ probability over the choice of $\mathsf{view} \leftarrow \mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, consistency (or T_{confirm}-liveness resp.) is satisfied:

- *Consistency*: A view satisfies consistency iff the following holds:
  - *Common prefix.* Suppose that in view, a so-far honest node i outputs LOG to $\mathcal{Z}$ in round t, and a so-far honest node j outputs LOG' to $\mathcal{Z}$ in round t' (i and j may be the same or different), it holds that either $\mathsf{LOG} \preceq \mathsf{LOG}'$ or $\mathsf{LOG}' \preceq \mathsf{LOG}$. Here the relation $\preceq$ means "is a prefix of". By convention we assume that $\emptyset \preceq x$ and $x \preceq x$ for any x.
  - *Self-consistency.* Suppose that in view, a node i is honest during rounds $[t, t']$, and outputs LOG and LOG' in rounds t and t' respectively, it holds that $\mathsf{LOG} \preceq \mathsf{LOG}'$.
- *Liveness*: A view satisfies T_{confirm}-liveness iff the following holds: if in some round $t \leq |\mathsf{view}| - T_{\text{confirm}}$, some forever honest node either received from $\mathcal{Z}$ an input set txs that contains some transaction tx or has tx in its output log to $\mathcal{Z}$ in round t, then, for any node i honest in any round $t' \geq t + T_{\text{confirm}}$, let LOG be the output of node i in round t', it holds that $\mathsf{tx} \in \mathsf{LOG}$.

  Intuitively, liveness says that transactions input to an initially honest node get included in forever honest nodes' LOGs within T_{confirm} time; and further, if a transaction appears in some forever honest node's LOG, it will appear in every forever honest node's LOG within T_{confirm} time.

6.2 Constructing State Machine Replication from Batch Agreement

It is relatively straightforward how to construct state machine replication from batch agreement: basically, all nodes start a batch agreement instance in round 0 henceforth denoted $\Pi_{\text{batch}}[0]$; as soon as the i-th batch agreement instance $\Pi_{\text{batch}}[i]$ outputs a batch, a node immediately starts a next batch agreement instance denoted $\Pi_{\text{batch}}[i+1]$. At any time, a node outputs a sequential concatenation of all batches that have been output so far by batch agreement instances[6]. For every instance of batch agreement, the confirmed set $\mathsf{TXs}_{\text{confirmed}}$ provided as input consists of all transactions that have been output by previous instances (i.e., these transactions need not be confirmed again).

[6]The state machine replication protocol above invokes many instances of batch agreement which may then invoke one or more instances of scoring agreement. Recall that each scoring agreement instance calls $\mathcal{F}_{\text{mine}}$. For composition, calls to $\mathcal{F}_{\text{mine}}$ are tagged with an instance identifier. Here the instance identifier contains a pair: first the identifier of the batch agreement instance and then the identifier of the scoring agreement.

Theorem 6.2 (State machine replication). *Let Δ be any polynomial function in κ and n and let $0 < \epsilon < 1$ be any positive constant. Suppose that $(\mathcal{A}, \mathcal{Z})$ is $(\frac{1}{3}-\epsilon, \Delta)$-respecting and moreover respects the assumptions stated in Section 6.1. Assume that the batch agreement protocol adopted satisfies consistency, T-liveness, and D-validity w.r.t. $(\mathcal{A}, \mathcal{Z})$, then the above state machine replication protocol satisfies consistency and $(2T + D)$-liveness w.r.t. $(\mathcal{A}, \mathcal{Z})$.*

Proof. Consistency follows directly from the consistency of batch agreement. Moreover, $(2T + D)$-liveness follows from the fact that if $\mathcal{Z}$ inputs a tx to some forever honest node in round r, then consider the first batch agreement instance that is started (by some honest node) in round $r + D$ or after: it takes up to T time till the this batch agreement instance ends by T-liveness of the batch agreement, and the immediate next batch agreement instance will surely output tx if tx is not output earlier by D-validity of the batch agreement. □

Communication efficiency. For analyzing communication efficiency, let us assume that we adopt the batch agreement protocol described in Section 5; further, assume that ϵ is an arbitrarily small constant and that $\lambda = \log^{1.1} \kappa$. For some transaction tx, suppose that round r is the first round in which some forever honest node observes tx. Then, starting from round r, the transaction tx will be confirmed after $\mathsf{poly} \log \kappa$ number of batch agreement instances, and thus it will contribute to the $\mathsf{TX}_{\text{active}}$ set of $\mathsf{poly} \log \kappa$ number of such instances. Recall that for each batch agreement instance, except with negligible in κ probability, only $\log^3 \kappa \cdot \Theta(|\mathsf{TX}_{\text{active}}| + \log \kappa)$ bits of honest messages are multicast. Thus, the bits of honest messages multicast, amortized to each tx, is bounded by $|\mathsf{tx}| \cdot \mathsf{poly} \log \kappa$ for some suitable polynomial $\mathsf{poly}(\cdot)$ except with negligible in κ probability.

7 Removing the Idealized Functionality $\mathcal{F}_{\text{mine}}$

So far, all our protocols have assumed the existence of an $\mathcal{F}_{\text{mine}}$ ideal functionality. In this section, we describe how to instantiate the protocols in the real world. Our techniques follow the approach described by Abraham et al. [2]. Although this part is not a contribution of our paper, for completeness, we describe all the building blocks and the approach in a self-contained manner, borrowing some text from Abraham et al. [2].

7.1 Preliminary: Adaptively Secure Non-Interactive Zero-Knowledge Proofs

We use $f(\kappa) \approx g(\kappa)$ to mean that there exists a negligible function $\nu(\kappa)$ such that $|f(\kappa) - g(\kappa)| < \nu(\kappa)$.

A non-interactive proof system henceforth denoted nizk for an NP language $\mathcal{L}$ consists of the following algorithms.

- $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa, \mathcal{L})$: Takes in a security parameter κ, a description of the language $\mathcal{L}$, and generates a common reference string crs.
- $\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w)$: Takes in crs, a statement stmt, a witness w such that $(\mathsf{stmt}, w) \in \mathcal{L}$, and produces a proof π.
- $b \leftarrow \mathsf{V}(\mathsf{crs}, \mathsf{stmt}, \pi)$: Takes in a crs, a statement stmt, and a proof π, and outputs 0 (reject) or 1 (accept).

Perfect completeness. A non-interactive proof system is said to be perfectly complete, if an honest prover with a valid witness can always convince an honest verifier. More formally, for any

$(\mathsf{stmt}, w) \in \mathcal{L}$, we have that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa, \mathcal{L}),\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w) : \mathsf{V}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1\right] = 1$$

Non-erasure computational zero-knowledge. Non-erasure zero-knowledge requires that under a simulated CRS, there is a simulated prover that can produce proofs without needing the witness. Further, upon obtaining a valid witness to a statement a-posteriori, the simulated prover can explain the simulated NIZK with the correct witness.

We say that a proof system $(\mathtt{gen}, \mathsf{P}, \mathsf{V})$ satisfies non-erasure computational zero-knowledge iff there exists a probabilistic polynomial time algorithms $(\mathtt{gen}_0, \mathsf{P}_0, \mathsf{Explain})$ such that

$$\Pr\left[\mathsf{crs} \leftarrow \mathtt{gen}(1^\kappa), \mathcal{A}^{\mathsf{Real}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \approx \Pr\left[(\mathsf{crs}_0, \tau_0) \leftarrow \mathtt{gen}_0(1^\kappa), \mathcal{A}^{\mathsf{Ideal}(\mathsf{crs}_0,\tau_0,\cdot,\cdot)}(\mathsf{crs}_0) = 1\right],$$

where $\mathsf{Real}(\mathsf{crs}, \mathsf{stmt}, w)$ runs the honest prover $\mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w)$ with randomness r and obtains the proof π, it then outputs (π, r); $\mathsf{Ideal}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w)$ runs the simulated prover $\pi \leftarrow \mathsf{P}_0(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, \varrho)$ with randomness ϱ and without a witness, and then runs $r \leftarrow \mathsf{Explain}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w, \varrho)$ and outputs (π, r).

Perfect knowledge extraction. We say that a proof system $(\mathtt{gen}, \mathsf{P}, \mathsf{V})$ satisfies perfect knowledge extraction, if there exists probabilistic polynomial-time algorithms $(\mathtt{gen}_1, \mathsf{Extr})$, such that for all (even unbounded) adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{crs} \leftarrow \mathtt{gen}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1\right] = \Pr\left[(\mathsf{crs}_1, \tau_1) \leftarrow \mathtt{gen}_1(1^\kappa) : \mathcal{A}(\mathsf{crs}_1) = 1\right],$$

and moreover,

$$\Pr\left[\begin{array}{ll} (\mathsf{crs}_1, \tau_1) \leftarrow \mathtt{gen}_1(1^\kappa); (\mathsf{stmt}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}_1); & \mathsf{V}(\mathsf{crs}_1, \mathsf{stmt}, \pi) = 1 \\ w \leftarrow \mathsf{Extr}(\mathsf{crs}_1, \tau_1, \mathsf{stmt}, \pi) & : \text{ but } (\mathsf{stmt}, w) \notin \mathcal{L} \end{array}\right] = 0$$

7.2 Adaptively Secure Non-Interactive Commitment Scheme

An adaptively secure non-interactive commitment scheme consists of the following algorithms:

- $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\kappa)$: Takes in a security parameter κ, and generates a common reference string crs.
- $C \leftarrow \mathsf{com}(\mathsf{crs}, v, \varrho)$: Takes in crs, a value v, and a random string ϱ, and outputs a committed value C.
- $b \leftarrow \mathsf{ver}(\mathsf{crs}, C, v, \varrho)$: Takes in a crs, a commitment C, a purported opening (v, ϱ), and outputs 0 (reject) or 1 (accept).

Computationally hiding under selective opening. We say that a commitment scheme $(\mathsf{gen}, \mathsf{com}, \mathsf{ver})$ is computationally hiding under selective opening, iff there exists a probabilistic polynomial time algorithms $(\mathtt{gen}_0, \mathsf{com}_0, \mathsf{Explain})$ such that

$$\Pr\left[\mathsf{crs} \leftarrow \mathtt{gen}(1^\kappa), \mathcal{A}^{\mathsf{Real}(\mathsf{crs},\cdot)}(\mathsf{crs}) = 1\right] \approx \Pr\left[(\mathsf{crs}_0, \tau_0) \leftarrow \mathtt{gen}_0(1^\kappa), \mathcal{A}^{\mathsf{Ideal}(\mathsf{crs}_0,\tau_0,\cdot)}(\mathsf{crs}_0) = 1\right]$$

where $\mathsf{Real}(\mathsf{crs}, v)$ runs the honest algorithm $\mathsf{com}(\mathsf{crs}, v, r)$ with randomness r and obtains the commitment C, it then outputs (C, r); $\mathsf{Ideal}(\mathsf{crs}_0, \tau_0, v)$ runs the simulated algorithm $C \leftarrow \mathsf{comm}_0(\mathsf{crs}_0, \tau_0, \varrho)$ with randomness ϱ and without v, and then runs $r \leftarrow \mathsf{Explain}(\mathsf{crs}_0, \tau_0, v, \varrho)$ and outputs (C, r).

Perfectly binding. A commitment scheme is said to be perfectly binding iff for every crs in the support of the honest CRS generation algorithm, there does not exist $(v, \varrho) \neq (v', \varrho')$ such that $\mathsf{com}(\mathsf{crs}, v, \varrho) = \mathsf{com}(\mathsf{crs}, v', \varrho')$.

Theorem 7.1 (Instantiation of our NIZK and commitment schemes [12]). *Assume standard bilinear group assumptions. Then, there exists a proof system that satisfies perfect completeness, non-erasure computational zero-knowledge, and perfect knowledge extraction. Further, there exist a commitment scheme that is perfectly binding and computationally hiding under selective opening.*

Proof. The existence of such a NIZK scheme was shown by Groth et al. [12] via a building block that they called *homomorphic proof commitment scheme*. This building block can also be used to achieve a commitment scheme with the desired properties. □

7.3 NP Language Used in Our Construction

In our construction, we will use the following NP language $\mathcal{L}$. A pair $(\mathsf{stmt}, w) \in \mathcal{L}$ iff

- parse $\mathsf{stmt} := (\mu, c, \mathsf{crs}_{\mathrm{comm}}, \mathsf{m})$, parse $w := (\mathsf{sk}, s)$;
- it must hold that $c = \mathsf{comm}(\mathsf{crs}_{\mathrm{comm}}, \mathsf{sk}, s)$, and $\mathsf{PRF}_{\mathsf{sk}}(\mathsf{m}) = \mu$.

7.4 Compilation to Real-World Protocols

We can remove the $\mathcal{F}_{\mathrm{mine}}$ oracle by leveraging cryptographic building blocks including a pseudo-random function family, a non-interactive zero-knowledge proof system that satisfies computational zero-knowledge and computational soundness, a perfectly correct and semantically secure public-key encryption scheme, and a perfectly binding and computationally hiding commitment scheme.

Earlier in Section 4, we informally have described the intuition behind our approach. In this section we provide a formal description of how to compile our $\mathcal{F}_{\mathrm{mine}}$-hybrid protocols into real-world protocols using cryptography. Using this compilation technique, we can compile our $\mathcal{F}_{\mathrm{mine}}$-hybrid state machine replication protocol to the real world. Our techniques are essentially the same as Abraham et al. [2], but we describe it in full for completeness.

- **Trusted PKI setup.** Upfront, a trusted party runs the CRS generation algorithms of the commitment and the NIZK scheme to obtain $\mathsf{crs}_{\mathrm{comm}}$ and $\mathsf{crs}_{\mathrm{nizk}}$. It then chooses a secret PRF key for every node, where the i-th node has key sk_i. It publishes $(\mathsf{crs}_{\mathrm{comm}}, \mathsf{crs}_{\mathrm{nizk}})$ as the public parameters, and each node i's public key denoted pk_i is computed as a commitment of sk_i using a random string s_i. The collection of all users' public keys is published to form the PKI, i.e., the mapping from each node i to its public key pk_i is public information. Further, each node i is given the secret key (sk_i, s_i). Remark 7.2 later mentions how multiple protocol instances can share the same PKI.

- **Instantiating $\mathcal{F}_{\mathrm{mine}}.\mathtt{mine}$.** Recall that in the ideal-world protocol a node i calls $\mathcal{F}_{\mathrm{mine}}.\mathtt{mine}(\mathsf{m})$ to mine a vote for a message m. Now, instead, the node i calls $\mu := \mathsf{PRF}_{\mathsf{sk}_i}(\mathsf{m})$, and computes the NIZK proof
$$\pi := \mathsf{nizk.P}((\mu, \mathsf{pk}_i, \mathsf{crs}_{\mathrm{comm}}, \mathsf{m}), (\mathsf{sk}_i, s_i))$$
where s_i the randomness used in committing sk_i during the trusted setup. Intuitively, this zero-knowledge proof proves that the evaluation outcome μ is correct w.r.t. the node's public key (which is a commitment of its secret key).

  The mining attempt for m is considered successful if $\mu < D_p$ where D_p is an appropriate difficulty parameter such that any random string of appropriate length is less than D_p with probability p — the probability p is selected in the same way as the earlier $\mathcal{F}_{\mathrm{mine}}$-hybrid world protocols.

  Recall that earlier in our $\mathcal{F}_{\mathrm{mine}}$-hybrid protocols, every message multicast by a so-far honest node i is a mined message of the form $(\mathsf{m} : i)$ where node i has successfully called $\mathcal{F}_{\mathrm{mine}}.\mathtt{mine}(\mathsf{m})$.

Each such *mined* message (m, i) that node i wants to multicast is translated to the real-world protocol as follows: we rewrite $(\mathsf{m} : i)$ as $(\mathsf{m}, i, \mu, \pi)$ where the terms μ and π are those generated by i in place of calling $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{m})$ in the real world (as explained above). Note that in our $\mathcal{F}_{\text{mine}}$-hybrid protocols a node $j \neq i$ may also relay a message $(\mathsf{m} : i)$ mined by i — in the real world, node j would be relaying $(\mathsf{m}, i, \mu, \pi)$ instead.

- **Instantiating $\mathcal{F}_{\text{mine}}.\texttt{verify}$.** In the ideal world, a node would call $\mathcal{F}_{\text{mine}}.\texttt{verify}$ to check the validity of mined messages upon receiving them, In the real-world protocol, we perform the following instead: upon receiving the mined message $(\mathsf{m}, i, \mu, \pi)$, a node can verify the message's validity by checking:

  1. $\mu < D_p$ where p is an appropriate difficulty parameter that depends on the type of the mined message; and
  2. π is indeed a valid NIZK for the statement formed by the tuple $(\mu, \mathsf{pk}_i, \mathsf{crs}_{\text{comm}}, \mathsf{m})$. The tuple is discarded unless both checks pass.

Remark 7.2 (Protocol composition in the real world.)**.** The real-world protocol may invoke multiple instances of scoring agreement each with a unique instance identifier. In the real world, all instances share the same PKI. Recall that in the $\mathcal{F}_{\text{mine}}$-hybrid world, every call to $\mathcal{F}_{\text{mine}}$ is prefixed with the instance identifier. Specifically, in the calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(\mathsf{m})$ and $\mathcal{F}_{\text{mine}}.\texttt{verify}(\mathsf{m}, i)$, one can imagine that the m part is tagged with the instance identifier. In the real world, this means that the message m passed to the PRF and the NIZK's prover and verifier is prefixed with the instance identifier too.

Now using the same proofs as Abraham et al. [2], we can prove that the compiled real-world protocols enjoy the same security properties as the $\mathcal{F}_{\text{mine}}$-hybrid protocols. Since the proofs follow identically, we omit the details and simply refer the reader to Abraham et al. [2]. We thus obtain the following theorem, by observing that in the real world protocol, each vote is of the form $(\mathsf{TXs}, r, i, \mu, \pi)$ where μ and π has length χ where χ is a cryptographic security parameter.

Theorem 7.3 (Real-world protocol: synchronous state machine replication)**.** *Under standard cryptographic hardness assumptions (more precisely, the existence of universally composable, adaptively secure non-interactive zero-knowledge and commitments [12]), there exists a synchronous state machine replication protocol that satisfies consistency and* $\mathsf{poly}\log\kappa \cdot \Delta$*-liveness against any non-uniform p.p.t.,* $(\frac{1}{3} - \epsilon, \Delta)$*-respecting* $(\mathcal{A}, \mathcal{Z})$ *where* $\mathsf{poly}(\cdot)$ *is a suitable polynomial function and* ϵ *is an arbitrarily small positive constant. Moreover, honest nodes only need to multicast* $\mathsf{poly}\log\kappa \cdot (\chi + |\mathsf{TXs}|)$ *bits of messages to confirm every batch of transactions denoted* TXs *where* χ *is a computational security parameter related to the strength of the cryptographic building blocks involved.*

Proof. Note our techniques for instantiating $\mathcal{F}_{\text{mine}}$ with actual cryptography is borrowed from Abraham et al. [2]. Their proof for showing that that the real-world protocol preserves the security properties proved in the ideal world is immediately applicable to our case. □

8 Deferred Proofs for Scoring Agreement

[Elaine: changed poly n to poly kappa]

In all of our proofs, we will by default assume that $(\mathcal{A}, \mathcal{Z})$ is non-uniform p.p.t., $(\frac{1}{3} - \epsilon, \Delta)$-respecting, and moreover, respects the assumptions stated in Section 4.1 (see the "Syntax" and "Constraints on $\mathcal{Z}$" paragraphs).

8.1 Consistency

We first prove that the protocol satisfies consistency.

Lemma 8.1 (Consistency). *Except with at most* $\mathsf{poly}(\kappa) \cdot \exp(-\Theta(\epsilon^2\lambda))$ *probability, no two so-far honest nodes can output different values in* $\mathcal{U}$.

Proof. We fix two values $v \neq v'$, and give an upper bound on the probability that both values are output by so-far honest nodes. Observe that this happens only if there are at least $\frac{2\lambda}{3}$ votes for each of the values, which means in total there are at least $\frac{4\lambda}{3}$ votes for either value.

We next consider how many mining attempts there can be for value v or v' among the $T := \lambda^2\Delta$ rounds. Observe that each forever honest node attempts to mine for at most one vote labeled with each round, where a (possibly adaptively) corrupted node can mine for both v and v' labeled with each round. Since the fraction of forever honest node is at least $\frac{2}{3} + \epsilon$, the total number of vote mining attempts for either v or v' is at most $(\frac{2}{3} + \epsilon) \cdot n \cdot T + (\frac{1}{3} - \epsilon) \cdot 2n \cdot T = (\frac{4}{3} - \epsilon) \cdot nT$.

Observing that the mining difficulty probability is $\frac{1}{\lambda\Delta n}$ and the mining events over different values, rounds and nodes are independent, by Chernoff Bound, the probability that there are at least $\frac{4\lambda}{3}$ votes for either v or v' is at most $\exp(-\Theta(\epsilon^2\lambda))$.

Independence Remark. Note that the outcome of a vote mining can affect what values an adaptively corrupted node will mine for. However, the important point is that the outcomes of the mining are independent, and there is a sure upper bound on the number of mining attempts, such that the Chernoff Bound can be applied above.

Since each node can perform only polynomial-time computation, and the entire transcript is polynomally bounded, it suffices to take union bound over $\mathsf{poly}(\kappa)$ number of unordered pairs of values to give the desired probability. □

8.2 Validity

We next prove that the protocol satisfies d-validity for $d = \frac{2\lambda}{3}$. When an execution is fixed, we can define B to be the largest value such that there exists a subset S of forever-honest nodes of size at least $(\frac{1}{3} + 0.5\epsilon)n$, such that for every $i \in S$, i received an input value v_i satisfying $\mathsf{score}_i(v_i) \geq B$.

[Elaine: i changed the lemma statement here. the reviewer complained about it. and i thought the complaint indeed made sense]

Lemma 8.2 (d-Validity). *Fix* $d = \frac{2\lambda}{3}$, *the following event happens with at most* $\mathsf{poly}(\kappa)\cdot\exp(-\Theta(\epsilon^2\lambda))$ *probability: there is some so-far honest node that outputs some value* $v^* \in \mathcal{U}$, *but for every initially honest node* j, *it holds that* $\mathsf{score}_j(v^*) < B - d$.

Proof. Fix some value v^* such that some so-far honest node outputs v^*. Suppose S is the set of $(\frac{1}{3} + 0.5\epsilon)n$ forever honest nodes in the hypothesis such that each $i \in S$ has some v_i such that $\mathsf{score}_i(v_i) \geq B$, but $\mathsf{score}_i(v^*) < B - d$. Then, we first show that the event in the lemma implies that there must be at least $d = \frac{2\lambda}{3}$ votes for v^* from nodes outside S.

Observe that for the first node i in S to vote for v^*, there must be at least d votes nodes outside S to compensate for the difference between $\mathsf{score}_i(v_i)$ and $\mathsf{score}_i(v^*)$. On the other hand, if there are no votes from S for v^*, then any so-far honest node that outputs v^* must see at least $\frac{2\lambda}{3} = d$ votes, which must all come from nodes outside S.

Recall that there are at most $(\frac{2}{3} - 0.5\epsilon)n$ nodes outside S. Within the $T = \lambda^2\Delta$ rounds, the mining difficulty for each node for v^* is $\frac{1}{\lambda\Delta n}$. Hence, the expected number of votes for v^* from nodes outside S is at most $(\frac{2}{3} - 0.5\epsilon)\lambda$. By Chernoff Bound, the probability that these nodes can produce at least $d = \frac{2\lambda}{3}$ votes for v^* is at most $\exp(-\Theta(\epsilon^2)\lambda)$.

Taking union bound over $\mathsf{poly}(\kappa)$ possible values contained in the polynomially sized transcript gives the result. [Elaine: i changed it: it doesnt say poly number of values in universe now] □

8.3 Liveness

For proving liveness, we assume that the "somewhat consistent initial score" condition is satisfied with the parameter $\vartheta = \frac{1}{\lambda}$ (see also Section 4.1).

Convergence Opportunity. We say that a round $t \geq \lambda\Delta$ is a *convergence opportunity*, if there is exactly one so-far honest node that successfully mines a vote in round t; moreover, no so-far honest nodes successfully mine votes within Δ rounds after t. No condition is placed on corrupted nodes.

The following lemma relates the number of convergence opportunities to the number of votes observed by a node for its most popular value.

Henceforth in Lemma 8.3 Corollary 8.4, when the execution we refer to is clear from the context, we use $\mathfrak{I}$ to denote the set of initially honest nodes, and let $M := \min_{j \in \mathfrak{I}} \mathsf{score}_i(v_j)$. [Elaine: i changed the definition of M, before, it was defined as $M := \min_{i \notin S} \mathsf{score}_i(v_i)$, with S undefined]

Lemma 8.3 (Convergence Opportunities and Votes)**.** *Consider any execution: for each $r \geq 1$, after Δ rounds following the r-th convergence opportunity, any so-far honest node i has observed at least $M - \mathsf{score}_i(v) + r(1-\vartheta)$ votes for its most popular value v.*

Proof. We show by induction on r. For $r = 0$, in order for a value v to be the most popular for node i, node i must have observed enough votes to compensate for the difference $\mathsf{score}_i(v_i) - \mathsf{score}_i(v) \geq M - \mathsf{score}_i(v)$.

Assume that claim is true for some $r \geq 0$. Suppose the $(r+1)$-st convergence opportunity due to a vote for value v by the so-far honest node i.

For node i, right before the $(r+1)$-st convergence opportunity, at least Δ rounds must have passed since the r-th convergence opportunity. The induction hypothesis says that node i has observed at least $M - \mathsf{score}_i(v) + r(1-\vartheta)$ votes [Elaine: note: hubert wrote 2 vartheta, i removed the 2 as reviewer suggested] for its most popular value v. Together with the new vote that node i has mined for the $(r+1)$-st convergence opportunity, node i has observed at least $M - \mathsf{score}_i(v) + 1 + r(1-\vartheta)$ votes for v. Observe that for any value v' to overtake v to be node i's most popular value, it will need at least $M - \mathsf{score}_i(v') + 1 + r(1-\vartheta)$ votes; hence, the result holds for node i.

For any other so-far honest node j, after Δ rounds following the $(r+1)$-st convergence opportunity, all the votes for v associated with the r-th convergence opportunity due to node i will have reached node j. Hence, if $\widehat{v}$ is a most popular value for node j at this moment, it must be the case that its popularity implies that the number of votes for $\widehat{v}$ observed by node j is at least:

$$(M - \mathsf{score}_i(v) + 1 + r(1-\vartheta)) + \mathsf{score}_j(v) - \mathsf{score}_j(\widehat{v}) \geq M - \mathsf{score}_j(\widehat{v}) + (r+1)(1-\vartheta)$$

where the last inequality follows because $\mathsf{score}_j(v) - \mathsf{score}_i(v) \geq -\vartheta$. Similarly, for any other value v' to overtake $\widehat{v}$ and become node j's most popular value, it must need at least $M - \mathsf{score}_j(v') + (r+1)(1-\vartheta)$ votes to be seen by node j. [Elaine: note: i changed M_j to M] This completes the inductive step and the proof of the lemma. □

Corollary 8.4 (Liveness)**.** *Suppose $\vartheta \leq \frac{1}{\lambda} \leq \Theta(\epsilon)$. Except with at most $\exp(-\Theta(\epsilon^2\lambda))$ probability, every forever-honest honest will have output some value by the end of round T_{end}. [Elaine: i changed so-far honest to forever-honest]*

Proof. For any forever-honest node i, from our "high initial scores" assumption that it will not see any value v' such that $\mathsf{score}_i(v') > \mathsf{score}_i(v_i)$, it follows that for any $u \in \mathcal{U}$, $M - \mathsf{score}_i(u) \geq \mathsf{score}_i(v_i) - \mathsf{score}_i(u) - \vartheta \geq -\vartheta$, where the inequality $M \geq \mathsf{score}_i(v_i) - \vartheta$ follows from the following:

$$M = \min_{j \in \mathcal{J}} \mathsf{score}_j(v_j) \geq \min_{j \in \mathcal{J}} (\mathsf{score}_i(v_j) - \vartheta) \geq \mathsf{score}_i(v_i) - \vartheta$$

where the first inequality follows from the "ϑ-somewhat-consistent initial scoring" assumption. [Elaine: i modified this line and added some explanation]

In view of Lemma 8.3, it suffices to show that except with at most $\exp(-\Theta(\epsilon^2\lambda))$ probability, the number of convergence opportunities is at least $\frac{2\lambda}{3}(1+\Theta(\epsilon))$, which implies that for each node, its most popular value has received at least $\frac{2\lambda}{3}$ votes.

Observe that if h is the number of so-far honest nodes not in S, the probability that a round is a convergence opportunity is around $hp(1-p)^{h\cdot\Delta+h-1} \geq \frac{2}{3\lambda\Delta}(1+\Theta(\epsilon))(1-\Theta(\frac{1}{\lambda}))$, where the inequality holds because $h \geq (\frac{2}{3}+\frac{1}{3}\epsilon)n$, and $p = \frac{1}{n\lambda\Delta}$ Hence, the expected number of convergence opportunities over the $T_{\text{end}} - \lambda\Delta$ rounds is at least $\frac{2\lambda}{3}(1+\Theta(\epsilon))$.

Even though that the events of different rounds being convergence opportunities are not independent, Lemma 8.5 shows that a measure concentration result still holds. Setting the parameters $h = (\frac{2}{3}+\epsilon)n$, $H = n$, $p = \frac{1}{\lambda\Delta n}$, $T = \lambda^2\Delta$ in Lemma 8.5, we have that except with $\exp(-\Theta(\epsilon^2\lambda))$ probability, the number of convergence opportunities is at least $\frac{2\lambda}{3}(1+\Theta(\epsilon))$.

Finally, during the last convergence opportunity by a so-far honest node, all the votes will be multicast to all nodes. Hence, all so-far honest nodes will see some value with at least $\frac{2\lambda}{3}$ votes after Δ more rounds. □

Lemma 8.5 (Measure Concentration for Convergence Opportunities)**.** *Suppose in each of $T+\Delta$ rounds, each of at least h but at most H so-far honest nodes mines a vote successfully with probability p.*

Then, except with probability at most $\exp(-\Theta(\epsilon^2 T p_0)$, the number of rounds in which there is exactly one successful vote and followed by Δ rounds of no votes is at least $(1+\Theta(\epsilon)-2Hp\Delta)Tp_0$, where $p_0 := hp(1-p)^H$.

Proof. The proof is adapted from [24, Lemma 1].

Let Y be the number of rounds within $[1..T]$ in which there is exactly one successful vote. Since the probability that a round has exactly one vote is at least $p_0 := hp(1-p)^H$ and the events for different rounds are independent, by Chernoff Bound, except with probability $\exp(-\Theta(\epsilon^2 T p_0))$, the random variable Y is at least $(1-\frac{\epsilon}{100})Tp_0$.

For $1 \leq i \leq R := \lceil (1+\epsilon)Tp_0 \rceil$, define the indicator random variable $Z_i \in \{0,1\}$ that equals 1 *iff* after the i-th round that has exactly one successful vote, there is at least one successful vote within the next Δ rounds. By the union bound, $\Pr[Z_i = 1] \leq Hp\Delta$. Define $Z := \sum_{i=1}^{R} Z_i$.

Next, observe that the random variables Z_i's are independent. The reason is that right after the ith round in which there is exactly one successful vote, when the next successful vote happens will determine the value of Z_i, but the $i+1$st round with exactly one successful vote will happen afterwards.

By Hoeffding's Inequality, we have that

$$\Pr[Z \geq Hp\Delta R + \frac{\epsilon}{100} \cdot R] \leq \exp(-\Theta(\epsilon^2 R))$$

By the union bound, except with probability at most $\exp(-\Theta(\epsilon^2 T p_0)$, $Y - Z \geq (1 - \frac{\epsilon}{10} - 2Hp\Delta)Tp_0$, which is also a lower bound on the number of convergence opportunities. □

8.4 Communication Efficiency

We now prove the communication efficiency of our scoring agreement protocol described in Section 4.3, that is, Theorem 4.3.

Recall that $\log^{1.1}\kappa \leq \lambda \leq \log^2\kappa$. By Chernoff bound, for every value $v \in \mathcal{U}$, except with negligible in κ probability. at most 1.1λ votes (honest or adversarial) are successfully mined for the value v during the course of execution.

So-far honest nodes only multicast a message whenever it successfully mines a vote. When it multicasts, it not only multicasts the newly mined vote, but also all votes it has already observed for the relevant value in $\mathcal{U}$ — recall that except with negligible in κ probability, there are at most 1.1λ such votes. Obviously, in the $\mathcal{F}_{\text{mine}}$-hybrid world, every vote can be encoded with $\Theta(\log\kappa + \ell)$ bits since both n and the number of rounds are polynomial in κ.

Summarizing the above, except with negligible probability, the total number of honest votes multicast is upper bounded by $\log^3\kappa$ for sufficiently large κ, and thus the total number of bits multicast by so-far honest nodes is upper bounded by $\log^3\kappa \cdot \Theta(\log\kappa + \ell)$.

9 Conclusion and Open Questions

In this paper, we proposed a novel paradigm for reaching consensus that is inspired by a social phenomenon called herding. Through this novel paradigm, we construct a state machine replication protocol that simultaneously achieves communication efficiency and adaptive security — to the best of our knowledge this was previously not possible with classical-style approaches without making strong assumptions such as erasures or the existence of proof-of-work oracles.

Our work naturally leaves open several questions:

1. Can we achieve a similar result for partially synchronous or asynchronous networks?
2. The best known small-round, communication-inefficient synchronous state machine replication protocol can tolerate minority corruptions [3,14,18]. Therefore, another natural question is: can we have a similar result for synchronous networks, but tolerating up to minority corruptions (thus matching the resilience of the best known small-round but communication inefficient protocol)?

References

[1] Aura - authority round. `https://wiki.parity.io/Aura`.

[2] I. Abraham, T.-H. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi. Communication complexity of byzantine agreement, revisited. *CoRR*, abs/1805.03391, 2018.

[3] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren. Efficient synchronous byzantine consensus. In *Financial Cryptography*, 2019.

[4] R. Canetti, D. Eiger, S. Goldwasser, and D.-Y. Lim. How to protect yourself without perfect shredding. Cryptology ePrint Archive, Report 2008/291, 2008. `https://eprint.iacr.org/2008/291`.

[5] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI*, 1999.

[6] J. Chen and S. Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

[7] P. Daian, R. Pass, and E. Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *Financial Cryptography*, 2019. First appeared on Cryptology ePrint Archive, Report 2016/919.

[8] B. M. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Eurocrypt*, 2018.

[9] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, 1992.

[10] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. In *SIAM Journal of Computing*, 1997.

[11] J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Eurocrypt*, 2015.

[12] J. Groth, R. Ostrovsky, and A. Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, June 2012.

[13] T. Hanke, M. Movahedi, and D. Williams. Dfinity technology overview seriesconsensus system. `https://dfinity.org/tech`.

[14] J. Katz and C.-Y. Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, Feb. 2009.

[15] A. Kiayias and A. Russell. Ouroboros-bft: A simple byzantine fault tolerant consensus protocol. Cryptology ePrint Archive, Report 2018/1049, 2018. `https://eprint.iacr.org/2018/1049`.

[16] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto*, 2017.

[17] S. Micali, S. Vadhan, and M. Rabin. Verifiable random functions. In *FOCS*, 1999.

[18] S. Micali and V. Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. MIT CSAIL Technical Report, 2017-004, 2017.

[19] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[20] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Eurocrypt*, 2017.

[21] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.

[22] R. Pass and E. Shi. Rethinking large-scale consensus (invited paper). In *CSF*, 2017.

[23] R. Pass and E. Shi. The sleepy model of consensus. In *Asiacrypt*, 2017.

[24] R. Pass and E. Shi. Rethinking large-scale consensus. *IACR Cryptology ePrint Archive*, 2018:302, 2018.

[25] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, Dec. 1990.

[26] E. Shi. Analysis of deterministic longest-chain protocols. `https://eprint.iacr.org/2018/1079.pdf`.

Supplementary Materials

A Removing the Transaction Diffusion Assumption

Recall that our definitions of batch agreement and state machine replication assumed that the following transaction diffusion assumption is respected with probability 1: if some forever honest node receives a transaction tx as input in some round t, then all so-far nodes must have received tx as input by the end of round $t + \Delta$. In other words, honest nodes observe any transaction within Δ rounds from each other. In this section, we show how to remove this assumption while preserving communication efficiency,

It is sufficient to describe how to remove the assumption in an $\mathcal{F}_{\text{mine}}$-hybrid world. We can then remove the $\mathcal{F}_{\text{mine}}$ assumption using adaptively-secure cryptography as we explained in Section 7.

Message types. We introduce two types of messages, `echo`, and `notify`:

- `echo` messages do not need to be mined. In our protocol honest nodes will flip local coins to probabilistically elect themselves to echo every new transaction they see;
- `notify` messages need to be mined for the message to be valid. If a node i calls $\mathcal{F}_{\text{mine}}.\texttt{mine}(\texttt{notify}, \mathsf{m})$ and the outcome is successful, then node i is eligible for echoing the message m by sending $(\texttt{notify}, \mathsf{m} : i)$.

  The mining difficulty for `notify` messages is set such that each mining attempt is successful with probability $\frac{\lambda}{n}$ where n is the total number of nodes and λ is large enough such that $\epsilon^2\lambda = \omega(\log \kappa)$. Without loss of generality we may assume that $\lambda < n$.

Diffusion protocol. We can adopt the following protocol to realize the transaction diffusion assumption, blowing up the effective network delay by only a polylogarithmic (in κ) factor:

For every transaction tx a node has newly observes, it immediately initiates a "diffusion attempt" for tx:

1. Whenever a node i sees a new tx that has not been observed before, call $\mathcal{F}_{\text{mine}}.\texttt{mine}(\texttt{notify}, \mathsf{tx})$, and if successful, multicast $(\texttt{notify}, \mathsf{tx} : i)$.
2. At any point of time if the node has observed $\frac{2\lambda}{3}$ `notify` messages for tx from distinct nodes, end the diffusion attempt for tx.
3. When the diffusion attempt for tx first starts (i.e., when tx is first observed), flip a local random coin and echo (i.e., multicast) the transaction tx to everyone with probability $p = \frac{\lambda}{n}$. With every 2Δ amount of time that elapses (until the diffusion attempt stops), double the probability of echo and retry echoing.

Theorem A.1 (Diffusion protocol in a synchronous network)**.** *Assume that $(\mathcal{A}, \mathcal{Z})$ is $(\frac{1}{3} - \epsilon, \Delta)$-respecting where ϵ is an arbitrarily small constant and Δ is some polynomial in κ and n. There exists an appropriate polynomial $\mathsf{poly}(\cdot)$ such that except for a negligible in κ fraction of execution traces, the following holds: if ever a forever honest node observes tx in round r, then by round $r + \mathsf{poly}\log\kappa \cdot \Delta$, all so-far honest nodes must have observed tx; moreover, only $\mathsf{poly}\log\kappa \cdot |\mathsf{tx}|$ bits of honest messages will have been multicast in total.*

Proof. Fix some tx. We first prove the following useful claim.

Claim A.2. *If ever a so-far honest node successfully sends an* `echo` *or a* `notify` *message for* tx*, then except with negligible probability, in* 2Δ *rounds, every so-far honest node will have terminated their diffusion attempt for* tx*.*

Proof. A so-far honest node would send its `notify` message to everyone. Thus in Δ rounds, all forever honest nodes would attempt to mine a `notify` message for tx. In another Δ rounds, except with negligible probability, every so-far node would have seen $\frac{2\lambda}{3}$ `notify` messages for tx. □

We now continue with the proof of Theorem A.1. Even if only a single forever honest node, denoted i, observes tx by round r, since it doubles its probability of echoing every 2Δ rounds till it terminates, in $\mathsf{poly}\log\kappa\cdot\Delta$ rounds, one of the following must have happened except with negligible probability:

- node i has terminated its diffusion attempt for tx — in this case node i must have observed $\frac{2\lambda}{3}$ `notify` messages for tx, and by Chernoff bound, except for a negligible fraction of the execution traces, one of these `notify` messages must have been sent by a forever honest node.
- node i has sent echo for tx.

In either case, we have that by round $r+\mathsf{poly}\log\kappa\cdot\Delta$, some forever honest node must have sent a `notify` or an `echo` message for tx. It follows by Claim A.2 that by round $r+\mathsf{poly}\log\kappa\cdot\Delta+\Delta$, all so-far honest nodes must have terminated their diffusion attempt for tx for some suitable polynomial $\mathsf{poly}(\cdot)$.

It remains to prove communication efficiency. Suppose that in some execution trace, so-far honest nodes try to send `echo` message for tx with probabilities $p_1, p_2, \ldots$ over time. Let $\mu(r)$ be the sum of these probabilities by some round r. Due to the Chernoff bound, except with negligible probability, when $\mu(r^*)$ first reaches $\mathsf{poly}\log\kappa$ in some round r^* for some fixed polynomial $\mathsf{poly}(\cdot)$, there is at least one honest successful attempt at echoing tx — thus by Claim A.2, in another 2Δ steps, every so-far honest node will have terminated their diffusion attempt for tx.

Also due to Chernoff bound, by this round r^*, the total number of honest echo messages for tx multicast by round r^* cannot exceed $2\mathsf{poly}\log\kappa$. The communication efficiency claim then follows by observing that the total honest echo messages for tx multicast by round $r+2\Delta$ cannot be more than $4\times$ larger than by round r; and moreover, by Chernoff bound, except with negligible probability, the total number of honest `notify` messages for tx is upper bounded by 1.1λ. □

B Additional Comparison with Related Work

As mentioned, the recent work by Abraham et al. [2] achieves adaptive security and communication efficiency without erasures or PoW, but their approach works only for agreement on a *single bit*. There does not seem to be any non-trivial method to extend their approach multi-valued agreement as we explain below. A most straightforward approach is to attempt parallel repetition of bit agreement. However, in Abraham et al. [2], to agree on each bit may require multiple epochs, and in each epoch a random node may be elected leader whose job, partly, is to propose the bit to be agreed upon. Thus, if we adopt parallel repetition, each bit agreed upon will likely be proposed by different leaders (a subset of which may even be malicious). When the multiple bits come from different leaders, there is no guarantee that their combination will be a valid value from the universe (e.g., in a real-world cryptocurrency system, a transaction need a well-formed signature to be valid).

Another straightforward attempt is to have the leader elected directly propose multiple bits. However, Abraham et al.'s technique [2] works only if the leader election is bit-specific, i.e., the coin that determines a node's eligibility for proposing the bit 0 is independent of the coin that determines its eligibility for proposing the bit 1. When the universe of valid values $\mathcal{U}$ is large, we cannot tie the value to the leader election — since otherwise the adversary can try many values from $\mathcal{U}$ and almost surely guarantee that for every epoch, some corrupt node will be elected to propose some valid value from $\mathcal{U}$ — on the other hand, Abraham et al. [2]'s technique works only if occasionally in some epoch, a single honest node makes a unique proposal.