Efficient Execution Auditing for Blockchains under Byzantine Assumptions

Researchers: Jeff Burdges[1], Alfonso Cevallos , Handan Kılınç Alper , Chen-Da Liu-Zhang[1], Fatemeh Shirazi[1], Alistair Stewart[1],
Implementors: Alex Gheorghe[2], Rob Habermeier , Robert Klotzner[2], Bastian Köcher[2], Andronik Ordian[2], Maciej Kris Żyszkiewicz[2], and Andrei Sandu[2]

[1]Web3 Foundation
[2]Parity Technologies

June 14, 2024

Abstract

Security of blockchain technologies primarily relies on decentralization making them resilient against a subset of entities being taken down or corrupt. Blockchain scaling, crucial to decentralisation, has been addressed by architectural changes: i.e., the load of the nodes is reduced by parallelisation, called sharding or by taking computation load off the main blockchain via rollups. Both sharding and rollups have limitations in terms of decentralization and security.

A crucial component in these architectures is a layer that allows to efficiently check the validity of incoming blocks in the system. We provide the first formalization and analysis of ELVES, the auditing layer that is currently deployed in the Polkadot and Kusama blockchains.

In this layer, "auditing committees" are formed independently for each block, and security relies on the fact that it is prohibitively expensive in expectation for an adversary to make ELVES to accept a block that is not valid. In addition, ELVES has the following characteristics: 1) Auditing committees wind up orders of magnitude smaller than pre-assigned committees. In fact, the size of the committees adapts automatically to network conditions but remains a low constant in expectation, in the order of tens or low hundreds; 2) Although synchronous per se, ELVES tolerates instant adaptive crashes, mirroring realistic network capabilities. Surprisingly, the committee-size analysis of our protocol is 'all but simple' and involves a novel strengthening of Cantelli's inequality, which may be of independent interest.

1 Introduction

Blockchains relieve us from having to trust single entities by offering decentralized solutions to agree on valid blocks of data. Block auditing, ensuring that we agree on valid and available blocks, is often one of the main bottlenecks. The naive approach for block auditing, where all blocks are acquired and executed by every node, requires significant computation power. Similarly, bandwidth is limited and storage is expensive on blockchains because all nodes bear these costs. The computational overhead constitutes a major bottleneck since it increases the machines' computational requirement capabilities, limiting the number of eligible participants, and therefore impacting decentralization and also the security and resilience of blockchains.

Our solution, ELVES: We propose an efficient block auditing layer ELVES[1], that dramatically reduces the required resources, and opportunity costs. At a very high level, ELVES operates in several phases (see a precise description of its underlying assumptions and security guarantees in Section 1.1).

[1]ELVES stands for "endorsing light validity evaluator system".

In the first "backing" phase, some initial auditor(s) called "backers" receive the block, check the block, and then sign a claim that the block is valid.

In the second "availability" phase, the backers distribute the block so that any auditor could check the block if they so choose. We keep this distribution process bandwidth efficient by giving each auditor one symbol from a Reed–Solomon encoding of the block. The validators agree that the block has been distributed in a way that ensures that the block can be reconstructed under Byzantine assumptions.

In the third "approvals" phase, we randomly assign an initial set of auditors to check the validity of the block, which then randomly grows whenever assigned auditors do not respond soon enough.

As a fourth phase, we finalize blocks only if and when the approvals phase concludes, including having every absent auditor replaced by enough additional random auditors. If, however, any auditors claims the block is invalid then all auditors check in a "disputes" phase, which resolves by a regular Byzantine vote. If a block is found invalid, then we eject its backing auditors permanently (in a proof-of-stake protocol, the stake backing these auditors should be confiscated e.g. with 100% slashing in Polkadot).

In the approvals phase, we choose these initial and growth parameters to balance costs, liveness, security/soundness, and network assumptions, so the existance of acceptable choices represents the primary technical result of this paper. At a high level, we only need them large enough to make attacks prohibitively expensive in expectation, thanks in large part to our ejection of bad auditors and delaying finality.

Existing architectures propose multiple approaches to solve the security/ performance trade-off [EGSVR16, KJG+16, LNZ+16a], *sharding* being a promising approach. In sharding, the execution is parallelized by splitting the validators into subsets that check the validity of blocks of that shard. Intuitively, the benefit of sharding is that the load to process each block is reduced, and the total throughput capacity of the system increases proportionally with the number of validators. Typical sharding approaches [LNZ+16a, ZMR18, KKJG+18] require a high number of validators per shard to preserve security, in the order of several hundreds or thousands of validators, since they rely on concentration bounds to ensure that every shard contains an honest majority with high probability. Our results show that our auditing layer improves upon current sharding solutions in terms of the number of validators that check a block (which is denoted as the shard size in a sharding architecture).

Another popular approach to tackle scalability are rollups that delegate the execution of blocks to outside the set of validators. As such, the bulk of the computation and storage is off-chain. In optimistic rollups (e.g. Arbitrum [KGC+18]), it is assumed that the off-chain blocks are valid and that there is a fraud-proving scheme to detect cases where blocks are not computed correctly. Even though in principle "any party" can attempt to check and detect invalid blocks, it is not clear how such schemes ensure provable security from a formal standpoint, since it is not specified how to ensure that the blocks are in fact checked.

It is worth additionally noting that current solutions for optimistic rollups and sharding are not resilient to adaptive crashes that may be deployed relatively fast and at large-scale: an adversary can simply perform a distributed denial-of-service (DDoS) attack to crash many or all validators of a shard, and the system loses liveness, or worse, safety. Our solution achieves safety and liveness under a fully adaptive adversary that can instantaneously crash parties.

Finally, it is worth mentioning the approach of so-called zero-knowledge rollups, where a cryptographic proof of validity for a batch of transactions is published towards the on-chain validators, who can verify it. However, producing such proofs of validity is computationally expensive (i.e., 50000 times slower than natively execution in current systems [Tha]) and this may lead to centralization of proving.

1.1 Our Contributions

We introduce ELVES, a novel provably-secure scaling auditing protocol that can be used to ensure the availability and validity of blocks. Our ideas and analysis can be applied to improve the scalability of current blockchain architectures (sharded or not sharded), and can also be applied to efficiently add a method to audit blocks in the context of optimistic rollups, while guaranteeing security.

The ELVES protocol has been successfully used in production on the Kusama and Polkadot blockchain networks since 2021, but no formal security proof of it has been provided so far.

The protocol achieves a minimal load per validator and is resilient against static Byzantine corruptions[2] and fully adaptive crashes controlled by a *rational* adversary. More concretely, in our protocol parties put down a collateral which can be slashed (i.e., lost) if misbehavior is detected, and the security of the protocol relies on the fact that the adversary behaves rationally, according to strategies that maximize their expected income.

Our design has security against a fraction of up to γ total corrupted parties, $0 \leq \gamma < 1/3$, under a synchronous network,[3] with access to a global state-machine replication (the primary chain). The protocol selects parties to check the validity of a block in a randomized manner, and achieves the following features.

Resilience to Fully Adaptive Crashes. ELVES is the first block-auditing solution that is safe and live under a fully adaptive adversary that can instantaneously crash parties. More concretely, ELVES ensures that 1) no rational adversary that follows strategies leading to non-negative expected profit can make the protocol accept invalid blocks and 2) if the initial auditing party receiving a valid block as input is honest, then every validator agrees that the block is valid.

Note that previous approaches to validate a block based on sharding immediately lose safety or liveness, since the adversary can simply adaptively crash all members of a shard and stall all the blocks that are checked by that shard.

Concrete Committee Sizes. ELVES achieves an optimal throughput by imposing a minimal load per party in the system. Given n parties and considering a period where the system processes a linear number $O(n)$ of blocks of length ℓ (which should correspond to a constant-time period of a few minutes), the average communication complexity incurred per block is $O(\ell + n\log(n))$ plus the cost of broadcasting $O(n)$ signatures on the primary chain, and the bandwith (per-party communication) is a $1/n$ fraction of it. In order words, the (amortized) communication is roughly asymptotically optimal.

Turning to concrete numbers, the incurred committee sizes are very small. Figure 1 shows an upper bound on the expected committee size per block (when n blocks are processed), as a function of the selected soundness failure probability ε and corruption bound γ. As we can see, our protocol achieves concretely efficient committee sizes for all reasonable parameters. In particular, for $\varepsilon = 2^{-60}$ and 33% of corruptions, the expected committee size is at most 466, improving over prior approaches such as Gearbox [DMM+22], which require a committee of around 1000.

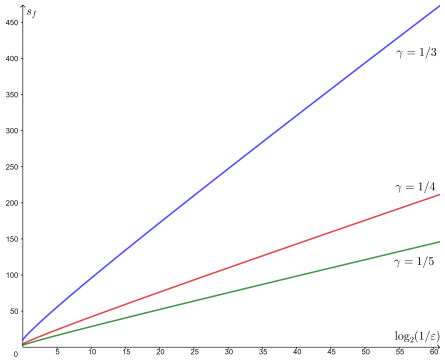

Figure 1: Bound on expected committee size per block, s_f, when n blocks are processed, as a function of the failure probability ε, for three values of corruption bound γ.

When considering rational adversaries, we show that one can in fact settle for a non-negligible soundness failure probability $\varepsilon \approx (20n)^{-1}$, making the expected committee size $s_f = O(\log n)$; see Section 9. Furthermore, the expected committee size not only depends on the fixed corruption bound γ assumed, but it

[2]Our protocol also works against mildly-adaptive Byzantine adversaries, that become corrupted after a sufficiently large delay, as long as the time it takes to corrupt a party is longer than the time it takes a block to be validated.

[3]Even though our protocols are designed in the synchronous network model, they tolerate fully adaptive crashes and therefore address some of the challenges that appear in partially synchronous or asynchronous networks.

automatically adapts to the current fractions α and β of static Byzantine corruptions and adaptive crash corruptions, respectively, with $\alpha + \beta \leq \gamma$. This leads to an even smaller committee size in expectation under typical network conditions, as shown in Figure 2. As an example, in an instance with $n = 1000$ parties, we argue that $\varepsilon = 20001^{-1}$ is secure enough, obtaining an expected committee size of at most 35 when there are up to 5% corruptions of any type, up to 97 with a 1/3 fraction of static Byzantine corruptions, and up to 129 with a 1/3 fraction of adaptive crash corruptions.

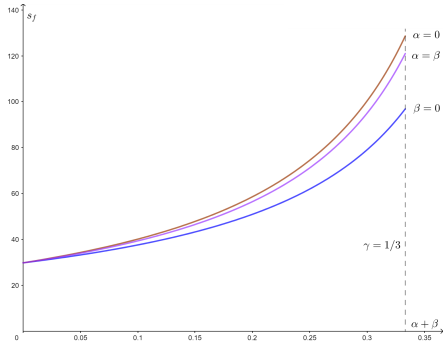

Figure 2: Bound on the expected committee size s_f as function of current fractions α and β of static Byzantine and adaptive crash corruptions respectively, when n blocks are processed, with $\gamma = 1/3$ and $\varepsilon^{-1} = 20001$.

In Section 9 we provide further discussion on Figures 1 and 2. Finally, we provide in our analysis a strengthening of Cantelli's inequality, which may be of independent interest; see Lemma 5.

2 Technical Overview

Model and Architecture. We consider a security model with n parties, where at any point in time a party can be honest or corrupted. Corrupted parties are controlled by a central adversary, who can perform two types of corruptions: up to an α fraction of *active* corruptions and also up to a β fraction of *crash* corruptions. Actively corrupted parties are fully controlled by the adversary and chosen statically, at the onset of the protocol. And crashed parties can be adaptively and instantaneously corrupted at any point in time. Crashed parties cannot receive or send messages. We assume that $\alpha + \beta \leq \gamma$, where $\gamma < 1/3$ is a fixed corruption bound.

The parties have access to a complete network of point-to-point authenticated channels, a diffusion network functionality to gossip messages, as well as a state machine replication system (the primary chain). The goal is to provide an auditing layer that checks the validity of incoming blocks in an efficient manner. Such a layer could then be employed on top of existing scaling solutions to support multiple independent chains.

We assume that the input to the auditing layer contains information about the block to verify: a) a header h, and b) a *proof of validity* $\pi(h)$, containing enough information to allow to efficiently check the validity of h.

The protocol then has a randomized procedure to select a subset of validators to check the validity of the block. Such validators can be thought of as auditors that check the validity of an input block. Let us first consider known approaches to audit blocks.

Strawman Approach. In the strawman approach, one simply randomly chooses a sufficiently large committee to check the block using, e.g., verifiable random functions (VRFs), so that with high probability the committee contains a two-thirds fraction of honest parties. The committee then takes a two-thirds majority decision on whether the block is valid. Unfortunately, this approach requires a substantially large committee size, in the order of high hundreds to thousands, so that concentration bounds apply. In order to improve the concrete efficiency, we will use the following two techniques.

Gradual Optimistic Paths. First, note that the strawman approach requires large sizes even when the actual number of corruptions is very low. In fact, this is true even when all parties are honest.

A simple idea to get around this is to first choose optimistically a smaller committee to check the block, where at least one honest party is elected with high probability. If there is no unanimous agreement, we revert back to the strawman approach with a larger committee and a two-thirds majority vote. By doing so, under optimistic conditions only the smaller committee is deployed. One can further develop this idea by adding parties across several steps: rather than falling back to the strawman approach directly, one can add parties gradually until the committee size reaches the strawman approach. In fact, such a multi-step idea has been recently proposed in the Gearbox protocol [DMM+22], which follows the safety-liveness dichotomy by guaranteeing safety in each step, but liveness only in sufficiently large committees, depending on the current level of corruption, with the very last step ensuring liveness for a 1/3 fraction of corruptions.

Unfortunately this protocol does not achieve liveness under fully adaptive crashes. This is because the adversary can trivially prevent a block from being checked by corrupting its elected committee in every step, as the committees remain small in size relative to the adversary's budget. To solve this, we first make the simple observation that one can keep increasing the committee sizes beyond a constant bound, until the adversary does not have enough budget to prevent liveness.

Even though the above approach with gradual optimistic paths is substantially better than the strawman approach, the fact that the first committee is chosen so that it contains at least honest party with high probability, still leads to committee size numbers that are substantial in practice. One would optimally expect to be able to settle with even smaller numbers, at least in the optimistic case.

Rational Adversary and Slashing Collateral. Choosing even smaller committees poses an important technical barrier, given that with a noticeable probability, the committee might entirely consist of actively corrupted parties, and therefore the security for that block is compromised.

To solve this problem, we consider the setting where each party puts a certain amount of tokens as collateral, and there is a rational adversary that follows strategies leading to higher income. The idea is to have a designated party, called the *backer*, be in charge of checking and distributing the block before the initial committee of checkers is revealed. This way, a corrupted backer runs the risk of getting caught and losing its collateral in the event of distributing an invalid block, since the initial committee is likely to contain an honest party. We then set a non-negligible soundness failure probability, and a collateral level, such that if having an invalid block be approved by the protocol earns the adversary a quantity M (say, the whole market capitalization of the tokens secured by the blockchain), its expected income is still negative.

This also means that, contrary to current protocols (e.g., in sharding) which elect fixed committees for some interval of time, we have to resample fresh committees for each processed block, so that the backer is committed to its decision **before** gaining any information on the composition of the corresponding initial committee. Similarly, we require that two blocks processed around the same time are distributed by different backers, each with its own collateral, so that the adversary gains no advantage from running multiple attacks simultaneously. To solve this last issue, we propose to make all n parties be collateralized and take turns acting as backers.

At a high level, our protocol consists of the following phases:

1. **Availability Distribution**: The backer distributes the initial block B towards all parties. More precisely, the backer uses an erasure coding with a reconstruction threshold of a 1/3-fraction of the parties, obtaining codewords $s_1, \ldots, s_n$, and also encodes them into a Merkle tree, obtaining a proof π_i for each codeword s_i. Each pair (s_i, π_i) is sent bilaterally to each party P_i, who then publishes a vote stating whether the received pair was correct. This phase ends when a set S of 2/3 fraction of parties publish their votes.

2. **Committee Elections and Checks**: After the block has been distributed, we use VRFs to elect an initial auditing committee C_0, to which each party is assigned independently with fixed probability, so that the expected size of C_0 is small, in the order of tens. Each party in C_0 publishes a proof of their assignment, receives in turn codewords from parties in S, reconstruct and audit block B, and publish their outcome. There are three options at this point: 1) some party claims the block is invalid, 2) all

parties that announced their assignment claim the block is valid, or 3) there are some *no-show* parties that announced their assignment but did not publish an outcome of their audit on time.

In the first case, the check escalates to all parties, a majority vote is performed on whether the block is valid, and parties on the minority side are slashed, with the backer included on the side that supports the block. In the second case, the block is accepted. In the third case, for each no-show party, a new *tranche* (subset) of parties of constant expected size is elected and added to the auditing committee, and rules 1 through 3 are applied again to the newly elected parties.

After the Availability Distribution, we know that among the parties that published votes, there is a 1/3-fraction of total parties that are honest. The codewords from honest parties completely determine a unique block B', with the guarantee that $B' = B$ if the backer is honest. For a block of length ℓ, the total communication cost of this step is $O(\ell + n\log(n))$ bits of communication plus the cost of publishing $O(n)$ votes (each with size independent of the block length); and the bandwith (per-party communication) is a $1/n$ fraction of it. As an example, for $n = 1000$ parties and a 1/3 fraction of corruption, blocks of size 1 MB and Merkle tree hash size of 256 bits, the bandwidth is expected to be around 3.32 KB.

Security is ensured against a rational adversary given that the committees are revealed after the backer distributes the block. Moreover, every no-show is replaced by several other parties, and as mentioned, we set the collateral and the probability that the block is accepted so that the expected adversarial income is negative. Our analysis shows that the load per node remains constant in the number of blocks to process. The total cost of the Committee Elections and Checks step, assuming that the final committee size is $\mathrm{s_f}$, corresponds to $O(\mathrm{s_f} \cdot (\ell + \log(n)))$ plus the publication of $\mathrm{s_f}$ VRF election claims and votes of validity.

Trade-offs Between Committee and Tranche Sizes. The idea of replacing each no-show auditor by a tranche of several new auditors is to protect the system against an adversary that has backed an invalid block, and is crashing honest auditors as soon as they announce their assignments. In this case, we want to cause a chain reaction that makes the committee grow quickly and extinguishes the adversary's budget to crash parties. Intuitively, for the same level of security, a larger tranche size alows for a smaller initial committee size, because larger tranches make it less likely that this chain reaction dies out early on. Figure 3 shows a tradeoff on concrete numbers for the expected initial committee size $\mathrm{s_0}$ compared to the expected tranche size s_δ per no-show, for an instance with a corruption bound of $\gamma = 1/3$ and a soundness failure probability $\varepsilon = 1/20001$. However, we also need to ensure that the tranche size is small enough not to trigger frequent chain reactions on valid blocks. We prove in Section 9 that the optimal value for the expected tranche size s_δ is between 2 and 2.5.

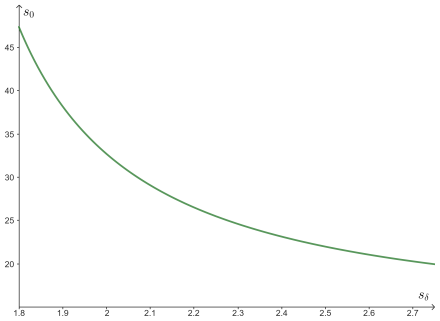

Figure 3: Expected initial committee size $\mathrm{s_0}$ as a function of the expected tranche size s_δ per no-show, for an instance where $\gamma = 1/3$ and $\varepsilon^{-1} = 20001$.

2.1 Related Work

The most related literature that focuses on techniques for choosing a small subsets of validators to check the validity of blocks is the sharding literature. Many works have investigated sharding approaches in the

literature. We refer to [WSNH19] and [LWZ23] for extensive surveys on sharding protocols in the literature.

One of the first sharding protocols, Elastico, was proposed in [LNZ+16b], where the authors introduce a synchronous protocol for a 1/4 fraction of static corruptions,[4] that uses a Proof-of-Work approach to elect the committees. However, the approach has a high failure probability when the committee size is around 100 parties. The work [KKJG+18] introduced the OmniLedger protocol, which gives improved security compared to Elastico for the same parameters, and considers *mildly adaptive* corruptions, which are corruptions that only take place after a prescribed number of epochs. The protocol Rapidchain [ZMR18] improved the corruption threshold, compared to the above protocols, to up to a 1/3 fraction of corruptions, while maintaining efficiency. Monoxide[WW19] is an asynchronous proof-of-work protocol that uniformly partitions the space of user addresses into shards (zones) according to the first fixed number of bits. Every party is permanently assigned to a shard uniformly at random and invokes an independent consensus protocol.

A framework to formalize sharding protocols was introduced in [AKKW19], and where the mentioned sharding protocols in the literature are analyzed. In particular, the work studies trade-offs between the achievable efficiency of a sharding protocol, compared to the speed the adversary can corrupt the parties.

The protocol Gearbox [DMM+22] improved the shard sizes by leveraging the safety-liveness dichotomy. In particular, the protocol starts choosing a smaller committee size that ensures safety (but possibly sacrificing liveness), but increasingly chooses larger committees to achieve liveness as well. The protocol works under the partially synchronous model and considers static corruptions. Our protocol ELVES takes this basic idea a step further and differs in two aspects: 1) Gearbox does not achieve liveness against an adaptive crash-adversary, since the adversary can simply crash all parties in a shard; this is fixed in a straightforward way in our protocol by scaling the shard size possibly towards all parties, if not enough valid responses are received; and 2) more importantly, our protocol achieves smaller committee sizes. In particular, for an error of 2^{-60} and 1/3 fraction of corruptions, our committee size is around 400 and Gearbox requires 1000 parties. Even more interestingly, our analysis shows that assuming a model where parties put down a collateral and security against a rational adversary (see Section 8), it is enough to settle for larger errors, in the order of 1/20001, drastically reducing the committee sizes to the order of less than 150 (see Figure 2).

New sharding protocols [LLZW23, XZD+23] have recently appeared achieving further trade-offs between security and efficiency. To the best of our knowledge, all the proposed sharding protocols lose their security guarantees (safety or liveness) under fast DDoS attacks, i.e., when considering a fully adaptive adversary that can instantaneously crash parties. We introduce the first protocol that is secure under such adversaries. Moreover, our analysis shows that our protocol incurs very small committee sizes in the presence of a rational adversary.

3 Preliminaries

We introduce preliminaries for the basic primitives used in our protocols.

Digital Signatures. A digital signature scheme consists of a tuple of three algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$, defined as follows:

1. $\mathsf{KGen}(\kappa)$ is the key generation algorithm that takes the security parameter and outputs the verification/signing key pair $(\mathsf{vk}, \mathsf{sk})$.
2. $\mathsf{Sign}(\mathsf{sk}, m)$ is a signing algorithm that takes as input a signing key and a message and outputs a signature σ.
3. $\mathsf{Ver}(\mathsf{vk}, m, \sigma)$ is a verification algorithm that takes as input the verification key, message and signature. The algorithm outputs 1 if σ is a valid signature on m under the verification key vk, and outputs 0 otherwise.

[4] The protocol can handle up to 1/3 fraction of corruptions, with decreased performance.

We require unforgeability, which guarantees that a PPT adversary cannot forge a fresh signature on a fresh message of its choice under a given verification key while having access to a signing oracle (that returns a valid signatures on the queried messages).

Erasure Correcting Code Schemes. We make use of erasure-correcting code schemes that tolerate a certain number of erasures.

Let n be the number of shares and t the number of erasures that can be tolerated. A (t, n) erasure-coding code scheme (ECCS) consists of two algorithms:

1. Enc takes a message m and produces a sequence of shares $s_1, \ldots, s_n$.
2. Dec takes a sequence of shares $s'_1, \ldots, s'_n$ such that for at least $n - t$ of the shares $s'_i = s_i$ and for the remaining shares $s'_i = \perp$, then the original message m is output.

We use standard Reed-Solomon codes [RS60] to instantiate a (t, n)-ECCS efficiently, where the size of each share is $O(\frac{\ell}{n-t})$, for a message of size ℓ bits.

4 Model

Our model consists of a set of parties $\mathcal{P}$ and a probabilistic polynomial time (PPT) adversary $\mathcal{A}$. Each party has the capability to communicate with one another, in accordance with a specified underlying network assumption.

Adversarial Model. We consider a mixed adversary $\mathcal{A}$ that can perform two types of corruption.

- **Active Corruption:** $\mathcal{A}$ can actively corrupt up to a fraction α of parties in $\mathcal{P}$. An actively called party, called malicious party, behave arbitrarily as prescribed by $\mathcal{A}$.
- **Crash Corruption:** $\mathcal{A}$ can crash up to a fraction β of parties in $\mathcal{P}$. Crashed parties cannot send or receive any further message from the time they were crashed. The adversary does not have access to the internal state of crashed parties.

We consider that actively corrupted parties are chosen statically, i.e., the adversary chooses the parties to actively corrupt at the start of the protocol. On the other hand, crash corruptions can be chosen fully adaptively, i.e., the adversary can instantly choose to corrupt a party during the protocol execution.

4.1 Clocks and Network

Parties have access to synchronized clocks. In addition, parties have access to a complete network $\mathcal{F}_{\mathtt{p2p}}^{\Delta}$ of point-to-point authenticated channels and a diffusion network functionality $\mathcal{F}_{\mathtt{flood}}^{\Delta}$ with known delay Δ. In the complete network of point-to-point channel functionalities, the adversary chooses when a message is delivered towards an honest recipient (within the prescribed delay). A crashed recipient is guaranteed to receive the messages when the delay is reached and the recipient becomes honest.

The diffusion functionality achieves the following properties:

1. If any party P that is honest at time τ inputs a message m to $\mathcal{F}_{\mathtt{flood}}^{\Delta}$, it gets delivered to all other parties that are honest by time $\tau + \Delta$. Crashed parties at time $\tau + \Delta$ are guaranteed to receive the message once they become honest.
2. If a party P is honest and receives a message m at time τ, then every other party that is honest at time $\tau + \Delta$ receives the same message. Crashed parties at time $\tau + \Delta$ are guaranteed to receive the message once they become honest.

4.2 State Machine Replication Resource

Parties have access to a resource $\mathcal{F}_{\mathtt{smr}}^{\Delta}$ (a.k.a., the primary chain). We model the resource as a *state machine replication* system that allows each party P_i to input a message and also keeps track of an array of messages $\mathsf{smr}_i = (\mathsf{smr}_i[1], \mathsf{smr}_i[2], \dots)$, one per index, that is output to party P_i, with the following security guarantees:

1. Δ-Liveness: If a message m is input by an honest party at time τ, then m appears in every array smr_i that belongs to an honest party P_i by time $\tau + \Delta$. Moreover, if a block m appears in an honest party P_i's array smr_i at time τ, then it also appears in every other honest party's array by time $\tau + \Delta$.
2. Safety: If an honest party outputs a message m in index i, then for all honest parties that have an output m' at index i, we have that $m' = m$.
3. Completeness: Every honest party outputs a message in all indices.

5 Availability Scheme

The first part of our protocol consists of a party distributing the input message block via a so-called *availability* scheme, which allows a party to distribute a block among all parties and make it available. More concretely, it has the guarantee that parties reach agreement on whether an initial message has been distributed, and if so, either the message can be consistenly reconstructed by any recipient R, or any recipient agrees that the dealer was actively corrupted.[5]

Syntax. An availability scheme consists of a pair $(\mathsf{Dist}, \mathsf{Rec})$ of distributed protocols. In protocol Dist, a designated party P, the dealer, holds an input message m to be distributed. At the end of the protocol, all parties agree on whether the protocol was successful, and if so, each party P_i outputs an encoded data piece (s_i, π_i) and a proof of correctness. In protocol Rec, each party P_i publishes its pair (s_i, π_i) towards the recipient R, who applies a function to reconstruct the original message m.

Security Guarantees. We will require that the scheme satisfies the correctness and soundness properties as defined below.

Definition 1. *A secure availability scheme consists of a pair of distributed protocols* $(\mathsf{Dist}, \mathsf{Rec})$*, where a designated dealer P holds an input m for the protocol* Dist *and every party agrees on whether* Dist *is successful, and afterwards parties run protocol* $\mathsf{Rec}(R)$ *towards any designated recipient R who can receive an output m', with the following guarantees.*

- *Correctness: If the dealer is honest, protocol* Dist *is successful.*
- *Soundness: If protocol* Dist *is successful, then there is a unique value m' such that either 1) in any execution of protocol* $\mathsf{Rec}(R)$*, R outputs m', or 2) in any execution of* $\mathsf{Rec}(R)$*, R outputs $\perp$, indicating that the dealer is corrupted. Moreover, if the dealer is honest, $m' = m$.*

5.1 Protocol Description

We describe a simple scheme, where parties have access to the functionalities $\mathcal{F}_{\mathtt{smr}}^{\Delta}$ and $\mathcal{F}_{\mathtt{p2p}}^{\Delta}$, and also a plain PKI infrastructure for digital signatures. For protocol Dist, the initial dealer P encodes the data m using an $(\gamma n, n)$-Reed-Solomon code, obtaining pieces $(s_1, \dots, s_n)$, and encodes the pieces using a Merkle tree, computing the path from each piece s_i to the root as the proof π_i. Party P then sends to each party P_i the pair (s_i, π_i) and publishes the commitment com in $\mathcal{F}_{\mathtt{smr}}^{\Delta}$.

Each party P_i, once they see com published in the SMR resource, waits Δ time to receive the pair (s_i, π_i). If received, P_i checks that the proof is correct, and if so, publishes a signed vote message v_i to $\mathcal{F}_{\mathtt{smr}}^{\Delta}$ indicating that P_i received a correct data piece. After Δ time, if $(1-\gamma)n$ signed votes appear in the SMR resource, the protocol is considered successful.

[5] This is similar to *weak secret sharing* (see, e.g., [Rab94]), except that there is no privacy requirement.

Protocol $\Pi^t_{\texttt{Dist}}$

A designated party P, the dealer, has an input m.

Encoding and Proofs

1: P computes a Reed-Solomon encoding of m with reconstruction threshold $\gamma n + 1$, obtaining data pieces $(s_1, \ldots, s_n)$. Furthermore, it also computes the path π_i for each codeword s_i to the root com. P then sends (s_i, π_i) to party P_i using the network $\mathcal{F}^{\Delta}_{\texttt{p2p}}$ and publishes com in $\mathcal{F}^{\Delta}_{\texttt{smr}}$.

Codeword Check and Availability Distribution

1: Each party P_i waits until com appears in $\mathcal{F}^{\Delta}_{\texttt{smr}}$. It then waits Δ time to receive pair (s_i, π_i). If received, P_i checks that the proof is correct, and if so, publishes a signed vote message v_i to $\mathcal{F}^{\Delta}_{\texttt{smr}}$.
2: After Δ time, if $(1-\gamma)n$ signed votes appear in $\mathcal{F}^{\Delta}_{\texttt{smr}}$, the protocol is considered successful.

For the protocol Rec towards a recipient R, each party P_i sends (s_j, π_j) to R. Upon receiving at least $\gamma n + 1$ correct pieces from parties that output a vote on $\mathcal{F}^{\Delta}_{\texttt{smr}}$, recipient R interpolates the rest of the pieces and checks if they are also consistent with the root. If so, it computes message m using Reed-Solomon decoding. Otherwise, it outputs $\perp$.

Protocol $\Pi^t_{\texttt{Rec}}$

A designated party R, the recipient, wants to reconstruct an output after $\Pi_{\texttt{Dist}}$ was successful.

Reconstruction

1: Each party P_i sends (s_i, π_i) to R.
2: Let S be the set of parties that output a signed vote in $\mathcal{F}^{\Delta}_{\texttt{smr}}$. Party R, upon receiving $\gamma n + 1$ pairs (s_i, π_i) from parties in S, that contain consistent paths from the codeword to the root, uses Lagrange interpolation to reconstruct all codewords $(s_1, \ldots, s_n)$, and checks if this tuple is consistent with the root published in $\mathcal{F}^{\Delta}_{\texttt{smr}}$. If so, R uses Reed-Solomon decoding and outputs the decoded message m'. Otherwise, it outputs $\perp$.

Lemma 1. *Protocol $(\Pi^t_{\texttt{Dist}}, \Pi^t_{\texttt{Rec}})$ is a secure availability scheme in the presence of up to a total of $\gamma < 1/3$ fraction of corruptions that can be static active or adaptive crashes.*

Proof. We first argue Correctness. If the dealer P is honest, then P computes a correct Reed-Solomon encoding of m and the root appears in $\mathcal{F}^{\Delta}_{\texttt{smr}}$. Moreover, every honest party receives a valid pair (s_i, π_i) after time at most Δ and publishes a signed vote in $\mathcal{F}^{\Delta}_{\texttt{smr}}$. Therefore, after Δ time, at least $(1-\gamma)n$ signed votes appear and the $\Pi_{\texttt{Dist}}$ is considered successful.

We now argue Soundness: assume that $\Pi_{\texttt{Dist}}$ was successful. In this case, at least $(1-\gamma)n$ signed votes appear in $\mathcal{F}^{\Delta}_{\texttt{smr}}$, and therefore at least $(1-2\gamma)n \geq \gamma n+1$ votes come from honest parties. Let us denote these honest parties by $\mathcal{H}$. During the reconstruction phase $\Pi_{\texttt{Rec}}$, each of these honest parties sends their correct pairs (s_i, π_i) to recipient R. These pairs $\{s_i\}_{i\in\mathcal{H}}$ uniquely define a tuple $(s'_1, \ldots, s'_n)$ lying on a degree-γn polynomial, which can be consistent with the root com or not.

In the positive case, note that any correct pair (s_j, π_j) received from P_j that is consistent with the root com satisfies that $s_j = s'_j$, by the security of the Merkle tree. In this case, the recipient reconstructs a message m' by decoding the tuple $(s'_1, \ldots, s'_n)$ with Reed-Solomon Decoding.

In the negative case, by the security of the Merkle tree, there cannot be $\gamma n+1$ pairs that, when interpolated into a tuple $(s''_1, \ldots, s''_n)$ lying on a degree-γn polynomial, are consistent with the root com. And in this case R outputs $\perp$ and the dealer was corrupted. □

6 Approval Scheme

The second part of the protocol consists of checking whether a block that has been distributed correctly is valid. Note that even though the availability scheme ensures that an honest proposer distributes a valid block, it does not guarantee that the block distributed by a corrupted proposer is valid. The general idea of

the protocol is to choose an initial small subset of parties randomly (e.g., using VRFs) to reconstruct and audit the block. If there are contradictory claims, or no-show auditors, more auditors are assigned to the block.

Syntax. At a high level, the protocol starts after a successful execution of Dist. Therefore, each party P_i receives a private input pair (s_i, π_i), and all parties receive a public input Com, where all pairs and Com satisfy a prescribed input predicate Q. In our specific example, Com is a Merkle root and π_i is a valid proof for each codeword s_i. At the end of the protocol, each party P_i outputs a bit 0 or 1, indicating whether the distributed block is valid according to a validity predicate $\mathsf{VerifyData}$.

Security Guarantees. We require that the approval scheme satisfies the following properties.

Definition 2. *A secure approval scheme for input predicate Q and validity predicate $\mathsf{VerifyData}$, where each party P_i holds an input (s_i, π_i) and a public input Com satisfying predicate Q, satisfies the following guarantees.*

- *Completeness: Let m' be the message defined by the reconstruction procedure Rec. If m' is valid, i.e., $\mathsf{VerifyData}(m') = 1$, then every honest party outputs 1.*
- *ϵ-Soundness: Let m' be the message defined by the reconstruction procedure Rec. If m' is invalid, then every honest party outputs 0 except with probability ε.*
- *Identification: If any honest party outputs 0, then every honest party has agreement on one party that is actively corrupted.*

6.1 Committee-Election Functionality

We describe the protocol in a hybrid world assuming an ideal functionality $\mathcal{F}_{ce}^{p}$ parameterized by a success probability p. This can be realized in a standard way assuming a trusted PKI for a verifiable random function (VRF) as setup. $\mathcal{F}_{ce}^{p}$ serves as a committee-election oracle functionality, and has the following interface:

1. Selection: For input x, when party P_i calls $\mathcal{F}_{ce}^{p}$ on input x for the first time, $\mathcal{F}_{ce}^{p}$ flips a fresh p-biased bit and returns the result $b \in \{0, 1\}$, where (1 indicates success, 0 indicates failure). Calling $\mathcal{F}_{ce}^{p}$ again in the future returns the same result b.
2. Query: any party P_i can call $\mathcal{F}_{ce}^{p}$ on an input x. If P_i has already called $\mathcal{F}_{ce}^{p}$ on input x and received output b, then $\mathcal{F}_{ce}^{p}$ returns b; otherwise, it returns 0.

6.2 Protocol Description

Let γ, $0 \leq \gamma < 1/3$, be a bound on the total fraction of corrupted parties, counting both static active or adaptive crash corruptions. Party P_i starts the protocol upon seeing for the first time that the r-th message m has been successfully distributed, i.e., $\Pi_{\texttt{Dist}}$ was successful. This means that the corresponding Merkle root value Com has sucessfully been published, $(1-\gamma)n$ signed votes have appeared in $\mathcal{F}_{\texttt{smr}}^{\Delta}$, and every honest party P_i holds a pair (s_i, π_i) of codeword and proof for message m. Let us denote this time by τ_i.

The protocol elects subsets of parties to check whether the block can be reconstructed and is valid. The subsets are elected in sequential epochs, called *tranches*, with the initial subset named *tranche* 0. The protocol operates with the following parameters: s_0, the expected size of the initial committee size (i.e., of tranche 0), and s_δ, the expected number of parties elected in each subsequent tranche.

Let $p_0 = \frac{\mathrm{s}_0}{n}$ and $p_\delta = \frac{\mathrm{s}_\delta}{n}$. In the initial step, parties invoke $\mathcal{F}_{ce}^{p_0}$ on input $(r, 0)$ (denoting block number r, tranche 0), and if P_i is elected to tranche 0, set $\mathsf{elected}_{r,0}^{i} = 1$ and input (a signed claim of) it to $\mathcal{F}_{\texttt{flood}}^{\Delta}$. Let us denote by $S_{r,0}^{i}$ the set of parties that were thus elected by time $\tau_i + 2\Delta$. Note that every party P_j sees the r-th message is correctly distributed at time $\tau_j \leq \tau_i + \Delta$ and starts tranche 0. Therefore, every honest P_j's election result appears in every honest party's view in $\mathcal{F}_{\texttt{flood}}^{\Delta}$ by time $\tau_j + \Delta \leq \tau_i + 2\Delta$.

After that, for each party in $P_j \in S^i_{r,0}$, party P_i starts the reconstruction procedure towards P_j for the r-th distributed block. In particular, if P_j is honest, its election claim is included in set $S^i_{r,0}$ for each honest party P_j, and therefore P_j can correctly reconstruct the original message m.

Each P_j then checks that $\mathsf{Verify}(m) = 1$, and that the codewords, proofs and Merkle root match, and propagates the result, i.e., inputs a signed claim into $\mathcal{F}^{\Delta}_{\texttt{flood}}$. Party P_i waits for time $\Delta_0 = 3\Delta$. Let $V^i_{r,0}$ denote the approval votes for the r-th block.

There are three options at this point. Each P_i does the following:

1. Some approval votes are negative. In this case, P_i initiates a so-called *dispute phase*, where all parties check the block.

2. No approval votes are negative, but some of them are missing. Let N_i denote the number of parties that were elected but did not vote (this is the size of the set $S^i_{r,0} \setminus V^i_{r,0}$). In this case, party P_i participates in the next N_i tranches, where for each tranche P_i repeats a new instance of the same procedure but with a committee election functionality $\mathcal{F}^{p_\delta}_{ce}$ parameterized with probability p_δ.

3. All parties elected so far have a unanimous positive approval vote. In this case, P_i outputs this answer.

Protocol $\Pi_{\texttt{Appr}}(P_i)$

Let $p_0 = \mathrm{s}_0/n, p_\delta = \mathrm{s}_\delta/n$. The protocol operates in the hybrid model where parties have access to the functionalities $\mathcal{F}^{\Delta}_{\texttt{flood}}$, $\mathcal{F}^{\Delta}_{\texttt{p2p}}$, $\mathcal{F}^{p_0}_{ce}$, $\mathcal{F}^{p_\delta}_{ce}$ and also a plain PKI infrastructure for digital signatures.
We describe the protocol from the point of view of party P_i.

Main Procedure

1: Start the protocol upon seeing block number r available at time τ_i. Initialize $k = 0$, $p = p_0$, $\texttt{cur}_i = -1$, $\texttt{total}_i = 0$.
2: **while** `true` **do**
3: Query $\mathcal{F}^{p}_{ce}$ with inputs (r, j) for $\texttt{cur}_i < j \leq \texttt{total}_i$. If selected in any instance, announce this using $\mathcal{F}^{\Delta}_{\texttt{flood}}$. Let $S^i_{r,k}$ denote the elected parties by $\tau_i + 2\Delta + 7k\Delta$.
4: Send to each $P_j \in S^i_{r,k}$ the codeword-proof pair (s_i, π_i) (from the availability phase) via $\mathcal{F}^{\Delta}_{\texttt{p2p}}$.
5: If elected, execute the RecCheck procedure. Let b_i be the output, denoted as the approval vote. Input the signed approval vote to $\mathcal{F}^{\Delta}_{\texttt{flood}}$.
6: Let $V^i_{r,k}$ denote the approval votes received by $\tau_i + 5\Delta + 7k\Delta$.
7: **if** there are negative approval votes **then**
8: Execute the Dispute Escalation Procedure and output the set that was obtained from this procedure and terminate.
9: **else if** all parties in $S^i_{r,k}$ or $(1-\gamma)n$ parties approved the block r **then**
10: Output 1 and terminate.
11: **end if**
12: Wait until $\tau_i + 7(k+1)\Delta$ time. Then, execute the following if no Dispute Escalation Procedure has been triggered. Set $\texttt{cur}_i = \texttt{total}_i$. Then, set $\texttt{total}_i = \texttt{total}_i + |S^i_{r,k} \setminus V^i_{r,k}|$, $k = k + 1$ and $p = p_\delta$.
13: **end while**

Dispute Escalation Procedure

1: All parties check the block: send to all parties the pair (s_i, π_i). Upon reconstruction, check if $\mathsf{VerifyData}(m) = 1$ and output the set of parties that set their approval votes to the opposite bit.

RecCheck Procedure

1: Upon receiving at least $\gamma n + 1$ valid pairs (s_i, π_i) that match the corresponding Merkle root, reconstruct the message m and output $\mathsf{VerifyData}(m)$.

Lemma 2. *Let Q be the input predicate induced by a Merkle tree. That is, each party has a private input (s_i, π_i) corresponding to the leaf value and a Merkle tree proof with respect to a public input* Com*, the Merkle root, which all parties receive as public input. Further let* VerifyData *denote a validity predicate. Then,*

protocol $\Pi_{\texttt{Appr}}$ *is a secure approval scheme in the presence of up to a fraction* $\gamma < 1/3$ *of corruptions that can be static active or adaptive crashes.*

6.3 Security Analysis

In this section we prove Lemma 2, i.e., the security of the approval scheme when executed after protocol Dist is successful.

6.3.1 Proof of Completeness

Let m' be the message defined by the reconstruction procedure Rec, so that $\mathsf{VerifyData}(m') = 1$. Observe that any negative approval votes come from corrupted parties. Moreover, any negative approval vote triggers a dispute phase that leads to all parties checking block m, which ends up being accepted since there are at least $(1-\gamma)n > 2n/3$ valid pairs that match the Merkle root and the reconstructed message is valid. Hence, either all approval votes are positive at some point, or some votes are missing, in which case more parties are added and eventually all parties check the block and output 1 as a result of the approval process.

6.3.2 Proof of Identification

Assume that an honest party outputs 0 as a result of the approval process. In this case, the block m' that has been initially distributed is invalid and therefore the initial dealer P is corrupted. Since a dispute escalation procedure will be triggered, and every honest party will check the block and claim it is invalid, no honest party will be on the minority side. But honest parties will agree that (at least) the dealer is on the minority side.

6.3.3 Proof of Soundness

Assume that the the block that has been distributed is invalid. We want to bound by ε the probability that it gets approved during the Approval Process, even if the adversary can adaptively and instantly crash parties.

Given that there is an honest supermajority among parties, a triggered dispute will lead to all honest parties checking the block and rejecting it. And any honest party who audits the block will trigger a dispute. Hence, a necessary condition for the invalid block to be approved is that every honest auditor who declares its assignment gets immediately crashed. However, each such auditor becomes a no-show and in turn activates a new tranche, which is likely to make new honest auditors declare their assignment, and so on. Hence, the adversary wins only its budget to crash corrupt is not extinguished by the time this chain reaction stops.

We model our protocol as if it could activate an indefinite number of tranches. In particular, we allow for a party to potentially be assigned to several tranches for the same block, and similarly to be declared a no-show multiple times, each activating a new tranche. Let T be the total number of tranches that get activated in total for the block in question, not counting tranche 0, and notice that this random variable has an unbounded range. The event that a validator gets assigned to an active tranche is independent across all validators and tranches, so we can represent the validator-to-tranche assignments with a bipartite graph with n validators on one side and T tranches on the other side, where each edge is a possible assignment and corresponds to an independent Bernoulli trial with probability $p_\delta = \mathrm{s}_\delta / n$. See Figure 4. We benefit the adversary by assuming that it wins whenever the total number of assignments is finite, even if this number is arbitrarily large.

Lemma 3. *If* s_0 *and* s_δ *are respectively the expected sizes of an initial committee and of each further tranche, and at most* γn *validators can be either actively corrupted or crash corrupted by the adversary, then*

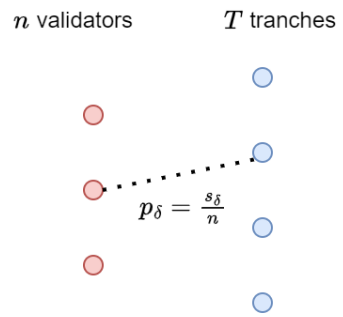

Figure 4: The validator-to-tranche assignment graph, when there are T tranches. We consider each possible assignment as an independent Bernoulli trial with probability $p_\delta = \mathrm{s}_\delta / n$.

the soundness failure probability is at most ε, provided that $(1-\gamma)\,\mathrm{s}_\delta > 1$, and

$$\mathrm{s}_0 \geq F \cdot \ln(1/\varepsilon), \quad \textit{where}$$
$$F := \frac{\mathrm{s}_\delta}{\ln\left(\frac{-(1-\gamma)\,\mathrm{s}_\delta}{W_0\left[-(1-\gamma)\,\mathrm{s}_\delta \cdot e^{-(1-\gamma)\,\mathrm{s}_\delta}\right]}\right)},$$

and where W_0 is the principal branch of the Lambert W function.

Proof. We assume pessimistically that if an auditor is corrupted in any way at assignment time, it will either approve the block, or not even announce its assignment. Conversely, an auditor who is honest at assignment time will make the announce and become crash corrupted immediately, thus becoming a no-show. Now, the number of auditors in a tranche follows a binomial distribution $\mathrm{Binom}(n, \mathrm{s}_\delta / n)$ (see Figure 4), which is well approximated by a Poisson distribution $\mathrm{Pois}(\mathrm{s}_\delta)$ when s_δ is small relative to n. Similarly, if X is the number of honest auditors in a tranche at assignment time, we can pessimistically model it as $X \sim \mathrm{Pois}(\lambda)$, for $\lambda := (1-\gamma) \cdot \mathrm{s}_\delta$, because the fraction of honest parties remains above $(1-\gamma)$ throughout the attack.

According to our model, once started the chain reaction of no-shows and tranche activations can possibly go on forever. Let q be the probability that this process eventually stops, when a single tranche is initially activated. If this tranche contains $X = i$ honest validators, this probability becomes q^i, because we will have i new no-shows, each activating a new tranche and thus starting a new independent sub-process. Hence, probability q must observe

$$q = \mathrm{E}[q^X] = e^{-\lambda(1-q)}, \tag{1}$$

where we applied the known formula for the probability generating function $\mathrm{E}[q^X]$ of a Poisson variable X. If $x := -\lambda e^{-\lambda}$, equation (1) is equivalent to

$$x = -\lambda e^{-\lambda} = -q\lambda e^{-q\lambda},$$

which, if $\lambda = (1-\gamma)\,\mathrm{s}_\delta > 1$, solves as $-\lambda = W_{-1}[x]$ and $-q\lambda = W_0[x]$, where W_{-1} and W_0 are the two real branches of the Lambert W function. Therefore,

$$q = -\frac{W_0\left[-\lambda e^{-\lambda}\right]}{\lambda}.$$

Finally, if Y is the number of honest auditors in tranche 0 at assignment time, we can also model this random variable as following a Poisson distribution, namely $Y \sim \mathrm{Pois}(\tau)$ with $\tau := (1-\gamma) \cdot \mathrm{s}_0$. If $Y = i$, there are i initial no-shows, and the probability that the committee eventually stops growing is q^i. Hence,

$$\begin{aligned}\Pr[\text{attack succeeds}] &\leq \Pr[\text{committee eventually stops growing}] \\ &= \mathrm{E}[q^Y] = e^{-\tau(1-q)} \\ &= [e^{-\lambda(1-q)}]^{\tau/\lambda} = q^{\tau/\lambda} = q^{s_0/s_T},\end{aligned}$$

where we applied equation (1). In order to bound the success probability by ε, it suffices to ensure that $q^{s_0 / s_\delta} \leq \varepsilon$. Solving for s_0, we obtain that $s_0 \geq s_\delta \cdot \ln(1/\varepsilon)/\ln(1/q) = F \cdot \ln(1/\varepsilon)$, as claimed. □

In Figure 3 we plot the above bound on the initial expected committee size s_0 as a function of the expected tranche size s_δ, for $\gamma = 1/3$ and $\varepsilon = 20001^{-1}$. We see that we can afford to have a smaller initial committee when tranches are bigger, which makes sense intuitively because in this case the chain reaction is less likely to stop early on, hence the attack success probability decreases.

7 ELVES Protocol

In this section, we define the properties of the main protocol ELVES for ensuring the availability and validity of input blocks. We categorize parties into two roles: the initial auditor, denoted as the "backer", who obtains the block as input (along with the proof of validity), and the "validators" who make the block available to other parties and check their validity.

Syntax. A designated party, denoted as backer B, has an initial input m, and every party P_i outputs a bit out_i at the end of the protocol. Afterwards, parties can invoke any (or both) sub-protocol gadgets Π_{rec} for *data recovery* and Π_{id} for *cheating identification*. Protocol Π_{rec}, with recipient R, outputs a message m' towards R, and Π_{id} outputs an index k identifying a party.

Definition 3 (Availability and Validity)**.** *Let* $\mathsf{VerifyData}$ *be a deterministic predicate. An availability and validity protocol* $\Pi_{\texttt{ELVES}}$ *is run by a backer* B *and validators* $\{P_1, P_2, \ldots, P_n\}$*, where* B *has an initial input* m *and each validator* P_i *outputs a bit* $\mathsf{out}_i \in \{0, 1\}$*, indicating whether the input is valid or not, according to the predicate. The protocol has two sub-protocol gadgets* Π_{rec} *for* data recovery *and* Π_{id} *for* cheating identification*, which can be executed after* $\Pi_{\texttt{ELVES}}$ *ends. A secure availability and validity protocol* $\Pi_{\texttt{ELVES}}$ *for predicate* $\mathsf{VerifyData}$ *and sub-protocols* Π_{rec} *and* Π_{id} *has the following properties:*

- ***Completeness:*** *If* B *is honest and* $\mathsf{VerifyData}(m) = 1$*, then every validator outputs* $\mathsf{out}_i = 1$*.*
- ***Availability:*** *We say that* m *is available towards an honest recipient party* R *if, after* $\Pi_{rec}(R)$ *is executed,* R *outputs* m*. The Availability property guarantees that if any honest party* P_i *has an output, then there exists a unique message* m *that is available towards any recipient* R*.*
- p_s***-Soundness:*** *With probability at most* p_s*, an honest party* P_i *outputs* $\mathsf{out}_i = 1$ *and* $\mathsf{VerifyData}(\Pi_{rec}(R)) = 0$ *for some honest* R*.*
- ***Identification:*** *If any honest party* P_i *outputs* $\mathsf{out}_i = 0$*, then after* $\Pi_{\texttt{Id}}$ *is executed, all honest parties output an index* k *such that* P_k *is actively corrupted.*
- ***Accountable High Bandwidth:*** *This property informally implies that either the per-party communication complexity is low or at least one actively corrupted party is identified. More concretely, given* $O(n)$ *parallel instances of* $\Pi_{\texttt{ELVES}}$ *with inputs of size* ℓ *bits, the communication complexity incurred per instance by each validator is either* $O(\frac{\ell}{n} + \log(n))$ *bits plus publishing* $O(1)$ *signatures on-chain or via flooding, or after* Π_{id} *is executed, all honest parties identify a party that is actively corrupted.*

7.1 Protocol Description

The protocol makes use of the sub-protocols described in the prior sections, which are secure up to $\gamma < n/3$ corruptions that can be static active or adaptive crashes:

- a secure availability scheme $(\Pi_{\texttt{Dist}}, \Pi_{\texttt{Rec}})$, and
- a secure approval scheme $\Pi_{\texttt{Appr}}$ with soundness ε.

The protocol is simple. The backer B with input m invokes $\Pi_{\texttt{Dist}}$. Then, if $\Pi_{\texttt{Dist}}$ is successful, the validators invoke $\Pi_{\texttt{Appr}}$ and each party outputs out_i as the output of $\Pi_{\texttt{Appr}}$. For the sub-protocol gadgets, we define $\Pi_{rec} = \Pi_{\texttt{Rec}}$, and Π_{id} is the protocol that outputs the backer's index if the output of $\Pi_{\texttt{ELVES}}$ outputs 0, and otherwise outputs $\perp$.

We now show that the simple protocol is a secure availability and validity protocol.

Theorem 1. *Let $(\Pi_{\texttt{Dist}}, \Pi_{\texttt{Rec}})$ be a secure availability scheme, and $\Pi_{\texttt{Appr}}$ be a secure approval scheme with soundness ε. Then, protocol $\Pi_{\texttt{ELVES}}$ with sub-protocol gadgets Π_{rec} and Π_{id} is a secure availability and validity protocol with parameter $p_s = \varepsilon$.*

7.2 Security Analysis

We now prove the security of the proposed protocol.

7.2.1 Proof of Completeness

Completeness follows by inspection. If the backer B is honest and the input m is valid, then correctness of the availability scheme ensures that $\Pi_{\texttt{Dist}}$ is successful. Moreover, completeness of the approval scheme ensures that all honest parties in $\Pi_{\texttt{Appr}}$ output 1.

7.2.2 Proof of Availability

Assume that an honest party has an output. This means that the protocol $\Pi_{\texttt{Dist}}$ was successful. Then, by the soundness property of the availability scheme, we have that there exists a unique message m' such that any recipient R that obtains the output from $\Pi_{\texttt{Rec}}$ outputs m'.

7.2.3 Proof of Soundness

Again, let us assume that the protocol $\Pi_{\texttt{Dist}}$ was successful in distributing a message m'. The probability that an honest party outputs 1 when m' is invalid is at most ε, due to soundness of the approval scheme.

7.2.4 Proof of Identification

If any honest party outputs $\mathsf{out}_i = 0$, the protocol outputs the index of the backer B, who is guaranteed to be corrupted, due to completeness of the protocol and the fact that an honest backer does not distribute invalid data.

7.2.5 Proof of Accountable High Bandwith

In this section we consider an adversary that attempts to maximize the communication complexity of the protocol as valid blocks are processed by the approval pipeline. First, the adversary can trigger a dispute by claiming that a valid block is invalid, but the protocol will identify the index of the corrupted party who made the false claim. Next, the adversary can make a block's committee grow beyond a constant size by crash corrupting its auditors as they announce their assignments. However, we shall prove that, if we consider a period of time in which b blocks are processed, and an adversary attempts to activate as many tranches as possible on any of these blocks, then the expected average final committee size $\mathrm{s_f}$ per block is bounded by a constant, provided that b is large enough.

We say a validator is *faulty* if it may become a no-show, i.e., it may announce its assignment to a block but not approve it within the given time window. Under optimistic conditions, we can expect a very low rate of faulty validators (say, below 0.05), in particular because any validator that is merely unresponsive would not announce their assignments and thus not become a no-show. However, this rate will increase in case of an attack, where faulty validators may be either actively corrupted or crash corrupted. For the sake of computing worst-case bounds on scalability, we assume in what follows that validators *always* announce their assignments, but may be either honest or faulty.

Static adversary. We first consider the case where the adversary can only cause faulty validators statically, i.e., an a-priori fixed set of αn validators is faulty throughout the attack.

Lemma 4. *If* $\mathrm{s_f}$ *is the expected final size of a committee with a static set of* αn *faulty validators, and* $\alpha \cdot \mathrm{s}_\delta < 1$*, then*

$$\mathrm{s_f} \leq \frac{\mathrm{s_0}}{1 - \alpha \cdot \mathrm{s}_\delta}.$$

Proof. Let the random variable T be the total number of tranches activated, summed over all b processed blocks, without counting any 0-th tranches. We consider again the model described in Section 6.3, with the validator-to-tranche assignment graph in Figure 4, where every possible assignment of a validator to one of the T tranches is an independent Bernoulli trial with probability $p_\delta = \mathrm{s}_\delta / n$. Thus, if X_v is the number of assignments of validator v to tranches (v's degree in the graph), it follows a binomial distribution $X_v \sim \mathrm{Binom}(T, p_\delta)$.

Let μ be the expected number of assignments per validator. It can be subdivided as $\mu = \mu_0 + \mu_\delta$, where μ_0 and μ_δ are respectively the expected number of assignments per validator to 0-th tranches, and to further tranches, and where

$$\mu_\delta = \mathrm{E}[X_v] = \frac{\mathrm{s}_\delta}{n} \cdot \mathrm{E}[T].$$

Likewise, let the random variable Z be the total number of no-shows summed over all b blocks, which we subdivide as $Z = Z_0 + Z_\delta$, where Z_0 and Z_δ are respectively the number of no-shows in 0-th tranches and in further tranches. Since one tranche is activated for each no-show:

$$\mathrm{E}[Z] = \mathrm{E}[T] = \frac{n}{\mathrm{s}_\delta} \cdot \mu_\delta. \tag{2}$$

Furthermore, we have that $E[Z_0] \leq \alpha n \cdot \mu_0$, and

$$Z_\delta \leq \sum_{v \text{ faulty}} X_v, \quad \text{so} \quad \mathrm{E}[Z_\delta] \leq \alpha n \cdot \mu_\delta, \tag{3}$$

where the two previous inequalities are tight in case every faulty validator becomes a no-show in every assignment, and where we highlight that each X_v has the same expectation μ_δ whether v is honest or faulty.

By taking expectations on identity $0 = Z - Z_0 - Z_\delta$:

$$\begin{aligned} 0 &= \mathrm{E}[Z] - \mathrm{E}[Z_0] - \mathrm{E}[Z_\delta] \\ &\geq \frac{n}{\mathrm{s}_\delta}\mu_\delta - \alpha n \cdot \mu_0 - \alpha n \cdot \mu_\delta, \end{aligned}$$

which gives the inequalities

$$\frac{\mu_\delta}{\mu_0} \leq \frac{1}{\frac{1}{\alpha \cdot \mathrm{s}_\delta} - 1} \quad \text{and} \quad \frac{\mu}{\mu_0} = 1 + \frac{\mu_\delta}{\mu_0} \leq \frac{1}{1 - \alpha \cdot \mathrm{s}_\delta}.$$

Notice that inequality $\alpha \cdot \mathrm{s}_\delta < 1$ ensures that μ/μ_0 remains bounded. Finally, we apply the trivial relation $\mathrm{s_f} \,/\, \mathrm{s_0} = \mu/\mu_0$ to obtain the claim. □

General case. We now consider a general attack, where a static set of αn validators is faulty, and an additional βn validators can be made faulty adaptively, by crash corrupting them at any point.

Notice that in this case, if the adversary focuses its attack on a single block, it can make its committee explode to a size linear in n. This is not however an issue for scalability, as every validator's load increases only by a small constant. The attack becomes problematic only if it is extended to several blocks, yet we will show that the expected committee size, averaged over a *sufficiently large* number b of processed blocks, remains bounded by a constant. Intuitively, the best adversarial strategy is to wait to gain partial information about the number of assignments each validator receives, and then crash corrupt those with the most assignments, in turn to cause the most no-shows. We will need the following result, whose proof is delayed to Appendix B.

Lemma 5. *Let X be a real-valued random variable with finite expectation and variance. For n and $0 < \beta \leq 1$ so that n and βn are positive integers, suppose we sample n independent values from X and keep the subset of βn highest-valued samples, and let $Y_{n,\beta}$ be the average within this subset. Then,*

$$E[Y_{n,\beta}] \leq E[X] + \sqrt{\frac{1-\beta}{\beta}\ Var[X]}.$$

We highlight that the result above is a strengthening of Cantelli's inequality. Indeed, if one uses Cantelli's inequality to upper bound the $(1-\beta)$-quantile in the distribution of X, we obtain exactly the same expression as the right-hand side above. But we prove that this expression actually bounds the expectation of the full distribution truncated to the right of the $(1-\beta)$-quantile.

Lemma 6. *Consider a scalability attack with αn static and βn adaptive faulty validators, for a period where b blocks are processed. If $(\alpha+\beta)\,\mathrm{s}_\delta < 1$, then*

$$\mathrm{s_f} \leq \frac{s_0 + G + G^{1/2}\left[G + 2(\alpha+\beta)\,\mathrm{s}_\delta \cdot s_0\right]^{1/2}}{1-(\alpha+\beta)\,\mathrm{s}_\delta}, \qquad \textit{where}$$
$$G := \frac{n}{2b} \cdot \frac{\beta(1-\beta)\,{\mathrm{s}_\delta}^2}{1-(\alpha+\beta)\,\mathrm{s}_\delta}.$$

Proof. We follow the same analysis as in the proof of Lemma 4. As before, we define variables $Z_0 + Z_\delta = Z$, where Z_0, Z_δ and Z are respectively the number of no-shows in 0-th tranches, in further tranches, and in total, summed over all b processed blocks. We still have identity (2) for $\mathrm{E}[Z]$, as well as inequality $\mathrm{E}[Z_0] \leq (\alpha+\beta)n \cdot \mu_0$, because there are at most $(\alpha+\beta)n$ faulty validators throughout the attack, and we can make our protocol give all validators roughly the same number of assignments to 0-th tranches, namely μ_0, with virtually no variance.[6]

Also, we have $Z_\delta \leq \sum_{v\ \text{faulty}} X_v$, where X_v is the number of assignments of validator v to further tranches. However, this time we cannot argue that X_v has the same expectation for honest and faulty validators, so inequality (3) fails. Instead, we use Lemma 5 to bound $\mathrm{E}[Z_\delta]$, where $X_v \sim \mathrm{Binom}(T, p_\delta)$ has expectation $\mathrm{E}[T] \cdot p_\delta = \mu_\delta$ and variance $\mathrm{E}[T] \cdot p_\delta(1-p_\delta) < \mu_\delta$:

$$\begin{aligned}
\mathrm{E}[Z_\delta] &\leq \mathrm{E}\left[\sum_{v\ \text{fixed faulty}} X_v + \sum_{v\ \text{adaptive faulty}} X_v\right] \\
&\leq \alpha n \cdot \mu_\delta + \mathrm{E}\left[\max_{V' \subset V:\ |V'| \leq \beta n} \sum_{v \in V'} X_v\right] \\
&\leq \alpha n \cdot \mu_\delta + \beta n \left(\mu_\delta + \sqrt{\frac{1-\beta}{\beta}\mu_\delta}\right) \\
&= (\alpha+\beta)n \cdot \mu_\delta + n \cdot \sqrt{\beta(1-\beta)\mu_\delta}.
\end{aligned}$$

We take expectations on identity $0 = Z - Z_0 - Z_\delta$:

$$\begin{aligned}
0 &= \mathrm{E}[Z] - \mathrm{E}[Z_0] - \mathrm{E}[Z_\delta] \\
&\geq \frac{n}{\mathrm{s}_\delta}\mu_\delta - (\alpha+\beta)n \cdot \mu_0 - (\alpha+\beta)n \cdot \mu_\delta - n \cdot \sqrt{\beta(1-\beta)\mu_\delta} \\
&= \frac{n}{\mathrm{s}_\delta}\left[(1-(\alpha+\beta)\,\mathrm{s}_\delta)\,\mu_\delta - (\alpha+\beta)\,\mathrm{s}_\delta \cdot \mu_0 - \mathrm{s}_\delta\sqrt{\beta(1-\beta)\mu_\delta}\right] \\
&= \frac{n}{\mathrm{s}_\delta}\left[(1-A)\mu_\delta - A \cdot \mu_0 - B\sqrt{\mu_0 \cdot \mu_\delta}\right],
\end{aligned}$$

[6] Notice our analyses only require assignments to further tranches to be independent from one another.

where $A := (\alpha + \beta)\,\mathrm{s}_\delta < 1$ and $B := \sqrt{\beta(1-\beta)/\mu_0} \cdot \mathrm{s}_\delta$. Reordering terms:

$$0 \geq (1-A)^2\mu_\delta^2 - [2A(1-A) + B^2]\mu_0 \cdot \mu_\delta + A^2\mu_0^2, \quad \text{so}$$
$$\frac{\mu_\delta}{\mu_0} \leq \frac{2A(1-A) + B^2 + \sqrt{4A(1-A)B^2 + B^4}}{2(1-A)^2},$$

where we applied the quadratic formula. From identity $\mathrm{s_f}\,/s_0 = \mu/\mu_0$:

$$\mathrm{s_f} = \left(1 + \frac{\mu_\delta}{\mu_0}\right) s_0 \leq \frac{2(1-A) + B^2 + B\sqrt{4A(1-A) + B^2}}{2(1-A)^2} \cdot s_0$$
$$= \frac{s_0 + G + G^{1/2}[2A \cdot s_0 + G]^{1/2}}{1-A},$$

where $G := \frac{B^2 s_0}{2(1-A)}$. Finally, we can check that G indeed equals its definition in the statement via the identity $b \cdot \mathrm{s}_0 = n \cdot \mu_0$, which is the expected number of assignments to the b 0-th tranches. This completes the proof. □

Notice the expected final committee size $\mathrm{s_f}$ is bounded by a constant as long as $G = O(1)$, which happens if $b = \Omega(n)$. Having $b = \Theta(n)$ corresponds to a constant time period of a few minutes in a blockchain system that processes $\Theta(n)$ blocks per minute. In the limit where $G \to 0$ (because either b is large or β is small), our bound is $s_0/[1 - (\alpha + \beta)\,\mathrm{s}_\delta]$, matching the bound in Lemma 4 for the static case.

8 Rational Model and Analysis

So far we have considered a polynomial-time adversary as described in Section 4, and where the soundness security property only holds with a constant probability. In this section we show that one can argue about the security of the presented protocols in a restricted rational model.

Accountable Protocols. For that, we first introduce the notion of *accountable protocols* which are protocols that have sub-protocol gadgets where violating a certain security property allows the gadget to identify an actively corrupted party.

Definition 4. *Let Π be a distributed protocol among n parties where each party P_i has an input x_i ($x_i = \perp$ if the party has no input), and at the end of the protocol each P_i outputs y_i. Further let Q be a binary predicate on the inputs and outputs of the protocol.*

We say that Π is accountable with respect to predicate Q via sub-protocol gadget Π', if after the execution of the protocol Π, we have: if $Q(x_1, \ldots, x_n, y_1, \ldots, y_n) = 0$, then honest parties can jointly execute Π', which outputs an index k such that P_k is actively corrupted.

It is easy to see that the protocol $\Pi_{\texttt{ELVES}}$ introduced in Section 7 is an accountable protocol with respect to the induced predicate from the soundness property, via the sub-protocol gadget $\Pi' = \Pi_{\texttt{Id}}$.

Rational Protocol. Consider a protocol Π that is accountable with respect to predicate Q via sub-protocol gadget Π'. Further consider the case that Π satisfies Q with probability at least p.

We can argue about the security of such a protocol in a rational model where each party P_i puts down a certain amount $C_i = C$ of tokens as collateral, for some protocol parameter C, and the adversary is rational and follows strategies leading to non-negative expected net profit.

We make three assumptions regarding the adversarial net profit in the model: First, if Π' identifies a corrupted party P_k, the adversary loses the collateral C_k. Second, we assume that M is an upper bound on the profit that the adversary gains if $Q(\cdot) = 0$. Finally, any adversary strategy that leads to $Q(\cdot) = 1$, gives the adversary a non-negative profit.

Consider the natural enhancement of the protocol $\Pi_{\texttt{ELVES}}$ as described above, where parties put down a collateral and the identification of a corrupted party P_k leads to the adversary losing C_k. We then have the following corollary of Theorem 1.

Corollary 1. *Let $0 < \nu < 1$ be a constant such that $C = \nu \cdot M/n$. Let ε be the soundness error parameter of $\Pi_{\texttt{ELVES}}$. Then, the protocol satisfies soundness with probability 1 in the rational model described above, if we have that $1/\varepsilon > n/\nu + 1$.*

Proof. The expected profit of an attack is at most

$$\varepsilon \cdot C - (1 - \varepsilon) \cdot \nu \cdot C/n,$$

which is negative under the assumption on ε. □

9 Committee Size Analysis

In this section we discuss our results on the expected final committee size $\mathrm{s_f}$, i.e., the expected average number of parties that verify a block, accounting for both the initial committee and all activated tranches. We also provide explanations for Figures 1 and 2.

The parameters needed to instantiate our main protocol are: 1) the soundness failure probability ε, 2) the bound γ on the total fraction of corrupted parties, 3) the expected initial committee size $\mathrm{s_0}$, and 4) the expected tranche size s_δ. In what follows we propose how to fix the last two parameters as functions of the first two. For starters, we fix $\mathrm{s_0}$ to its bound given in Lemma 3, i.e., $\mathrm{s_0} := F(\gamma, \mathrm{s}_\delta) \cdot \ln(1/\varepsilon)$.

Our choice of the tranche size s_δ is bounded by the constraints $1/(1-\gamma) < \mathrm{s}_\delta < 1/\gamma$, coming from Lemmas 3 and 6, respectively. For $\gamma = 1/3$, for instance, we obtain the bounds $1.5 < \mathrm{s}_\delta < 3$. Now, notice that the bound on the expected final committee size $\mathrm{s_f}$ given in Lemma 6 depends on the instantiation parameters γ, ε and s_δ, as well as on the current corruption fractions α and β, and the number b of blocks processed during the period in consideration. In the particular case that $\alpha + \beta = \gamma$ and $b \to \infty$ (corresponding to a worst-case attack during a long period of time), we have that $G = 0$ and the bound simplifies to the expression $\mathrm{s_f} \leq F(\gamma, \mathrm{s}_\delta) \cdot \ln(1/\varepsilon)/(1 - \gamma \cdot \mathrm{s}_\delta)$. For simplicity, we suggest selecting the unique value of $\mathrm{s}_\delta = \mathrm{s}_\delta(\gamma)$ that minimizes this particular expression. Some of these values are given in Table 1.

γ	s_δ	$\mathrm{s_f} \,/\, \ln(1/\varepsilon)$
1/5	2.3105	2.2737
1/4	2.1972	4.4376
1/3	2.0794	9.7767

Table 1: Value of s_δ that minimizes bound given in Lemma 6, and corresponding bound on $\mathrm{s_f} \,/\, \ln(1/\varepsilon)$, for three values of γ, assuming that $\alpha + \beta = \gamma$ and $b \to \infty$.

Once $\mathrm{s_0} = \mathrm{s_0}(\gamma, \varepsilon)$ and $\mathrm{s}_\delta = \mathrm{s}_\delta(\gamma)$ are so fixed, notice that the bound on $\mathrm{s_f}$ given by Lemma 6, under the constraints $\alpha + \beta \leq \gamma$ and $b = \Omega(n)$, is worst when $\alpha = 0$ and $\beta = \gamma$. We select these values plus the canonical value of $b = n$ to plot Figure 1.

Now, what value of ε is "secure enough" for a realistic implementation? Consider as an example a blockchain network maintained by $n \leq 1000$ nodes, securing tokens with a combined market capitalization M. We consider it realistic to require nodes to have a sum of collaterals of at least 5% of M,[7] i.e., each party puts down a collateral $C_i \geq C := \nu \cdot M/n$ with $\nu = 0.05$. By Corollary 1, we have that $1/\varepsilon > 1000/0.05 + 1 = 20001$ is secure enough. This justifies the instantiation values proposed in Figure 2, with $\gamma = 1/3$, $\mathrm{s}_\delta = 2.0794$, and $b = n$.

10 Conclusions

We have introduced ELVES, a provably-secure efficient gadget to check the validity of blocks. Our protocol scales essentially optimally in terms of computation, communication and storage, and can be applied on top

[7] For example, as of January 2024, the sum of validators' collaterals subject to being lost as punishment in case of misbehavior, is more than 25% of all ETH in Ethereum, and more than 40% of all DOT in Polkadot.

of current practical blockchain systems. In contrast to all prior scaling solutions, our gadget tolerates instant adaptive crashes, a desirable feature due to the fact that realistic distributed DDoS attacks can be deployed relatively fast in practice. Moreover, compared to prior sharding approaches, our protocol allows to obtain smaller committee sizes to check the blocks. This is in part due to the fact that we consider a model where the adversary is rational. However, note that even without the rationality argument, our synchronous model with instant crashes allows to still obtain better numbers. In particular, for the setting with 30% corruptions (static active or adaptive crashes), our committees have size around 400 for an error of 2^{-60} (compared to state of the art solutions like Gearbox [DMM+22] that require 1000 parties).

Our gadget can also be considered a contribution to the literature of rollups. On one hand, our protocol provides 1) a formal way to select the validators to check the blocks without relying on additional optimistic rollup assumptions, and 2) a way to balance the load equally among the system validators, improving the main bottleneck of ZK-rollups, which require some nodes, the operators, to have higher hardware capabilities to generate and process the proof systems.

References

[AKKW19] Georgia Avarikioti, Eleftherios Kokoris-Kogias, and Roger Wattenhofer. Divide and scale: Formalization of distributed ledger sharding protocols. *arXiv preprint arXiv:1910.10434*, 2019.

[DMM+22] Bernardo David, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 683–696, 2022.

[EGSVR16] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. {Bitcoin-NG}: A scalable blockchain protocol. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 45–59, 2016.

[KGC+18] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, Baltimore, MD, August 2018. USENIX Association.

[KJG+16] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th usenix security symposium (usenix security 16)*, pages 279–296, 2016.

[KKJG+18] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE symposium on security and privacy (SP)*, pages 583–598. IEEE, 2018.

[LLZW23] Mingzhe Li, You Lin, Jin Zhang, and Wei Wang. Cochain: High concurrency blockchain sharding via consensus on consensus. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023.

[LNZ+16a] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 17–30, 2016.

[LNZ+16b] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 17–30, 2016.

[LWZ23] Yi Li, Jinsong Wang, and Hongwei Zhang. A survey of state-of-the-art sharding blockchains: Models, components, and attack surfaces. *Journal of Network and Computer Applications*, page 103686, 2023.

[Rab94] Tal Rabin. Robust sharing of secrets when the dealer is honest or cheating. *Journal of the ACM (JACM)*, 41(6):1089–1109, 1994.

[RS60] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[Tha] Justin Thaler. A technical faq on lasso, jolt, and recent advancements in snark design. `https://a16zcrypto.com/posts/article/a-technical-faq-on-lasso-jolt-and-recent-advancements-in-snark-design`.

[WSNH19] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[WW19] Jiaping Wang and Hao Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, pages 95–112, 2019.

[XZD+23] Yibin Xu, Jingyi Zheng, Boris Düdder, Tijs Slaats, and Yongluan Zhou. A two-layer blockchain sharding protocol leveraging safety and liveness for enhanced performance. *arXiv preprint arXiv:2310.11373*, 2023.

[ZMR18] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

A Sharding Analysis for Strawman and Optimistic Paths

In this section we compute estimates of required committee sizes for the strawman sharding approach, and its variant with optimistic paths.

In the analysis we assume there are parameters α and β such that $t = \alpha + \beta < 1/3$, and an adversary A that controls a fraction α of parties statically, and can make an additional fraction β of parties crash adaptively. In particular, at any instant there are at least $(1-t)n$ parties who are both honest and online, hence can announce their assignments to shards. We also assume that the number of parties is $n \leq 3000$, and that they get partitioned into T subsets, each serving one shard, with an average subset size of $m = n/T$.

Strawman Sharding Approach. We partition parties into T subsets, each of size m. Each subset serves a shard, and we establish the validity of a block by taking the majority vote within its shard subset. Hence, the protocol is safe as long as each subset has an honest majority.

Fix a shard S. We model the protocol as if S samples k parties from V one by one uniformly at random. Let X_i be a variable with value 0 if the i-th party is honest, and 1 if adversarial, and let $\bar{X} := \sum_i X_i/k$ be the mean value. Clearly, the expected value of $\bar{X}$ is α, and we can apply Hoeffding's inequality to establish a bound on the probability that $\bar{X}$ is greater than $1/2$:

$$P[\bar{X} \geq 1/2] \leq e^{-2(1/2-\alpha)^2 m}.$$

We remark that this bound is usually applied to variables with binomial distributions, and that variable $\bar{X}$ rather follows a hypergeometric distribution because the sampling was performed *without replacement*; however, this bound is known to hold for both distributions.

An advantage of this protocol is that it does not need to churn parties across shards frequently, but it should do it from time to time in response to changes in the global party set V. If we assume that the protocol performs a new random assignment to shards once per day, and we want adversary A to have to wait 1000 years in expectation until any of the T shards is compromised, then we need the above value to be bounded by $\varepsilon := (365 \cdot 1000 \cdot T)^{-1}$, i.e.

$$e^{-2(1/2-\alpha)^2 m} \leq (365 \cdot 1000 \cdot n/m)^{-1},$$

where we used the fact that $T = n/m$. For $n = 3000$ and $\alpha = 1/3$, we obtain that each shard should be of size $m \geq 274$.

Gradual Optimistic Paths. In this approach, each party gets assigned to one of the T shards uniformly at random, and we consider this procedure to be safe if for any shard, there is at least one honest online party assigned to it, except with probability at most ε for an error parameter ε.

Fix a shard S: each honest online party is assigned to S independently with probability $1/T$. Hence, the probability that none of the $(1-\alpha)n$ honest online parties gets assigned to S is

$$\left(1 - \frac{1}{T}\right)^{(1-\alpha)n} = \left[\left(1 - \frac{1}{T}\right)^{T}\right]^{(1-\alpha)\frac{n}{T}} \leq e^{-(1-\alpha)\frac{n}{T}} = e^{-(1-\alpha)m},$$

where we recall that $m = n/T$, and we notice that the inequality above becomes tight for large values of T, which is the case in this approach. If we want this probability to be at most ε, we obtain the inequality $e^{-(1-\alpha)m} \leq \varepsilon$.

Now, what is an adequate error parameter ε? If we do not assume an adversarial cost per attack attempt, we propose to consider the protocol as safe if it takes the adversary A over a thousand years in expectation to succeed in an attack. In Polkadot there are 10 slots per minute, hence A has $10T$ opportunities to attack per minute, each succeeding if the corresponding shard subset is fully adversarial. If the success probability per attack is $\varepsilon := (5.26 \cdot 10^9 \cdot T)^{-1}$, then a success will take a thousand years in expectation. We obtain

$$e^{-(1-\alpha)m} \leq \varepsilon = \left(5.26 \cdot 10^9 \cdot \frac{n}{m}\right)^{-1} \quad \text{or,} \quad m \cdot e^{(1-\alpha)m} \geq 5.26 \cdot 10^9 \cdot n.$$

If $\alpha = 1/3$ and $n = 3000$, then $m \geq 41$.

B Delayed proofs

Proof of Lemma 5. Let X_1 and X_2 be the unique random variables such that a) the probability distribution of X is the mixture of the probability distributions of X_1 and X_2 with weights $1-\gamma$ and γ respectively, and b) every sample from X_1 is smaller than or equal to every sample from X_2. Hence, the ranges of X_1 and X_2 intersect in at most value, namely the $(1-\gamma)$-quantile of X. Clearly, as n grows the empirical distribution of the n sampled values converges to the distribution of X, and in turn the distribution of the γn highest-valued samples converges to the distribution of X_2. Hence, $\mathrm{E}[Y_{N,\gamma}]$ converges to $\mathrm{E}[X_2]$.

In fact, we claim that $\mathrm{E}[Y_{n,\gamma}] \leq \mathrm{E}[X_2]$ for any n, i.e., this expectation is bounded by its limit. To see this, consider running the experiment twice: given two sets of n values $\{x_i\}_{i\in[1,n]}$ and $\{x_i\}_{i\in[n+1,2n]}$ sampled independently from X, if $\{x_i\}_{i\in I}$ and $\{x_i\}_{i\in I'}$ are respectively their subsets of γn highest-valued samples, we see that their union $\{x_i\}_{i\in I\cup I'}$ is of size $2\gamma n$, but does not necessarily contain the $2\gamma n$ highest-valued samples within $\{x_i\}_{i\in[1,2n]}$. This proves that $\mathrm{E}[Y_{n,\gamma}] \leq \mathrm{E}[Y_{2n,\gamma}]$, and hence proves the claim.

Assuming without loss of generality that $\mathrm{E}[X] = 0$, the statement follows from the previous claim and the fact that $\mathrm{E}[X_2] \leq \sqrt{(1-\gamma)\,\mathrm{Var}[X]/\gamma}$, which is our second claim that we prove next. To this end, we express the expectation and variance of X in terms of those of its mixture components. We have

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}[(1-\gamma)X_1 + \gamma X_2] \\
&= (1-\gamma)\,\mathrm{E}[X_1] + \gamma\,\mathrm{E}[X_2] = 0, \quad \text{so} \\
\mathrm{E}[X_1] &= -\frac{\gamma}{1-\gamma}\,\mathrm{E}[X_2]. \qquad (4)
\end{aligned}$$

Similarly, the formula for the variance gives

$$\begin{aligned}
\mathrm{Var}[X] = \mathrm{E}[X^2] &= (1-\gamma)\,\mathrm{E}[X_1^2] + \gamma\,\mathrm{E}[X_2^2] \\
&= (1-\gamma)(\mathrm{Var}[X_1] + \mathrm{E}^2[X_1]) + \gamma(\mathrm{Var}[X_2] + \mathrm{E}^2[X_2]) \\
&\geq (1-\gamma)\,\mathrm{E}^2[X_1] + \gamma\,\mathrm{E}^2[X_2] \\
&= (1-\gamma)\cdot\frac{\gamma^2}{(1-\gamma)^2}\,\mathrm{E}^2[X_2] + \gamma\,\mathrm{E}^2[X_2] \\
&= \frac{\gamma}{1-\gamma}\,\mathrm{E}^2[X_2],
\end{aligned}$$

proving the second claim, and where the applied identity (4). □