Milkomeda Rollup

Zeta Avarikioti, Makis Arsenis,
Dimitris Karakostas, Orfeas Thyfronitis-Litos
Common Prefix

December 2022

1 Introduction

Milkomeda aims to provide EVM functionality to blockchains that do not provide this functionality via executing a rollup on top of such blockchains. The following exposition proposes a design proposal for this project. In a nutshell, the goal is to use the underlying blockchain, e.g., Algorand or Cardano, for data availability and the election of the operators of the rollup and move all the execution from the chain to the rollup. To do so in a secure manner, three protocols are necessary:

- **Validator election:** Determines the entities, namely the validators, that are responsible for sequencing the data and publishing it on the chain. The validators also are responsible for determining the state in which a party can exit the rollup.
- **Sequencer election:** Determines which one of the validators is responsible for publishing the rollup data at any point in time.
- **Safe exit:** Determines how a participant can exit the rollup.

In the following sections, we first explain the model and security goals (Section 2). Then, we propose specific solutions and corresponding sketch proofs under the assumptions stated in the model, as well as discuss possible alternative solutions and trade-offs. In particular, we describe and analyze our proposal for the validator election in Section 3. Subsequently, we present the sequencer election protocol in Section 4, and we conclude with Section 5 where possible safe exit protocol designs are investigated.

2 The model

2.1 Assumptions

System model. Parties can lock their money in the Milkomeda rollup in order to continue the execution off-chain thus gaining access to functionalities that are not available on-chain (EVM). These parties will be referred to as *clients*. For the secure operation of the rollup, parties can volunteer to participate as *validators*. The validators lock stake in the rollup contract and are responsible for the correct protocol execution, i.e., data is published timely on-chain and clients can exit the rollup in the correct state.

The protocol proceeds in (asynchronous, event-based) rounds that are determined by the underlying blockchain – one block on L1 defines one round. s consecutive rounds determine an epoch.

Network model. We assume a partially synchronous communication model as the rollup is intended to operate on blockchains that also remain secure under this network model such as Algorand. This means that there is a fixed upper bound on the communication delay but this is unknown to the protocols. Equivalently, we can assume arbitrarily long periods of asynchrony – during which the protocols must remain safe – followed by long enough periods of synchrony (after GST which is unknown) – during which the protocols shall make progress.

Cryptographic assumptions. We make the usual cryptographic assumptions, that secure communication channels, digital signatures, and PKI exist.

Threat model. We assume the adversary respects the cryptographic and network assumptions, meaning she cannot break cryptography and she can reorder messages but cannot drop them. Furthermore, the adversary may control up to 1/3 of the validators, as well as any client. Lastly, we assume the adversary is slowly adaptive with respect to the validators, i.e., within an epoch the adversary is static, that is, she can choose which validators to corrupt at the beginning of an epoch but cannot change the corruption set on the fly within the epoch.

Incentives. Our goal is to eventually design protocols that are secure under rational participants, that is, both validators and clients act in order to maximize a utility function, i.e., their profit. In this model, clients and validators are allowed to collude with each other.

2.2 Desired properties

In this section, we determine what security means for the Milkomeda rollup under the aforementioned model.

- **Liveness:** Any transaction submitted on the rollup becomes stable, i.e., is executed on the rollup by an honest client, within u blocks, where u is the liveness security parameter.

- **Bridging:** Any participant of the L1 can enter or exit the rollup upon request within w blocks, where w is a security parameter, according to the state that an honest rollup client reports.

Bridging encompasses the safety of the rollup. It expresses that no party loses money, i.e., any transaction that is executed on the rollup will remain confirmed at any future time. Bridging also implies that any party that wishes to exit the rollup will leave it in the state that corresponds to the sequence of relevant transactions, meaning the rollup parties will not gain or lose money when bridging to the blockchain.

Bridging, in addition, encompasses a liveness notion, as for the rollup to be functional and make meaningful progress, parties must be able to join and leave on demand. Liveness, on the other hand, expresses that the rollup ledger will make progress, thus transactions submitted in the rollup will be eventually executed. Note that both liveness and bridging security parameters can be expressed in time units when the network is synchronous. However, the L1 may be operating in partial synchrony (e.g., Algorand), in which case we have no liveness guarantees. For this reason, we express the liveness and bridging security parameters for the rollup in blocks.

Our approach. To show the desired properties, we will first determine the individual protocols and their corresponding desiderata, and subsequently, we will prove the three desired properties: liveness and bridging for the composition of the protocols. In particular, the proposed design enables a clear separation of liveness and bridging hence we will first show that during an epoch the rollup remains live (Sections 3,4) and later we will show that at the end/beginning of each epoch bridging is satisfied (Section 5). We will show the desiderata hold under an honest supermajority of stake, and then examine under which conditions or assumptions we can relax the honesty assumption and prove the desiderata under rational parties.

3 Validators Election

In this section, we isolate the validator election process from the rest of the protocol and treat it initially as a separate component. We propose a novel proof-of-stake auction for the validator election. In particular, we first define the setting and then determine the desired properties for a proof-of-stake auction, i.e., an auction that correlates the staked amounts to validator slots that operate the rollup. Finally, we present a novel auction mechanism and prove it satisfies our desired properties.

3.1 Setting

There exist n parties $[P_0, \ldots, P_n]$. Each party P_i holds an amount S_i (stake) which is publicly known at the start of the execution to all other parties.[1]

The execution comprises "epochs". Each epoch consists of e slots where each slot corresponds to the generation of a block in the underlying L1.[2] At the end of each epoch, an auction takes place to map the e slots of the next epoch to the parties. In particular, the aim of the auction is to allocate one party to each slot in a fair manner according to the parties' stakes. We will also refer to slots as items, the number of which is denoted by I; we have $I = e$.[3]

All parties that are awarded at least one slot via the auction are eligible to take part in two mechanisms:

i) the *safe exit* protocol which enables clients to exit the rollup – in other words, an L2 to L1 bridge. Participation in the safe exit protocol may yield rewards if the party acts correctly, or slash its stake if the party misbehaves.

ii) the *sequencer election* protocol, where each party assigned in a slot, namely the sequencer, is collecting and publishing on-chain the rollup data. A correct sequencer is rewarded with revenue r.

Note that all rewards are assigned at the end of the epoch and are given in the native token.

Locking the stake & delegation. Before each auction, a party can choose one of the following participation options: i) participate in the auction directly; ii) *delegate* their participation rights to another party;[4] iii) not participate in the auction *at all*.

If a party chooses to participate, either directly or via delegation, their funds (stake) are temporarily locked (at least until the end of the auction).

[1]For simplicity, we assume that S_i is fixed throughout the execution.

[2]Each slot may correspond to one L1 block or any pre-specified constant number of L1 blocks, i.e., the elected validator may have 5 blocks time to publish its batch.

[3]We maintain the different notation because the auction mechanism is of independent value and does not need to correlate to the epoch length.

[4]A party can choose multiple delegates and split its stake arbitrarily among them. If a delegator has been delegated some stake as well, its delegated stake is redelegated to its delegees – i.e., delegation is transitive.

After the stakeholders lock their stake, each party P_i is associated with an amount of stake that is either 0 if the party opted with delegation or equal to the sum of the delegated stake to this party. We abuse the notation S_i to express the total amount of stake with which the party P_i participates in the auction, including the delegated stake to P_i. With s_i we denote the proportional stake P_i owns, that is $s_i = \frac{S_i}{\sum_{j=1}^{n} S_j}$, where n is the parties that participate in the auction – i.e., excluding the delegators.[5]

Auction characteristics. The auction has the following characteristics: i) a number I of *indivisible, identical* items (i.e., the slots) is auctioned; ii) each party[6] can bid an amount $B_i \leq S_i$ for acquiring items; iii) each party can only bid once and its bid is committing (e.g., locks an amount of funds on-chain); iv) each bid becomes public instantly.

Identical items. For simplicity, we assume that all slots in an epoch are identical. This assumption is very strong as the order of slots is, in fact, critical for the liveness of the rollup. If the adversary manages to acquire more than u consecutive slots – via biasing the randomness or successful grinding attacks – he may effectively censor transactions in the rollup, violating the liveness property. In this case, the bundle of consecutive slots will infinitely increase the utility of the adversary. To address these attack vectors without incorporating them into the utility function, we later introduce a property for the auction, namely unpredictability, that guarantees that any PPT adversary cannot bias or predict the auction outcome in a meaningful manner. Due to this property, we can assume that the items of the auction are identical.

3.2 Auction Properties

The overarching goal of the allocation mechanism should ensure that the parties are allocated a fraction of the slots that is on average equal to their staking percentage. Towards that end, definitions 1, 2, 4, 5 and 6 distill the necessary properties that the slot auction mechanism should satisfy.

First, market clearance guarantees that all items are allocated to parties. In the validator election setting, this ensures that each slot will be assigned a validator. Second, symmetry guarantees that the items will be allocated solely based on the stakes of the parties and not their public keys. Third, unpredictability ensures that an adversary cannot predict the outcome of the auction beforehand, and thus cannot grind on potential keys and/or divisions of its stake that may result in acquiring consecutive slots (liveness attack). Finally, Sybil and collusion resilience guarantee that the auction mechanism is not vulnerable to Sybil and collusion attacks, respectively. Intuitively, a party cannot increase its allocated items by combining its stake with other parties

[5] We also abuse the notation n as it was initially defined as all the parties. We do so, in order to avoid the use of unnecessary notation.

[6] Each party refers to an address, not an entity; each entity may have multiple addresses.

(collusion-resilience) or splitting its stake across multiple "fake" identities (Sybil-resilience).

An allocation rule is an algorithm $\mathbf{A}$ that, given as input the stake percentages of the parties $\mathbf{s} = \{s_1, s_2, \ldots, s_n\} \in \mathbb{R}$ and a random seed r, returns a two-dimensional matrix of random variables $Y_{i,j}$ that denotes the allocation of the items. In particular, $Y_{i,j} = 1$ if item $j \in I$ is allocated to validator P_i.

Definition 1 (Market Clearing). *The allocation rule should allocate exactly I items, $\sum_{\forall i} \sum_{\forall j} Y_{i,j} = I$.*

Definition 2 (Symmetry). *An allocation rule is symmetric if for every permutation π on any stake distribution $\mathbf{s}$ resulting in Y', it holds $Pr[Y_{i,j} = 1] = Pr[Y'_{\pi(i),j} = 1]$.*

A symmetric allocation rule also ensures that two parties with the same stake win a slot with the same probability.

Definition 3 (Slot Symmetry). *An allocation rule is slot-symmetric if any two slots are equally likely to be allocated to a party: $\forall P_i, \forall j \neq j' \in I, Pr[Y_{i,j} = 1] = Pr[Y_{i,j'} = 1]$.*

Corollary 1. *A symmetric allocation rule of identical items is also slot-symmetric.*

Definition 4 (Unpredictability). *Before the allocation of the auction on-chain, any PPT adversary must not be able to distinguish between two (plausible) allocations that are sampled from the same stake distribution. Formally, given any pair of allocations $Y_{i,j}$ and $Y'_{i,j}$ with $\|\left(\sum_i (Y_{i,j} - Y'_{i,j})\right)_j\|_\infty < \epsilon$, an adversary cannot distinguish between the two outcomes with probability greater than $1/2 - f(\epsilon)$.*

Definition 5 (Sybil Resilience). *For every stake distribution $\mathbf{s}$ and every stake distribution $\mathbf{s}'$ that can be derived from $\mathbf{s}$ by replacing a party P_i with stake percentage s_i by a set $\mathcal{P}'$ of parties with total stake percentage at most s_i, the total expected items allocated to the parties $\mathcal{P}'$ (denoted by Y') is at most that of party P_i in the initial stake distribution (denoted by Y):*

$$\sum_{i \in \mathcal{P}'} \sum_{j \in I} Y'_{i,j} \leq \sum_{j \in I} Y_{i,j}$$

Definition 6 (Collusion Resilience). *For every stake distribution $\mathbf{s}$ and every stake distribution $\mathbf{s}'$ that can be derived from $\mathbf{s}$ by replacing a set $\mathcal{P}$ of parties by a new party $P_{i'}$ with stake percentage at most $s_{i'} \leq \sum_{i \in \mathcal{P}} s_i$, the total expected items allocated to the party $P_{i'}$ under s' (denoted by Y') is at most the total expected items of the parties $\mathcal{P}'$ (denoted by Y):*

$$\sum_{j \in I} Y'_{i',j} \leq \sum_{i \in \mathcal{P}} \sum_{j \in I} Y_{i,j}$$

Proving that a mechanism is symmetric and market clearing is typically a direct result of the definitions. However, proving that the mechanism satisfies Sybil resilience and collusion resilience can be significantly trickier. To make the analysis more straightforward, we define a third property, *fairness* (Definition 7). Intuitively, fairness guarantees that each individual party is allocated their fair share of items, which is equal to their stake percentage. Following closely the results of [CPR19], Lemma 1 shows that fairness is equivalent to Sybil and collusion resilience in any symmetric and market clearing mechanism. Therefore, to guarantee Sybil and collusion resilience, it suffices to show that a mechanism allocates to each party an average number of items equal to its staking percentage.

Definition 7 (Fairness). *The mechanism should allocate to party P_i* on expectation *$s_i \cdot I$ items.*

Lemma 1. *A market clearing allocation mechanism (cf. Definitions and 1) is symmetric, Sybil resilient, and collusion resilient (cf. Definitions 2, 5 and 6) if and only if it is fair (cf. Definition 7).*

Proof. The proof of the lemma is similar to the proof of the uniqueness of the proportional allocation rule for block rewards presented in [CPR19] (Theorem 1). It is straightforward to adapt the proof to show that any allocation mechanism is fair if and only if it is market clearing, symmetric, Sybil-resilient, and collusion-resilient.

□

Sybil and collusion resilience (and its equivalent, fairness) is a core property to guarantee proportional representation and, eventually, security against adversarial takeovers. However, requiring the *expected* items per party to be equal to their power is not necessarily enough to make the allocation mechanism appealing. For instance, consider Bitcoin's reward allocation mechanism. In Bitcoin, a party produces a block on each round of execution with probability proportional to its power. Thus, *on expectation in the long term*, each party produces a proportion of blocks equal to their power. However, in the short term, a party with small power will typically not produce any blocks. This is particularly problematic when considering temporal discounting [RL11], that is the tendency to disfavor rare or delayed rewards. In Bitcoin, this arguably translates into the formation of mining pools, which promise slightly lower, but more frequent, rewards to smaller miners.

In our setting, to make the allocation more appealing, we define the *concentration property* (Definition 8). This property sets a lower limit to the number of items each party is allocated per auction, which is equal to the floor of its (stake-based) proportion. For instance, if a party controls 1.7% of power, it should receive at least 1% of all items *in all cases* — and possibly more items with some probability. Consequently, the income rate (in terms of items) for each party is steadier and the variance depends only on the decimal part of its stake percentage. Nonetheless, we will treat concentration as optional, since it is not a prerequisite for preventing a malicious party from acquiring a disproportionate amount of items (in the scope of the entire execution).

Definition 8 ((Optional) Concentration)**.** *The mechanism should allocate to party P_i* at least $\lfloor s \cdot I \rfloor$ *items.*

A correlated notion that captures the behavior of miners in the Bitcoin network is risk-neutrality and risk-averseness. In the former, the miners' utility is proportional to their expectation of rewards, while in the latter, the miners' utility is a strictly concave function of the expected rewards. However, this approach rules out the proportional allocation rule for risk-averse parties [CPR19]. In contrast, our concentration property captures the desired behavior of bidders and at the same time allows for a more flexible design space.

To summarize, the properties our allocation mechanism must satisfy are: *market clearance (Def. 1), unpredictability, (Def. 4), fairness (Def. 7), and concentration (Def. 8).* We will also show, *slot-symmetry (Def. 3)* as we will need it in the next section.

3.3 Proof-of-stake auction

In this section, we present our solution that consists of two components: (a) a functionality that maps a vector of reals to a vector of integers with the same expectation on the sum and a corresponding algorithm that implements it; (b) an allocation algorithm that distributes the items to the bidders such that the desired properties are satisfied.

Random Sampling Functionality (RSF) $\mathcal{F}(\mathbf{x})$ An RSF is a function that, given n non-negative real numbers x_i which sum to a *positive integer* $x = \sum_{i=1}^{n} x_i \in \mathbb{Z}^+$, outputs a random vector of non-negative integers $\mathbf{V} \in \mathbb{N}^n$, such that: i) $\mathbb{E}[V_i] = x_i$, and ii) $\sum_{i=1}^{n} V_i = x$.

Allocation Algorithm $\mathcal{A}$ The inputs to $\mathcal{A}$ are as follows:

- I: the number of items;
- n: the number of parties;
- s_i: the stake percentage of party P_i , s.t. $\sum_{i=1}^{n} s_i = 1$.

$\mathcal{A}$ operates as follows:

1. Deterministically allocate $\lfloor s_i \cdot I \rfloor$ items to each bidder P_i. Let $x = I - \sum_{i=1}^{n} \lfloor s_i \cdot I \rfloor$ be the remaining, unallocated items.
2. Call a Random Functionality $\mathcal{F}$ sub-routine with input a vector $\mathbf{x}$ of n values, where $x_i = s_i \cdot I - \lfloor s_i \cdot I \rfloor$. For each value V_i in the output of $\mathcal{F}$, assign V_i extra items to bidder P_i.

Interval Sampling Algorithm We now realize the random functionality $\mathcal{F}$ via the following algorithm $\mathcal{S}$. The input to $\mathcal{S}$ is (as with $\mathcal{F}$):

- $\mathbf{x}$: a vector of n real numbers $[x_1, x_2, \ldots, x_n]$.

$\mathcal{S}$ operates as follows:

1. Consider an interval of the real line which consists of the concatenation of n intervals each of length x_i. In other words, split the interval $[0, x]$, where $x = \sum_{i=1}^{n} x_i$, into n parts like $I_1 = (0, x_1], I_2 = (x_1, x_1 + x_2], \ldots, I_n = (x - x_n, x]$.
2. For each integer $j \in \{1, \ldots, x\}$, sample P_j uniformly at random on the interval $(0, x]$.
3. Let $Y_{i,j}$ be the indicator random variable of the event that $P_j \in I_i$.
4. Define $V_i = \sum_{j=1}^{x} Y_{i,j}$ for all $i \in [n]$.
5. Output $\mathbf{V} = (V_1, V_2, \ldots, V_n)$.

Lemma 2. *The interval sampling algorithm $\mathcal{S}$ implements a random sampling functionality $\mathcal{F}$.*

Proof. For every sample P_j, the probability of sampling a number on the i-th interval at any stage is exactly x_i/x. Therefore, by linearity of expectation, for each vector position i it holds that:

$$\mathbb{E}[V_i] = \sum_{j=1}^{x} \mathbb{E}[Y_{i,j}] = \sum_{j=1}^{x} \Pr[Y_{i,j} = 1] = \sum_{j=1}^{x} \frac{x_i}{x} = x \cdot \frac{x_i}{x} = x_i$$

so the first RSF condition holds.

Regarding the second RSF condition, for the entire vector $\mathbf{V}$ it holds:

$$\sum_{i=1}^{n} V_i = \sum_{i=1}^{n} \sum_{j=1}^{x} Y_{i,j} = \sum_{j=1}^{x} \sum_{i=1}^{n} Y_{i,j} = \sum_{j=1}^{x} 1 = x$$

□

Random seed (r). Our allocation mechanism needs some source of randomness to realize the Interval Sampling Algorithm. We propose to use the underlying blockchain to acquire a shared random seed among the participants. In particular, we propose that r is set as the VRF solution included in the block generated exactly after the end of the auction. The reason we chose this value is that it is unbiasable and unpredictable by any participant in the auction.

3.4 Analysis

We now prove our proposed solution described in Section 3.3 satisfies the desired properties of Section 3.2.

Theorem 1. *Algorithm $\mathcal{A}$ satisfies the market clearing property (Definition 1).*

Proof. The number of items that the algorithm allocates is as follows.

First, $\mathcal{A}$ by definition allocates $x = \sum_{i=1}^{n} \lfloor s_i \cdot I \rfloor$ items.

Second, $\mathcal{A}$ calls the random functionality $\mathcal{F}(\mathbf{x})$ with input the remaining items $y = I - \sum_{i=1}^{n} \lfloor s_i \cdot I \rfloor$. By definition of $\mathcal{F}$, i.e., since $\sum_{i=1}^{n} V_i = x$, these are all allocated.

Therefore, in total $x + y = \sum_{i=1}^{n} \lfloor s_i \cdot I \rfloor + I - \sum_{i=1}^{n} \lfloor s_i \cdot I \rfloor = I$ items are allocated by $\mathcal{A}$. □

Theorem 2. *Algorithm $\mathcal{A}$ satisfies the unpredictability property (Definition 2).*

Sketch proof. The unpredictability of the algorithm depends solely on the selection of the random seed r. Assuming that the auction participants have no influence on the underlying blockchain, hence they cannot affect the random values produced by the L1, the random seed is unpredictable and unbiasable because it is produced after the bids are placed and it is not affected/correlated with the operations of the L2.

We note here, that even if the stakeholders that bid for slots in the rollup are also validators for the L1, they cannot know in advance the random seed as the outcome of the VRF cannot be precomputed.

In general, we assume there is always a way to extract from the L1 unpredictable and unbiasable randomness because otherwise, the L1 proof of stake consensus suffers similar problems. □

Theorem 3. *Algorithm $\mathcal{A}$ satisfies the slot-symmetry property (Definition 3).*

Proof. Given an unbiasable and unpredictable random seed (Theorem 2), the algorithm assigns the validators to slots uniformly at random, therefore each party is assigned to each slot with equal probability. Thus, slot-symmetry is satisfied in algorithm $\mathcal{A}$ – even if the slots are not "identical" items. □

Theorem 4. *Algorithm $\mathcal{A}$ satisfies the fairness property (Definition 7).*

Proof. Each party P_i is allocated by algorithm $\mathcal{A}$ the following items.

First, it gets $\lfloor s_i \cdot I \rfloor$ items (deterministically).

Second, it gets (on expectation) $s_i \cdot I - \lfloor s_i \cdot I \rfloor$ items.

Therefore, on expectation, on each auction a party P_i is allocated $\lfloor s_i \cdot I \rfloor + s_i \cdot I - \lfloor s_i \cdot I \rfloor = s_i \cdot I$, so the fairness property holds. □

Theorem 5. *Algorithm $\mathcal{A}$ satisfies the concentration property (Definition 8).*

Proof. The property is satisfied directly by the algorithm's definition as, in its first step, allocates $\lfloor s_i \cdot I \rfloor$ items to each party P_i. □

4 Sequencer Election

In this section, we leverage the auction designed previously to elect sequencers for each epoch. We define the desired properties of the sequencer election, describe the reward mechanism, determine the utility function of the sequencers, and show that the desiderata are met under rational validators that aim to maximize their utility.

4.1 The Model

There exist n parties $[P_0, \ldots, P_n]$ that want to participate in the rollup execution. The execution comprises "epochs". Each epoch consists of e slots where each slot corresponds to the generation of a block in the underlying L1. Each slot shall be assigned to a party called the sequencer that is responsible for periodically publishing the rollup data on the L1, typically for a reward. The sequencers are responsible for the liveness of the rollup.

Each party P_i has a valuation V_i for participating in the rollup; this valuation expresses the willingness of the party to become sequencer, and its monetary value depends on the mechanism design, i.e., what "price" does the party pay to participate as sequencer. We denote the valuation percentage of party P_i by $v_i = \frac{V_i}{\sum_{j=1}^{n} V_j}$. The parties' valuations are not publicly known. Instead, what is publicly known at the beginning of each epoch, are the stakes reported by the parties: S_i and s_i denote the stake amount and stake percentage of party P_i, respectively.

Threat model. We consider all parties to be rational, aiming to maximize their utility function as will be defined later in Section 4.4. We further assume an adversary may control up to 1/3 of the total valuation of all the parties that want to participate in the rollup. This means that the adversary may control more than 1/3 of the sequencers if the mechanism is not truthful, meaning that the parties report stakes that do not correspond to their true valuation for participation in the rollup.

4.2 Desired Properties

The sequencers of the Milkomeda rollup are responsible for periodically publishing the rollup data, therefore for the liveness of the rollup. To ensure the liveness threshold is not violated, even under our threat model (a hybrid model of a static adversary corrupting up to 1/3 of the stake valuation while the rest is rational) we must show the following properties: *slot clearance*, *truthfulness*, *egalitarianism*, and *individual rationality.*

Slot clearance ensures that all slots will be assigned a unique sequencer, and therefore the mechanism has the possibility to always make progress as fast as possible. Note that this property is not necessary as an absolute; if some slots are left empty or multiple sequencers are allocated to some slots the protocol may still make progress, just slower. Imposing slot clearance may minimize the liveness parameter (u) during synchrony. But to guarantee liveness a relaxed probabilistic notion of slot clearance is strictly necessary (much like chain growth for the blockchain).

Definition 9 (Slot clearance)**.** *The mechanism allocates exactly one party to each slot.*

Truthfulness encapsulates that the parties should report (via bidding) their true valuations on the rollup participation. Egalitarianism on the other hand, conveys that parties are awarded fairly according to their reported bids, which implies that they are elected sequencers proportionally to their bids and that long-range attacks (e.g., acquiring more than u consecutive slots) are not possible. Intuitively truthfulness in combination with egalitarianism guarantee that the stake valuations correspond to uniformly distributed valuation-proportional slots; meaning that the security threshold on the valuations is transferred to the number of elected sequencers which are securely distributed in the epoch. Hence, the adversary cannot gain a stake-disproportional advantage which may lead to a liveness violation.

Definition 10 (Truthfulness)**.** *Parties report their true valuation for the rollup participation, i.e.,* $V_i = S_i$*, for all* $i \in [n]$*.*

Definition 11 (Egalitarianism)**.** *Parties are rewarded (on expectation) proportionally to their self-reported stake percentage* s_i*.*

Individual rationality captures that the dominant strategy of a rational validator is to be active during all its slots as a sequencer and help the rollup make progress by publishing the data on-chain. This property ensures that rational sequencers will not go idle, enabling the adversary to violate liveness.

Definition 12 (Individual Rationality)**.** *Parties benefit from being sequencers, i.e., the rewards minus the costs of participation should be positive.*

The rewards minus the costs are typically expressed through the utility function of a party. By participation costs we do not mean the operating costs of the rollup (running the execution and publishing data on-chain); we consider these costs already calculated in the reward and willingness of parties to take part in the protocol. Instead, with operating costs we mean any potential monetary cost incurred by the mechanism, e.g., opportunity costs from long-term locked stake.

4.3 Sequencer Reward Mechanism

We now describe the exact protocol that realizes the sequencer election. At the beginning of each epoch, parties publicly lock their stake on the rollup-auction smart contract; this stake (S) corresponds to their bid. Then, we leverage the allocation mechanism described in Section 3.3, to elect validators and allocate them to slots. After the auction is over, all parties' stakes are unlocked, whether they are validators or not.

During each slot, the corresponding uniquely assigned validator called the *sequencer*, is responsible for batching the rollup data and publishing them on-chain. The rollup data do not have to be valid – the sequencer simply publishes an order. Hence, a malicious sequencer can only choose which data to include or stay inactive.

At the end of the epoch, the validators (or sequencers – used interchangeably) are called to lock up a specific amount of stake for each assigned slot. This amount is determined via a typical second-price auction (VCG mechanism for multi-item auctions), in order to guarantee incentive compatibility, i.e., that the parties will report as their stake-bid their truthful valuation. For each slot, if the sequencer locks the necessary stake (S') and promptly published rollup data within its slot (i.e., in the block the sequencer was assigned to), the validator will be awarded the sequencer reward r; otherwise, the sequencer will *not* be awarded the reward r, *even if it acted correctly at its slot.*

Apart from receiving the sequencers' rewards, locking the required stake at the end of the epoch enables the validators to participate in the safe exit protocol. Nevertheless, we will not examine this aspect of the mechanism in this section.

4.4 Utility function

The utility function of a party depends on the following:

- The *opportunity cost* of the stake to be locked at the end of the epoch, S'. We denote this opportunity cost for party P_i by $q \cdot S'_i$, where q is a global, constant factor that expresses the opportunity cost of locking up one unit of funds for a bounded time period.

- The expected *sequencer rewards* to be awarded at the end of the epoch, denoted by $r \cdot k$, where k is the number of slots during which party P_i was active. k may be at most equal to the number of slots the party was awarded from the allocation algorithm in this epoch.

- Whether a validator may acquire more than u consecutive slots, effectively *violating the liveness* bound. In this case, we consider the utility of a party infinite.

In short, the utility of a party P_i is

$$u_i := \begin{cases} 0, & \text{if } P \text{ is not assigned to any slots} \\ r \cdot k - q \cdot S'_i, & \text{if } P \text{ is active during } k \text{ slots \& locks } S'_i \text{ at the end of epoch} \\ 0, & \text{if } P \text{ did not lock } S'_i \text{ at the end of epoch} \\ \infty, & \text{if } P \text{ is awarded } k > u \text{ consecutive slots} \end{cases} \tag{1}$$

Unlocking the stake. The auction mechanism should guarantee that the parties bid truthfully, meaning they lock the maximum amount of stake they are willing to "maintain" during an epoch for their participation in the rollup. We note that the stake is not locked during the epoch and thus the opportunity cost of the stakeholder is very low: the only requirements are to be able to lock a specific amount per slot at the end of the epoch to collect the rewards, and of course, that these rewards are more beneficial to the stakeholder than the operating costs of being a sequencer. We further note that unlocking the funds encourages much larger participation of stakeholders in the rollup which significantly enhances the rollup security. Potential drawbacks are discussed later in Section 4.6.

Utility from participation in safe exit protocol. We do not consider in this section the dual functionality of validators as both sequencers and safe exit validators. In particular, in this protocol, we only examine the fair and randomized allocation of stakeholders in rollup slots. Next, we will show that these properties guarantee the liveness of the rollup while bridging demands the design of a safe exit protocol that is incentive compatible and not manipulable by an adversary. We will address the safe exit in Section 5.

4.5 Analysis

In this section, we prove our proposed sequencer mechanism satisfies the desired properties under our threat model, and conclude the section by proving the liveness of the Milkomeda rollup that employs our design.

Theorem 6. *The mechanism satisfies slot clearance.*

Proof. This property is directly derived from the market clearance property of the auction mechanism, see Theorem 1. □

Theorem 7. *The mechanism satisfies truthfulness.*

Proof. The parties' utility can be increased if their locked stake at the end of the epoch decreases. However, to determine this locked stake, a VCG mechanism is used. Hence, the stake of a slot does not depend on the bid of this slot's sequencer and thus the parties cannot increase their utility by misreporting their valuation. □

Theorem 8. *The mechanism satisfies individual rationality.*

Proof. To show that parties may only benefit from being active, it is enough to show that $r \cdot k - q \cdot S'_i > 0$. We know that $r \cdot k > q \cdot S_i$, since the mechanism is truthful (Theorem 7) and the party P_i wants to participate in the auction. Moreover, from VCG we have that $S_i \geq S'_i$, hence $r \cdot k > q \cdot S'_i$. □

Lemma 3. *In a slot-symmetric mechanism, a party with stake percentage s_i ($<1/3$) may be awarded k consecutive out of e total slots with probability $(e-k) \cdot s_i^k$.*

Proof. By the slot-symmetry property (Theorem 3), each slot is equally likely to be assigned to any validator, $\forall P_i, \forall j \neq j' \in I, Pr[Y_{i,j} = 1] = Pr[Y_{i,j'} = 1] = s_i$. Hence, the probability of a party P_i with stake s_i to have the consecutive slots $1, 2, \ldots, k$ is s_i^k. Applying union bound to take into account all possible consecutive slots, we have that the probability of party P_i to have any k consecutive slots is $(e - k) \cdot s_i^k$. □

Theorem 9. *The mechanism satisfies egalitarianism.*

Proof. Suppose our mechanism is not egalitarian, i.e., there is at least one party P_i that is awarded an amount of rewards that is not proportional to their bid s_i. There are two cases: a) the party is awarded less than its fair share, or b) the party is awarded more than its fair share.

The parties are awarded a reward r per slot they have won in the slot auction and they were active (see utility function – equation 1). Furthermore, from the fairness property of the auction (Theorem 4), each party is assigned an average number of slots that is proportional to their stake. Therefore, if active on all slots, a party's expected reward is proportional to their bid s_i. So case (a) can only occur if the party was inactive in one or more of their assigned sequencer slots; contradicts individual rationality (Theorem 8).

For case (b), a party can only increase their rewards more than their fair share by acquiring $k > u$ consecutive slots from the auction (see equation 1). By Lemma 3), the probability of a party P_i with stake $s_i < 1/3$ to have k consecutive slots is $(e - k) \cdot s_i^k$.

Thus, the parties are rewarded proportionally to their self-reported stake percentage s_i, and our mechanism is egalitarian. □

Liveness. Next, we show that the Milkomeda rollup that employs the sequencer mechanism described in Section 4.3 satisfies liveness.

Theorem 10. *The Milkomeda rollup satisfies liveness with probability* $1 - (e - u) \cdot a^u$*, where* $e \in \mathbb{N}$ *is the number of slots in an epoch,* $a \in [0, 1]$ *the adversarial threshold, and* $u \leq e$ *the liveness security parameter.*

Proof. To show the Milkomeda rollup satisfies liveness we must show that there is a finite upper bound of blocks u on the L1 within which a transaction that is submitted in the rollup will be included on-chain and thus executed.

We assume that active (or correct) sequencers will include all pending transactions. Since rational validators will always be active due to individual rationality (Theorem 8), it is enough to bound the expected number of slots for having a correct sequencer. With any adversary controlling less than half the stake, the expected number of slots for having a correct sequencer is two, while bounding the existence of a correct sequencer with a probability dependent on the epoch length e can be derived by Lemma 3. We note that from the same lemma we can deduce that for large enough security parameter u (dependent on e), no party can control more than u consecutive slots, so liveness will not be violated. □

4.6 Elective features and alternatives

Fully-adaptive adversaries. To defend against fully adaptive adversaries, we can employ VRFs for the sequencer election, among the validators that are responsible for each epoch. In particular, we may employ the cryptographic sortition of Algorand [GHM$^+$17]. However, this solution means that there will be some empty slots and some with multiple leaders which may require more rigorous analysis to show liveness.

Locking the stake. In our proposal, the stakeholders lock their stakes only during the auction and at the final stage of each epoch. This feature allows stakeholders to participate in the rollup with very low opportunity cost and hence motivates the participation of all L1 stakeholders. However, this design choice bears some drawbacks.

First, it enables an adversary to loan stake to break the security threshold; this attack is out of the scope of this work, as rational agents would never loan such amounts but instead participate in the protocol themselves. Nevertheless, it is an attack vector that can be mitigated if the validators keep their stake locked during the entire epoch.

Second, entering and exiting the rollup is only allowed during the change of an epoch where the safe exit protocol operates. This allows us to separate totally the liveness from the bridging of the protocol, which makes our analysis under rational players more approachable. We can, however, design a protocol where parties may leave and join the rollup on demand and not only during specific time periods. We anticipate that this approach would require sequencers to play an active role in the bridging of the protocol. In this case, the stake should be locked throughout the execution for accountability purposes and a composable incentive analysis of the sequencer and safe exit protocols would be needed to show bridging.

5 Safe Exit

In this section, we investigate potential protocols for implementing the safe exit. First, we provide an ideal safe exit protocol and prove the liveness and bridging properties of the ideal protocol. Then in Section 5.2, we provide a realization of the ideal protocol under the assumption of an honest supermajority among the elected validators. In Section 5.3 we examine the case where the validators are rational, and indicate that designing a safe exit protocol under rational validators may be impossible[7]. In the remaining sections, we discuss possible ways to circumvent the implied impossibility and design a safe exit protocol secure under rational validators.

5.1 Ideal safe exit

A safe exit protocol has the following functionality: it takes as input the request of a client $\mathcal{C}$ to exit the rollup along with the ordered rollup transactions $\mathcal{T}$, and returns a state $t_{\mathcal{C}}$. $t_{\mathcal{C}}$ is the correct state that $\mathcal{C}$ can leave the rollup, i.e., the state that would be derived by any correct executor of all transactions $\mathcal{T}$ with respect to $\mathcal{C}$.

Suppose there is an oracle $\mathcal{O}$ that realizes the aforementioned functionality, that is, $\mathcal{O}(\mathcal{C}, \mathcal{T}) = t_{\mathcal{C}}$, where $\mathcal{C}$ is an existing client of the rollup, $\mathcal{T}$ the rollup ledger (ordered list of transactions), and $t_{\mathcal{C}}$ is a state that is consistent with the language of the L1. Whenever a client wants to exit the rollup, the client calls the safe exit oracle and exits the rollup in the returned state.

Given this oracle, we will show next that the Milkomeda rollup is secure under our model. In the following sections, we will discuss various directions to realize such an oracle.

Security analysis. We now show that given a safe exit oracle $\mathcal{O}$, the Milkomeda rollup is secure, meaning it satisfies liveness and bridging.

Theorem 11. *Given a safe exit oracle $\mathcal{O}$, the Milkomeda rollup satisfies bridging for $w = e$.*

Proof. Any participant can enter the rollup at the beginning of a new epoch. Moreover, any client that wants to leave the rollup may call the safe exit oracle at the end of each epoch. The oracle always returns the state of the client that an honest client would report, i.e., in accordance with the rollup ledger $\mathcal{T}$. Hence, bridging is satisfied for $w = e$. $\square$

Theorem 12. *Given a safe exit oracle $\mathcal{O}$, the Milkomeda rollup satisfies liveness.*

Proof. Directly derived from Theorem 10, as the safe exit protocol does not affect the liveness of the rollup. $\square$

[7]The analysis is not exhaustive, but it covers all known approaches. We use it as an indication of what is feasible with the methods that are known so far.

5.2 Safe exit under honesty assumptions

We first show a simple way to realize a safe exit oracle under honest supermajority assumption. We then discuss the case where 1/3 of the validators are honest.

The protocol. A simple safe exit protocol that satisfies the desiderata is as follows: When a client wants to exit the rollup at the end of an epoch, the client must provide the smart contract with at least 2/3 of the validators' signatures. Then, the client can exit the rollup on that state. This protocol realizes the safe exit oracle.

5.2.1 Honest supermajority

It is straightforward to see that assuming an honest supermajority, i.e., 2/3 of all validators, guarantees liveness and bridging.

Liveness. Liveness is guaranteed as long as one honest validator is chosen to act as a sequencer within u blocks. Assuming 2/3 honest validators makes the probability of breaking liveness (i.e., electing malicious validators for u consecutive blocks) reasonably small (cf. Theorem 10).

We note that proving liveness in this setting is straightforward as no argumentation for rational validators is necessary.

Bridging. Bridging is satisfied as the protocol always returns the correct exit state upon request within e blocks. Specifically, since 2/3 of the validators are honest, they will always sign exit transactions correctly (i.e., allowing users to exit the rollup with the correct amounts of funds) and, equivalently, the malicious (1/3) validators cannot produce a valid signed exit transaction.

5.2.2 Honest 1/3

Assuming 1/3 of all validators are honest guarantees liveness but does not guarantee bridging.

Liveness. Liveness is guaranteed if an honest validator is elected as sequencer within u blocks. Since now 1/3 of all validators are honest, meaning that 1/3 of all the sequencers are honest either the probability of violating liveness increases (but not significantly – cf. Lemma 3) or the security parameter u must increase (to maintain the same level of security). In any case, liveness will be satisfied.

Bridging. Bridging, however, is not guaranteed. Specifically, since the honest validators control only 1/3 of the votes for the exit protocol, a coalition of more than 1/3 (up to 2/3) of validators can block (i.e., not sign) any exit transaction they choose. Nonetheless, a valid exit transaction *has to* be signed by at least one honest validator. Therefore, no malicious coalition can violate the correctness conditions for exiting, i.e., a party cannot exit the protocol with more than the appropriate funds. Interestingly, the safety aspect of bridging cannot be violated but instead the liveness aspect of bridging may.

5.3 Impossibility of rational security without fraud-proofs

In this section, we show that bridging the rollup with the L1, i.e., designing a safe exit protocol, that is secure under rational validators is *impossible without any notion of fraud or validity proofs*.

The challenge lies in the fact that the rollup operations are not recognizable by the L1 language and thus the L1 consensus participants, which we will call the L1-validators[8], cannot verify if a state is correct or not according to the rollup state transitions (aka transactions). For instance, the Milkomeda rollup allows for EVM computations while the underlying chain it is deployed on (e.g., Algorand) is not EVM-compatible, hence the Algorand validators cannot evaluate the exiting state of a Milkomeda client.

Naturally, the first idea is to leverage the validators[9] to vote on the exit state of a rollup client (as described in Section 5.2), encourage them to participate via rewards, and disincentivize them to cheat via slashing. However, the slashing must be initiated either by the client exiting the rollup (cheater) or the other validators, and the scheme should be secure, such that no party loses money, e.g., via bribery attacks. In this section, we show that any scheme that only relies on monetary incentives, rewards, and punishments, is not secure under rational validators.

Bridging attack. Suppose there is a safe exit protocol, where the realization of the safe exit oracle is done by the elected validators or a subset thereof, denoted V. This means that the correct state for any address that exits the rollup is determined by the set V under a pre-specified rule R, denoted R_V. For instance, the correct exiting state may be the output of a consensus among the V validators, or the state that is signed by the majority of V, etc. All these are different rules that realize the safe exit oracle, $R_V(\mathcal{C}, \mathcal{T}) = t_{\mathcal{C}}$ for any client $\mathcal{C}$ that wants to exit the rollup.

Note now that there is no way to prove fraud or validity of an exit state to the underlying L1. This means that the "truth" is indeed determined by the rule R_V and no one can dispute it, since the fraud is not provable. Allowing disputes that cannot be verified creates the opposite problem of disputing truthful statements. Therefore, the set V has total control over the "truth" of the rollup, i.e., the exit state of any client. This means that the validators V can create a multisig address and a smart contract on-chain that divides the earnings of this address equally among them. Then, they may enter with this address the rollup as client $\mathcal{C}$. This must be allowed otherwise the bridging property is violated. Then, $\mathcal{C}$

[8]L1-validators are different than the validators that are the elected rollup operators.

[9]We assume clients are not part of this mechanism since they have the same access with the validators but no locked stake that may be slashed for misbehavior.

requests to exit the rollup with all the money of the rollup M (or any maximum amount allowed). Regardless of the selected rule, the set V outputs $t_{\mathcal{C}} = M$ in total agreement and exits the rollup with the total amount. Note that, no one can prove fraud occurred, hence the validators will leave with their stake intact as well as their share of the rollup money.

This attack demonstrates that a naive realization of the safe exit protocol under rational parties is vulnerable to bridging attacks. Moreover, it is indicated that no protocol can be secure without some notion of validity-proof or fraud-proof if the exit rule is determined solely by the validators of the rollup. The intuition behind this impossibility stems from the typical operation of rollups that inherit their safety from the blockchain. Removing this guarantee leaves the rollup vulnerable to bridging attacks. This fact also indicates that possible solutions may involve the L1-validators either via the L1 consensus or by creating some notion of fraud-proof that is understandable to the L1. In the next sections, we explore various such directions to bypass this impossibility.

5.4 Safe exit via witness encryption

In this section, we explore the idea of using witness encryption to circumvent the impossibility of Section 5.3.

Consider an NP language $\mathcal{L} \subset \{0,1\}^*$ with an efficient witness relation R (by definition, statement $x \in \mathcal{L} \Leftrightarrow \exists w : R(x,w) = 1$). A witness encryption scheme allows one to encrypt a message in $\mathcal{M}$ with a statement such that it can be decrypted with a corresponding witness:

$$\forall m \in \mathcal{M}, \Pr[\mathrm{Dec}(\mathrm{Enc}(x,m),w) = m] = 1$$

For our application, we set $x_{P,R}$ to be “there exists a series of actions under which P has committed fraud against R” (for a suitable definition of “fraud”, expressed in EVM) – a witness would be such a fraudulent series of actions. When joining the rollup, P encrypts her private key (which carries her coins at the AVM level) with R’s public key and then witness-encrypts the resulting ciphertext under $x_{P,R}$ to obtain C. C is sent to R, who can decrypt it and punish P only if the latter commits fraud.

At a high level, the exit protocol is as follows. A party P asks to exit. This starts a timelock, within which other parties may prove fraud. If the timelock expires, P can unilaterally exit the rollup with its fair share of coins. The low-level exit mechanism is to be determined but can copy existing rollups and/or commit-chains in a straightforward manner.

Open problems:

- Avoid the need for one witness encryption per counterparty while still ensuring that the defrauded party is the one to be compensated with the fraudster’s coins. Otherwise, $O(n^2)$ ciphertexts are exchanged and interaction with all existing parties is needed when new party joins.
- Ensure that the encrypter encrypts the correct thing – some kind of “verifiable” witness encryption is needed for this.

5.5 From EVM to AVM

One possible avenue is to investigate the differences between the expressiveness of EVM and AVM. The goal here would be to build a mechanism that extracts, from EVM-based contracts, AVM-compatible fraud proofs. This would allow to circumvent the impossibility of Section 5.3 and provide an incentive mechanism for safe exit. This is one of the currently leading directions of research. Nonetheless, such a mechanism would not be generic, i.e., L1-agnostic like the other alternatives, but would be tailored to AVM-compatible ledgers.

5.6 Leveraging Ethereum

One possible solution to the design of a safe exit oracle is to leverage Ethereum. This solution demands the rollup operators/validators hold stake in both the rollup's L1, e.g., Algorand, and Ethereum. The initial auction would take place as described in Sections 3, 4 in the main chain of the rollup, let this be Algorand. Then, the sequencers will publish their data as described in Section 4 on the Algorand blockchain. At the end of the epoch though, the elected validators will be asked to lock their stake in Ethereum and in case anyone exits on a fraudulent state a fraud-proof will be submitted within a timelock on Ethereum and slash the validators responsible for the fraud. The exact design for this solution is still to be determined.

There are many open problems such as:

- Determine and minimize the relayed information across chains.
- Determine the minimum number of parties in the rollup that must act as light clients on Ethereum. Can we improve this by leveraging other techniques, such as super-light clients?
- At the current design the validators must hold stake in both chains, which is a heavy requirement. Can we enable some type of loan in a secure manner in order to accommodate the liquidity on Ethereum, and if slashed claim their funds on Algorand?

5.7 Leveraging the L1-validators

In this section, we describe a solution that circumvents our impossibility by involving the L1-validators in the rollup operation.

In contrast to the proposed solutions so far, the sequencers must now verify the transactions, so only valid rollup transitions are included. The L1-validators only include valid rollup transactions, hence they also validate the ordering of the sequencer. If the sequencer and the L1-validator both commit fraud, the next L1-validator will fork out the block of the malicious L1-validator. Hence, the current L1-validator will not include fraudulent/invalid rollup transactions. Instead, the L1-validator will slash the sequencer and claim his reward.

Some remarks and open questions on this approach are:

- This solution changes altogether the proposed construction as the sequencer is now responsible for publishing only valid data and therefore may tamper with the bridging of the rollup. In our constructions so far the sequencer could only interfere with liveness, which allowed a clear separation of operations and a simpler analysis.

- To prove that the next L1-validator will fork out the fraud block of the main chain in a rational setting, we have to show that the expected reward will drop if at least one future L1-validator is honest and may fork out his block.

- The slashing of a sequencer awards the L1-validator his stake in case of fraud. However, if the protocol is designed naively other L1-validators will exclude the current block to get the slashed amount themselves. This is the same problem with transactions with very high fees. To address this issue, we demand the sequencer be able to publish only on a specific block height and subsequently get punished by only one specific L1-validator.

- This approach meddles with the main consensus incentives. Is it enough to assume the underlying chain has an honest majority or shall we study the existing incentives of the chain? Are there any major issues or attack vectors introduced by our involvement?

References

[CPR19] Xi Chen, Christos Papadimitriou, and Tim Roughgarden. An axiomatic approach to block rewards. In *ACM Conference on Advances in Financial Technologies*, AFT '19, page 124–131, New York, NY, USA, 2019. Association for Computing Machinery. `doi:10.1145/3318041.3355470`.

[GHM$^+$17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery. `doi:10.1145/3132747.3132757`.

[RL11] Derek D. Reed and James K. Luiselli. *Temporal Discounting*, pages 1474–1474. Springer US, Boston, MA, 2011. `doi:10.1007/978-0-387-79061-9_3162`.

About Common Prefix

Common Prefix is a blockchain research, development, and consulting company consisting of a small number of scientists and engineers specializing in many aspects of blockchain science. We work with industry partners who are looking to advance the state-of-the-art in our field to help them analyze and design simple but rigorous protocols from first principles, with provable security in mind.

Our consulting and audits pertain to theoretical cryptographic protocol analyses as well as the pragmatic auditing of implementations in both core consensus technologies and application layer smart contracts.