AUCIL: An Inclusion List Design for Rational Parties

Sarisht Wadhwa*, Julian Ma†, Thomas Thiery†, Barnabé Monnot†, Luca Zanolini†, Fan Zhang‡ and Kartik Nayak*
*Duke University
{sarisht.wadhwa@, kartik@cs.}duke.edu
†Ethereum Research
{julian.ma, thomas.thiery, barnabe.monnot, luca.zanolini}@ethereum.org
‡Yale University
f.zhang@yale.edu

***Abstract*—The decentralized nature of blockchains is touted to provide censorship resistance. However, in reality, the ability of proposers to completely control the contents of a block makes censorship resistance relatively fragile. Inclusion list mechanisms have been informally discussed in blogs and forums as a means to improve censorship resistance in blockchains. However, the only formal previous design—FOCIL (Fork-choice enforced Inclusion Lists)—lacks a rigorous incentive analysis of its committee-based approach. This paper presents the first analysis of an inclusion list design– AUCIL. Our inclusion list design leverages multiple proposers to propose transactions and improve censorship resistance. The design has two key components. The first component is a utility-maximizing input list creation mechanism that allows rational proposers to achieve a correlated equilibrium while prioritizing high-value transactions. The second component, AUCIL (auction-based inclusion list), is a mechanism for aggregating the input lists from the proposers to output an inclusion list based on the auctions. We prove that the resulting two-phase design is resilient against censorship attacks and report on a prototype implementation to verify the claims.**

I. Introduction

Transaction censorship is the act of refusing to process valid transactions. Centralized finance services are prone to censorship because important intermediaries, such as banks and exchanges, can heavily influence which transactions are allowed. This power can be abused to suppress dissent, curb protests, and harm freedom of speech [1], [2], [3], [4].

Blockchains are often touted as censorship-resistant, yet block creators (e.g., proposers and builders in Ethereum) can readily censor transactions because they control block contents. Two factors typically drive censorship. First, self-censorship because of regulation: for example, transaction censorship on Ethereum surged after an OFAC sanction [5], affecting all users regardless of jurisdiction. Second, financial incentives: a malicious actor can bribe block creators to exclude a target transaction by offering a bribe slightly more than the transaction fee to be received by the block creator. The rise of Decentralized Finance (DeFi) has created various avenues where such bribed censorship is profitable [6], [7], [8]. Moreover, sustaining censorship for an extended time is feasible when the set of block creators is small, as in case of Ethereum, where more than 90% of all blocks in the Ethereum blockchain is built by only three dominant builders [9], [10].

Inclusion lists. To tackle this problem, the idea of Inclusion Lists (ILs) has been proposed [11], [12], [13]. The high-level idea is to introduce a new protocol to create so-called *inclusion list* and modify the blockchain protocol so that a block creator must include transactions in IL; otherwise, the block is considered invalid. The key challenge, of course, is to ensure that the protocol creating ILs has stronger censorship resistance than the blockchain itself. Fox et al. [8] show that using multiple parties for block creation can scale the censorship resistance by the number of parties who include that transaction. Building on this idea, proposals such as COMIS [14] and FOCIL [15], [16] distribute the creation of ILs to a set of parties, each outputting an inclusion list, and the union of their lists forms the *final* inclusion list. When properly designed, this can significantly increase the bribery cost because the adversary must now bribe multiple parties.

Challenges to the design. We observe that these designs can be problematic when composed with other validity constraints of a blockchain protocol, in particular, the block size limit. Most blockchains enforce a block size limit so that blocks can be efficiently processed by the network. To be compatible with block size limits, the above designs allow a block creator to not have to include *anything* in the inclusion list if it creates a full block. This allows for various ways to bypass the inclusion list protocols. While full blocks are not common in Ethereum (since its transaction fee mechanism targets half-empty blocks in expectation [17]), other blockchains still suffer from such a scenario being common instead.

To achieve stronger guarantees, we instead focus on *unconditional* inclusion lists [18], where all transactions in the final inclusion list must be included. This requires that the combined inclusion lists of all parties cannot exceed the block size limit. Unconditional designs are attractive because once a transaction is on the list, the only way to censor it is to forgo the block reward entirely.

Two challenges arise in the design of unconditional inclusion lists. First, since parties creating individual inclusion

lists are rational and strategic, we need to ensure that the equilibrium has ideal properties (e.g., robust to bribery). For instance, in a design where parties independently select the highest paying transactions, the outcome may not be an equilibrium, so they can deviate to improve their own rewards, leading to inefficient and potentially easy-to-censor outcomes. Second, the aggregation process, where individual inclusion lists are combined to form the final inclusion list, creates an attack vector: e.g., in a design with a designated aggregator, the aggregator may be bribed to censor transactions. Also, if the designated aggregator crashes, the protocol loses liveness.

Introduction to the solution—AUCIL. We address these challenges through two techniques. First, we use the solution concept of *correlated equilibrium* [19] to govern the transaction selection process. Specifically, parties in our protocol, called IL proposers, are given a suggested transaction selection designed in a way that if all IL proposers follow the suggestion, their utility is maximized, thus forming an equilibrium where no party has an incentive to deviate unilaterally from the suggestion. This mechanism ensures that parties prioritize high-value transactions and avoid any strategic deviation that might undermine censorship resistance.

Following the construction of individual inclusion lists, an aggregator aggregates these lists, combining the IL proposers' submissions into a final inclusion list. The aggregation process is reinforced by an *auction*, in which IL proposers compete to collect the maximum number of input lists to aggregate into a bid. The value of this bid is proportional to the number of input lists used in its construction. This bidding process ensures that IL proposers are incentivized to aggregate inclusion lists. For any adversary to exclude a broadcasted input list, it must censor the bid of an IL proposer or bribe the IL proposers to exclude the input from its bid an amount equal to the prize for winning the auction; otherwise, any IL proposer might choose to reject the bribe and win the auction instead, which it would have likely lost if it accepted the bribe.

Summary of Results. The resulting two-phase design for creating the inclusion list can be used as an add-on to any existing blockchain network (like Ethereum) with minimal changes to the consensus (requires only a change to add verification of a proof to the validity condition of the blocks). In summary, this paper:

1) provides the first formal definition for inclusion lists and introduces formal properties required for inclusion lists.
2) proposes a new definition for censorship resistance based on the amount of bribe required to censor each IL output.
3) proposes AUCIL, a novel inclusion list protocol combining correlated equilibrium and auction-based aggregation.
4) presents a rigorous analysis and proves significant improvement over the state-of-the-art (e.g., FOCIL [15]).
5) develop a simulation-based evaluation of the proposed system to assess the security guarantees.

A. Paper Outline

Section II gives an overview of AUCIL and the properties it achieves. Section III formalizes censorship resistance and outlines the motivation for using an inclusion list to improve censorship resistance for blockchain protocols. In Section IV, we present the first phase of the design, where we propose a greedy allocation of transactions to IL proposers to achieve input lists that follow a correlated equilibrium. The second phase is shown in Section V, where we use auctions to ensure that maximum input lists get used while aggregating input lists into inclusion lists. Analysis of adversarial censorship attack is shown in Section VI. Section VII discusses various open problems with the solution. Section VIII compares previous related work, and discusses properties offered by various previous related works.

II. Design Overview

To reiterate from Section I, the two major problems when constructing a censorship-resistant inclusion list design are the games created due to the interplay of 1) block limits and transaction selection, and 2) crash resistance and maximizing the number of non-censored participants. Thus, any inclusion list design must have the following two properties apart from censorship resistance: 1) there exists an equilibrium in the transaction selection strategy (formally introduced as Input Selection Equilibrium), and 2) it tolerates crash faults (θ-Crash Fault Tolerance). In designs involving multiple parties, censorship can arise at three distinct stages: (i) transaction selection by each party, (ii) aggregation of input lists into a candidate inclusion list, and (iii) final selection of an inclusion list by the block proposer. These motivate three separate censorship resistance properties: β_1-Input Censorship Resistance, β_2-Aggregation Censorship Resistance, and β_3-Blockchain Censorship Resistance.

To satisfy our two required properties and maximize the minimum of the three censorship resistance parameters, we introduce AUCIL, an auction-based inclusion list protocol. AUCIL employs a two-phase inclusion list design that maximizes utility for IL proposers while still ensuring strong censorship resistance.

A. Phase I: Input Selection

The goal of this phase is to achieve a *correlated equilibrium* [20], in which IL proposer are incentivized to select transactions according to the protocol's recommendations. This ensures that no IL proposer deviates from the recommendation without reducing their utility.

The reason we choose to induce a correlated equilibrium is as follows. If no predefined strategy were provided, a default action for rational IL proposers is to choose transactions probabilistically, which can be shown to be a mixed Nash equilibrium. This makes bribing seemingly more costly, because any transaction may be picked by any IL proposers, but it gives rise to undesirable strategic actions as rational IL proposers are incentivized to avoid choosing the same transactions as others, which reduces their utility. For example, rational parties

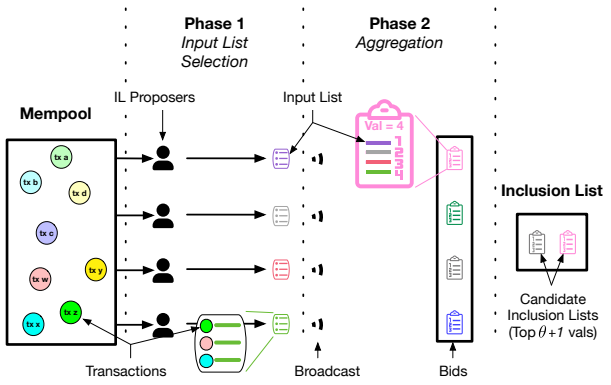


Fig. 1: **AUCIL outline:** A two-phase inclusion list protocol. β_1-Input CR is the cost for censorship before the end of Phase 1, β_2-Aggregation CR is the cost for censorship from the start of Phase 2. β_3-Blockchain CR is the cost of censorship during enforcement of the inclusion list.

might prefer waiting for other parties to declare their selections first, leading to latency games. Moreover, a coordination-based attack is possible, where an adversary acts as a coordinator to make suggestions that improve the utilities of IL proposers (by helping avoid choosing the same transactions as others) while censoring a target transaction.

These strategic actions can be avoided if the protocol induced a correlated equilibrium approach where the protocol plays the role of the coordinator and suggests a utility-maximizing selection. To do so, we propose a two-step transaction selection algorithm, which first greedily chooses high-utility transactions, and then allocates them to IL proposers in a round-robin fashion. We will show in Section IV, that the resulting allocation follows a correlated equilibrium. Note that when the number of transactions exceeds the capacity of the inclusion list, the lower-paying transactions will not be selected and cannot be offered any censorship resistance, fundamental in any design. Thus, AUCIL provides higher-paying transactions with higher censorship resistance at the input layer.

At the end of this phase, IL proposers broadcast their inclusion lists, which serve as the input to the aggregation phase. For this reason, we also call each IL proposer's list an *input list*. We now describe the next phase.

B. Phase II: Aggregation

In this phase, we introduce a novel auction-based aggregation protocol to incentivize IL proposers to maximize transaction inclusion.

We start with the following idea. All IL proposers produce aggregations of input lists, and the number of input lists in an aggregation is considered as its *bid*. The aggregation with the highest bid is selected by the block producer (this is a simplification; see below for details), and the IL proposer who produces it is awarded a reward. Using auctions allows the protocol to distinguish a crashed IL proposer from one whose input list is censored by some aggregators, because censored input lists will still be included by other rational aggregators.

To tolerate up to θ crash faults (i.e., θ parties may not submit a bid), we relax and allow the block producer to select any bid that is larger than $n-\theta-1$ other bids. By ensuring that all of the top $\theta+1$ bids include all the inputs they receive from other parties, we maximize censorship resistance at the aggregation layer in AUCIL.

However, the above design encourages IL proposers to withhold their input lists to get a bid higher than those who publicly broadcast. To avoid this, the bidding process incorporates a randomized bias factor generated using Verifiable Random Functions (VRFs) [21]. Using randomness helps most bidders identify that the probability with which they win the auction (in the absence of censorship) is low, and thus broadcasting their list to earn a fee from transactions yields a higher utility. This ensures unpredictability in the auction outcome, making it rational for most IL proposers to share their input lists.

Therefore, AUCIL ensures robustness through five key properties. It is *crash fault tolerant* (Property 1), allowing progress as long as at most θ parties fail to participate. The *input selection phase* achieves a *correlated equilibrium* (Property 2), ensuring no proposer benefits from deviating from their assigned transactions. At the *input stage*, censorship is deterred (Property 3) since rational parties include high-fee transactions to maximize utility. During *aggregation*, rational bidders incorporate all known inputs to strengthen their bids, preventing exclusion (Property 4). Finally, *blockchain-level censorship* (Property 5) is constrained, as the block proposer must choose from among the top $n-\theta$ bids, all of which include shared inputs—leaving full block omission as the only censorship avenue.

III. Problem Statement

As illustrated in Figure 2, our setting involves two protocols. The first protocol is a **blockchain protocol**, which realizes the abstraction of a state machine replication system [22]. The central building block of state machine replication is multi-shot consensus, where a set of parties (a subset of which may be faulty) agree on a dynamically growing sequence of log. Such a protocol provides: (i) safety: non-faulty parties agree on each log position, and (ii) liveness: every input that arrives at all parties is eventually recorded in the log. Typically, the log is referred to as a blockchain, and a log entry is referred to as a block. Each block contains a set of transactions that external clients submit to the parties. We refer to the transactions that are not included in the log yet but received by the parties as their mempool. In this paper, we assume a class of blockchain protocols where there is a designated party that is responsible for producing the block at a given position in the log; we refer to these parties as proposers (or leaders).[1]

The second protocol is an **inclusion list protocol**, which operates simultaneously with the blockchain protocol. This

[1]We make this assumption for concreteness. At a high level, our result is more generally applicable.

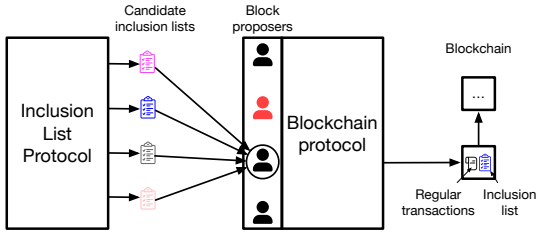


Fig. 2: **The setting of an inclusion list with a blockchain protocol.** The inclusion list protocol outputs a set of inclusion lists for each log position, and a block proposed in the blockchain protocol must include one of these lists.

protocol is run among a set of parties (that may or may not be the same as the ones running the blockchain protocol) that receive transactions in their mempool, and the protocol outputs a set of lists (of transactions). An inclusion list protocol instance is synchronized with the blockchain protocol execution (e.g., running the protocol instance for each position of the log). The key requirement enforced on a blockchain protocol augmented with an inclusion list is that, if an instance for a position j outputs a set S of lists, then the proposer of position j must include some $s \in S$ (s would be the inclusion list, enforcement of transactions within s is based on the inclusion list protocol design).

Motivation for using an inclusion list. The key motivation for maintaining an inclusion list is censorship resistance. Observe that if all of the parties in the blockchain are honest, then the mempool effectively is an inclusion list. This is because the next proposer will include the set of pending transactions (modulo block size constraints). However, if the proposers are Byzantine or rational, they may not be incentivized to include all of the pending transactions.

The inclusion list is thus used as an enforcement mechanism to ensure that proposers include a specific set of transactions; looking ahead, the goal would be ensure that the inclusion list protocol provides the desired censorship resistance property.

Definition 1 (Inclusion List)**.** *An inclusion list, denoted as $IncL^j$, for a log position j is a set of lists of transaction identifiers such that a proposer of log position j must prioritize including transactions of one of the lists in a block within the blockchain protocol.*

In practice, the transaction identifiers could be the transaction hash or a header containing some data that identifies the transaction, like transaction nonce and a sender's address. To simplify, we will assume that the output would contain the entire transaction. We state that transactions in one of the lists are *prioritized* since the transactions can be dropped if the block is full. A related definition is that of an unconditional inclusion list, which requires the list to be smaller than the block size.

Definition 2 (Unconditional Inclusion List)**.** *An unconditional inclusion list $IncL^j$ for a log position j is a set of lists of transaction identifiers such that a proposer of log position j must include the entirety of one of the lists in the block, i.e., $\exists i : IncL_i^j \in IncL^j$ such that all $tx \in IncL_i^j$ are also in the produced block.*

Here each IncL_i^j is termed as a *candidate inclusion list*. This paper mainly deals with creating an unconditional inclusion list, and any references to an IL refer to an unconditional inclusion list.

Model. All parties in the paper are assumed to be rational, i.e., they try to maximize their utility. The parties' utility is defined as the amount of stake they earn via protocol rewards and external bribes, subtracting protocol-level penalties like slashing. To account for external bribes, we consider an external adversary as a non-participating protocol member with undefined utility from censoring a transaction in the system. The adversary aims to censor a particular transaction (tx_τ which we will refer to as a target transaction) while minimizing the cost incurred. This cost is incurred when the adversary bribes a rational party into an action that otherwise reduces the said party's utility. In practice, the adversary may control some consensus parties, but to simplify the context, we can assume the controlled nodes to be rational and bribed. Throughout the paper, we assume that all parties are in sync, i.e., all parties have the same mempool, and the network is also synchronous.

Desired properties. Before discussing the desired properties of an inclusion list protocol, we introduce some recurring terms in the paper. Each party chosen to add transactions to a candidate inclusion list (in the set of output lists) is referred to as an *IL Proposer*. We refer to the input that each IL proposer selects as an *input list*. We label the input list as InpL_i^j for log position j and IL proposer P_i. These terms are illustrated in Figure 1. Looking ahead, these lists will be aggregated with other input lists from other IL proposers to create candidate inclusion lists.

Previous works [14], [15] use the term Local Inclusion List to refer to the inputs of IL proposers. We deviate from this terminology since input lists for our protocol do not resemble the output inclusion lists in many cases. Contrary to previous work, since this paper deals with unconditional inclusion lists, we restrict the size of each input list as k, such that $n \cdot k \leq$ block size.

Next, we introduce the properties that an inclusion list protocol must satisfy to improve the censorship resistance of a blockchain. Our protocol achieves all these properties.

Crash fault tolerance: State Machine Replication systems require some degree of crash fault tolerance. Since an inclusion list protocol is an addendum to an SMR protocol, it should tolerate crash faults without degrading the liveness guarantees of the combined system. We assume that all parties assume other parties to be online for analyzing the game. This is justified by the analysis that on the Ethereum chain consistently more than 99% live validators [23].

Property 1 (θ-Crash Fault Tolerance)**.** *Let j denote the log position, and let $Q_j \subseteq N$ be the set of IL proposers that crash during log position j (at any point of the protocol). We say the protocol satisfies* θ-Crash Fault Tolerance *if, for any log position j such that $|Q_j| \leq \theta$, the protocol is still live, i.e., the protocol produces at least one candidate inclusion list for log position j.*

In other words, the protocol continues to process transactions as long as the number of crashed proposers per log position does not exceed the threshold θ.

Input selection: The next property to consider would be to determine which transactions IL proposers would choose to add to their input list. Since we are dealing with rational IL proposers, an equilibrium condition must be established. To achieve this, we assume that each party has received the same set of transactions from the network similar to [24]. This assumption, however, does not exclude any party from including a transaction outside this set of transactions.

Property 2 (Input Selection Equilibrium)**.** *Let M be the set of transactions visible to all parties at log position j, and let N be the set of input list proposers. Each proposer $P_i \in N$ selects an input list $InpL_i^j \subseteq M$ of size k, following a strategy Ψ_i. Let $\mathbb{U}_i(InpL_i^j, InpL_{-i}^j)$ denote the utility obtained by proposer i when submitting $InpL_i^j$ while all others submit $InpL_{-i}^j = \{InpL_k^j\}_{k \neq i}$.*

The protocol satisfies Input Selection Equilibrium *if the strategy profile $\{\Psi_i\}_{i=1}^n$ forms an equilibrium. That is, for all $P_i \in N$, and for all alternative input lists $InpL_i^{j'} \subseteq M$ of size k, we have:*

$$\mathbb{E}_{InpL_{-i}^j \sim \Psi_{-i}} \left[\mathbb{U}_i(InpL_i^j, InpL_{-i}^j) \right] \geq \max_{InpL' \subseteq M} \mathbb{E}_{InpL_{-i}^j \sim \Psi_{-i}} \left[\mathbb{U}_i(InpL_i^{j'}, InpL_{-i}^j) \right].$$

Censorship resistance during input selection: This property deals with the cost of censorship during the input selection phase. This value is not just governed by the protocol being built, but also by the transaction fee. For example, if a transaction pays no fee, and the amount that an adversary would need to censor it would only be $\varepsilon \to 0^+$, regardless of the protocol being built. Similarly, if only 10 transactions can be accommodated by the protocol (due to block size limits), then the lowest-paying transaction amongst 11 different transactions will not have any censorship resistance, irrespective of the chosen protocol. Thus, when discussing censorship resistance, it is crucial to consider only transactions that are included at equilibrium.

Property 3 (β_1-Input Censorship Resistance)**.** *Let j denote the log position, $InpL_i^j$ be the input of IL proposer P_i. In the absence of an adversary, suppose all IL proposers follow an equilibrium strategy, i.e.,*

$$\forall P_i \in N, \quad \widetilde{InpL_i}^j \sim \Psi_i.$$

Then, if a transaction (tx_τ) that pays a fee (f_τ) appears in the input of an IL proposer at this equilibrium, then the transaction appears in the input of the IL proposer even in the presence of an adversary, that is

$$\forall P_i \in N, \quad tx_\tau \in \widetilde{InpL_i}^j \implies tx_\tau \in InpL_i^j.$$

unless the adversary incurs a cost of at least $\beta_1(f_\tau)$ to prevent its inclusion in any of the inputs to the inclusion list.

Censorship resistance in aggregation: The next property we aim to ensure is that each candidate inclusion list must include every transaction that is declared by any party in their input to the inclusion list construction process. In some cases, more than one party would include this transaction; thus, the cost of censorship would depend on this number.

Property 4 (β_2-Aggregation Censorship Resistance)**.** *Let j denote the log position and let $IncL^j$ be the set of candidate inclusion lists. Let N be the set of all IL proposers, and for each $P_i \in N$, let $InpL_i^j$ denote their input to the inclusion list construction process. Define n_t as the number of IL proposers whose inputs include transaction tx, i.e.,*

$$n_t = \left| \left\{ P_i \in N \;\middle|\; tx \in InpL_i^j \right\} \right|.$$

If $n_t \geq 1$, then the tx must appear in all candidate inclusion lists, i.e.,

$$\forall IncL_i^j \in IncL^j, \quad tx \in IncL_i^j.$$

unless the adversary incurs a cost of at least $\beta_2(n_t)$ to prevent its inclusion in any of the candidate inclusion lists.

Censorship resistance in enforcement Lastly, this property analyzes the cost to censor a transaction while enforcing the inclusion list design. If a target transaction tx_τ is included in all the candidate inclusion lists for the log position, it should be included in the block proposed for that same log position. If not, then the adversary must bribe the block proposer at least β_3 to remove the transaction tx_τ.

Property 5 (β_3-Blockchain Censorship Resistance)**.** *Let j denote the log position and let $IncL^j$ be the set of candidate inclusion lists. Suppose a transaction tx satisfies:*

$$\forall IncL_i^j \in IncL^j, \quad tx \in IncL_i^j.$$

Then tx must appear in the block for log position j, unless the adversary incurs a cost of at least β_3 to prevent its inclusion.

Note that throughout the paper, probabilistic attacks are not considered, i.e, if an adversary wants to censor a transaction, then the transaction should not be accepted on-chain with any probability. A probabilistic equivalent of all these properties could be considered, but is left for future work.

Theorem 1. *All unconditional inclusion list designs have a β_3-Blockchain Censorship Resistance of R (i.e., the reward received for block production).*

The proof follows from the fact that once a transaction is included in the inclusion list, the only way to censor is to

drop the block, foregoing the reward that the block producer receives. In Ethereum, the set of attesters can choose not to vote for a block that includes the target transaction without any consequences. However, any such manipulations by attesters are hard to find and model, and therefore considered out of the scope of this paper.

Censorship resistance. Informally, we define the censorship resistance of an inclusion list protocol for a set of transactions to be a list of bribe amounts (one for each transaction in the input) that an adversary must spend to remove them from sufficiently many parties' output, as formalized below.

Definition 3 ((B, θ, T)-Censorship Resistance)**.** *Given an adversary with an arbitrary bribing budget b, a protocol running with a mempool M is said to be (B, θ, T)-censorship resistant with a tolerance for bribes represented by B, for a set of target transactions $T(M)$ if there exists a set of at least $n-\theta$ parties such that for all $i : t_i \in T(M), b_i \in B$ if $b < b_i$ all parties in this set include the transaction t_i in their output.*

That is, given transactions in mempool M, a (B, θ, T)-Censorship Resistance protocol provides the following guarantee: for any transaction $t_i \in T(M)$, the same set of at least $n - \theta$ parties must include each transaction t_i in their respective outputs unless the bribe budget exceeds the tolerance corresponding to the transaction b_i. Note that θ is a knob to adjust crash tolerance ($\theta = 0$ means no crash fault tolerance). $T(M)$ is a subset of transactions in the mempool M for which the protocol can provide censorship resistance.

A transaction is said to be (B, θ, T)-Censorship Resistance if at equilibrium based on strategy Ψ (Property 2 value is Ψ_i) the transactions included in the inclusion list is T. For all transactions $tx_i \in T$ which pays f_i, b_i is the minimum of $(\beta_1, \beta_2, \beta_3(f_i))$ (Properties 3 to 5) while tolerating crash faults up to θ nodes (Property 1).

Example 1. *Consider the following examples for using the above definition to describe censorship resistance guarantees.*

Suppose there exist six transactions $M = \{m_1, m_2, m_3, m_4, m_5, m_6\}$ paying a fee of $\{10, 9, 8, 3, 2, 1\}$ respectively. Suppose we don't tolerate crash faults ($\theta = 0$) that can output at most, say, 4 transactions set by an exogenous block size limit. Our protocol (to be presented later) can guarantee censorship resistance with $B = \{20, 9, 8\}$ and $T(M) = \{m_1, m_2, m_3\}$. There may exist an alternative protocol that guarantees censorship resistance with $B = \{10, 9, 8, 3\}$ and $T(M) = \{m_1, m_2, m_3, m_4\}$. Note that in this case, the censorship resistance of the two protocols is not necessarily comparable.

IV. Input List Building Protocol

This section presents a protocol for how an IL proposer should build an input list.

A. A Naïve Approach

Consider that the only action an IL proposer can take to increase its contribution score for a transaction is to include it in the input list. Thus, given the required properties of the contribution score, the only satisfying contribution score mechanism would be if the fee is divided equally amongst all parties that include the transaction in their input list, and the input list is included in the set of final inclusion lists. Let us consider IL proposers naïvely choosing the transactions that pay the highest fee independent of what other IL proposers might choose. This naïve scheme of greedily picking transactions without considering other parties' actions is not a Nash equilibrium. Given all other IL proposers' input lists that consist of greedily selected transactions, the rational choice for an IL proposer may not be to construct its own input greedily. The example in Table I confirms the stipulation.

Strategy	Objects Picked	Utility
Pick Top Paying	m_1, m_2	7
Alternate	m_3, m_4	15

TABLE I: Picking top-paying objects is not a Nash equilibrium. Consider three parties ($n = 3$), selecting two transactions ($k = 2$) each from $\{m_1, m_2, m_3, m_4, m_5, m_6\}$ with utilities $\{11, 10, 9, 6, 4, 3\}$ respectively. Other parties are assumed to follow the strategy of picking the top-paying transaction.

A natural equilibrium would be to consider mixed strategies where neither party knows the prior of the other. In this, each party would choose transactions such that a mixed Nash Equilibrium (MNE) is achieved. In this case, an MNE would be to choose each transaction with a probability $f_i / \sum(f_i)$ for a transaction tx_i paying a fee f_i (More details in Appendix A) such that selecting transactions from outside the probability distribution would result in lower expected revenue. However, we cannot rely on a mixed Nash equilibrium as is (without the use of commit and reveal primitives).

Primarily, two flaws exist in using a Mixed Nash Equilibrium strategy. The first is that the game is not coordinated. This leads to a scenario where an adversary can coordinate the game while censoring. Consider the following adversarial strategy: Contact all parties and tell each IL proposer that an external coordination has been reached, and choosing the suggested list would yield a revenue greater than what the IL proposer expects at Mixed Nash Equilibrium. To ensure that the IL proposer believes it, the adversary offers a conditional bribe in which if the IL proposer follows its list and receives a revenue lower than expected, then it would make it whole by paying the loss in revenue. This strategy is rational for each individual IL proposer not only to accept, but also initiate even in absence of an adversary.

A second flaw is that the IL proposers may engage in a timing game. Each IL proposer can wait for others to broadcast their input lists before creating their own. This creates a complex game where actions are influenced by the time taken to broadcast the input list.

B. Input List Building Protocol

As alluded to in the overview Section II, we use the notion of *correlated equilibrium* where parties are suggested to take

a particular action. A third party usually does this, but in our context, it will be a publicly known algorithm simulated by each party that would suggest appropriate action to the party. This action corresponds to the maximum utility that can be received by such a party, given that all the other parties follow the suggested action.

The idea behind using a correlated equilibrium is simple. If you know what the other parties will do, there is no point in waiting for them to produce and declare their input. Let us define this game more formally. Consider a setting with m objects (transactions) and n parties (IL proposers). Each party can select a maximum of k distinct objects. Each object m_i is associated with a utility value u_{m_i}, which is uniform across all parties. When multiple parties select an object, the utility u_{m_i} is shared (split) equally among the selecting parties. Further, after allocation, with probability $1-\gamma$, the allocation for each party is dropped (and given an empty set instead). The reason for this probabilistic dropping will be clear in the next section - since not all parties would want to broadcast their lists. The objective for each party is to maximize its expected utility. We assume that all IL proposers value transactions solely based on the fees they offer, i.e., $u_{m_i} = f_i$.

Define n_i as the number of parties selecting object m_i and N_i as the set of parties that have chosen m_i at the end of the phase. The expected number of parties that select the object m_i (and don't drop later) is γn_i. Let L_j denote the set of objects allocated to party P_j, and let $L = \{L_1, \ldots, L_n\}$ represent an allocation satisfying the problem's constraints. We define $\mathbb{U}_j(L_j, L)$ as the utility derived by party P_j from selecting objects in L_j given all other parties accept the allocation in L. [2]

$$\mathbb{U}_j(L_j, L) = \sum_{m_i \in L_j} \frac{u_{m_i}}{\gamma(n_i - 1) + 1} \tag{1}$$

Notice the denominator. For an external party viewing the problem, the expected number of parties that select the object is γn_i. However, for the party choosing it, the expected number of parties, apart from itself, would be $\gamma(n_i - 1)$. When considering the utility it will receive from the object, it will always include itself in the number of parties receiving the object, and thus, the utility it receives would be the fee distributed amongst $\gamma(n_i-1)+1$ parties. For ease of notation, let $\widetilde{n_i}$ represent the denominator ($\widetilde{n_i} = 1 + \gamma(n_i - 1)$).

We aim to find a strategy for the allocation of the objects, such that the party selects all the objects in the allocation as its action. The actions of all parties correspond to a correlated equilibrium (i.e. no other choice of selection of objects by the party could result in a higher utility, given all other players follow the action suggested by the allocation). This subsection aims to design an algorithm that is publicly known to all parties such that simulating the algorithm leads to an allocation that all parties would choose to accept as their selection. All parties observe their own set of objects and the sets of all other parties' sets of objects as recommended by the algorithm. Given this information, the party should have no incentive to deviate from the recommended objects.

Let us represent the constraints mathematically. The number of times each object can be selected is limited by the total number of parties.

$$\forall m_j \in M : n_j \leq n \tag{2}$$

, and each party can be allocated at most k objects

$$\forall L_i \in L : |L_i| \leq k \tag{3}$$

The last constraint we want to represent is for achieving correlated equilibrium. Suppose L is suggested to all parties. Correlated equilibrium states that choosing any other set of objects would lead to a utility less than or equal to the utility from the set that the algorithm recommends. Formally, let $M^{[\leq k]}$ represent a subset of $M = \{m_1, \ldots, m_m\}$ with cardinality less than or equal to k, we require

$$\forall L_{i'} = M^{[\leq k]}, L_i \in L : \mathbb{U}_i(L_i, L) \geq \mathbb{U}_i(L_{i'}, L) \tag{4}$$

Let $T(M)$ represent all the objects the protocol assigns to at least one party $T(M) = \bigcup L_i$. Consider the following constraint:

$$\begin{aligned} &\forall m_a \in T(M), m_b \in M : \\ &\exists i : (L_i \in L,\ m_a \in L_i, m_b \notin L_i) \implies \frac{u_{m_a}}{\widetilde{n_a}} \geq \frac{u_{m_b}}{\widetilde{n_b} + \gamma} \end{aligned} \tag{5}$$

In words, if there exists an object allocated to some party P and another object not allocated to P, then for P, swapping the objects cannot lead to a better utility. This is because swapping would increase the expected number of times the object appears by γ from the perspective of the party.

Lemma 1. *(Small Mempool) If the mempool contains fewer than k transactions, i.e., $|M| < k$, then any utility maximizing algorithm should assign every transaction to every proposer: $L_i = M$ for all $i \in \{1, \ldots, n\}$.*

Consider to the contrary, $|L_i| < |M|$. In such a case, there exists an object m_j not in the allocation L_i. Since L_i is a subset of M, $|L_i| < k$, and thus the object L_i can be added to the allocation.

Lemma 2. *(Correlated Equilibrium Reduction) If $|M| \geq k$, then any algorithm that satisfies constraint (5) also satisfies the constraint (4).*

Proof. Since $|M| \geq k$, $|L_i| = k$. This is because if $|L_i| < k$, then there exists an object in M, not in L_i, which can be added to increase the utility gained by the selection. Now let's assume, to the contrary, that constraint (5) holds but constraint (4) does not. The negation of constraint (4) states

$$\exists L_i \in L, \exists L_{i'} = M^{[\leq k]} : \mathbb{U}_i(L_i) < \mathbb{U}_i(L_{i'})$$

Let $L_{i'} \neq L_i$ represent the allocation with the highest utility (strictly greater than allocation L_i). If multiple such allocations

[2]The true utility is given by $\sum_{m_i \in L_j} \sum_{r=0}^{n_i-1} \binom{(n_i-1)}{r} \gamma^r (1-\gamma)^{n_i-1-r} \frac{u_{m_i}}{r+1}$. The used function is a good estimate given $\gamma \sim 1$

exist with the highest utility, consider $L_{i'}$ as the set that differs in the least number of objects from L_i. There exists an object m_a such that $m_a \in L_i$ but $m_a \notin L_{i'}$. (If no such object exists, then $L_{i'} \supseteq L_i$. However since $|L_{i'}| \leq k = |L_i|$, this is possible only if $L_i = L_{i'}$)

Consider $|L_{i'}| < k$. Adding m_a would only add to the party's utility; however, $L_{i'}$ was considered as the maximal such allocation, and thus such an $L_{i'}$ can only exist if $|L_{i'}| = k$. Since both $L_{i'}$ and L_i are of size k and there exists $m_a \in L_i$ but $m_a \notin L_{i'}$, there must exist $m_b \in L_{i'}$ but $m_b \notin L_i$.

Given all other selections $L_j \neq L_i$ remain the same as governed by the allocation, the number of times the object m_b is chosen increases by 1 in $L_{i'}$ compared to L_i. From(1),

$$\mathbb{U}_i(L_i) = \mathbb{U}_i(L_i \setminus m_a) + \frac{u_{m_a}}{\widetilde{n_a}}$$
$$\mathbb{U}_i(L_{i'}) = \mathbb{U}_i(L_{i'} \setminus m_b) + \frac{u_{m_b}}{\widetilde{n_b} + \gamma}$$

From constraint(5), in this case (where $\exists i : (L_i \in L, m_a \in L_i, m_b \notin L_i)$),

$$\frac{u_{m_a}}{\widetilde{n_a}} \geq \frac{u_{m_b}}{\widetilde{n_b} + \gamma}$$

Consider the set $L_{i''} = (L_{i'} \setminus \{m_b\}) \cup \{m_a\}$. $|L_{i''}| = |L_{i'}|$, and thus satisfies $M^{[\leq k]}$ property. The utility of this set is

$$\begin{aligned}\mathbb{U}_i(L_{i''}) &= \mathbb{U}_i(L_{i'} \setminus m_b) + \frac{u_{m_a}}{\widetilde{n_a}} \\ &= \mathbb{U}_i(L_{i'}) + \frac{u_{m_a}}{\widetilde{n_a}} - \frac{u_{m_b}}{\widetilde{n_b} + \gamma} \\ &\geq \mathbb{U}_i(L_{i'})\end{aligned}$$

This is a contradiction since $L_{i'}$ was considered as the set with the highest utility that differed in the least elements compared to L_i; however, we show the existence of another set with one less element differing from L_i, which has a utility greater than or equal to $L_{i'}$. □

The above set of constraints forms the basis of our problem. Any solution that satisfies the above constraints can be used as a third party that informs IL proposers of the input to pick.

C. A Greedy Algorithm

If we use the above constraints as invariants in a greedy, step-by-step selection and then allocate selected objects in a round-robin allocation, the final allocation would satisfy the required constraints by default.

Algorithm 1 presents a greedy algorithm that chooses and assigns the objects to parties while satisfying the invariants defined in Eq (2), (3) and (5). To do so, we choose objects one at a time, picking the object with the highest local utility $U \oslash N$, where $\oslash$ denotes element-wise division of the utility by the number of times the object would have been picked if selected now (in expectation). By picking the object, we increase the number of times it is selected by the probability that this selection of the object is broadcast. This updates N such that $U \oslash N$ represents the utility when the object is next selected. Next, if the object has been selected n times (making

Algorithm 1 A 2-Step algorithm for transaction inclusion

Require: $n \geq 0$, $m \geq 0$, $k \geq 0$, $\gamma \geq 0$
▷ number of parties, transactions, input list size, probability of broadcast

Ensure: L_i arrays for all $i \in N$
▷ final inclusion arrays for each party

Step 1: Choose Objects

$U \leftarrow [u_{m_1}, \ldots, u_{m_m}]$ ▷ Utility values for each transaction
$N_c \leftarrow [1, \ldots, 1]$ ▷ Count array (corresponding to $\widetilde{n_i}$), initialized to 1 for each transaction
$S \leftarrow \{\}$
for $\_\_ \in 1$ to $n * k$ **do**
  $U_{curr} \leftarrow U \oslash N_c$ ▷ Compute current utility by element-wise division of utility by the number of times the object has been selected
  $s \leftarrow \text{argmax}(U_{curr})$ ▷ Find the index of the maximum value in U_{curr}
  $S \leftarrow S \cup \{s\}$
  if $U[s] = -1$ **then** break
  ▷ if the maximum utility for objects is -1, then all objects have been selected n times (Line 10 sets -1 for objects selected n times)
  $N_c[s] \leftarrow N_c[s] + \gamma$ ▷ Increment the expected number for the next time this object is selected
  if $N_c[s] + \gamma \geq n\gamma + 1$ **then** $U[s] \leftarrow -1$
  ▷ Set utility of object to -1 if it has been allocated n times

Step 2: Allocate Objects

$U \leftarrow [u_{m_1}, \ldots, u_{m_m}]$ ▷ Reset utilities
$N \leftarrow [1, \ldots, n]$ ▷ Array of party identifiers
$\forall i \in N : L_i \leftarrow \{\}$ ▷ Inclusion sets for each party, initialized to empty
$U_f \leftarrow U \oslash_s (N_c \ominus [\gamma]^m)$ ▷ $\oslash_s$ is element-wise safe division (returns 0 if dividing by 0). This computes utility for each selected object, adjusting for the extra γ added for the preparation of the next selection
$A \leftarrow sort(S, \text{key} = (-U_f[s] \text{ for } s \in S, s))$
▷ Get indices of objects in descending order of utilities ($[U_f[A[0]], U_f[A[1]], \ldots]$ is in descending order)
for $j \in 1$ to $|A|$ **do**
  $L_{(j \bmod n)+1} \leftarrow L_{(j \bmod n)+1} \cup \{A[j]\}$
  ▷ Assign objects in round-robin fashion
return $\forall i \in N : L_i$ ▷ Return the inclusion sets for all parties

Description: This algorithm iteratively selects objects with the highest available utility. After all objects have been selected, they are assigned to parties in a round-robin format, with the highest utility object being assigned first. A follow-along example is shown in Appendix C.

$N[i] = n\gamma + 1$), the utility is set as negative. This ensures the object is not picked again.

For each object, such that $N[i] \neq 1$, i.e., the number of times it has been selected is at least one, the invariant in Eq (5) is maintained. The constraint Eq (2) is satisfied by setting the object's utility to -1 after the object is selected n times and ensuring it is never picked again.

Having chosen a list of objects and decided the number of times they will be chosen, the second step is to assign/allocate them to parties. Any allocation that assigns objects uniquely to all parties and ensures that all parties receive the same number of objects (at most k objects are assigned per Eq (3)) would satisfy the correlated equilibrium since all the constraints are satisfied. However, to allocate fairly, we use a round-robin allocation, assigning the highest-valued object one at a time to each party. The fairness guarantees are shown in Appendix D.

In the following theorem, we show that the above algorithm gives a correlated equilibrium.

Theorem 2. (Correlated Equilibrium) *Given the assignment of input lists $L = \{L_1, \ldots, L_n\}$ according to algorithm 1, then following the strategy $\Psi = \{\Psi_1, \ldots, \Psi_n\}$, where $\Psi_i =$ select L_i, is a correlated equilibrium, i.e., for all parties $P_i \in N$, P_i cannot obtain a better utility from selecting any other list than L_i given every other party P_j follows $\Psi_j =$ select L_j.*

To understand the proof of the above theorem, consider the following. Let some object m_i be picked for the last time (i.e. the utility after this pick does not decrease for m_i). For this object, the utility to whichever party it is given to is greater than all other candidate objects. Further, the utility of other objects could decrease, but the utility for this object does not decrease (since we are considering the last instance of selection for the object). Thus, if after the assignment is complete, any party tries to switch away from object m_i to any other object m_j, the utility would be lower. This would then satisfy constraint (5). By Lemma 2 , this would also imply (4), proving that correlated equilibrium is established.

Proof. To prove the theorem statement, we first prove that Eq(5) holds for Algorithm 1. Let $U(m_i, r)$ represent the utility from object m_i in round r of the selection. Let $O(r)$ represent the object chosen by the algorithm in round r. Let $R(m_i)$ represent the last round in which m_i was chosen (For objects never chosen, this value is not defined). Let $N_c(m_i, r)$ represent the expected number of times object m_i has been selected by round r (considering that with probability $1 - \gamma$ the object might be dropped)[This value is γ less than the corresponding N_c value in the algorithm]. At the end of round $r = n \cdot k$, $T(M) = S$ and utility from each $O(r)$ is $U(O(r), R(O(r)))$, i.e., the utility value it had on its last selection. Note that

$$r_1 < r_2 \implies U(m_i, r_1) \geq U(m_i, r_2) \tag{6}$$

That is, the utility of an object can only decrease or remain the same as the rounds progress.

Also, note that

$$U(m_i, r_1) \geq U(m_i, r_2) \tag{7}$$

Rewriting Eq (5) in these new terms, we have

$$\forall r, m_i \in M : \exists j : (L_j \in L,\ O(r) \in L_j, m_i \notin L_j) \implies U(O(r), R(O(r))) \geq \frac{u_{m_i}}{N_c(m_i, n \cdot k) + \gamma}$$

We claim that no matter what the allocation rule is, after the choice of the selection algorithm (Step 2), the resulting allocation always satisfies this equation, as long as an object is not allocated to the same party twice. In other words, we will prove the following stronger statement

$$\forall r \in \{1, \ldots, n \cdot k\},\ m_i \in M : \exists j : m_i \notin L_j \implies U(O(r), R(O(r))) \geq \frac{u_{m_i}}{N_c(m_i, n \cdot k) + \gamma}$$

We prove this by contradiction. Let's assume, to the contrary, that for some m_i and some $O(r) \neq m_i$, such that $\exists j : m_i \notin L_j$,

$$U(O(r), R(O(r))) < \frac{u_{m_i}}{N_c(m_i, n \cdot k) + \gamma} \tag{8}$$

Since $m_i \notin L_j$ for at least some party, and we are assuming that the same object is not allocated to the same party twice, the number of times m_i has been chosen must be less than n implying that $N_c(m_i, n \cdot k) < n\gamma + 1 - \gamma$. This means that line 10 never triggers for m_i.

Case 1: For all $m_i \neq O(n \cdot k)$, the R.H.S. of (8) corresponds to the utility of object m_i in round $n \cdot k$, i.e., $U(m_i, n \cdot k)$. Thus, we are given,

$$U(O(r), R(O(r))) < U(m_i, n \cdot k)$$

For simplicity, let $r = R(O(r))$, i.e., r is the last round in which the object $O(r)$ is selected. Thus,

$$U(O(r), r) < U(m_i, n \cdot k)$$

By line 6,

$$\forall m_i : U(O(r), r) \geq U(m_i, r)$$

Thus,

$$U(m_i, r) \leq U(O(r), r) < U(m_i, n \cdot k)$$

which is a contradiction to (6) since for some $r \leq n \cdot k - 1$, $U(m_i, r_1 = r) < U(m_i, r_2 = n \cdot k)$

Case 2: $m_i = O(n \cdot k)$. Consider round r' such that for all rounds $\{r' + 1, \ldots, n \cdot k\}$, object m_i is chosen. In this case, the R.H.S. of the equation (8) is $< U(m_i, r' + 1)$ since the utility only reduces with further inclusions of m_i. Following steps similar to Case 1 and considering a round r which is the last round for $O(r)$ to be selected,

$$U(O(r), r) < U(m_i, r' + 1)$$

By line 6,

$$\forall m_i : U(O(r), r) \geq U(m_i, r)$$

Thus,

$$U(m_i, r) \leq U(O(r), r) < U(m_i, r' + 1)$$

which is a contradiction to (6) since for some $r \leq r'$, $U(m_i, r_1 = r) < U(m_i, r_2 = r' + 1)$.

This proves that Algorithm 1 follows the constraint of Eq (5), and since by Lemma 2, Eq (5) implies (4), a correlated equilibrium is established.

□

Now that we have established an equilibrium for the input selection, we move towards computing an aggregate for the lists. Since all IL proposers should follow $\Psi_i = \text{Select } L_i$, we say that each party chooses its input list $\text{InpL}_i \sim \Psi_i$ and $\text{InpL}_i = L_i$.

V. Aggregation Protocol

In Section IV, we presented a protocol for IL proposers to build their input list and achieve a correlated equilibrium. The next step in the design is to aggregate these input lists into inclusion lists, which would be used to constrain the builder. In this section, we will present a protocol to aggregate the input lists generated by parties such that the overall inclusion list design outputs a set of candidate lists.

Key design challenges. There are two key challenges we need to address. First, since the lists are being aggregated, the adversary can censor a target transaction by somehow ensuring that the input lists containing the target transaction are not aggregated. This can be achieved, for example, by bribing a party that aggregates.

Second, some of the parties in the protocol may crash, and the protocol needs to tolerate a certain absence of input lists. Suppose the aggregator is required to include all transactions. In that case, even if one IL proposer goes offline or chooses not to broadcast, an honest aggregator cannot create an inclusion list by including all other input lists. Thus, our crash-tolerant protocol would allow the aggregator to exclude a threshold θ of the input lists. However, in doing so, we allow the aggregator to censor complete input lists willfully without any penalties. To mitigate this, we introduce multiple aggregators, each competing to become the aggregator for the block, and proving that aggregating all input lists is the best strategy for all IL proposers. We will show such a design in this section, and in Section VI we will analyze the properties of the design under the threat of censorship. To incentivize the parties to follow the protocol, we describe a reward distribution mechanism that maximizes the cost for the adversary to exclude a transaction from the inclusion list (i.e., maximizes Property 4).

A. Outlining our Solution: AUCIL

The design we will discuss is inspired by a winner-take-all game like an auction, where the bid submitted is the length of the inclusion list. We name this design AUCIL.[3] In AUCIL, all parties work as aggregators. All IL proposers can aggregate the input lists created and broadcasted by IL proposers. After aggregation, each party declares their bids as the size (in terms of the number of input lists included) of the inclusion list they created. A natural way of collecting these bids is through the block proposer of the next block in the blockchain. This block proposer would accept all bids and add the inclusion list with the highest bid.

Algorithm 2 AUCIL outline

Participants: All IL proposers $P_1, P_2, \ldots, P_n$

Step 1: IL proposers broadcast input lists

1: **for all** P_i **do**

2: $P_i \rightarrow_B$ all parties : InpL_i

Step 2: Parties aggregate input lists into an inclusion list and broadcast it

3: **for all** P_j **do**

4: $\text{IncL}_j = \bigcup_{i=1}^{n} \text{InpL}_i$

5: $P_j \rightarrow_B$ all parties : $(\text{IncL}_j, \ell_j = \text{size}(\text{IncL}_j))$

Step 3: Proposer selects the highest bid inclusion list

6: Proposer receives:

$$\{(\text{IncL}_1, \ell_1), (\text{IncL}_2, \ell_2), \ldots, (\text{IncL}_n, \ell_n)\}$$

7: Proposer selects the highest bid where ℓ_i denotes the bid for P_i.

Incentive Structure: IL Proposer of the selected bid receives u_{agg}

As described in Algorithm 2, the outline has three major steps – (i) InpL broadcast, (ii) bid creation and broadcast, and (iii) collection of bids. While the second step is a competition between bidders and is thus incentivized by a reward for winning (u_{agg}), the other two steps are not incentivized. For the first step, what is the incentive for each IL proposer to share its input list? Suppose all parties share the input lists and the proposer picks the inclusion list with the highest bid. In that case, it is strictly dominant for a party not to share its input list since it could aggregate others' input lists with its own and create the largest-size inclusion list, and win the auction. Thus, sharing the input list is not an equilibrium for such a party. This highlights the first problem with the approach.

Moreover, even if all parties declare their input lists, the auctions also suffer from censorship problems, as also highlighted in [8]. The next block proposer could ignore the competitive bids in favor of an adversarial bid, which bribes the proposer to exclude other bids. However, distinct from on-chain auctions, this off-chain auction has the essential property of having a fixed number of bidders, all of which have a positive utility to bid. While some parties could crash - or choose to feign a crash as an adversarial action - at least a threshold $n - \theta$ would broadcast (or have the option to broadcast) their bid, and the block proposer is required to prove that the bid included is greatest amongst $n-\theta$ bids. If the proposer cannot create such a proof, then the block generated would be invalid, and the block proposer would be slashed. This proof would be verified on-chain (by the consensus as a part of the validity condition or by using fraud proofs[25]). We label each bid that has a

[3] AUCIL stands for Auction-based Inclusion List.

valid proof of having a value larger than $n-\theta-1$ other bids is labeled as a *candidate inclusion list*. Thus, by construction θ-Crash Fault Tolerance (Property 1) is θ for our design.

Given this threshold, θ, of parties whose bids are not required to create an inclusion list, the adversary could censor some bids that include the target transaction after the bids have been sent. To incentivize the bids to be as high as possible, despite the censorship, the reward distribution for aggregation (u_{agg}) would take place some blocks after the block for which the inclusion list is being designed. The bids would continue to be collected in the next round(s). The rewards would be distributed as ($u_{agg}/2$) for the highest bid (observed across multiple blocks) and the rest ($u_{agg}/2$) for the winning bid (observed during the block for the creation of the inclusion list). This reward distribution separates incentives for two targets - place bid as soon as possible (such that the protocol remains live) and place bid as high as possible (to ensure that even if the bid is censored in the slot, it still wins a partial reward in future slots). This distribution of rewards is arbitrarily chosen, and other reward distributions could exist.

To solve the first problem, let's consider the other extreme situation. If no party shares its input list, it is dominant for most IL proposers to share it. This is because it would not be able to create a winning bid; if the tie is broken randomly, there is a $1/n$ probability of winning. Releasing its input list thus allows the winning bid to include this input list and thus give a transaction fee reward as described in Section IV. This proves that the Nash equilibrium for the above game is mixed and probabilistically lies somewhere between sharing the input list and not sharing the input list. The exact equilibrium would depend on the probability of the IL proposer winning the game by not sharing the input list.

To discourage proposers from hiding their input lists, we introduce a simple form of randomness into the bidding process. Each IL proposer generates a bias b value using a verifiable random function (VRF) [26]. This bias is drawn uniformly from the range $[0, b_{max}]$ and is added to the IL proposer's bid (the bid is the length of the inclusion list). A small bias $(0 \leq b < b_{max}-1)$ means the proposer's adjusted bid is likely to be low compared to others. In that case, the proposer has little chance of winning the auction outright. Instead, the best strategy is to broadcast its input list so that other proposers can include those transactions, allowing it to earn inclusion fees from them. A large bias $b \geq b_{max}-1$ increases the adjusted bid and the chance of winning, so the proposer might consider withholding its list to keep an advantage. However, because biases are unpredictable, the expected outcome is that only proposers with the highest biases will contemplate withholding; the rest will find it optimal to publish their lists. This mechanism makes sharing the default behaviour for most IL proposers, and for those with high bias values, AUCIL encourages them not to broadcast their input lists.

In order to separate the notion of broadcasting the input list and others adding it to their bid, we introduce a metadata value associated with the input list, which we label as flag $\mathbf{F}$. If $\mathbf{F}$ is set to 1, then the input list is available to all parties and increases the bid size by 1 when included in the bid. If $\mathbf{F}$ is set to 0, then even if the input list is included in a bid, it would not increase the bid size. In return, only those input lists that have $\mathbf{F}$ set to 1 would receive their share of transaction inclusion rewards as described in the previous section. In a rational world, setting the flag to 1 is equivalent to broadcasting, and 0 is the same as withholding its input list. In further sections, an input list is considered available if it has been broadcast and $\mathbf{F}$ is set to 1.

B. Analysis

Algorithm 3 AUCIL for log position pos

Participants: All IL proposers $P_1, P_2, \ldots, P_n$

Step 0: IL proposers generate their auction bias

for all P_i **do**

$P_i : (b_i, \pi_i) \leftarrow \text{VRF}_{sk_i}(\text{pos})$ scaled to a range of $[0, b_{max}]$

▷ Generate the random bias (uniform dist.) between 0 and b_{max} to add to the bid

Step 1: IL proposers broadcast input lists

for all P_i **do**

$P_i : \mathbf{F}_i \leftarrow \text{checkAvailable}(b_i)$

▷ Check whether P_i should make its input list available

$P_i \rightarrow_B$ all parties : $\text{InpL}_i, \mathbf{F}_i$

▷ Parties broadcast their InpL while choosing to make it available or not

Step 2: Aggregate inputs into inclusion list and broadcast

for all P_j **do**

$\text{IncL}_j, y_j = \bigcup_{i=1}^{n} \text{InpL}_i, \sum_{i=1}^{n} \mathbf{F}_i$

▷ If some IL Proposer's value is missing take $\{\}, 0$ as InpL and $\mathbf{F}$

$P_j \rightarrow_B$ all parties : $(\text{IncL}_j, \ell_j = (y_j + b_j))$

▷ Parties declare their bid with added bias

Step 3: Block proposer selects highest bid inclusion list

Proposer receives:
$\{(\text{IncL}_1, \ell_1), (\text{IncL}_2, \ell_2), \ldots, (\text{IncL}_n, \ell_n)\}$

Proposer selects the highest bid and adds it to the block (IncL, ℓ).

Proposer verifies π_i, includes proof for the bid being greater than $n-\theta$ bids.
- Block is considered verified if the proof is valid.

Description: Algorithm for aggregating input lists into an inclusion list. The basis of AUCIL is an auction design, where all parties try to compete with the largest size inclusion list. Even after next block is created, the bids are still collected in case a bid higher than the winning bid is found.

Incentive Structure: IL Proposer of selected bid receives $0.5u_{agg}$. IL Proposer of highest bid across multiple slots receives $0.5u_{agg}$.

In this section, we will analyze the utility of IL proposers in the absence of adversarial censorship. In such a case, the proposer will select the highest bid amongst all the bids. The first thing to observe is that not all parties may have an incentive to broadcast, so let's assume that η (At equilibrium, $\gamma = \eta/n$) InpLs are publicly available. Consider that the IL proposer includes all the input lists it receives and its own.

The IL proposer has two options: make its InpL available (**F** = 1) or withhold it (**F** = 0). If the IL proposer makes its input list available, then the following lemma holds.

Lemma 3. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b \leq 1$. If P chooses to make its InpL available, its expected utility is $u_{il} + \mathsf{negl}(n)$.*

Proof. The bid generated by P is calculated as $\eta + 1 + b$. All other IL proposers also receive the input list of P. There exist two classes of other IL proposers: 1) those that made their input list available (there exist η such IL proposers) and those that did not $(n - \eta - 1)$. Let b_i represent the bias generated through VRF for IL proposer P_i.

The bid for each IL proposer who did not make its input list available is $\eta + 2 + b_i$ ($\eta + 1$ from publicly available lists, and 1 private). Similarly, the bid for each IL proposer who chose to make its input list available is $\eta + 1 + b_i$ (Its list is included in the publicly available lists).

The probability that P wins the auction is the same as the bid generated by P being greater than all other bids.

$$\begin{aligned}
\mathbb{P}(P \text{ wins}) &= \prod_{i=0}^{\eta} \mathbb{P}(\eta + 1 + b \geq \eta + 1 + b_i) \\
&\quad \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(\eta + 1 + b \geq \eta + 2 + b_i) \\
&= \prod_{i=0}^{\eta} \mathbb{P}(b \geq b_i) \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(b \geq 1 + b_i) \\
&= \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \prod_{i=0}^{n-\eta-1} \{\mathbb{P}(b \leq 1)\mathbb{P}(b \geq 1 + b_i | b \leq 1) \\
&\qquad + \mathbb{P}(b > 1)\mathbb{P}(b \geq 1 + b_i | b > 1)\} \\
&= \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \prod_{i=0}^{n-\eta-1} (\mathbb{P}(b > 1)\mathbb{P}(b - 1 \geq b_i | b > 1)) \quad (9)
\end{aligned}$$

If $b \leq 1$, the probability of winning the auction is 0, unless all parties $(\eta = n - 1)$ make their input lists available.

The utility in this case is given by u_{il}. If all parties make their list available, then the utility would increase by $(\frac{b}{b_{max}})^n \cdot u_{agg}$, which is negligible in n. □

Lemma 4. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b > 1$. If the IL proposer chooses to make its InpL available, then its expected utility is $u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg}$*

Proof. From (9), the probability of winning the auction is

$$\mathbb{P}(P \text{ wins}) = \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1}$$

If P wins the auction, then it will receive both input list inclusion and aggregation rewards, while if it loses the auction, then the reward earned is only the input list inclusion reward.

$$\begin{aligned}
u_P &= \mathbb{P}(P \text{ wins})(u_{agg} + u_{il}) + (1 - \mathbb{P}(P \text{ wins}))\, u_{il} \\
&= u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg} \quad (10)
\end{aligned}$$

□

Lemma 5. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b < b_{max} - 1$. If the IL proposer chooses not to make its InpL available, then its expected utility is $\left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il})$*

Proof. The bid generated by P is $\eta + 1 + b$. All other IL proposers can not extend their bid with the input list of P. There exist two classes of other IL proposers - those that made their input lists available (there exist η such IL proposers) and those that did not $(n-\eta-1)$. Let b_i represent the bias generated through VRF for IL proposer P_i.

The bid for each IL proposer who made their input list available is $\eta + b_i$. Similarly, the bid for each IL proposer who did not is $\eta + 1 + b_i$.

The probability that P wins the auction is the same as the bid generated by P being greater than all other bids.

$$\begin{aligned}
\mathbb{P}(P \text{ wins}) &= \prod_{i=0}^{\eta} \mathbb{P}(\eta + 1 + b \geq \eta + b_i) \\
&\quad \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(\eta + 1 + b \geq \eta + 1 + b_i) \\
&= \prod_{i=0}^{\eta} \mathbb{P}(b + 1 \geq b_i) \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(b \geq b_i)
\end{aligned}$$

$$\begin{aligned}
\mathbb{P}(b + 1 \geq b_i) &= \mathbb{P}(b \leq b_{max} - 1)\mathbb{P}(b + 1 \geq b_i | b \leq b_{max} - 1) \\
&\quad + \mathbb{P}(b > b_{max} - 1)\mathbb{P}(b + 1 \geq b_i | b > b_{max} - 1)
\end{aligned}$$

If $b > b_{max} - 1$, then $b + 1$ is always $> b_i$. Thus,

$$\begin{aligned}
\mathbb{P}(b + 1 \geq b_i) &= \mathbb{P}(b \leq b_{max} - 1)\mathbb{P}(b + 1 \geq b_i | b \leq b_{max} - 1) \\
&\quad + \mathbb{P}(b > b_{max} - 1) \quad (11)
\end{aligned}$$

Since $b < b_{max} - 1$,

$$\begin{aligned}
\mathbb{P}(b + 1 \geq b_i) &= \mathbb{P}(b + 1 \geq b_i | b \leq b_{max} - 1) \\
&= \left(\frac{b+1}{b_{max}}\right)
\end{aligned}$$

Similarly,

$$\mathbb{P}(b \geq b_i) = \frac{b}{b_{max}}$$

Thus, given $b \leq b_{max} - 1$, the probability of winning the auction is

$$\mathbb{P}(P \text{ wins}) = \left(\frac{b}{b_{max}}\right)^{n-\eta-1} \cdot \left(\frac{b+1}{b_{max}}\right)^{\eta}$$

If P wins the auction, then it will receive both input list inclusion and aggregation rewards, while if it loses the auction, then no reward is earned since the input list was not available to others.

$$\begin{aligned} u_P &= \mathbb{P}(P \text{ wins})(u_{agg} + u_{il}) \\ &= \left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il}) \end{aligned}$$

□

Lemma 6. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b \geq b_{max} - 1$. If the IL proposer chooses not to make its InpL available, then its expected utility is $\left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il})$*

Proof. Equation (11) still holds for the analysis of this Lemma. Given $b > b_{max} - 1$, we get

$$\mathbb{P}(b + 1 \geq b_i) = 1$$

Thus,

$$\begin{aligned} \mathbb{P}(P \text{ wins}) &= \left(\frac{b}{b_{max}}\right)^{n-\eta-1} \\ u_P &= \mathbb{P}(P \text{ wins})(u_{agg} + u_{il}) \\ &= \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il}) \end{aligned}$$

□

Theorem 3. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), $b_{max} > 2$ and the total number of IL proposers is n.*

1) *Given P generates a bias $b \leq 1$. Except with a negligible probability, making its input list available is the dominant action for P.*
2) *Given P generates a bias $1 < b < b_{max} - 1$. Except with a negligible probability, making its input list available is the dominant action for P. Consequently (from parts 1 and 2 of the theorem), at least $\frac{b_{max}-1}{b_{max}}$ of the parties broadcast their input list in expectation.*
3) *Given P generates a bias $b \geq b_{max} - 1$. The Nash equilibrium for the game would be a mixed strategy, i.e., make its input list available with some probability and withhold with some probability.*

Proof. 1) From Lemma 3, the utility from making its input list available is u_{il}. From Lemma 5, the utility from making its input list available is

$$\left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il})$$

Since $b_{max} > 2$ and $b \leq 1$, this utility tends to 0. Thus, the utility from making its input list available (u_{il}) is greater than that of not making its input list available (0). Thus, all such parties would make their input list available.

2) From Lemma 4, the utility from making its input list available is

$$u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg}$$

From Lemma 5, the utility of not making its input list available is

$$\left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il})$$

Consider the difference between the utilities.

$$\begin{aligned} &u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg} \\ &\quad - \left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il}) \\ &\geq u_{il} \left(1 - \left(\frac{b+1}{b_{max}}\right)^{n-1}\right) \\ &\quad + u_{agg} \left(\left(\frac{b}{b_{max}}\right)^{n-1} - \left(\frac{b+1}{b_{max}}\right)^{n-1}\right) \\ &\geq u_{il} \left(1 - \left(\frac{b+1}{b_{max}}\right)^{n-1}\right) - u_{agg} \left(\frac{b+1}{b_{max}}\right)^{n-1} \\ &= u_{il} \left(1 - \left(1 + \frac{u_{agg}}{u_{il}}\right) \left(\frac{b+1}{b_{max}}\right)^{n-1}\right) \end{aligned}$$

To ensure this is ≥ 0, we require

$$\left(\frac{b+1}{b_{max}}\right)^{n-1} < \frac{u_{il}}{u_{il} + u_{agg}}$$

The probability for this is given by

$$\mathbb{P} = \frac{b_{max}\left(\frac{u_{il}}{u_{il}+u_{agg}}\right)^{1/n-1} - 1}{b_{max} - 1}$$

which is approximately 1 if n is large. Thus, parties with $b < b_{max} - 1$ are incentivized to make their input list available. This occurs with a probability of $\frac{b_{max}-1}{b_{max}}$, which implies that in expectation, at least $\frac{b_{max}-1}{b_{max}} \cdot n$ parties would make their input lists available.

3) From Lemma 4, we know that the utility from making its input list available is

$$u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg}$$

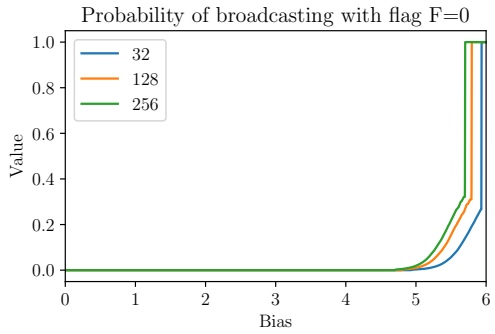


Fig. 3: An example for the probability of withholding input list with the bias generated for IL Proposers. Example parameters: $n = 36, b_{max} = 6, u_{il} \simeq 32$. The input list reward (u_{il}) comes from running algorithm 1 on transactions chosen from a Beta distribution, with parameters $(1, 5)$ and scaled by a factor of 30. The individual lines represent the additional reward (u_{agg}) from winning the auction from Algorithm 3.

From Lemma 6, the utility of not making its input list available is

$$\left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg} + u_{il})$$

From Part 2, we know that η is in expectation more than $\frac{b_{max}-1}{b_{max}}n$. Substituting in Lemma 6, we get

$$\left(\frac{b}{b_{max}}\right)^{\frac{n}{b_{max}}-1}(u_{agg} + u_{il})$$

As b approaches b_{max}, the difference in utility is

$$u_{il} + 0 - \left(1 - \left(\frac{n}{b_{max}} - 1\right)\left(\frac{b_{max} - b}{b_{max}}\right)\right)(u_{agg} + u_{il})$$
$$= \left(\left(\frac{n}{b_{max}} - 1\right)\left(\frac{b_{max} - b}{b_{max}}\right)\right)(u_{agg} + u_{il}) - u_{agg}$$

which is negative since the first term approaches 0.

Thus, at least some parties are incentivized not to make their input list available, and a mixed Nash equilibrium follows.

□

The chart in Figure 3 shows the (simulated) probability with which the IL Proposer would withhold its input list at equilibrium versus the bias drawn through the VRF (not make it available to other IL Proposers) [4].

Note that we establish a Mixed Nash Equilibrium here. A coordination of the game is not feasible since all parties are competing for a single reward and thus there is no opportunity for coordination.

[4]Simulation code repo: https://anonymous.4open.science/r/Implementation-4CF6/

VI. Censorship Resistance

Combining the protocols in Sections IV and V, we get our complete end-to-end design for an inclusion list creation as follows:

- All IL proposers create an input list (InpL) each from a set of transactions as determined by Algorithm 1. ($\text{InpL}_i \sim \Psi_i$, where Ψ_i is selecting list L_i from Algorithm 1)
- All IL proposers create and broadcast inclusion lists trying to maximize the number of available input lists as described in Algorithm 3.

The above-described scheme works well in the absence of any external incentives. However, an adversary trying to censor a transaction can manipulate the incentives of the IL proposers with the following actions (for the rest of the paper, we would refer to the transaction the adversary wants to exclude as a *target* transaction m_t, and the utility IL proposers get from including the transaction is u_{m_t}, which is suggested to a set of parties N_t, where $|N_t| = n_t$):

1) Add transactions in the mempool to naturally manipulate the number of parties that are suggested to include m_t.
2) Bribe IL proposers to exclude the target transaction from the suggested input list.
3) Bribe IL proposers to exclude all input lists that contain the target transaction from the generated bid.
4) Exclude a bid containing at least one input list with the target transaction by crashing the party creating the bid.

In the following subsections, we will look at each of these actions. Initially, assume that Actions 1, 2 affect the input list building scheme (their values determine β_1-Input Censorship Resistance) and Actions 3, 4 affect the aggregation scheme (their values determine β_2-Aggregation Censorship Resistance). The two pairs of actions are exclusive and do not affect each other; later, after looking at the results, we will defend why this assumption is justified. As mentioned in Theorem 1, the β_3-Blockchain Censorship Resistance of the unconditional inclusion list design remains R.

A. Censorship Resistance for Input-List building

We would first analyze Action 2 and then look at Action 1 and the interplay between the two. The first thing to note is that since some parties choose not to make their input lists available, this reduces the number of expected parties from the number of actual parties suggested to include in the target transaction; the adversary does not know which parties would not broadcast. Thus, when considering the parties to bribe, the adversary would consider all parties in the set $|N_t|$ as opposed to the expected number of parties that broadcast ($N[s]$ in algorithm 1).

However, we need to consider the expected number of broadcast input lists when considering the utility that the IL proposers receive from transactions. Let n_t be the number of input lists suggested to include the transaction ($n_t = |N_t|$), $\overline{n_t}$ as the expected number of broadcast input lists that contain the target transaction ($\overline{n_t} = n_t \cdot \gamma$, where γ is the probability that an IL proposer broadcasts), $\widetilde{n_t} = 1 + \gamma(n_t - 1)$ represents

the denominator as specified in Eq (5) and $\widehat{n}_t$ represents the actual number of input lists that broadcast the transaction.

The expected utility for each IL proposer from including the target transaction is $\frac{u_{m_t}}{\widehat{n}_t}$ and excluding it and adding another transaction m_s gives some utility $\frac{u_{m_s}}{\widetilde{n}_s+\gamma}$. If m_s is available in the global mempool, then $\frac{u_{m_s}}{\widetilde{n}_s+\gamma} \leq \frac{u_{m_t}}{\widehat{n}_t}$, otherwise the cost u_{m_s} would be an additional cost to the adversary (and $n_s = 0$). Also, consider the utility for including the input list as suggested, but without the target transaction to be u_{il}^-, which is the sum of the utility that each transaction in the list provides. The strategies that each IL proposer follows, given it receives some bribe from the adversary, are shown in Lemma 7.

Lemma 7. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary, and a set N_t ($|N_t| = n_t$) of IL proposers who are suggested to include the target transaction m_t (i.e., $L_i = m_t \cup L_i^-$). Consider a bribe (br) from the adversary to IL proposers in the set N_t to censor the target transaction (action 2) and replace it with a replacement transaction m_s.*

- *i) If $br \leq \frac{u_{m_t}}{\widehat{n}_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$, the IL proposer would reject the bribe and propose the suggested input list.*
- *ii) If $\frac{u_{m_t}}{\widehat{n}_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma} < br < u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$, the IL proposers would reject the bribe with some non-zero probability.*
- *iii) If $br \geq u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$, the IL proposer would always accept the bribe and ignore the transaction.*

Proof. The utility received by following the equilibrium strategy (or rejecting the bribe) is $\mathbb{U}_r = u_{il}^- + \frac{u_{m_t}}{\widehat{n}_t}$, where $u_{il}^- = \mathbb{U}(L_i^-, \_)$ is the utility received from selecting list L_i^- and $\widehat{n}_t = \widetilde{n}_t$, if all IL proposers include the transactions as suggested. If some IL proposers accept the bribe and exclude the transaction from their input list, then $\widehat{n}_t < \widetilde{n}_t$. The utility received for accepting the bribe is $\mathbb{U}_a = u_{il}^- + \frac{u_{m_s}}{\widetilde{n}_s+\gamma} + br$. Thus, $\mathbb{U}_r - \mathbb{U}_a = \frac{u_{m_t}}{\widehat{n}_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma} - br$. Since $1 \leq \widehat{n}_t \leq n_t$, we have that $\frac{u_{m_t}}{n_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma} - br \leq \mathbb{U}_r - \mathbb{U}_a \leq u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma} - br$. We now prove each part of the lemma separately.

i) If $br \leq \frac{u_{m_t}}{\widehat{n}_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$, $\mathbb{U}_r - \mathbb{U}_a \geq 0$. Since the utility gained by the IL proposer from accepting the bribe is less than rejecting the bribe, the IL proposer would always reject the bribe.

ii) If bribe $\frac{u_{m_t}}{\widehat{n}_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma} < br < u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$, then the bribe is higher than the individual utility gained by the IL proposer if all other IL proposers choose to include the transaction (i.e., reject the bribe, $\widehat{n}_t = \widetilde{n}_t$). However, if all IL proposers choose to accept the bribe, then the utility received from rejecting the bribe and being the only IL proposer to include the target transaction is u_{m_t}. Thus, it is not rational for all IL proposers to accept or reject the bribe. A mixed Nash equilibrium would exist since both pure strategies are not an equilibrium, implying a non-zero probability of rejecting the bribe.

iii) If $br \geq u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$, $\mathbb{U}_r - \mathbb{U}_a \leq 0$. Since the bribe available is greater than the utility that the IL proposer could receive, even if any other party does not include the transaction, the IL proposer always chooses to accept the bribe.

□

Lemma 8. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in the absence of an adversary, the adversary must pay at least $u_{m_t} \cdot (n_t - 1)$ to ensure (with probability = 1) that the target transaction does not appear in any input lists.*

Proof. From Lemma 7, if the adversary bribes the IL proposers in the set N_t in such a way that the IL proposers always censor the transaction, then the bribe has to be at least $u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma}$. Now, if m_s is a public transaction, i.e., available in mempool, then $u_{m_t} - \frac{u_{m_s}}{\widetilde{n}_s+\gamma} \geq u_{m_t} - \frac{u_{m_t}}{\widehat{n}_t} \geq u_{m_t} - \frac{u_{m_t}}{n_t}$. Since at least n_t parties receive this bribe, the total cost to the adversary is at least $u_{m_t} \cdot (n_t - 1)$ to bribe all the IL proposers in set N_t. However, if the transaction is private, then the cost to the adversary is $u_{m_s} + br = u_{m_t}$. This amount must be given to all n_t parties, and thus the total cost would be $u_{m_t} n_t > u_{m_t} \cdot (n_t - 1)$ □

If the adversary chooses a strategy to bribe IL proposers to exclude the target transaction (Action 2), it must pay a cost of at least $u_{m_t} \cdot (n_t - 1)$ as shown in Lemma 8.

Consider Action 1. The adversary may choose to add spam transactions in such a way that the target transaction does not appear in any input list, or it may choose to reduce the number of n_t, and then follow up with Action 2.

Before we proceed with the analysis of this action, we need to observe the dependence of n_t on the fee paid by the transaction u_{m_t} and the fee paid by other transactions. Let $T(M)$ represent all the transactions the input list-building mechanism chooses. From Eq (5) for a correlated equilibrium, we know that,

$$\forall m_i \in T(M) : \frac{u_{m_i}}{\widetilde{n}_i} \geq \frac{u_{m_t}}{\widetilde{n}_t + \gamma}$$

Since $\widetilde{n}_i > 0$ for all $m_i \in T(M)$, we have

$$\forall m_i \in T(M) : u_{m_i}(\widetilde{n}_t + \gamma) \geq u_{m_t}\widetilde{n}_i$$

Taking the sum over all $m_i \in T(M)$, and noting that $\frac{u_{m_t}}{\widehat{n}_t} > \frac{u_{m_t}}{\widetilde{n}_t+\gamma}$

$$\sum_{i \in T(M)} u_{m_i}(\widetilde{n}_t + \gamma) > u_{m_t} \sum_{i \in T(M)} \widetilde{n}_i$$

$$\sum_{i \in T(M)} u_{m_i}(\widetilde{n}_t + \gamma) > u_{m_t} \cdot n \cdot k \cdot \gamma$$

Let σ represent $\sum_{i \in T(M)} u_{m_i}$. Also, $\widetilde{n}_t = 1 + \gamma(n_t - 1)$. Thus,

$$1 + \gamma n_t > \gamma \cdot n \cdot k \frac{u_{m_t}}{\sigma}$$

$$n_t > n \cdot k \frac{u_{m_t}}{\sigma} - \frac{1}{\gamma} \tag{12}$$

Using Action 1, the adversary can censor the target transaction by adding transactions to the mempool. If more transactions are to be chosen, then the algorithm would suggest the transaction to fewer parties. If we view these extra transactions as a sequential addition of transactions to the mempool, we will reach a point where the target transaction is suggested to only one IL proposer. In other words, the Lemma 9 compares using Action 1 to censor completely and a hybrid of Action 1 and Action 2 to first reduce the number of IL proposers and then bribe the rest. The lemma prooves that the latter is a dominant action.

Lemma 9. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary, if the adversary reduces the number of times the transaction is suggested to $n_t = 1$ (Action 1), then after reaching this state, the cost incurred to an adversary in bribing the IL proposer (Action 2) is lower than adding further transactions to reduce the number of parties that are suggested to include the transaction (Action 1).*

Proof. To displace the target transaction, the adversary would have to displace all selections of the transactions after the target transaction is chosen for the first time in Algorithm 1 since each of the transactions that are selected after the target transaction gave lower utility than the first selection of the target transaction. Thus, in the worst case for $n_t = 1$, the target transaction is the last transaction chosen in Step 1 of Algorithm 1 such that there are no other transactions to displace. To displace the last chosen object (which in this case is m_t), the adversary would need to add a transaction m_a such that $\frac{u_{m_a}}{\widetilde{n_a}+\gamma} \geq \frac{u_{m_t}}{\widetilde{n_t}}$. Since $n_t = 1$, $\widetilde{n_t} = 1$, this implies $u_{m_a} \geq u_{m_t}$. Thus, Action 1 costs at least u_{m_t}.

If the adversary instead chooses to bribe the IL proposer, then from Lemma 7 the minimum bribe it would have to pay the IL proposers is $u_{m_t} - u_{m_s} \leq u_{m_t} \leq u_{m_a}$, where u_{m_s} represents the utility of some replacement transaction that the IL proposer could include. Thus, Action 2 costs at most u_{m_t}.

Thus, bribing the IL proposer dominates the action of displacing the transaction through added adversarial transactions when the target transaction appears only once. □

Lemma 10. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary, if the adversary adds adversarial transactions (Action 1) with a total fee of u_{m_a} and pays a bribe br_1 to all the IL proposers which are suggested to add the target transaction (Action 2). If $u_{m_t} \leq \frac{\sigma}{\sqrt{nk}}$, then the cost incurred by the adversary is greater than $u_{m_t}(n \cdot k\frac{u_{m_t}}{\sigma} - 1 - \gamma)$*

Proof. If the adversary does not add any adversarial transaction, then the cost to the adversary by only bribing is given from Lemma 8 and Eq.(12). This cost is represented by

$$C = u_{m_t}(n \cdot k\frac{u_{m_t}}{\sigma} - 1 - \frac{1}{\gamma})$$

Since no additional transactions are added, the cost to the adversary is only the bribe. Thus, $br_1 + u_{m_a} \geq C$ in this case.

If the adversary adds some transactions to reduce the number of times the target transaction appears in algorithm 1, and then censors the rest (hybrid of action 1 and action 2) then the cost to the adversary is given by the fees paid plus the bribe cost to remove the target transaction from the reduced number of input lists.

$$C_1 = u_{m_a} + br_1$$

From Lemma 8 and Eq.(12), we have

$$\begin{aligned} br_1 &\geq u_{m_t}(n_t{}' - 1) \\ n_t{}' &\geq n \cdot k \cdot \frac{u_{m_t}}{\sigma - \sum(u_l) + u_{m_a}} - \frac{1}{\gamma} \\ &\geq n \cdot k \cdot \frac{u_{m_t}}{\sigma + u_{m_a}} - \frac{1}{\gamma} \end{aligned}$$

, where $\sum(u_l)$ represents the sum of any transactions removed. Thus,

$$C_1 \geq u_{m_a} + u_{m_t}(n \cdot k \cdot \frac{u_{m_t}}{\sigma + u_{m_a}} - 1 - \frac{1}{\gamma})$$

The difference in this cost to the adversary and the minimum cost we claim is given by

$$\begin{aligned} C_1 - C &\geq u_{m_a} - u_{m_t}(nku_{m_t}) \cdot \left(\frac{1}{\sigma} - \frac{1}{\sigma + u_{m_a}}\right) \\ &\geq u_{m_a} - nku_{m_t}^2 \left(\frac{u_{m_a}}{\sigma(\sigma + u_{m_a})}\right) \end{aligned}$$

If $u_{m_t} \leq \frac{\sigma}{\sqrt{nk}}$, then $u_{m_t}^2 \leq \frac{\sigma^2}{nk}$. Thus,

$$C_1 - C \geq u_{m_a} - \sigma^2 \left(\frac{u_{m_a}}{\sigma(\sigma + u_{m_a})}\right) \geq 0$$

Thus, $C_1 \geq C$ and the minimum cost that the adversary must pay is $u_{m_t}(n \cdot k\frac{u_{m_t}}{\sigma} - 1 - \frac{1}{\gamma})$.

□

Theorem 4. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary, AUCIL has β_1-Input Censorship Resistance value of $u_{m_t}(n \cdot k\frac{u_{m_t}}{\sigma} - 1 - \frac{1}{\gamma})$*

The proof follows from Lemma 10 where the adversary uses a hybrid of Actions 1 and 2 to exclude a transaction that exists in the input lists at equilibrium.

B. Censorship Resistance for Aggregation Step

The next set of actions that an attacker can take to censor a transaction is to target the aggregation algorithm and ensure the aggregated list does not contain the target transaction in any of the input lists. We first give the adversary the advantage that any input list that is broadcast but not made available (i.e., **F** is set to 0), then the cost to remove such a list from the bid is 0. Let $\widehat{n_t}$ be the number of input lists made available with the target transaction, m_t. This could be less than the number of parties suggested to include the target transaction in the input list due to the effect of Actions 1 and 2 by the adversary. To censor the transaction in Algorithm 3, the adversary must ensure that the highest bid selected by the block proposer

excludes all the input lists containing the target transaction. Let's parameterize the blockchain's requirement to include a proof that the bid is greater than $n-\theta-1$ other bids. We give the adversary absolute control over which bids are dropped due to threshold requirements. In other words, the adversary must ensure that at least one of the bids within the top $\theta+1$ bids does not contain any input lists with the target transaction.

Verifiable Random Functions (VRFs) guarantee the privacy of the bias generated by each IL proposer. Thus, we assume that the adversary would not know the bias generated by the IL proposer. This does not prevent the adversary from bribing the IL proposer to get this information. The first thing to note here is that the adversary can infer from Theorem 3 that if a proposer has not made its input list available, then the bias for such a party must be larger than one less than the maximum bias, i.e., $b > b_{max} - 1$. However, it cannot tell that if an IL proposer made its input list available that the bias for the party is $\leq b_{max} - 1$, since an IL proposer may still choose to make its input list available even if the bias for it is $> b_{max} - 1$ (since it is a mixed Nash equilibrium). Next, we also note that Action 4 cannot censor the target transaction since, under honest conditions, each bid would contain all input lists, including those containing the target transaction; excluding θ of them would not censor the target. Thus, we would look at Action 4 as a sub-routine within Action 3.

Lemma 11. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary, given $\widehat{n_t}$ is the number of input lists that contain the target transaction, m_t. Given an IL proposer P which generates a bias $b \geq b_{max} - \widehat{n_t}$. If $br < u_{agg}/2$, then P would reject the bribe with some non-zero probability. If $br \geq u_{agg}/2$, then P would accept the bribe.*

Proof. In order to remove the target transaction m_t, the adversary requires the IL proposers to exclude all $\widehat{n_t}$ input lists that contain it. This would reduce the bid the IL proposer can send by $\widehat{n_t}$. Let's consider the case where $br \leq u_{agg}/2$. At equilibrium, let the probability with which the bribe is rejected be p.Consider the case of $p = 0$. If all IL proposers decide to accept the bribe, then if P rejects the bribe, the adversary would drop its bid amongst the θ crash faults tolerated. However, in subsequent blocks, this bid would be included with proof that the bid was higher than the winning bid. (There is no incentive for the adversary to censor it in later rounds). This yields a utility of $u_{agg}/2$ for P. Thus, the incentive from rejecting the bribe is at least $u_{agg}/2$. If the bribe is less than $u_{agg}/2$, then all parties would have an incentive to reject the bribe with some non-zero probability.

For the case that $br \geq u_{agg}/2$, if the IL proposer rejects the bribe, the maximum utility it can get is by winning the highest bid reward of $u_{agg}/2$ (while it would also have to pay a fee to get its bid included in the later round). Thus, it would always accept the bribe if $br \geq u_{agg}/2$. □

As a consequence of Lemma 11, if the bribe offered is $< u_{agg}/2$, then the number of bids submitted by IL proposers that do not accept bribes is (with some probability) greater than θ. Thus, the bribery fails with some probability.

Lemma 12. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary, given $\widehat{n_t}$ is the number of input lists that contain the target transaction, and η is the total number of input lists available. If an adversary wants to censor the target transaction (with 100% probability) by bribing during the aggregation phase, then the total cost incurred by the adversary is at least $(n-\theta) \cdot u_{agg}/2$.*

Proof. From Lemma 11, the minimum bribe required to bribe an IL proposer who draws a bias $> b_{max} - \widehat{n_t}$ would be $u_{agg}/2$. However, the adversary does not know which parties draw such a bias. The adversary can identify that each IL proposer that did not broadcast the input list would have (with a high probability) a bias greater than $b_{max} - 1$; however, this does not give any information about an IL proposer drawing a bias less than $b_{max} - n_t$. This implies that the adversary would have to bribe all but θ IL proposers regardless of the value of bias drawn. Thus, the total bribe the adversary has to pay is $(n-\theta) \cdot u_{agg}/2$. □

From Lemma 12, we observe that any bribery for parties in the aggregation phase (hybrids of Actions 3 and 4 is independent of the number of times the target transaction appears in the input lists. Thus, a reduction of the number of times the target transaction appears in input lists by Actions 1 and 2 has no reduction in the cost to an adversary when it takes Actions 3 and 4. Thus, the two sets of actions are independent.

Theorem 5. *Given all IL proposers follow strategy Ψ as described in Algorithm 1 in absence of adversary such that n_t is the expected number of IL proposers that would include tx_τ, then AUCIL has a β_2-Aggregation Censorship Resistance of $(n-\theta) \cdot u_{agg}/2$.*

The proof follows from Lemma 12.

C. Overall Censorship Resistance

Consider the following parameterization of the protocol. $b_{max} = \sqrt{n}, u_{agg} = \sqrt{n} \cdot u_{il}$. The sum of rewards for the input list across all parties is the same as the sum of fees paid by all transactions in the inclusion list, i.e., $\sum_{j \in N} u_{il}^j = \sum_{i \in T(M)} f_i = \sigma$. And thus, the expected u_{il} for each party is σ/n. Also, by Theorem 7 in Appendix D, the reward distribution is roughly the same across all parties, and thus, we can say that the reward for each party does not deviate from the expected reward by much. For this protocol, we claim the following:

Theorem 6. *Given n parties running the protocol, M represents the transactions available to all parties in the mempool, f_j represents the fee paid by a transaction $m_j \in M$, $\theta - 1$ represent the number of crash faults tolerated, and T represent the union of all lists L_j when Algorithm 1 is run on M. Consider $B = \{br_1, \ldots, br_{|T(M)|}\}$ such that $br_j = \min((n \cdot k\frac{f_j}{\sigma} - 1 - \frac{1}{\gamma})f_j, \frac{(n-\theta)}{\sqrt{n}}\sigma, R)$. The protocol satisfies (B, θ, T)-censorship resistance.*

Proof. Consider an adversary with a bribery budget of br. From Lemma 10, we know that if the adversary attempts to censor a transaction m_j from the input list, the least amount of bribe it must pay is

$$(n \cdot k\frac{f_j}{\sigma} - 1 - \frac{1}{\gamma})f_j$$

Note that here, $k \cdot f_j < \sigma$ and $f_j \leq \frac{\sigma}{\sqrt{nk}}$. If the adversary attempts to censor the transaction in the aggregation phase, then the total cost, as governed by Lemma 12 is

$$\begin{aligned}(n-\theta)u_{agg} = (n-\theta)\sqrt{n}u_{il} &\geq (n-\theta)\sqrt{n}\sigma/n \\ &\geq \frac{(n-\theta)}{\sqrt{n}}\sigma\end{aligned}$$

Now, let $br_i = \max((n \cdot k\frac{f_j}{\sigma} - 1 - \frac{1}{\gamma})f_j, \frac{(n-\theta)}{\sqrt{n}}\sigma, R)$.

If $br < br_i$, then at least θ parties will output the inclusion list, which includes the transaction m_i, implying that the proposer will select the inclusion list with transaction m_i at least once. Thus, the protocol is (B, θ, T)–censorship resistant. □

Figure 4 shows the bribes required when simulated over chosen mempools. The simulation matches the results in the theorem.

VII. Discussion

On unconditional inclusion lists. An essential property for inclusion lists we consider is that all transactions in the list must be included in the next block. This limits the maximum size of the inclusion list to be less than the block size. Since each party can choose to propose transactions with no overlap, the maximum size of the input list needs to be restricted to the size of the block divided by the number of parties in the system. We consider the notion of unconditional inclusion lists since if there is a way for a proposer to exclude the transactions in the inclusion list, then it creates a single point of failure that the adversary can exploit. If the number of transactions available in the inclusion list is greater than the size of the block, then in such cases, the cost to censor the transaction is just the difference in the fee paid by the target transaction to the proposer and that for its replacement (which may not be from the IL).

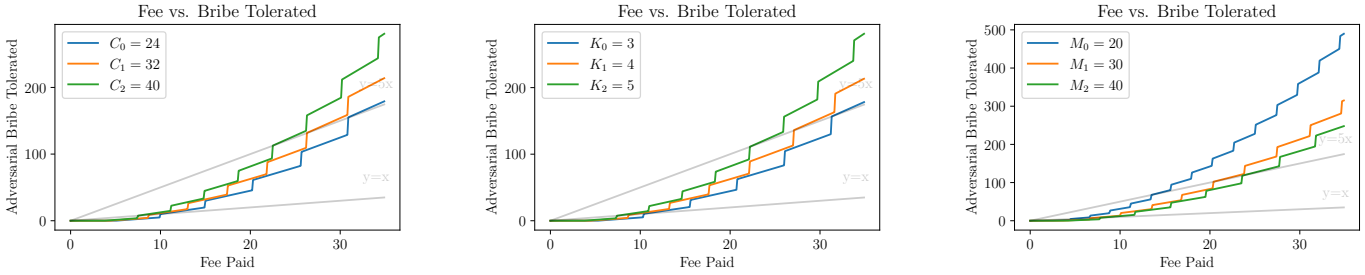
Inclusion lists with EIP 1559. In Ethereum, due to the presence of the EIP-1559 fee mechanism [17], in expectation, block sizes fluctuate around half of the limit. This counters the previous discussion point since, in most cases, there would be enough space for transactions in the block. Suppose the adversary adds spam transactions to fill the leftover block space. In that case, the adversary will incur an additional cost corresponding to the transaction fee (base fee in EIP 1559) of half the block size limit. However, in doing so, the adversary would also increase the base fee for the next block, which may lead to censorship by raising the base fee above the fee. In the absence of these, the input list-building algorithm used could be replaced with a much simpler rule to include all transactions (or capped to block limit to prevent spamming, like in [16]). This does not affect the second part of the design - AUCIL, where inclusion lists are formed by aggregation of input lists. Compared to prior designs such as FOCIL [15], in AUCIL, we do not rely on the honesty of the attestors to collect and aggregate the list locally. Compared to FOCIL, where attesters need to receive all local inclusion lists and compute a running aggregate locally, AUCIL only requires a simple verification of proof. If they do not correctly verify, they could be slashed for incorrectly voting on the progress of the block. In an alternate design where fraud-proofs [25] is used, the proposer who adds the incorrect proof would be slashed. In this case, the attestors are not involved in verifying the validity of the proof either.

Common mempool assumption. In our protocol for input list building algorithm 1, we assume that all parties have the same view of transactions. We note that transactions that pay a fee for obtaining censorship resistance guarantees would necessarily be transmitted through public channels (as opposed to transactions sent as private order flows to only some providers). Hence, any transaction received by one party will be received by all parties soon enough. There may still be minor differences in the mempool of parties due to the time required to transmit transactions to others. This assumption is supported by similar assumptions in previous and parallel works like [24], [27] which build fee structure for multiple leader protocols. Accounting for these differences in the protocol design is an important future work.

Practical considerations. Compared to other inclusion list designs [14], [15], AUCIL has a lower overhead largely in part due to the restricted size of data that each party has to share. However, being a 2-step protocol where the first input lists are broadcast and then the inclusion lists are created as bids, the number of communication rounds increases. Extrapolating numbers from EIP-7805 [16] for inclusion lists in Ethereum, the limits on the size of the input list could be set to $k = 5$ average-sized transactions (or 3 kB of data), and the number of parties in the committee is $n = 32$. This would, on average, imply an inclusion list containing 160 transactions (not an upper bound, just a parameter chosen for practicality).

To be robust against bribery, AUCIL introduces a new fee for each transaction and a reward for aggregating lists. Such a fee can replace the tip paid by the user in case an inclusion list route is taken, or could be introduced in addition to the tip paid. Similarly, the aggregation reward is currently treated as an out-of-protocol reward, i.e., the protocol would generate rewards. However, this could be replaced such that the reward for aggregation is extracted from the user's fee. This design is left as future work.

Commit reveal scheme with a mixed Nash Equilibrium for input list building. As an alternative to using a correlated equilibrium scheme as described in algorithm 1, we could use a commit reveal scheme that avoids the timing games. This potentially could lead to better censorship resistance. However, one of the major properties of the inclusion list-building scheme is that no consensus needs to be reached,



(a) Committee size varied

(b) Input List size varied

(c) scaling factor of transaction fee varied (Average fee is 4, 6, 8 respectively).

Fig. 4: Bribe tolerated vs. user fee under varying parameters. Default parameters: committee size of 32, input list size of 5, and 200 transactions. Transaction fees are chosen from a beta distribution with parameters $(1, 4)$, scaled by a factor of 30.

and without consensus, committing to a bid before revealing it is infeasible.

Secret ordering of IL proposers. In Section VI for algorithm 1, we assume that the adversary knows the ordering of IL proposers, i.e., if it simulates the algorithm, it would know exactly which party would include the target transaction in its input list. However, in a design where this information is unknown to the adversary, it would have to bribe all the parties to ensure that none of the parties include the target transaction. Making this order secret has its challenges. All IL proposers would need to know their exact position but not have any information about the position of others IL proposers to avoid single-party bribery, revealing the entire sequence. This does not have to be verifiable since no party has incentives to switch its position (due to the proven correlated equilibrium). A secret ordering mechanism would thus ensure the theoretically maximum censorship resistance (linear in the number of parties) while also ensuring the maximum utility for each party.

VIII. Related Work

Bribery-based censorship attacks. Blockchains are vulnerable to bribery-based censorship attacks [28], [29], [7]. These attacks have been known to affect the security of various applications like AMMs [30], [31], atomic swaps [7], [32], [33], [34], and auctions [8]. Censorship resistance was formally studied in [8], where they show the extent of the problem, modeling censorship into the consensus can bring for financial applications like an auction.

Inclusion list designs. To our knowledge, this paper is the first to introduce an inclusion list formally designed to combat censorship in literature. However, there are some ideas presented in research posts specific to particular blockchains [12], [14], [15], [35]. In forward Inclusion List designs [12], inclusion list is published by the previous block's proposer. All transactions in the list must be considered in the next block if block space remains unused. This forces builders to fill blocks, making censorship costly, as any unused block space must be used to include transactions in the inclusion list. However, owing to a single proposer-based proposal, this faces major bribery-based censorship issues in which an adversary can bribe the proposer to remove the transaction from the inclusion list.

Multi-party designs like COMIS [14] and FOCIL [15] address the low cost of bribery by relying on a committee to create the inclusion list. Intuitively, they argue that if more parties include the transaction in their inclusion lists, the amount of bribes the adversary pays increases. A practical version of this design-FOCIL has been pushed as EIP-7805 [16], which limits the size of each input; however, the total size limit is still greater than the capacity of the block and could thus overflow. However, such designs fail to account for how these committee members will create their inputs to the inclusion list, with a basic assumption that the highest paying transactions would be chosen, and, thus, fail to provide guarantees when the network is busy, i.e., there are enough transactions to fill the block.

To introduce incentives in the FOCIL inclusion list design, [36] and [37] have been proposed. IncluderSelect [36] attempts to use auctions for includers (IL Proposers) as well; however, it opens the auction to all users, and since each user has an incentive to participate in the auction (to get its transaction accepted). FOCILR [37] attempts to quantify what the proposer must do in case the inclusion list overflows the block size. It also reallocates the burnt base fee as a part of EIP-1559 [17] as rewards for the includers. Table II represents the various properties of previously known inclusion list schemes.

In Flashbots report [35], inclusion lists protocols are studied for censorship resistance in blockchains, albeit under an honest and Byzantine model, where a known threshold of parties can be Byzantine, and the rest are considered honest. The definition of censorship resistance used differs from our definition. They consider the time required to include a transaction in the chain as a parameter for censorship and try to reduce it with various protocols. They analyze leader-based protocols with inclusion lists and note that using a data availability layer or reliable broadcast can help reduce censorship in their design. While they mention the number of parties that need to be bribed in a world where all parties are rational, they do not show what amount of bribes is required or a formal analysis of why the number of bribes cannot be reduced.

Multi-proposer based designs. In [8], in addition to modeling bribery in auctions, they propose mitigating censorship by adding multiple proposers to produce a block simultaneously. While doing so, they propose a dual fee structure in which if the transaction is proposed only once, the tip paid to the party that includes it is higher. This would increase the cost of censorship for the adversary to be proportional to the higher tip (which is rarely paid by the user) instead of the general tip. However, in doing so, they inadvertently prioritize solo inclusion of transactions, reducing the number of times the transaction would be repeated in case more transactions are pending than the size of the block and not all proposers can add all transactions in their local block.

In Flashbots report [35], they also introduce a multi-proposer system called Partially Ordered Dataset (POD) to partially order the available set of transactions. Unlike traditional consensus mechanisms that impose a strict transaction order, POD assigns timestamps that loosely order transactions across replicas. Transactions can be submitted to any replica. They will be recorded as long as they reach a quorum of honest replicas, making it difficult for any single entity to block or censor them entirely. Additionally, POD includes mechanisms for detecting and documenting censorship attempts, creating accountability for malicious behavior. By supporting high throughput and rapid transaction propagation, POD reduces the window in which censorship could occur.

A. Further Details of Inclusion List Designs

In this subsection, we will look at individual protocols that have a (semi-formal) design for the inclusion list. We will compare the 5 major properties described earlier in the model section. We will also clarify what assumptions each of the protocols makes. In all designs, assume there exists a transaction tx_τ, which pays a transaction fee f_τ^p to the proposer (For EIP-1559-like mechanisms [17], this is just the tip paid), and an inclusion fee f_τ for the inclusion.

Base Blockchain with Proposer Builder Separation. In current Ethereum design, block building is auctioned to a builder that provides the highest return to the proposer. Due to private order flows [38], the ethereum builder market has become heavily centralized with 95% of all blocks being built by two builders. In such a system, censorship resistance guarantees are minimal. Since there is no inclusion list in the current design and the block is built by a single proposer, aggregation censorship Property 4, input equilibrium Property 2 and input censorship Property 3 are not applicable in the system. Since there is only one party proposer (and builder is selected from a live auction), θ-Crash Fault Tolerance (Property 1) should be 0. In order to exclude a transaction from the block, an external adversary would need to bribe the builder $br > f_\tau^p$. This would be the value for β_3-Blockchain Censorship Resistance (Property 5). However, going forward we would assume both these values as the base i.e. that the proposer is always live, and the bribe paid to builder to remove the transaction is 0 (unless otherwise constrained).

Single Proposer Forward Inclusion List. To compare the single proposer forward inclusion list design, we will specifically look at EIP-7547[13]. Proposer publishes an inclusion list (IL) each slot, listing transactions that *must* be included by the builder in the next slot (or its own slot). The protocol assumes that the proposer honestly follows the prescribed protocol, and adds all transactions not included in their own current slot in the inclusion list of the next slot. Attesters enforce inclusion list compliance via fork-choice (they will reject a block if the builder leaves space while eligible IL transactions exist). Since this system is still a single-proposer model, the value of θ-Crash Fault Tolerance (Property 1) remains the same as the base blockchain. If the transaction is included in the inclusion list, then the cost of excluding the transaction is either letting go of the block itself (R), or completely filling the block space (due to the conditional nature of the inclusion list). In EIP-1559[17] based blockchains, in expectation, the block is half full and thus the remaining half needs to be filled. The cost of doing so is claimed to be less than $r \cdot s$, where r is the base fee of the block and s is half the size of the block. Thus, value of β_3-Blockchain Censorship Resistance (Property 5) is $\min(R, r \cdot s)$. Since the protocol involves only a single proposer creating an inclusion list, there is no equilibrium or aggregation phase. Thus β_2-Aggregation Censorship Resistance and Input Selection Equilibrium are not applicable. Lastly, if an adversary wants to censor the transaction from the inclusion list, then it would need to pay f_τ to the proposer of the previous block, along with the base bribe to the builder. Thus, β_1-Input Censorship Resistance (Property 3) value is f_τ.

Single-Proposer Unconditional Inclusion Lists To compare this design, we assume an unconditional [18] variant of EIP-7547[13]. In such a case, if the transaction makes it to the inclusion list, then there is no way of removing the transaction from the block except for foregoing the block itself, and thus incurring a cost of R. Thus, value of β_3-Blockchain Censorship Resistance (Property 5) is R. The rest of the values remain the same as its conditional variant.

COMIS: Committee-enforced Inclusion Sets [14][5] Instead of a single proposer, a committee of multiple validators jointly constructs an inclusion set (IS) for each slot. Every slot pos, a committee of size n is randomly chosen; each member picks transactions from the mempool for their local set. These are combined into a global inclusion set IS_{pos}. If IS_{pos} is available in time, the block in slot pos (or pos + 1) must commit to IS_{pos} (e.g. include a summary on-chain), and the next block must include all transactions required by IS_{pos} (subject to conditions). They introduce the concept of Inclusion Threshold (IT) to avoid spam transactions. They require that for a transaction to be in IS_{pos}, it needs to be included by at least $(1 - IT) \cdot n$ parties. While the incentives in COMIS were not clearly defined, it relies on distributing the reward for

[5]Many of the protocol choices are not defined concretely, like the incentive structure, crash tolerance, etc. We make the best we could think of fixes to each of these.

inclusion based on the contribution that each party makes. For our analysis, we will assume that the reward distributed is equal to the inclusion fee f_τ for the transaction. COMIS can tolerate up to IT parties crashing; however, in such a case, censorship resistance for a transaction would be 0. In their model, they assume the existence of an aggregator (which can be the block proposer, and for the numbers here, we assume this to be the case). Let's look at COMIS, which can tolerate up to θ crashes, and requires $ITx = (1 - IT)$ fraction of includes to include a transaction to be valid. The value of θ-Crash Fault Tolerance (Property 1) is thus θ. If a transaction is included in the inclusion list by the aggregator, then depending on whether COMIS is considered to be an unconditional inclusion list or not, the cost of censorship is R and $\min(R, r \cdot s)$ respectively (β_3-Blockchain Censorship Resistance, Property 5). Since the aggregator is a single party collecting all inputs, and this aggregation is not rewarded, the cost to bribe the aggregator to exclude a transaction that is in the input of a single party would be $\epsilon \to 0^+$. In order to avoid this, each transaction must be included in $\theta + ITx$ different includers' list. If so, the aggregator would not be able to censor the transaction, given that COMIS implementation requires proof of $n - \theta$ inputs. Thus the value of β_2-Aggregation Censorship Resistance (Property 4) is 0 if $n_t < \theta + ITx$ and ∞ otherwise.

Since the inclusion committee is assumed to consist of an honest majority of includers, no equilibrium is established in the game. However, if all members are considered rational, then an equilibrium would be reached in which all parties maximize their revenue. Assuming that an optimal revenue is reached, the members would receive at most $\frac{f_\tau}{ITx}$ revenue from a transaction. With $n_t < n$ and using the reference $ITx = n/5$, then in order to bribe all the parties from excluding the transaction from the input, an adversary would require $< 5f_\tau$ (β_1-Input Censorship Resistance, Property 3). The actual cost is far lower than this, since the adversary can use an aggregator to threaten the exclusion of the includer's inputs if it includes a particular transaction. This leads to an equilibrium, where includers would choose not to include the transaction for a small bribe of $\varepsilon \to 0^+$.

The cost for the unconditional variant of COMIS is much lower, since n_t at equilibrium would be close to ITx (Since censorship is not expected during the equilibrium calculation). At this value of n_t, β_2-Aggregation Censorship Resistance would be 0 for most transactions.

FOCIL: Fork Choice enabled Inclusion List [15],[16] FOCIL mostly follows the basic design underlined by COMIS. They use an Inclusion List committee to declare inputs that need to be included in the Inclusion List. They fix the aggregator as the block producer (Builder in the post-PBS[39], [40] world, or the block proposer). They remove the inclusion threshold (i.e., set $ITx = 1$). To verify the validity of the inclusion list, they rely on attesters to locally compute the list 3 seconds before the block producer creates the inclusion list, allowing enough time for the block producer to receive the inputs from all IL proposers. However, this relies on an honest majority of attesters, which differs from Ethereum's general fork choice, where a single honest attester is sufficient to prove correctness. FOCIL does not have any incentives defined for the IL proposers. For analysis, we use the double fee mechanism and assume that all parties that include transaction tx_τ share f_τ equally.

Here, it is tricky to assume any θ-Crash Fault Tolerance. They rely on attesters to justify whether a party was actually crashed and did not broadcast an inclusion list, or whether the aggregator censored the input it submitted. This reliance is justified when assuming that the attesters behave honestly. However, there is no proof that the aggregation was honest. If enough attesters agree with a censored inclusion list, then there is no proof of censorship. Thus, if all parties are considered rational, then no crash can be tolerated, i.e., θ-Crash Fault Tolerance has a value $n-1$. Under this condition, FOCIL achieves β_1-Input Censorship Resistance (Property 3) of $n \cdot f_\tau$. This is because, being a conditional inclusion list, the equilibrium for input selection involves all parties choosing all valid transactions in the mempool (Input Selection Equilibrium, Property 2 is Select M). Like all conditional inclusion lists, since EIP-1559 targets a half full block, the value for β_3-Blockchain Censorship Resistance (Property 5) is less than $r \cdot s$, where r is the base fee of the block and s is half the size of the block.

Also note that a recent parallel work analyzing a transaction fee mechanism design for FOCIL [27] shows that even when all parties are considered rational, it is an equilibrium for the users to distribute the fee such that $f_\tau = 0, f_\tau^p = f$, where f is the maximum fee that the user pays. This implies that under rationality of all IL proposers, the overall censorship resistance is still bounded by f. This result is a derivative of an interesting method to spam the mempool, such that the target transaction is excluded by the block proposer and the includers, without incurring the fee for spamming.

Protocol Name	Comments	Property 1	Property 2	Property 5	Property 4	Property 3
Base Ethereum	Low CR for multi-block	-	-	f_τ^p	-	-
Single-Proposer Forward IL	Each block has independent CR	-	M	$\min(R, r \cdot s)$	-	f_τ
Single-Proposer Unconditional		-	top k	R	-	f_τ
COMIS Forward IL	$n_\tau < \theta + ITx$ $n_\tau \geq \theta + ITx$	θ θ	M M	$\min(R, r \cdot s)$ $\min(R, r \cdot s)$	0 ∞	$\frac{n f_\tau}{ITx}$ $\frac{n f_\tau}{ITx}$
COMIS Unconditional	$n_\tau < \theta + ITx$ $n_\tau \geq \theta + ITx$	θ θ	MNE MNE	R R	0 ∞	≈ 0 ≈ 0
FOCIL	Any-honest committee, Majority honest attesters	$n-1$	M	$\min(R, r \cdot s)$	∞	-
FOCIL	Rational parties	0	M	$\min(R, r \cdot s)$	≈ 0	$n\ f_\tau$
AUCIL (w/o input selection)		θ	MNE	R	$(n-\theta)(u_{agg}/2)$	≈ 0
AUCIL (with input selection)		θ	Alg. 1	R	$(n-\theta)(u_{agg}/2)$	$O(f_\tau^2 \cdot n/\sigma)$

TABLE II: Comparison to previous protocols [6]

REFERENCES

[1] M. Holden, "WikiLeaks says "blockade" threatens its existence | Reuters." [Online]. Available: https://www.reuters.com/article/us-britain-wikileaks/wikileaks-says-blockade-threatens-its-existence-idUSTRE79N46K20111024/?feedType=RSS&feedName=topNews

[2] "Canada convoy protest," Sep. 2024, page Version ID: 1243947900. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Canada_convoy_protest&oldid=1243947900

[3] Robinhood, "Keeping Customers Informed Through Market Volatility," Jan. 2021. [Online]. Available: https://newsroom.aboutrobinhood.com/keeping-customers-informed-through-market-volatility/

[4] A. Robertson, "Robinhood is facing dozens of lawsuits over GameStop stock freeze," Feb. 2021. [Online]. Available: https://www.theverge.com/2021/2/1/22254656/robinhood-gamestop-stonks-trade-freeze-class-action-lawsuits

[5] U.S. Department of the Treasury, "OFAC Specially Designated Nationals Data," 2022. [Online]. Available: https://www.treasury.gov/ofac/downloads

[6] A. Wahrstätter, J. Ernstberger, A. Yaish, L. Zhou, K. Qin, T. Tsuchiya, S. Steinhorst, D. Svetinovic, N. Christin, M. Barczentewicz, and A. Gervais, "Blockchain Censorship," Jun. 2023, arXiv:2305.18545 [cs]. [Online]. Available: http://arxiv.org/abs/2305.18545

[7] F. Winzer, B. Herd, and S. Faust, "Temporary censorship attacks in the presence of rational miners," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 357–366.

[8] E. Fox, M. Pai, and M. Resnick, "Censorship resistance in on-chain auctions," 2023. [Online]. Available: https://arxiv.org/abs/2301.13321

[9] "Builder landscape — ethereum mainnet," 2025. [Online]. Available: https://explorer.rated.network/builders?network=mainnet

[10] S. Yang, K. Nayak, and F. Zhang, "Decentralization of Ethereum's Builder Market," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, May 2025, pp. 1512–1530.

[11] V. Buterin, "State of research: increasing censorship resistance of transactions under proposer/builder separation (pbs)," 2021. [Online]. Available: https://notes.ethereum.org/s3JToeApTx6CKLJt8AbhFQ#Hybrid-PBS-can-we-use-proposers-only-for-inclusion-of-last-resort

[12] Francesco. [Online]. Available: https://notes.ethereum.org/@fradamt/forward-inclusion-lists

[13] Michael, Vitalik, Francesco, Terence, potuz, and Manav, "Eip-7547: Inclusion lists," Oct 2023. [Online]. Available: https://eips.ethereum.org/EIPS/eip-7547

[14] Thomas, Francesco, and Barnabe, "The more, the less censored: Introducing committee-enforced inclusion sets (comis) on ethereum," Feb. 2024. [Online]. Available: https://ethresear.ch/t/the-more-the-less-censored-introducing-committee-enforced-inclusion-sets-comis-on-ethereum/18835

[15] "Fork choice enforced inclusion lists," Jun. 2024. [Online]. Available: https://ethresear.ch/t/fork-choice-enforced-inclusion-lists-focil-a-simple-committee-based-inclusion-list-proposal/19870

[16] E. I. Proposals, "Eip-7805: Fork-choice enforced inclusion lists (focil) [draft]," Nov. 2024. [Online]. Available: https://eips.ethereum.org/EIPS/eip-7805

[17] Vitalik, Eric, Rick, Matthew, Ian, and Abdelhamid, "Eip-1559," Apr 2019. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1559

[18] "Unconditional inclusion lists," Jan. 2024. [Online]. Available: https://ethresear.ch/t/unconditional-inclusion-lists/18500

[19] R. J. Aumann, "Subjectivity and correlation in randomized strategies," *Journal of mathematical Economics*, vol. 1, no. 1, pp. 67–96, 1974.

[20] ——, "Subjectivity and correlation in randomized strategies," *Journal of mathematical Economics*, vol. 1, no. 1, pp. 67–96, 1974.

[21] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[22] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: a tutorial," *ACM Comput. Surv.*, vol. 22, no. 4, p. 299–319, Dec. 1990. [Online]. Available: https://doi.org/10.1145/98163.98167

[23] [Online]. Available: https://beaconscan.com/stat/networkparticipation

[24] P. Garimidi, L. Heimbach, and T. Roughgarden, "Transaction fee mechanism design for leaderless blockchain protocols," *arXiv preprint arXiv:2505.17885*, 2025.

[25] M. Al-Bassam, A. Sonnino, and V. Buterin, "Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities," *arXiv preprint arXiv:1809.09044*, vol. 160, 2018.

[26] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.
[27] A.-P. Stouka, J. Ma, and T. Thiery, "Multiple proposer transaction fee mechanism design: Robust incentives against censorship and bribery." [Online]. Available: http://arxiv.org/abs/2505.13751
[28] J. Bonneau, "Why buy when you can rent? bribery attacks on bitcoin-style consensus," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 19–26.
[29] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*. Springer, 2019, pp. 3–18.
[30] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," *arXiv preprint arXiv:1904.05234*, 2019.
[31] C. F. Torres, R. Camino, and R. State, "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1343–1359. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/torres
[32] S. Wadhwa, J. Stoeter, F. Zhang, and K. Nayak, "He-htlc: Revisiting incentives in htlc," *Cryptology ePrint Archive*, 2022.
[33] H. Chung, E. Masserova, E. Shi, and S. A. Thyagarajan, "Rapidash: Foundations of side-contract-resilient fair exchange," *Cryptology ePrint Archive*, 2022.
[34] I. Tsabary, M. Yechieli, A. Manuskin, and I. Eyal, "Mad-htlc: because htlc is crazy-cheap to attack," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1230–1248.
[35] O. Alpos, B. David, N. Kamarinakis, and D. Zindros, "Flashbots report: System requirements, existing and new solutions, and their efficiency." *Flashbots Report*, 2024.
[36] J. Ma. [Online]. Available: https://mirror.xyz/julianma.eth/G15Gs2TGfnU93t8R7fuyFjTmZGIwwhRFhNhH_M0dgGE
[37] A. Elowsson, Feb. 2025. [Online]. Available: https://ethresear.ch/t/rainbow-roles-incentives-abps-focilr-as/21826
[38] S. Wang, Y. Huang, W. Zhang, Y. Huang, X. Wang, and J. Tang, "Private order flows and builder bidding dynamics: The road to monopoly in ethereum's block building market," 2024. [Online]. Available: https://arxiv.org/abs/2410.12352
[39] "Proposer/block builder separation-friendly fee market designs - Economics," Jun. 2021, section: Economics. [Online]. Available: https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725
[40] L. Heimbach, L. Kiffer, C. Ferreira Torres, and R. Wattenhofer, "Ethereum's proposer-builder separation: Promises and realities," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, 2023, pp. 406–420.
[41] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," *Journal of Political Economy*, vol. 119, no. 6, pp. 1061–1103, 2011.
[42] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum nash welfare," *ACM Transactions on Economics and Computation (TEAC)*, vol. 7, no. 3, pp. 1–32, 2019.

APPENDIX A
NASH EQUILIBRIUM

To compute the Nash Equilibrium for the input list-building scheme (with $\gamma = 1$), consider the following:

Let p_i represent the probability of choosing object m_i in a Nash equilibrium. In a Nash equilibrium, all parties follow the same probability distribution of choosing the object. Given this information, let party n_j be the decision-making party when picking objects and try to deviate from the given Nash Equilibrium Probability. For this party, let x_{p_i} represent the probability of choosing an object m_i. The utility for such a party is given by

$$u_j = \sum_{i=1}^{m} x_{p_i} f_i \sum_{k=0}^{n-1} \frac{\binom{n-1}{k}(p_i)^k(1-p_i)^{n-k-1}}{k+1}$$
$$= \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \sum_{k=0}^{n-1} \frac{1}{k+1}\binom{n-1}{k}\left(\frac{p_i}{1-p_i}\right)^k$$

Let $c = \frac{p_i}{1-p_i}$

$$u_j = \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \sum_{k=0}^{n-1} \binom{n-1}{k}\frac{c^k}{k+1}$$
$$= \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \cdot \frac{1}{c} \cdot \sum_{k=0}^{n-1} \binom{n-1}{k}\frac{c^{k+1}}{k+1}$$

Now, from the integral of binomial expansion of $(1+c)^{n-1}$(by parts from 0 to c)

$$\sum_{k=0}^{n-1} \binom{n-1}{k}\frac{c^{k+1}}{k+1} = \frac{(1+c)^n - 1}{n}$$

From this and substituting $c = \frac{p_i}{(1-p_i)}$,

$$u_j = \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \cdot \frac{1}{c} \cdot \left(\frac{(1+c)^n - 1}{n}\right)$$
$$= \sum_{i=1}^{m} \frac{x_{p_i}}{n \cdot p_i} f_i (1-p_i)^n \cdot \left((\frac{1}{1-p_i})^n - 1\right)$$
$$= \sum_{i=1}^{m} \frac{x_{p_i}}{n \cdot p_i} f_i \cdot (1-(1-p_i)^n)$$

We can ignore the higher-order term for any $0 < p_i < 1$ (As long as p_i is greater than $O(1/n)$). Thus,

$$u_j = \sum_{i=1}^{m} \frac{x_{p_i}}{np_i} f_i$$

For the Nash equilibrium to exist at values $x_{p_i} = p_i, \forall i \in m$, all terms should be individually equal (so that multiplying by x_{p_i} yields the same value across all m terms). Thus, $\frac{f_i}{p_i} = \frac{f_j}{p_j}$. Since each party selects k objects, we have $\sum p_i = k$, this gives us

$$p_i = \frac{k f_i}{\sum_i f_i} \tag{13}$$

However, some p_i can also be 0. For this to be the case, we require that the utility term corresponding to i should be less than the utility of all other utility terms. Thus, $f_i \leq \frac{f_j}{np_j}$ for all j such that $p_j \neq 0$. This implies if $f_i \leq \frac{\sum f_j}{nk}$ then $p_i = 0$.

Some p_i can exceed 1 as well, given the calculation needed to find the value. In such a case, the probability will be capped at 1, and all parties will choose that object.

The expected utility for the input list proposer would be less than or equal to $\frac{\sum\limits_{i:p_i \neq 0} f_i}{n}$(in cases where due to randomness,

one or more m_i with $p_i > 0$ did not get selected). As an example, if there exist two objects with values 3 each, and two parties select one object each. In this case, if both parties select the same object (occurs with probability 0.5), the utility for both parties is 1.5. In this case, the expected utility for both parties is 2.25, which is < 3.

Appendix B
Allocation Examples

To look at various allocations, consider the following example: 15 transactions exist with utilities 15, 14, 13, 11, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1. Let there be 5 IL proposers, choosing three transactions each.

Let's consider 5 different allocations as shown in Table III. A1, A2, A3, and A4 represent the correlated equilibrium at various

	Party	Index	Utility			
			U_{100}	U_{70}	U_{50}	U_0
A0	Party 1	[0, 5, 10]	21	21	21	21
	Party 2	[1, 6, 11]	19	19	19	19
	Party 3	[2, 7, 12]	17	17	17	17
	Party 4	[3, 8, 13]	14	14	14	14
	Party 5	[4, 9, 14]	8	8	8	8
A1	Party 1	[1, 4, 6]	14.6667	15.8333	17	24
	Party 2	[0, 1, 3]	13.9167	17.1426	20.3333	40
	Party 3	[0, 2, 3]	13.5833	16.726	19.8333	39
	Party 4	[0, 2, 5]	13.0833	15.2554	17.5	33
	Party 5	[0, 1, 2]	12.75	16.0887	19.5	42
A2	Party 1	[0, 2, 4]	14.0833	16.2554	18.5	34
	Party 2	[0, 1, 2]	12.75	16.0887	19.5	42
	Party 3	[1, 3, 5]	13.3333	15.4167	17.5	30
	Party 4	[0, 1, 3]	12.0833	15.2554	18.5	40
	Party 5	[0, 2, 3]	11.75	14.8387	18	39
A3	Party 1	[0, 1, 2]	11.5833	14.7715	18.1	42
	Party 2	[0, 1, 2]	11.5833	14.7715	18.1	42
	Party 3	[2, 3, 4]	14	16	18	30
	Party 4	[0, 1, 3]	10.9167	13.9382	17.1	40
	Party 5	[0, 1, 3]	10.9167	13.9382	17.1	40
A4	Party 1	[0, 1, 2]	8.4	11.0526	14	42
	Party 2	[0, 1, 2]	8.4	11.0526	14	42
	Party 3	[0, 1, 2]	8.4	11.0526	14	42
	Party 4	[0, 1, 2]	8.4	11.0526	14	42
	Party 5	[0, 1, 2]	8.4	11.0526	14	42

TABLE III: Utility Received by Each Party for Different Allocations. U_p represents the utility of broadcasting the list if the probability of broadcasting is $p\%$

Appendix C
Example for Algorithm 1

This example demonstrates the operation of algorithm 1 with $n = 3$ parties, $m = 5$ objects, $k = 2$ size of inclusion list and $U = [8, 6, 5, 3, 1]$ utility values of the objects.

[6] σ is the sum of all fees paid by transactions in the Inclusion list, u_{agg} is the issuance reward for the aggregator. The analysis of these protocols is based on our best knowledge. There is no formal analysis available for any previous work.

In Step 1, the algorithm iteratively selects the highest-value objects from U, dynamically adjusting selection counts in N. This process continues until $n \cdot k$ selections are made or all objects are fully allocated.

In Step 2, the algorithm distributes the selected objects among the players. The objects are allocated based on their adjusted utilities U_f and assigned in decreasing order of U_f. Each round assigns objects in order, updating the players' inclusion arrays L_i. Step 2 starts with $U = [8, 6, 5, 3, 1]$, $N = [2, 2, 1, 1, 0]$, $U_f = [4, 3, 5, 3, 0]$, and $A = [2, 0, 1, 3, 4]$

Appendix D
Fairness of Algorithm 1

Randomness has its limitations. On-chain randomness is prone to grinding attacks and thus is not enough to guarantee that the rewards from the input list-building scheme are equally distributed. Thus, we prove that the round-robin allocation of objects through algorithm 1 with $\gamma = 1$ follows multiple definitions of fairness known in the literature. The definition as proposed by [41]

Lemma 13. *(EF1-Fairness) The allocation achieved in algorithm 1 is 1-Envy Free.*

Proof. The proof for the algorithm is simple. During Step 2, objects are allocated in a round-robin after sorting. Let o_i^r be allocated to party i in round r. The following two properties hold due to sorted allocation.

$$\mathbb{U}(o_i^r) \leq \mathbb{U}(o_j^r)\ \forall r; i < j$$
$$\mathbb{U}(o_i^r) \leq \mathbb{U}(o_j^k)\ \forall j; k < r$$
$$\text{where, } \mathbb{U}(o_i^r) = \frac{u_{o_i^r}}{n_{o_i^r}}$$

Let party j envy party i. If $j < i$, then in each round, the utility gained by party j is greater than party i, and thus, there is no envy. For $j > i$, let's remove the first object party i received. Thus, for each object received by party i in round r, there exists an object in round $r - 1$ that party j receives. Since the utility gained from an earlier round is always more than that gained in the previous round, the utility of party j is greater than that of party i if the last object allocated to party i is removed. □

However, the property of EF1 does not determine any bound on the difference in utilities for all parties. In our allocation, since a correlated equilibrium is maintained while choosing the objects, we can also prove an absolute bound on the utility difference between parties.

Lemma 14. *(Bounded Envy) In Algorithm 1, the utility for each party is at least half the utility of all other parties.*

Proof. We maintain the following invariant while selecting objects in algorithm 1, except when $n_j = n$.

$$\frac{u_i}{n_i} \geq \frac{u_j}{n_j + 1}$$

Loop	U_{curr}	s	N	U after update
1	[8, 6, 5, 3, 1]	0	[1, 0, 0, 0, 0]	[8, 6, 5, 3, 1]
2	[4, 6, 5, 3, 1]	1	[1, 1, 0, 0, 0]	[8, 6, 5, 3, 1]
3	[4, 3, 5, 3, 1]	2	[1, 1, 1, 0, 0]	[8, 6, 5, 3, 1]
4	[4, 3, 2.5, 3, 1]	0	[2, 1, 1, 0, 0]	[8, 6, 5, 3, 1]
5	[2.66, 3, 2.5, 3, 1]	1	[2, 2, 1, 0, 0]	[8, 6, 5, 3, 1]
6	[2.66, 1.5, 2.5, 3, 1]	3	[2, 2, 1, 1, 0]	[8, 6, 5, 3, 1]

Round	Description	Variable State
1	Assign $A[0]$ once, $A[1]$ twice	L = [[0, 0, 1, 0, 0], [1, 0, 0, 0, 0], [1, 0, 0, 0, 0]]
2	Assign $A[2]$ twice, $A[3]$ once	L = [[0, 1, 1, 0, 0], [1, 1, 0, 0, 0], [1, 0, 0, 1, 0]]

If $n_j = n$, then the object would be allocated to all n parties since all parties are allocated an object only once. Let's say some party that picks object o_i in the last round envies the party that picked o_j in the first round.

$$\frac{u_{o_i}}{n_{o_i}} \geq \frac{u_{o_j}}{n_{o_j}+1}$$

$$\frac{u_{o_j}}{n_{o_j}} \geq \frac{u_{o_i}}{n_{o_i}}$$

In the proof for Lemma 13, we have that utility for a party with object o_j is lower than the utility for a party with object o_i if both o_j and o_i are removed.

$$\mathbb{U}(L_i \setminus \{o_i\}) \geq \mathbb{U}(L_j \setminus \{o_j\})$$

The difference for the party with object o_i who envies the party with object o_j is given by,

$$\begin{aligned}
&\mathbb{U}(L_j) - \mathbb{U}(L_i) \\
&\mathbb{U}(L_j \setminus \{o_j\}) + \frac{u_{o_j}}{n_{o_j}} - \mathbb{U}(L_i \setminus \{o_i\}) - \frac{u_{o_i}}{n_{o_i}} \\
&\leq \mathbb{U}(L_j \setminus \{o_j\}) + \frac{u_{o_j}}{n_{o_j}} - \mathbb{U}(L_i \setminus \{o_i\}) - \frac{u_{o_j}}{n_{o_j}+1} \\
&\leq \frac{u_{o_j}}{n_{o_j}} - \frac{u_{o_j}}{n_{o_j}+1} \leq \frac{u_{o_j}}{n_{o_j}\cdot(n_{o_j}+1)} \leq \frac{u_{o_j}}{n_{o_j}\cdot 2} \leq \frac{\mathbb{U}(L_j)}{2}
\end{aligned}$$

□

Now that we have bounded the envy, can we do better? In another definition of Envy Freeness in [42], we have

Lemma 15. *(EFx-Fairness) The allocation achieved in Algorithm 1 is EFx Fair.*

Proof. Consider party j envies party i, i.e. $\mathbb{U}(L_i) \geq \mathbb{U}(L_j)$. Let u_p^r be the utility from r^{th} object allocated to party $p \in \{i, j\}$. We know that

$$\frac{u_i^r}{n_i^r} \geq \frac{u_j^r}{n_j^r} \ \forall r$$

$$\frac{u_p^{r-1}}{n_p^{r-1}} \geq \frac{u_{p'}^r}{n_{p'}^r} \ \forall p, p' \in \{i, j\}, \forall r$$

Consider all objects allocated to party i in rounds $2, \ldots, k-1$.

$$\begin{aligned}
&\frac{u_j^{r-1}}{n_j^{r-1}} \geq \frac{u_i^r}{n_i^r} \ \forall r \in \{2, \ldots, k-1\} \\
&\sum_{r\in\{1,\ldots,k-2\}} \frac{u_j^r}{n_j^r} \geq \sum_{r\in\{2,\ldots,k-1\}} \frac{u_i^r}{n_i^r} \qquad (14)
\end{aligned}$$

Further,

$$\begin{aligned}
\frac{u_i^1}{n_i^1} &\geq \frac{u_j^k}{n_j^k} \\
\frac{u_j^k}{n_j^k} \geq \frac{u_i^1}{n_i^1+1} &\geq \frac{u_i^1}{2n_i^1} \\
\frac{u_j^{k-1}}{n_j^{k-1}} \geq \frac{u_j^k}{n_j^k} &\geq \frac{u_i^1}{2n_i^1} \\
\frac{u_j^{k-1}}{n_j^{k-1}} + \frac{u_j^k}{n_j^k} &\geq \frac{u_i^1}{n_i^1} \qquad (15)
\end{aligned}$$

Adding (14) and (15),

$$\sum_{r\in\{1,\ldots,k\}} \frac{u_j^r}{n_j^r} \geq \sum_{r\in\{1,\ldots,k-1\}} \frac{u_i^r}{n_i^r} \qquad (16)$$

Which implies $\mathbb{U}(L_j) \geq \mathbb{U}(L_i \setminus o_i^k)$ where o_i^k is the lowest utility item in L_i, and thus proving the algorithm satisfies EFx Fairness. □

Thus, we achieve the following fairness definition.

Theorem 7. *(Fairness) Algorithm 1 achieves an EFx fair allocation with the utility of each party being at least half of any other party participating in the allocation.*

The proof follows from Lemma 14 and Lemma 15.

Allocation	Bribery Budgets							
Fee tx	15	14	13	11	6	5	4	3,2,1,1,1,1,1,1
A0	8	6.5	5.5	3.5	0	0	0	0
A1	46.583	32.083	28.833	15.25	2.333	1.333	0.333	0
A2	46.25	31.75	28.5	22.25	2	1	0	0
A3	44.75	40.75	27	20.75	1	0	0	0
A4	49	44	39	0	0	0	0	0

TABLE IV: Censorship Resistance provided by each allocation.