Prooφ: A ZKP Market Mechanism

Wenhao Wang[1], Lulu Zhou[1], Aviv Yaish[1], Fan Zhang[1], Ben Fisch[1], and Benjamin Livshits[2]

[1] Yale University, New Haven, CT, USA
[2] Imperial College London, London, UK

Abstract. Zero-knowledge proofs (ZKPs) are computationally demanding to generate. Their importance for applications like ZK-Rollups has prompted some to outsource ZKP generation to a market of specialized provers. However, existing market designs either do not fit the ZKP setting or lack formal description and analysis.

In this work, we propose a formal ZKP market model that captures the interactions between users submitting ZKP tasks and provers competing to generate proofs. Building on this model, we introduce *Prooφ*, an auction-based ZKP market mechanism. We prove that *Prooφ* is incentive compatible for users and provers, and budget balanced. We augment *Prooφ* with system-level designs to address the practical challenges of our setting, such as Sybil attacks, misreporting of prover capacity, and collusion. We analyze our system-level designs and show how they can mitigate the various security concerns.

1 Introduction

Zero-knowledge proofs (ZKPs) enable efficient verification of computation and are used by blockchain scalability solutions (e.g., ZK-Rollups [31, 29, 27]), user authentication [2], data oracles [33], and many more. Despite recent efficiency improvements, generating ZKPs remains computationally expensive [6], often requiring specialized infrastructure (e.g., GPU or ASIC). This has naturally led to the emergence of *ZKP markets* [19, 11, 15] that allow users to outsource proof generation to specialized provers. Ideally, an effective market will not only improve user experience but also lower user costs by fostering competition among provers. However, since such markets operate in an open and decentralized environment, challenges arise because users and provers might be malicious.

Market designs by both industry and academia [26, 11, 19, 29] are limited. For example, the elegant work of Gong et al. [12] considers a market comprising one user and multiple provers. However, proving multiple user transactions in batches is a key performance optimization in ZK-Rollups. Moreover, ZKP applications, such as ZK-Rollups, are already serving a substantial user base [17], highlighting the need for mechanisms that address multi-user scenarios. On the other hand, the mechanism advanced by the commercial prover market Gevulot [11] supports multiple users but relies on a fixed fee for all tasks, implying users willing to pay more may have to wait longer than those willing to pay less. A survey of the

literature (see Section 2) shows that other proposed mechanisms either do not apply to our setting or lack formal specifications and analysis.

This work. We formally model ZKP markets and dissect several designs. In particular, we propose *Proo*φ, a ZKP market mechanism that comprises a *core auction mechanism* and *system-level designs*. At a high level, the core mechanism of *Proo*φ runs an auction between users and provers and allocates a set of low-cost provers with another set of high-value user tasks. We formally analyze the properties of the core mechanism and show that it is budget-balanced and guarantees incentive compatibility of market participants. We then introduce system-level designs to enhance the core mechanism and mitigate security threats that the core auction mechanism alone cannot address, and provide an analysis of how these designs defend against untruthful capacity bids, Sybil attacks, and collusion. The main challenges lie in modeling the ZKP market to capture the setup of practical systems, achieving desired security properties through game-theoretic designs (e.g., with auctions) and system-level designs (e.g., using cryptography), and developing rigorous formal analysis.

1.1 ZKP Market Model

We consider a model comprising users, who have ZKP generation tasks, and provers, who finish user tasks for rewards. The market operates in rounds of auctions, and each auction handles a specific *type* of tasks, i.e., tasks of the same circuit and ZKP scheme. In each auction, users submit tasks and specify a fee f for each task. The fee reflects how much the user values this task, also called the *value* of a task. Meanwhile, each prover specifies a capacity s, i.e., the number of tasks they can handle within a predefined time frame (tailored to application needs), and their unit cost p i.e., the cost to finish one given task. Given user tasks and provers' capacities and unit costs, a market mechanism decides *allocation* and *payments*: it selects a subset of tasks that are to be proven, and for each one, the prover that would generate the associated proof, the user's payment, and the amount that can be collected as revenue by the prover. Note that user payments and the prover's revenue may differ from user-chosen fees.

Our model explicitly captures the fact that practical provers produce proofs in batch, by allocating user tasks in batches. Assigning tasks of the same type to one prover is more efficient than assigning them to different provers, as it avoids context-switching costs (e.g., loading proving keys and circuits to the memory) and can leverage efficient batch proofs available in some ZKP schemes [16, 3, 8]. Although we use ZK-Rollup terminologies, our market model applies to other systems that include a market of verifiable services, such as zkLogin [2], where users need to generate ZKPs to authenticate their identity.

1.2 Our Core Market Mechanism: *Proo*φ

Inspired by second-price auction mechanisms such as VCG [30] and classic double auction mechanisms [18], we present our core ZKP market mechanism named

Prooφ. In this mechanism, we guarantee incentive compatibility by paying each allocated prover the "second price", i.e., the price reported by the unallocated prover with the lowest cost, and ensuring each allocated prover's capacity is used in full. Concretely, the mechanism first hypothetically allocates the highest-paying user tasks to the lowest-cost provers and stops when reaching a task with a fee that no longer covers the cost of the prover. The mechanism then selects this hypothetical set of provers excluding the one with the highest unit cost, and selects the highest value tasks up to the capacity of the selected provers. The core mechanism of *Proo*φ is specified below.

Core Mechanism of *Proo*φ

- The mechanism collects user bids $\{f_i\}_{i=1}^{n}$, and prover bids $\{(s_j, p_j)\}_{j=1}^{N}$ with $f_1 \geq \cdots \geq f_n$ and $p_1 \leq \cdots \leq p_N$. Let $\bar{S}_j := \sum_{i=1}^{j} s_i$ denote the sum of the first j provers' capacities.
- The mechanism determines the largest ℓ such that $p_{j+1} \leq f_{(\bar{S}_j+1)}$, i.e. $\ell = \arg\max_j \{p_{j+1} \leq f_{(\bar{S}_j+1)}\}$. The first ℓ provers are allocated in full, with the first $\bar{S}_\ell$ user tasks.
- Each selected user is charged $f_{\bar{S}_\ell+1}$. Each selected prover with index $j \in [1, \ell]$ is $s_j \cdot p_{\ell+1}$.

To provide intuition, we run the mechanism through a simple example.

Example 1. Suppose a market with 8 tasks with values $(10, 10, 10, 10, 9, 9, 1, 1)$, and 3 provers with capacities s_i and costs p_i such that $(s_1, p_1) = (4, 0), (s_2, p_2) = (2, 1), (s_3, p_3) = (2, 10)$. When all parties are bidding honestly, we have $\ell = 1$. The first $\bar{S}_\ell = \sum_{j=1}^{\ell} s_j = 4$ task are allocated, and are charged $f_{\bar{S}_\ell+1} = 9$. Prover 1 is paid $p_{\ell+1} = 1$ per task, and has utility $(1 - 0) \times 4 = 4$.

One interpretation is that we first greedily match high-paying tasks with low-cost provers until none are left. E.g., p_1 is matched f_1 through f_4, and so on. Then, we compare a prover k's cost p_k with the highest fee of her matched tasks (which is precisely $f_{\bar{S}_{k-1}+1}$). E.g., compare p_1 with f_1, p_2 with f_5, and p_3 with f_7. Let's call a prover *feasible* if $p_k \leq f_{\bar{S}_{k-1}+1}$. Finally, the mechanism allocates *all but the last* feasible provers to their full capacity and calls the number of allocated provers ℓ. In this example, p_1 and p_2 are feasible, so $\ell = 1$. The last feasible prover's cost and its highest-paying task's fee set the prices for provers and users, respectively. I.e., provers are paid $p_{\ell+1}$ and users are charged $f_{\bar{S}_\ell+1}$.

We prove *Proo*φ achieves several desirable properties. It is *budget-balanced* (Proposition 1), i.e., the payments to provers are always covered by the fees collected from users. Moreover, it is *incentive compatible* for users and provers to bid honestly (Propositions 2 and 3).

1.3 Implementing *Proo*φ in an Open ZKP market

The core mechanism alone leaves certain security threats open, thus we enhance it with system-level solutions, including slashing, fixing prover capacity bids over

multiple rounds, and encrypting bids to hide information required for profitable deviations such as misreporting of prover capacity, Sybil attacks, and collusion. When considering risk-averse actors (in line with previous work making the same assumption [7, 5, 23, 4]), our solutions thwart possible risk-free threats.

Incorrect or missing proofs. To ensure the correctness and timeliness of the completion of the ZKP tasks, we require provers to deposit collateral when joining the protocol, and seize the collateral of provers who generate incorrect ZKPs or do not generate them in a timely manner (as detailed in Section 5.1).

Misreporting capacity. The core *Proo*φ mechanism is incentive compatible for provers, meaning that they truthfully report their *proving costs*. However, provers may profitably deviate in other ways, such as by misreporting their *capacity*. In Proposition 4, we prove that such deviations are worthwhile only when the bids of other users and provers satisfy a narrow condition, implying that dishonest provers would have to monitor user bids and adjust their reported capacities correspondingly. Following this observation and given that a prover's capacity is not likely to change drastically in a short amount of time, the threat can be mitigated by restricting the changes of the capacity bid over time; for example, a prover can only change the capacity every n rounds by m folds (where n, m are parameters, possibly adjusted via, e.g., community votes in a DAO).

Sybil attacks. A malicious prover could pose as multiple actors (provers or users) and mount *Sybil attacks*: a prover may create several Sybil provers or submit "fake" user tasks [9]. In Proposition 6, we prove that the threat of such attacks is limited. Particularly, a prover splitting into multiple provers never harms the social welfare if such an attack is profitable. In the other case, where a prover generates fake user bids to gain more profit, we show that such attacks are only profitable when the bids of the other users and provers satisfy certain conditions. We show in Proposition 5 that if the prover has no information on the bids submitted by other parties, there always exists a possible configuration of these bids that would result in the prover having a loss. Therefore, this type of Sybil attack can be mitigated by concealing user and prover bids from the provers, i.e., making the auctions sealed-bid.

Collusion. Collusion can occur among different actors, e.g., users and provers or among a group of provers. In Propositions 7 and 8, we show that for the collusion to be definitely profitable, the colluders need to have full information of the other parties' bids. Therefore, these types of collusion can be intuitively mitigated using a similar approach as we use for Sybil attacks, i.e., hiding the bids of other parties from provers. Still, our protocol cannot properly circumvent all collusion (e.g., all provers can collude with each other and form a monopoly), but it is common for mechanism designs to disregard collusion [12, 13].

2 Related Work

Commercial mechanisms for ZKP markets. Several commercial platforms suggested ZKP market mechanisms without formal analysis. Gevulot [11] uses

Table 1: Summary of prover market designs.

Protocol	User Payment	Prover Selection	Payment to Provers
Proofφ	second price	lowest price	second price × allocated capacity
=nil;	limit order book	limit order book	limit order book
Taiko	posted price	random selection	prover-specified payment
Scroll	first price	random selection	partially subsidized
Gevulot	posted price	random selection	posted price × allocated capacity

posted prices to determine the set of provers (i.e., the same predetermined fee per task is paid to selected provers); Taiko [26] lets provers set their prices; Scroll [29] selects and partially subsidizes the lowest-cost provers and pays them with their bid; =nil; [19] facilitates price discovery by maintaining a limit order book for each circuit with buy orders from users and sell orders from provers. An order is executed when the buying and selling prices meet. In Table 1, we list the protocols across three dimensions: how provers are selected ("Prover Selection"), how user payments are determined ("User Payment"), and how the profit of provers is decided ("Payment to Provers").

Other computation outsourcing mechanisms. Thyagarajan et al. [28] propose OpenSquare, a Verifiable Delay Function (VDF) market that incentivizes maximum server participation, which is resistant to censorship and single-point failures. In a broader sense, ZKP markets are computation outsourcing markets. In V3rified [12], the fee mechanism for computational tasks is categorized into revelation ones (i.e., an auction where provers bid their costs) and non-revelation ones (the client posts its task with a given fee) and characterize their power and limitations. The protocols in V3rified cannot be directly applied to our model, as they are tailored for multiple provers proving one user task.

Double auctions. Our setting is of a two-sided market: on the demand side, users submit transactions that require proving, while on the supply side, provers provide proving capacity. Double auctions are commonly used to coordinate trade in such markets, i.e., to choose which agents get to trade and at what prices. While there is some overlap between the "traditional" setting explored by auction theory literature and the ZKP market setting, we note that the latter introduces new challenges: provers may collude, either amongst themselves or with users, and provers have a capacity parameter to bid. McAfee [18] presents a mechanism that is incentive compatible, individually rational, and budget-balanced for unit-demand buyers and unit-supply sellers (i.e., each buyer wishes to purchase a single item, and each seller offers a single item for sale). Huang, Scheller–Wolf, and Sycara [13] show a mechanism that is also incentive compatible, individually rational, and budget-balanced for multi-unit buyers and sellers (i.e., buyers may want multiple items, and suppliers may sell multiple items). However, their work assumes public capacity information and no collusion, and they are not a direct generalization of our mechanism to multi-item buyers. Although outside the scope of our work, we refer readers interested in a broad review of the literature

to the survey of Parsons, Rodriguez-Aguilar, and Klein [21], which covers a variety of auction formats, including double auctions.

3 Model

3.1 ZKP Market

In a ZKP market, user-specified ZKP generation tasks are outsourced to specialized parties (i.e., provers) for a price. The market is specified by a core mechanism and a system-level protocol. We now elaborate on these components.

Roles. We use n to denote the number of users in the market, and each user has a ZKP generation task. Each user task has a value f that the user is willing to pay to complete the task. There are N provers in the market willing to complete ZKP generation tasks for rewards. Each prover has a ZKP generation capacity s, i.e., the number of tasks it can complete within a fixed time window, and a unit cost p, i.e., its cost to complete one ZKP task. Note that in our model, we assume that each prover's total cost is the sum of the costs of all its tasks. Besides users and provers, a coordinator (or auctioneer) is responsible for collecting bids from users and provers and executing the market system and mechanism. For instance, when applied to ZK-Rollups, the centralized sequencer can be the auctioneer.

Core market mechanism. In each round, the market mechanism selects a set of user tasks that can be proven and an allocation of the tasks to the provers. Additionally, the core mechanism determines the amount of fee that needs to be collected from each user and determines the payment to each prover.

Market system. The market system specifies the additional steps and requirements for users and provers during the execution of the market mechanism. An example is that the provers deposit some collateral before they are eligible to submit bids in the market. The goal of the market system is to work in synergy with the market mechanism to secure the market against security threats.

Utilities. Here we introduce the utility of the market participants. We assume that all parties are myopic, i.e., they are only concerned with their utility within one round of market. We further assume that the auctioneer is trusted and therefore does not have a utility to maximize. For a user whose valuation of its task is f and pays f', if the task is selected, its utility is $f - f'$; otherwise, if the task is not selected, its utility is $-f'$. For a prover whose unit cost is p, and is paid w and allocated s' tasks, then its utility is $w - s' \cdot p$.

3.2 Threat Model

Both users and provers can act strategically and deviate from the protocol arbitrarily. The attacks on the market may occur in the following forms.

Incorrect or missing ZKPs. Provers may fail to generate correct ZKPs in the allotted time, whether intentionally or otherwise.

Misreporting bids. Both users and provers may not bid truthfully. If any actor can benefit from misreporting its bid, we consider this as an attack.

Sybil attacks. Besides misreporting bids, a user or a prover may pose as multiple actors and submit fake bids. Sybil attacks can occur when a prover submits bids as fake users or a prover submits bids as fake provers.

Collusion. Users and provers may collude to bid strategically to increase their joint utility. Not all forms of collisions harm players outside the coalition, but those that do are considered a security threat.

Remark 1. Previous work on TFMs typically adopted a non-Bayesian perspective, including when analyzing different possible threats, i.e., misreporting values, Sybil attacks, and collusion [4, 10, 1, 22]. In practice, blockchain actors such as Bitcoin miners who invest resources to receive rewards may prefer to minimize the financial risk entailed in their operation, with prior work seeing this as an explanation for the popularity of mining pools. Therefore, we follow previous work that modeled actors as risk-averse [7, 5, 23, 32], who will deviate from the honest behavior only if it would certainly improve their payoffs [4]. We note that alternative models could lead to interesting future work (see Section 6).

3.3 Desiderata

Following prior work [22, 18], we devise a desiderata for ZKP market mechanisms.

Budget balance. In a ZKP market, the fees collected from users in each round should cover the payment to provers, a property called *budget balance*.

Definition 1 (Budget Balance (BB)). *A mechanism is* budget-balanced *if the fees collected from the users are no less than the sum of payment to provers.*

Incentive compatibility. Our mechanism should incentivize actors to truthfully report their values, i.e., it should satisfy Definition 2. In Definitions 3 and 4 we provide the corresponding definitions for users and provers, respectively.

Definition 2 (Dominant Strategy Incentive Compatibility (DSIC)). *A mechanism is dominant strategy incentive compatible if it is always best for each participant to bid its true valuation.*

Definition 3 (User DSIC (UDSIC)). *A mechanism UDSIC if the mechanism is DSIC for users.*

Definition 4 (Prover DSIC (PDSIC)). *A mechanism is PDSIC if it is DSIC for provers to bid their costs honestly.*

Here we note that with PDSIC, it is still possible that a prover may misreport its capacity and get extra utility.

Sybil proofness. Provers submitting bids under fake identities should not negatively impact social welfare if creating Sybils is profitable. Recall that provers can bid under fake prover or user identities. We define them respectively as *prover Sybil proofness* (Definition 5) and *user Sybil proofness* (Definition 6).

Definition 5 (Prover Sybil Proofness). *A mechanism is prover-Sybil-proof if other parties' utilities are not reduced when any prover profits from creating prover Sybils.*

Definition 6 (User Sybil Proofness). *A mechanism is user-Sybil-proof if others' utilities are not reduced when any prover profits from creating user Sybils.*

Collusion resistance. A ZKP market mechanism is collusion resistant if a coalition of provers and users (Definition 7) or a coalition of provers (Definition 8) results in higher joint utility and harms the utility of other parties.

Definition 7 (User Collusion Resistance). *A ZKP market mechanism is user collusion resistant if a coalition of provers and users cannot simultaneously result in higher joint utility and harm the utility of other parties.*

Definition 8 (Prover Collusion Resistance). *A ZKP market mechanism is prover collusion resistant if a coalition of provers cannot simultaneously result in higher joint utility and harm the utility of other parties.*

4 Core Mechanism of *Prooφ*

In this section, we analyze the game-theoretic properties of the core mechanism (as specified in Section 3.3): we prove that it satisfies Budget Balance (Proposition 1), UDSIC (Proposition 2), and PDSIC (Proposition 3). In the interest of space, we defer some of the proofs to Appendix A.

Budget balance. A ZKP market mechanism is budget-balanced if the fees collected from all users are no less than the total payments to the provers.

Proposition 1. *The Prooφ mechanism is BB.*

Proof. According to the *Prooφ* mechanism, each allocated task is charged $f_{\bar{S}_{\ell}+1}$, and each allocated prover is paid the unit price $p_{\ell+1}$. Since the mechanism guarantees that $f_{\bar{S}_{\ell}+1} \geq p_{\ell+1}$, it must be budget-balanced. □

User incentive compatibility. A ZKP market mechanism is UDSIC if the mechanism is DSIC for users (Definition 3). In Proposition 2, we prove that the core mechanism of *Prooφ* is UDSIC.

Proposition 2. *The Prooφ mechanism is UDSIC.*

Proof. We show that for any user, bidding differently than its valuation is not profitable. Suppose user i changes its bid from f_i to f. We re-label the modified user bids as $f'_1 \geq \cdots \geq f'_n$. Similarly, we use primed variables throughout the proof to denote variables after user i changes its bid, and those without primes denote variables before that. I.e., ℓ' is the number of allocated provers and $\bar{S}'_{\ell'}$ the allocated number of tasks after i's change. Note that prover bids do not change by assumption, so $p'_j = p_j$ and $\bar{S}'_j = \bar{S}_j$ for all $j = 1, \cdots, N$.

We say a task has *rank* i if its fee is the i-th highest among all tasks. We denote the rank of user i's task before and after the deviation with $R(i)$ and $R'(i)$, respectively. We now discuss the utility of user i in two cases: when the task is allocated before the change (i.e., $R(i) \in [1, \bar{S}_\ell]$), and when it is not.

Case 1: User i is allocated before changing the bid. By the assumption that user i's task is allocated before deviation, the fee f_i is at least $f_{\bar{S}_\ell+1}$, i.e., $f_i \geq f_{\bar{S}_\ell+1}$. By the definition of our mechanism, $p_{\ell+1} \leq f_{\bar{S}_\ell+1}$ (as shown in Fig. 1) We now consider two sub-cases, based on whether $f < f_{\bar{S}_\ell+1}$ or not, i.e., whether the modified fee is less than the "second" price before deviating.

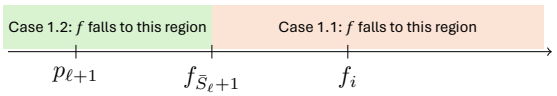


Fig. 1: Two of the sub-cases examined in the proof of Proposition 2.

If $f \geq f_{\bar{S}_\ell+1}$, we show that the task of user i is still allocated, and the second price does not change. Otherwise, if $f < f_{\bar{S}_\ell+1}$, we show that the task of user i is not allocated. In both, the utility of user i does not increase.

Case 1.1: $f \geq f_{\bar{S}_\ell+1}$. We now show that in this case, user i is still allocated and the payment $f'_{\bar{S}'_{\ell'}+1}$ remains the same as $f_{\bar{S}_\ell+1}$.

Firstly, we show that $R'(i) \leq \bar{S}_\ell$. By assumption ($f \geq f_{\bar{S}_\ell+1}$ and $f_i \geq f_{\bar{S}_\ell+1}$), tasks with a rank greater than $\bar{S}_\ell$ before the deviation will have the same rank after the deviation. Figure 2 shows an intuitive example.

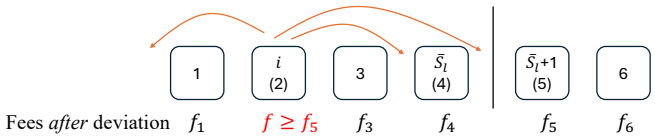


Fig. 2: Example used in case 1.1 to show $R'(i) \leq \bar{S}_\ell$. Labels show the ranks of tasks before i's deviation. After the deviation, i's new rank $R'(i) \leq 4$, because $f \geq f_{\bar{S}_\ell+1}$.

In this example, $i = 2$ and $\bar{S}_\ell = 4$. After user i changes its bid such that $f \geq f_{\bar{S}_\ell+1}$, tasks with rank 5 and 6 will still have the same rank, so $R'(i)$ can only be in $[1, 4]$. In general, it follows that $R'(i) \leq \bar{S}_\ell$.

Secondly, we prove that $\ell' = \ell$, by showing that $\ell' \geq \ell$ and $\ell' \leq \ell$. To see that $\ell' \geq \ell$, recall that $\ell \stackrel{\text{def}}{=} \arg\max_j \{p_{j+1} \leq f_{\bar{S}_j+1}\}$, and $\ell' \stackrel{\text{def}}{=} \arg\max_j \{p'_{j+1} \leq f'_{\bar{S}'_j+1}\}$. As argued previously, $f'_{\bar{S}_\ell+1} = f_{\bar{S}_\ell+1}$ and $R'(i) \leq \bar{S}_\ell$ (c.f. Fig. 2). In addition, note that $p'_{\ell+1} = p_{\ell+1}$ and $\bar{S}'_\ell = \bar{S}_\ell$ since prover bids are not changed,

indicating $f'_{\bar{S}_\ell+1} = f'_{\bar{S}'_\ell+1}$. It follows then $p'_{\ell+1} = p_{\ell+1} \leq f_{\bar{S}_\ell+1} = f'_{\bar{S}_\ell+1} = f'_{\bar{S}'_\ell+1}$. Since ℓ' is the maximum of all such ℓ's, we have $\ell' \geq \ell$.

To see $\ell' \leq \ell$, consider any $\ell^* > \ell$. We have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ by the definition of ℓ. Note that $f'_{\bar{S}_{\ell^*}+1} = f_{\bar{S}_{\ell^*}+1}$, since $\bar{S}_{\ell^*}+1 > \bar{S}_\ell$ by the definition of $\bar{S}$, and that tasks with a rank greater than $\bar{S}_\ell$ will have the same rank after the deviation (as argued previously). Further recall that $p'_{\ell^*+1} = p_{\ell^*+1}$ and $\bar{S}_{\ell^*} = \bar{S}'_{\ell^*}$ since prover bids are not changed. It follows that $p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}$. Since $p'_{\ell^*+1} > f'_{\bar{S}'_{\ell^*}+1}$ for any $\ell^* > \ell$, ℓ must be at least maximum, i.e., $\ell' \leq \ell$.

In summary, $\ell' = \ell$. We now can derive $\bar{S}'_{\ell'}$ the number of allocated tasks after the deviation: $\bar{S}'_{\ell'} = \bar{S}'_\ell = \bar{S}_\ell$. Since $R'(i) \leq \bar{S}_\ell = \bar{S}'_{\ell'}$ as argued previously, task i remains allocated after the deviation. The payment after deviating is $f'_{\bar{S}'_{\ell'}+1} = f'_{\bar{S}_\ell+1} = f_{\bar{S}_\ell+1}$, as before. Thus, the utility of user i remains the same.

The proof for case 1.2 (when $f < f_{\bar{S}_\ell+1}$, user i's task will not be allocated after the user changes its bid) is similar to that for case 1.1; the proof for case 2 is more involved as it incorporates more sub-cases. □

Prover incentive compatibility. Considering the prover side, recall that a ZKP market is PDSIC if the mechanism is DSIC for the provers to bid their costs honestly. We show that the *Proo*φ mechanism is PDSIC.

Proposition 3. *The Proo*φ *mechanism is PDSIC.*

Proof. We show that it is not profitable for any prover to bid other than its valuation. Recall that the users' bids satisfy $f_1 \geq \cdots \geq f_n$, and the provers' bids satisfy $p_1 \leq \cdots \leq p_N$. We assume that the prover j changes its bid from p_j to p, and suppose after the bid change, the new user bids are $f'_1 \geq \cdots \geq f'_n$, and the new prover bids are $p'_1 \leq \cdots \leq p'_N$. (All notations with primes denote the variables after prover j changes its bid, but note that the user bids are not changed, i.e., $f'_i = f_i$ for all $i = 1, \cdots, n$.) Let ℓ' be the number of allocated provers is and $\bar{S}'_{\ell'}$ be the number of allocated tasks.

A prover is said to have *rank* j if its cost is the j-th lowest among all provers. We denote the rank of prover j before and after changing its bid with $R(j)$ and $R'(j)$. We now discuss the utility change of prover j in two cases: when the prover is allocated before the change (i.e., $R(j) \leq \ell$), and when it is not.

Case 1: Prover j is allocated before changing the bid. By the assumption that prover j is allocated, the cost p_j is at most $p_{\ell+1}$, i.e., $p_j \leq p_{\ell+1}$. By the mechanism we have $p_{\ell+1} \leq f_{\bar{S}_\ell+1}$. The proof proceeds in two cases depending on whether $p > p_{\ell+1}$, i.e., the cost p is greater than the "second-price" of the provers. If $p \leq p_{\ell+1}$, we show that prover j is still allocated, and the second price does not change. Otherwise, we prover that prover j is not allocated. In either case, prover j's utility does not increase.

Case 1.1: $p \leq p_{\ell+1}$**.** We now show that in this case, prover j will still be allocated and the payment $s_j \cdot p'_{\ell'+1}$ remains the same as $s_j \cdot p_{\ell+1}$.

First, we show that $R'(j) \leq \ell$. By assumption, we have $p \leq p_{\ell+1}$ and $p_j \leq p_{\ell+1}$, thus provers with rank greater than ℓ before prover j changes its bid will have the same rank after it changes its bid. It follows that $R'(j) \leq \ell$.

Second, we show that $\ell' = \ell$, as $\ell' \geq \ell$ and $\ell' \leq \ell$. Recall that by definition, $\ell = \arg\max_k \{p_{k+1} \leq f_{\bar{S}_k+1}\}$, and $\ell' = \arg\max_k \{p'_{k+1} \leq f'_{\bar{S}'_k+1}\}$. To see that $\ell' \geq \ell$, note that $\bar{S}'_\ell = \bar{S}_\ell$ and $p'_{\ell+1} = p_{\ell+1}$, because the provers with rank greater than ℓ before prover j changes its bid will have the same rank after prover j changes its bid (as argued previously). Further recall that $f'_{\bar{S}_\ell} = f_{\bar{S}_\ell}$ since user bids are not changed. It follows that $p'_{\ell+1} = p_{\ell+1} \leq f_{\bar{S}_\ell+1} = f'_{\bar{S}_\ell+1} = f'_{\bar{S}'_\ell+1}$. So, $\ell' \geq \ell$ by the definition of ℓ'.

To see $\ell' \leq \ell$, consider any $\ell^* > \ell$, and $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ by the definition of ℓ. Because the provers with rank greater than ℓ before prover j changes its bid will have the same rank after prover j changes its bid (as argued previously), $p'_{\ell^*+1} = p_{\ell^*+1}$ and $\bar{S}'_{\ell^*} = \bar{S}_{\ell^*}$. Note that $f'_{\bar{S}_{\ell^*}+1} = f_{\bar{S}_{\ell^*}+1}$ since user bids are not changed. It follows that $p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}$. Since $p'_{\ell^*+1} > f'_{\bar{S}'_{\ell^*}+1}$ for any $\ell^* > \ell$, $\ell' \leq \ell$ by the definition of ℓ'. In summary, $\ell' = \ell$.

We now have the allocation after prover j changes its bid: $p'_{\ell'+1} = p'_{\ell+1} = p_{\ell+1}$. Therefore, prover j will still be allocated. Moreover, since the payment to of prover j is $p'_{\ell'+1} \cdot s_j = p_{\ell+1} \cdot s_j$, the utility of prover j will remain the same.

The proof of case 1.2 (when $p > p_{\ell+1}$, prover j will not be allocated after it changes its bid) and the proof of case 2 are more involved. □

5 Implementing *Proo*φ in an ZKP Market

In the previous section, we have shown that *Proo*φ has desirable game-theoretic properties. However, the core mechanism alone leaves certain security threats open. In this section, we present system-level designs to mitigate them. As before, missing proofs are given in Appendix A.

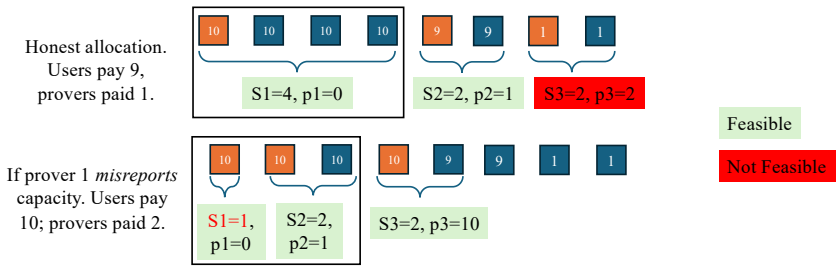
5.1 Missing or Incorrect ZKPs

It is possible that an allocated prover does not generate the required ZKPs. To address this concern, we require each prover to deposit collateral when joining the protocol. The collateral is used to refund users whose tasks were not generated on time. Specifically, suppose the market designer sets a public "refund" limit for user tasks that are not completed, up to a value of $\bar{p}$. This effectively serves as an upper bound on user valuations: if the refund $\bar{p}$ seems too low to some users, they can choose not to send their task to the ZKP market. Then, a prover with capacity s needs to maintain a deposit of at least $\bar{p} \cdot s$ when it bids (s, p). Note that $p < \bar{p}$. If the prover does not finish the allocated task in time, its deposit will be confiscated (causing it to lose $s \cdot \bar{p}$ utility) and used to refund $\bar{p}$ to each affected user, which is no less than the fees they paid. We note that this way of setting the collateral matches industry-adopted practices such as [25].

5.2 Misreporting Capacity

Proposition 3 proved that provers are incentivized to bid their costs honestly. However, a prover may misreport its *capacity*, as demonstrated in Example 2.

Example 2. We continue with Example 1, where when all parties bid honestly, $\ell = 1$ and prover 1 gets $(1-0) \times 4 = 4$ utility. However, when prover 1 bids capacity 1 instead of 4, $\ell = 2$ and prover 1 gets $(10-0) \times 1 = 10$ utility. The allocation is shown pictorially as follows.



This example suggests that reporting a lower capacity in some cases can render more provers feasible, increasing the prover's payment. Worst yet, fewer users are allocated in the market, and those allocated must pay more. However, in Proposition 4, we observe that the above attack is profitable only under specific market conditions. The mitigation we propose is based on this observation.

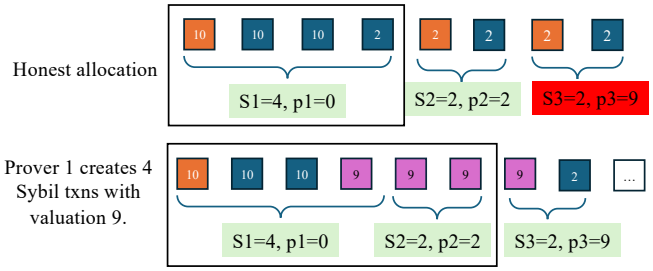
Proposition 4. *When a prover $j < N$ submits a smaller capacity bid than its true capacity, there will always be some prover and user bids that result in the prover receiving lower utility compared to if it had bid honestly, assuming all other provers are bidding honestly.*

Mitigation: Fixing capacity. As the profitability of the above attack depends on user bids, one mitigation is to have provers submit their capacity bids before users. The uncertainty of user bids creates a risk for provers. Another approach is to fix prover capacity for several rounds, to prevent provers from strategically adjusting capacities based on market conditions. In practice, a prover's capacity is unlikely to change drastically in a short amount of time.

5.3 User Sybil Proofness

We show in Example 3 the core mechanism alone is not user Sybil proof, i.e., provers can profit by creating fake user bids.

Example 3. Suppose there are 8 user tasks with values $(10, 10, 10, 2, 2, 2, 2, 2)$, and 3 provers with capacities s_i and costs p_i $(s_1, p_1) = (4, 0), (s_2, p_2) = (2, 2), (s_3, p_3) = (2, 9)$. When all parties bid honestly, we have $\ell = 1$ and prover 1 can make $(2-0) \times 4 = 8$. However, prover 1 can create 4 Sybil tasks with a bid of 9. Then we have $\ell = 2$, and prover 1 will make $\underbrace{(9-0) \times 4}_{\text{prover reward}} - \underbrace{9 \times 3}_{\text{task cost}} = 9$.



Intuitively, creating high-paying tasks can sometimes render more provers feasible, increasing all provers' utility. This can also reduce user utility. We observe that creating user Sybils benefits the attacker only under specific conditions and that the attack is possible only if the malicious prover knows all user bids.

Proposition 5. *When a prover $j < N$ submits Sybil user bids, there will always be some prover and user bids that result in prover j receiving lower utility compared to if it had acted honestly, assuming all other provers are honest.*

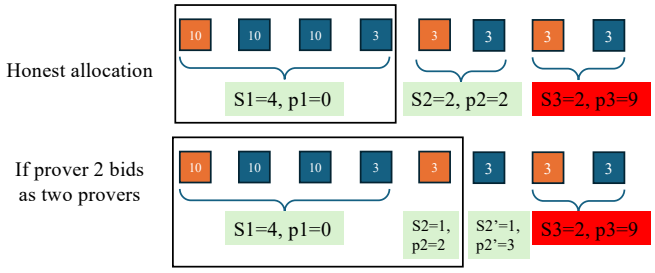
Mitigation: hiding bids. Based on this observation, the above attacks can be mitigated by *encrypting all user and prover bids* to hide the necessary information required by profitable Sybil attacks, i.e., making the auctions sealed-bid.

Remark 2. If provers are risk-averse and thus would try to avoid worst-case outcomes (as assumed by prior work, see Remark 1), Proposition 5 suffices to thwart the threat of a deviating prover sending Sybil user bids. One may consider a Bayesian setting, where different solutions may be needed to ensure that such deviations lower an attacker's *expected* utility (see Section 6).

5.4 Prover Sybil proofness

Recall that a mechanism is prover Sybil proof if no prover finds it profitable to submit bids as multiple provers. We show that the *Prooφ* mechanism is not prover Sybil proof, as demonstrated in Example 4. Intuitively, prover ℓ may split into multiple provers and get allocated some proof generation capacity.

Example 4. Suppose there are 8 user tasks, where 3 of them have value 10, and 5 of them have value 3. Also, suppose there are 3 provers in the ZKP market. The capacities s_i and costs p_i of the provers are $(s_1, p_1) = (4, 0), (s_2, p_2) = (2, 2), (s_3, p_3) = (2, 9)$. When all parties are bidding honestly, we have $\ell = 1$, and prover 2 will get zero utility. Instead, prover 2 can split into two provers, one with bid $(1, 2)$ and the other with bid $(1, 3)$. Then ℓ will increase to 2, and the split prover with bid $(1, 2)$ will be allocated and get 1 utility.



We analyze prover Sybil attacks and show in Proposition 6 that profitable attacks contribute to social welfare and thus do not harm users.

Proposition 6. *Suppose a prover j with capacity s_j splits into p Sybil provers, where the k-th split has cost bid $p_{j,k}$ and capacity bid $s_{j,k}$, with $\sum_k s_{j,k} = s_j$. Then if the Sybil attack is profitable, the social welfare will increase.*

5.5 Collusion

A mechanism is vulnerable to collusion if a coalition of provers and/or users finds it profitable to coordinate and bid strategically. In this paper, we focus on two types of collusion in a ZKP market; one is a prover colluding with a user, and the other is a set of provers colluding with each other. We note that potential collusion among market agents is typically not discussed in related work [12, 13]. Here, we consider the possibility of collusion and show that it is not much of a threat with proper implementation of the mechanism. Specifically, we assume a coalition of colluding participants can freely communicate; i.e., colluders have private communication channels with each other, while honest actors not in the coalition are independent and do not communicate with any other party.

In case a prover colludes with a user, we show in Proposition 7 that this benefits the attacking coalition only under specific conditions.

Proposition 7. *When a prover and a user collude by deviating from their honest bids, there will always be other prover and user bids that cause the coalition to receive lower joint utility, assuming all other parties are honest.*

In the proof, we show in a case-by-case manner that all profitable deviations require the coalition to know the bids of the other parties.

Following this result, the previous mitigation (hiding bids) also mitigates such collusion by creating uncertainty about profitability. I.e., a prover-user coalition cannot be sure that they will profit by bidding strategically.

When a set of provers form a coalition and bid strategically, if the provers in the coalition have the same cost then this is equivalent to a single prover splitting into prover Sybils. As in Proposition 6, such coalitions do not harm the ZKP market. On the other hand, we show in Proposition 8 that two provers colluding can only benefit the provers in the coalition when the other parties have specific bids, which again is mitigated by encrypting user and prover bids.

Proposition 8. *When two provers with different costs collude by changing their bids, there will always be other prover and user bids that result in the coalition receiving lower joint utility compared to if they had bid honestly, assuming all other parties are bidding honestly.*

6 Conclusions and Future Work

We have presented *Proo*φ, a mechanism designed to address the demands of ZKP markets, where proofs are generated in batches. *Proo*φ consists of a core auction mechanism that matches low-cost provers with high-value user tasks. We formally analyze the mechanism and show that it ensures incentive compatibility for both users and provers and that it is budget-balanced. We also introduce system-level designs such as slashing, fixing prover capacity bids over multiple auction rounds, and encrypting bids, to address security challenges that are not handled by the core mechanism alone. We analyze how our measures mitigate security concerns including untruthful capacity bids, Sybil attacks, and collusion.

Future work. Following the literature [4, 10, 1, 22], we consider a non-Bayesian setting. One possible direction for future work is to extend the model to a Bayesian one where actors may deviate to increase *expected* payoffs. For example, provers can send Sybil user bids to "simulate" a reserve price for users, thus potentially increasing revenue. Analyzing this manipulation could be interesting from the perspective of the adversaries, who should surmount several practical difficulties. First, attackers should have robust estimates for the distribution of user valuations and the distribution of the number of users. Second, in a permissionless setting, price gouging may not be profitable in the long run: it can allow new actors to offer lower prices while still making a profit [14]. Moreover, long-term dynamics may result in lower-paying transactions accumulating until the pent-up demand becomes profitable for provers [20].

Acknowledgements

The authors wish to thank Matter Labs for research discussions and support with several aspects of this work. Wenhao Wang was supported in part by the ZK Fellowship from Matter Labs.

References

[1] Maryam Bahrani et al. "Transaction Fee Mechanism Design in a Post-MEV World". In: *6th Conference on Advances in Financial Technologies (AFT '24)*. 2024, 29:1–29:24. DOI: 10.4230/LIPIcs.AFT.2024.29.

[2] Foteini Baldimtsi et al. "zklogin: Privacy-preserving blockchain authentication with existing credentials". In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 3182–3196. DOI: 10.1145/3658644.3690356.

[3] Binyi Chen et al. "Hyperplonk: Plonk with linear-time prover and high-degree custom gates". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023. DOI: 10.1007/978-3-031-30617-4_17.

[4] Hao Chung and Elaine Shi. "Foundations of transaction fee mechanism design". In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms*. 2023. DOI: 10.1137/1.9781611977554.ch150.

[5] Lin William Cong et al. "Decentralized Mining in Centralized Pools". In: *The Review of Financial Studies* 34.3 (Apr. 2020), pp. 1191–1235. ISSN: 0893-9454. DOI: 10.1093/rfs/hhaa040.

[6] Jens Ernstberger et al. "zk-Bench: A Toolset for Comparative Evaluation and Performance Benchmarking of SNARKs". In: *Security and Cryptography for Networks*. Ed. by Clemente Galdi and Duong Hieu Phan. Cham: Springer Nature Switzerland, 2024, pp. 46–72. ISBN: 978-3-031-71070-4. DOI: 10.1007/978-3-031-71070-4_3.

[7] Ben Fisch et al. "Socially Optimal Mining Pools". In: *Web and Internet Economics*. Ed. by Nikhil R. Devanur and Pinyan Lu. Cham: Springer International Publishing, 2017, pp. 205–218. DOI: 10.1007/978-3-319-71924-5_1.

[8] Ariel Gabizon et al. "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge". In: *Cryptology ePrint Archive* (2019). URL: https://ia.cr/2019/953.

[9] Yotam Gafni and Moshe Tennenholtz. "Optimal Mechanism Design for Agents with DSL Strategies: The Case of Sybil Attacks in Combinatorial Auctions". In: *Electronic Proceedings in Theoretical Computer Science* 379 (2023), pp. 245–259. ISSN: 2075-2180. DOI: 10.4204/eptcs.379.20.

[10] Yotam Gafni and Aviv Yaish. "Barriers to Collusion-resistant Transaction Fee Mechanisms". In: *Proceedings of the 25th ACM Conference on Economics and Computation*. EC '24. 2024. DOI: 10.1145/3670865.3673469.

[11] *Gevulot Docs: Economics*. https://docs.gevulot.com/gevulot-docs/network/fees.

[12] Tiantian Gong et al. *V3rified: Revelation vs Non-Revelation Mechanisms for Decentralized Verifiable Computation*. 2024. arXiv: 2408.07177 [cs.GT].

[13] Pu Huang et al. "Design of a Multi–Unit Double Auction E–Market". In: *Computational Intelligence* 18.4 (2002), pp. 596–617. DOI: 10.1111/1467-8640.t01-1-00206.

[14] Gur Huberman et al. "Monopoly without a Monopolist: An Economic Analysis of the Bitcoin Payment System". In: *The Review of Economic Studies* (2021). DOI: 10.1093/restud/rdab014.

[15] *Introducing Succinct Network: The Protocol for Programmable Truth*. https://blog.succinct.xyz/succinct-network/.

[16] Aniket Kate et al. "Constant-Size Commitments to Polynomials and Their Applications". In: *Advances in Cryptology - ASIACRYPT*. Springer, 2010.

[17] *Layer 2 Activity*. https://l2beat.com/scaling/activity.

[18] R.Preston McAfee. "A dominant strategy double auction". In: *Journal of Economic Theory* (1992). DOI: 10.1016/0022-0531(92)90091-u.
[19] *=nil; Proof Market*. https://docs.nil.foundation/proof-market/market/economics.
[20] Noam Nisan. *Serial Monopoly on Blockchains*. 2023. arXiv: 2311.12731.
[21] Simon Parsons et al. "Auctions and bidding: A guide for computer scientists". In: *ACM Comput. Surv.* (2011). DOI: 10.1145/1883612.1883617.
[22] Tim Roughgarden. "Transaction Fee Mechanism Design". In: *J. ACM* 71.4 (2024). ISSN: 0004-5411. DOI: 10.1145/3674143.
[23] Tim Roughgarden and Clara Shikhelman. "Ignore the Extra Zeroes: Variance-Optimal Mining Pools". In: *Financial Cryptography and Data Security*. Springer, 2021, pp. 233–249. DOI: 10.1007/978-3-662-64331-0_12.
[24] Erel Segal-Halevi et al. "MUDA: a truthful multi-unit double-auction mechanism". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018. DOI: 10.1609/aaai.v32i1.11450.
[25] *Succinct Network: Prove the World's Software*. www.provewith.us.
[26] *Taiko Tokenomics*. https://github.com/taikoxyz/taiko-mono/blob/42bbc5/packages/protocol/docs/tokenomics_staking.md.
[27] *The Starknet Book*. https://book.starknet.io/title-page.html.
[28] Sri Aravinda Krishnan Thyagarajan et al. "Opensquare: Decentralized repeated modular squaring service". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021. DOI: 10.1145/3460120.3484809.
[29] *Transaction Fees on Scroll*. https://docs.scroll.io/en/developers/transaction-fees-on-scroll/.
[30] William Vickrey. "Counterspeculation, Auctions, and Competitive Sealed Tenders". In: *The Journal of Finance* 16.1 (1961). DOI: 10.2307/2977633.
[31] *Workflow of a zkSync Era transaction: from generation to finalization*. https://blog.quarkslab.com/zksync-transaction-workflow.html.
[32] Aviv Yaish and Aviv Zohar. "Correct Cryptocurrency ASIC Pricing: Are Miners Overpaying?" In: *5th Conference on Advances in Financial Technologies*. Vol. 282. 2023, 2:1–2:25. DOI: 10.4230/LIPIcs.AFT.2023.2.
[33] Fan Zhang et al. "Deco: Liberating web data using decentralized oracles for tls". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 1919–1938. DOI: 10.1145/3372297.3417239.

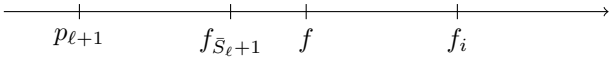
Appendix

A Proofs

A.1 Full Proof of Proposition 2

Proof. We need to show that it is not profitable for any user to bid other than its valuation. Recall that the users' bids satisfy $f_1 \geq \cdots \geq f_n$, and the provers' bids satisfy $p_1 \leq \cdots \leq p_N$. Suppose that f_i is user i's valuation of its task. We assume that user i changes its bid from f_i to f. After the bid change, the new user bids are $f'_1 \geq \cdots \geq f'_n$, and the new prover bids are $p'_1 \leq \cdots \leq p'_N$. The allocated number of provers are ℓ' and the allocated number of tasks are $\bar{S}'_{\ell'}$ Since no prover's bid has changed, we have $p'_j = p_j$ and $\bar{S}'_j = \bar{S}_j$ for all $j = 1, \cdots, N$. We discuss the utility change of user i depending on whether or not it is allocated, i.e., whether or not $i \in [1, \bar{S}_\ell]$. A task is said to have *rank* i if its fee is the i-th highest among all tasks. Let $R(i)$ and $R'(i)$ be the rank of user i's task before and after changing her bid. In each case, a figure is used to demonstrate the relation between variables, and there could be more possibilities.

– If $i \in [1, \bar{S}_\ell]$, i.e., task i is allocated, then we consider whether $f < f_{\bar{S}_\ell+1}$.
  - If $f \geq f_{\bar{S}_\ell+1}$, the concern is that user i can somehow decrease its payment while its task is still allocated after the bid change.



    We show that in this case task i is still allocated, while the payment $f'_{\bar{S}'_{\ell'}+1}$ remains the same.
    Firstly, we show that $R'(i) \leq \bar{S}_\ell$ after user i changes its bid to f. This is because $f \geq f_{\bar{S}_\ell+1}$ and $f_i \geq f_{\bar{S}_\ell+1}$, which indicates that tasks with rank greater than $\bar{S}_\ell$ before user i changes its bid will have the same rank after user i changes its bid. It follows that $R'(i) \leq \bar{S}_\ell$
    Secondly, we show that $\ell' = \ell$. We show this with $\ell' \geq \ell$ and $\ell' \leq \ell$. To see that $\ell' \geq \ell$, recall that $\ell = \arg\max_j \{p_{j+1} \leq f_{\bar{S}_j+1}\}$, and $\ell' = \arg\max_j \{p'_{j+1} \leq f'_{\bar{S}'_j+1}\}$. Note that $f'_{\bar{S}_\ell+1} = f_{\bar{S}_\ell+1}$, since $R'(i) \leq \bar{S}_\ell$. Further note that $p'_{\ell+1} = p_{\ell+1}$ and $\bar{S}'_\ell = \bar{S}_\ell$. It follows that

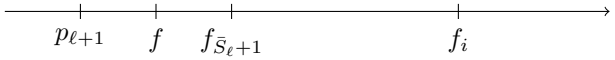
$$p'_{\ell+1} = p_{\ell+1} \leq f_{\bar{S}_\ell+1} = f'_{\bar{S}_\ell+1} = f'_{\bar{S}'_\ell+1}.$$

    So $\ell' \geq \ell$ by the definition of ℓ'. To see $\ell' \leq \ell$, for any $\ell^* > \ell$ we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ by the definition of ℓ. Note that $f'_{\bar{S}_{\ell^*}+1} = f_{\bar{S}_{\ell^*}+1}$, since $\bar{S}_\ell < \bar{S}_{\ell^*} + 1$ by the definition of $\bar{S}$, and tasks with rank greater than $\bar{S}_\ell$ before user i changes its bid will have the same rank after user i changes its bid (as argued previously). Further recall that $p'_{\ell^*+1} = p_{\ell^*+1}$ and $\bar{S}_{\ell^*} = \bar{S}'_{\ell^*}$. It follows that for any $\ell^* > \ell$,

$$p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}.$$

Therefore, $\ell' \leq \ell$ by the definition of ℓ'. Above all, $\ell' = \ell$.
We have $\bar{S}'_{\ell'} = \bar{S}'_{\ell} = \bar{S}_{\ell}$. Since the rank of task i is no more than $\bar{S}_{\ell}$, task i will still be allocated after user i changes its bid to f. Also, note that the payment of this task is $f'_{\bar{S}'_{\ell'}+1} = f'_{\bar{S}_{\ell}+1} = f_{\bar{S}_{\ell}+1}$, we have the utility of user i will remain the same after user i changes its bid.

- If $f < f_{\bar{S}_{\ell}+1}$, the concern is also that user i can somehow decrease its payment while its task is still allocated after it changes its bid to f.



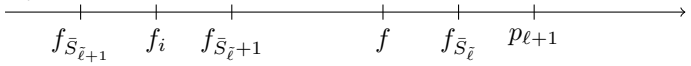
We show that in this case task i is not allocated, so user i cannot increase its utility after it changes its bid.
Firstly, we show that $R'(i) > \bar{S}_{\ell}$. This is because $f < f_{\bar{S}_{\ell}+1}$ and $f_i \geq f_{\bar{S}_{\ell}+1}$, which indicates that there will be at least $\bar{S}_{\ell}$ tasks with bids greater than f after user i changes its bid. It follows that $R'(i) > \bar{S}_{\ell}$.
Secondly, we show that $\ell' \leq \ell$. To see this, for all $\ell^* > \ell$ we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ by the definition of ℓ. Note that $f'_{\bar{S}_{\ell^*}+1} \leq f_{\bar{S}_{\ell^*}+1}$, since $f < f_i$ and when there is no increasing element in a sequence, element in the new sequence will be no more than the element with the same rank in the original sequence. Further recall that $p'_{\ell^*+1} = p_{\ell^*+1}$ and $\bar{S}_{\ell^*} = \bar{S}'_{\ell^*}$. It follows that

$$p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} \geq f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}.$$

Therefore, $\ell' \leq \ell$ by the definition of ℓ'.
We have $\bar{S}'_{\ell'} \leq \bar{S}'_{\ell} = \bar{S}_{\ell}$. Since $R'(i) > \bar{S}_{\ell}$, task i will not be allocated after user i changes its bid to f. Therefore, the utility of user i will become 0 after user i changes its bid.

– If $i \in [\bar{S}_{\ell} + 1, n]$, i.e., task i is not allocated, we consider the cases depending on the value of i. We have there exist some $\tilde{\ell} \geq \ell$ such that $i \in [\bar{S}_{\tilde{\ell}} + 1, \bar{S}_{\tilde{\ell}+1}]$ for some $\tilde{\ell} \geq \ell$, or $i > \sum_{j=1}^{N} s_j$.

- If $f < f_{\bar{S}_{\tilde{\ell}}}$, we argue that task i will not be included.

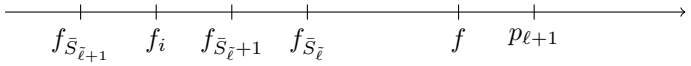


Firstly, we show that $R'(i) > \bar{S}_{\tilde{\ell}}$. This is because $f < f_{\bar{S}_{\tilde{\ell}}}$, which indicates that there are at least $\bar{S}_{\tilde{\ell}}$ tasks with bids greater than f after user i changes its bid. It follows that $R'(i) > \bar{S}_{\tilde{\ell}}$.
Secondly, we show that $\ell' \leq \tilde{\ell}$. To see this, for any $\ell^* > \tilde{\ell} \geq \ell$ we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ by the definition of ℓ. Then we show that $f_{\bar{S}_{\ell'+1}} \geq f'_{\bar{S}_{\ell'+1}}$. If $f \geq f_i$, since $\bar{S}_{\ell'+1} \geq i$ we have $f_{\bar{S}_{\ell'+1}} = f'_{\bar{S}_{\ell'+1}}$; if $f < f_i$, we have $f_{\bar{S}_{\ell'+1}} \geq f'_{\bar{S}_{\ell'+1}}$, since when there is no increasing element in a sequence, element in the the new sequence will be no more than the element with the same rank in the original sequence. It follows that $f_{\bar{S}_{\ell'+1}} \geq f'_{\bar{S}_{\ell'+1}}$. Further recall that $p'_{\ell^*+1} = p_{\ell^*+1}$ and $\bar{S}'_{\ell^*} = \bar{S}_{\ell^*}$. Therefore, for any $\ell^* > \tilde{\ell}$,

$$p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} \geq f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}.$$

So $\ell' \leq \tilde{\ell}$ by the definition of ℓ'. We have $\bar{S}'_{\ell'} \leq \bar{S}'_{\tilde{\ell}} = \bar{S}_{\tilde{\ell}}$. Since $R'(i) > \bar{S}_{\tilde{\ell}}$, task i will not be allocated after user i changes its bid to f. Therefore, the utility of user i will remain 0 after user i changes its bid.

- If $f \geq f_{\bar{S}_{\tilde{\ell}}}$, the concern is that user i can somehow get its task allocated will paying less than f_i.



We show that in this case either task i is not allocated or task i is charged at least f_i.
Firstly, we show that $\ell' \leq \tilde{\ell}$. To see this, for any $\ell^* > \tilde{\ell} \geq \ell$ we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ by the definition of ℓ. Then we show that $f_{\bar{S}_{\ell'+1}} \geq f'_{\bar{S}_{\ell'+1}}$. If $f \geq f_i$, since $\bar{S}_{\ell'+1} \geq i$ we have $f_{\bar{S}_{\ell'+1}} = f'_{\bar{S}_{\ell'+1}}$; if $f < f_i$, we have $f_{\bar{S}_{\ell'+1}} \geq f'_{\bar{S}_{\ell'+1}}$, since when there is no increasing element in a sequence, element in the the new sequence will be no more than the element with the same rank in the original sequence. It follows that $f_{\bar{S}_{\ell'+1}} \geq f'_{\bar{S}_{\ell'+1}}$. Further recall that $p'_{\ell^*+1} = p\ell^* + 1$ and $\bar{S}'_{\ell^*} = \bar{S}_{\ell^*}$. Therefore, for any $\ell^* > \tilde{\ell}$,

$$p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} \geq f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}.$$

So $\ell' \leq \tilde{\ell}$ by the definition of ℓ'.
We have $\bar{S}'_{\ell'} \leq \bar{S}'_{\tilde{\ell}} = \bar{S}_{\tilde{\ell}}$. If task i is allocated, it is charged $f'_{\bar{S}'_{\ell'}} \geq f'_{\bar{S}'_{\tilde{\ell}}}$. Since $f > f_i$, and when there is no decreasing element in a sequence, the element in the new sequence will be no less than the element with the same rank in the original sequence. So, $f'_{\bar{S}'_{\tilde{\ell}}} \geq f_{\bar{S}'_{\tilde{\ell}}} = f_{\bar{S}_{\tilde{\ell}}} \geq f_i$. It follows that the payment of task i is at least f_i if task i is allocated. Therefore, user i cannot get positive utility if it changes its bid to f.

Above all, the mechanism is UDSIC. □

A.2 Full Proof of Proposition 3

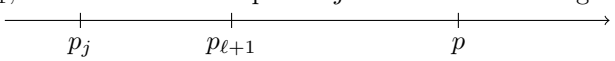
Proof. We prove it is unprofitable for any prover to bid other than its valuation. Recall that the users' bids satisfy $f_1 \geq \cdots \geq f_n$, and the provers' bids satisfy $p_1 \leq \cdots \leq p_N$. Suppose that p_j is prover n's valuation of its proof generation cost per unit capacity. We assume that the prover j changes its bid from p_j to p. After the bid change, the new user bids are $f'_1 \geq \cdots \geq f'_n$, and the new prover bids are $p'_1 \leq \cdots \leq p'_N$. The allocated number of provers is ℓ' and the allocated number of tasks is $\bar{S}'_{\ell'}$. Since no user's bid has changed, we have $f'_i = f_i$ for all $i = 1, \cdots, n$. We discuss the utility change of prover j depending on whether or not it is allocated, i.e., whether or not $j \in [1, \ell]$. A prover is said to have *rank* j if its cost is the j-th lowest among all provers. We denote the rank of prover j before and after changing its bid with $R(j)$ and $R'(j)$. Each case's figure demonstrates the relation between variables, but there could be more possibilities.

– If prover j is allocated ($j \in [1, \ell]$), consider possible relations between $p, p_{\ell+1}$.
  - If $p \leq p_{\ell+1}$, the concern is that prover j can increase its revenue while remaining allocated.

    We show that in this case prover j remains allocated, while the payment $p'_{\ell'+1} \cdot s_j$ stays the same, so changing the bid to p is not profitable.
    First, we show that $R'(j) \leq \ell$. This is because $p \leq p_{\ell+1}$ and $p_j \leq p_{\ell+1}$, which indicates that provers with rank greater than ℓ have the same rank before and after prover j changes its bid. Thus, $R'(j) \leq \ell$.
    Second, we prove that $\ell' = \ell$ by showing that both $\ell' \geq \ell$ and $\ell' \leq \ell$. Recall that $\ell = \arg\max_k \{p_{k+1} \leq f_{\bar{S}_k+1}\}$, and $\ell' = \arg\max_k \{p'_{k+1} \leq f'_{\bar{S}'_k+1}\}$. To see that $\ell' \geq \ell$, note that $\bar{S}'_\ell = \bar{S}_\ell$ and $p'_{\ell+1} = p_{\ell+1}$, since the provers with rank greater than ℓ before prover j changes its bid will have the same rank after prover j changes its bid. Further recall that $f'_{\bar{S}_\ell} = f_{\bar{S}_\ell}$. It follows that

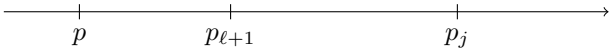
$$p'_{\ell+1} = p_{\ell+1} \leq f_{\bar{S}_\ell+1} = f'_{\bar{S}_\ell+1} = f'_{\bar{S}'_\ell+1}.$$

    So $\ell' \geq \ell$ by the definition of ℓ'. To see $\ell' \leq \ell$, we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ for all $\ell^* > \ell$ by the definition of ℓ. Note that $p'_{\ell^*+1} = p_{\ell^*+1}$ and $\bar{S}'_{\ell^*} = \bar{S}_{\ell^*}$ since the provers with rank greater than ℓ before prover j changes its bid will have the same rank after prover j changes its bid. Also recall that $f'_{\bar{S}_{\ell^*}+1} = f_{\bar{S}_{\ell^*}+1}$. It follows that: $p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}$. Therefore, $\ell' \leq \ell$ by the definition of ℓ'. Above all, $\ell' = \ell$.
    We have $p'_{\ell'+1} = p'_{\ell+1} = p_{\ell+1}$ as argued previously. Therefore, prover j will be allocated and its payment is $p'_{\ell+1} \cdot s_j = p_{\ell+1} \cdot s_j$. Therefore, prover j will not increase its utility after it changes its bid.
  - If $p > p_{\ell+1}$, the concern is that prover j can increase ℓ and get allocated.



    We prove by contradiction that in this case prover j is not allocated, thus getting zero revenue. Before that, we show $R'(j) > \ell$. This is because $p > p_{\ell+1}$ and $p_j \leq p_{\ell+1}$, which indicates that there will be at least ℓ provers with bids lower than p after prover j changes its bid. It follows that the rank of prover j is more than ℓ after prover j changes its bid. Suppose by contradiction that prover j is allocated, then $R'(j) \leq \ell'$, so $\ell' > \ell$. By the definition of ℓ', $p'_{\ell'+1} \leq f'_{\bar{S}'_{\ell'}+1}$. Since $R'(j) \leq \ell'$, then $\bar{S}'_{\ell'} = \bar{S}_{\ell'}$ and $p'_{\ell'+1} = p_{\ell'+1}$. Recall that $f'_{\bar{S}_{\ell'}+1} = f_{\bar{S}_{\ell'}+1}$. It follows that: $p_{\ell'+1} = p'_{\ell'+1} \leq f'_{\bar{S}'_{\ell'}+1} = f_{\bar{S}_{\ell'}+1}$. However, $p_{\ell'+1} > f_{\bar{S}_{\ell'}+1}$ by the definition of ℓ since $\ell' > \ell$, leading to a contradiction. Therefore, prover j is not allocated after changing its bid to p. Thus, prover j receives zero utility.

– If $j \in [\ell+1, N]$, we consider the cases depending on whether or not $p \geq p_{\ell+1}$.
  - If $p < p_{\ell+1}$, the concern is that prover j can somehow get allocated while being paid more than p_j.

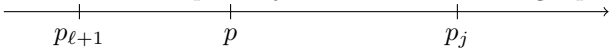


We show that in this case, prover j will either not be allocated or be paid at most p_j while being allocated.
First, we show that $R'(j) \leq \ell + 1$. This is because $p < p_{\ell+1}$ and $p_j \geq p_{\ell+1}$, which indicates that provers with rank strictly lower than $\ell+1$ before prover j changes its bid will have rank lower than $\ell + 1$ after prover j changes its bid, and that the prover $\ell + 1$ before prover j changes its bid will have rank $\ell + 2$ after prover j changes its bid. It follows that $R'(j) \leq \ell + 1$.
Secondly, we show that $\ell' < j$. By the definition of ℓ, we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ for all $\ell^* \geq j > \ell$. Note that $p'_{\ell^*+1} = p_{\ell^*+1}$ for all $\ell^* \geq j$ since prover j lowers its bid. Also recall that $f'_{\bar{S}_{\ell^*}+1} = f_{\bar{S}_{\ell^*}+1}$. It follows that for all $\ell^* \geq j$

$$p'_{\ell^*+1} = p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}.$$

Therefore, $\ell' < j$ by the definition of ℓ'.
Note that after prover j changes its bid to p, it is either allocated or not allocated. If prover j is not allocated, it gets zero utility. If prover j is allocated, we have $p'_{\ell'+1} \leq p'_j \leq p_j$. The last inequality holds because prover j lowers its bid, and when there is no increasing element in a sequence, any element in the new sequence is at most equal to the element with the same rank in the original sequence. Therefore, prover j at most earns $p_j s_j$ and gets zero utility. In both cases, prover j will not receive above zero utility.

- If $p \geq p_{\ell+1}$, the fear is that prover j can be allocated and get positive utility.



We show that in this case prover j cannot be allocated.
Firstly, we show that $R'(j) \geq \ell+1$. This is because $p \geq p_{\ell+1}$ and $p_j \geq p_{\ell+1}$, which indicates that provers with rank lower than $\ell + 1$ before prover j changes its bid will have rank lower than $\ell + 1$ after prover j changes its bid. It follows that $R'(j) \geq \ell + 1$.
Second, we show that $\ell' \geq \ell$. By the definition of ℓ, we have $p_{\ell^*+1} > f_{\bar{S}_{\ell^*}+1}$ for any $\ell^* > \ell$. Note that $p'_{\ell+1} \geq p_{\ell+1}$, $f'_{\bar{S}_{\ell^*}+1} = f_{\bar{S}_{\ell^*}+1}$. Thus, for any $\ell^* > \ell$

$$p'_{\ell^*+1} \geq p'_{\ell+1} \geq p_{\ell+1} > f_{\bar{S}_{\ell}+1} \geq f_{\bar{S}_{\ell^*}+1} = f'_{\bar{S}'_{\ell^*}+1}.$$

Therefore, $\ell' \geq \ell$ by the definition of ℓ'. Since $R'(j) > \ell$, it is not allocated. Therefore, the utility of prover j remains zero after changing its bid.

Above all, the mechanism is PDSIC □

A.3 Proof of Proposition 4

Proof. We discuss the utility of prover $j < N$. Suppose there are $\bar{S}_N$ tasks all with fee $f = p_{j+1}$, and $p_{j+1} > p_j$. When prover j is bidding honestly, $\ell = j$ and prover j would receive $s_j \cdot (p_{j+1} - p_j)$ utility. If prover j bids a lower capacity s'_j, still we have $\ell = j$ and prover j would receive $s'_j \cdot (p_{j+1} - p_j)$ utility, which is less than the utility when the prover bids honestly. □

A.4 Proof of Proposition 5

Proof. We discuss the utility of prover $j < N$. Suppose there are $\bar{S}_N$ tasks, all with fee $f = p_{j+1}$, and $p_{j+1} > p_j$. When prover j does not create Sybils, $\ell = j$ and prover j receives $s_j \cdot (p_{j+1} - p_j)$ utility. If prover j submits Sybil user bids, then prover j is always allocated, and will receive the same payment. If any of the Sybil bids are allocated, prover j will lose at least f utility. Therefore, in this case, prover j does not profit by submitting Sybil user bids.

□

A.5 Proof of Proposition 6

Proof. Suppose $p_{j,1} \leq p_{j,2} \leq \cdots \leq p_{j,r}$. After the Sybil attack, suppose the allocated number of provers is ℓ' and the allocated number of tasks is $\bar{S}'_{\ell'}$. Also suppose the new prover bids are $p'_1 \leq p'_2 \leq \cdots$ after the Sybil attack.

If $j \leq \ell$, such Sybil attacks are never profitable.

- If $p_{j,r} \leq p_{\ell+1}$, then $\ell' = \ell + r - 1$, thus prover j still gets the same utility.
- If there exists $q \leq r - 1$ such that $p_{j,q} \leq p_{\ell+1} < p_{j,q+1}$, we show that all the splits with index greater than q will not be allocated by contradiction. Assume by contradiction that the split with index $q + 1$ will be allocated. It follows that $p'_{\ell'+1} \geq p_{\ell+2}$. Recall that $\ell = \arg\max_k \{p_{k+1} \leq f_{\bar{S}_k+1}\}$, and $\ell' = \arg\max_k \{p'_{k+1} \leq f_{\bar{S}'_k+1}\}$. Therefore, $p_{\ell+2} \leq p'_{\ell'+1} \leq f_{\bar{S}'_{\ell'}+1} \leq f_{\bar{S}_{\ell}+1}$. However, by the definition of ℓ we have $p_{\ell+2} > f_{\bar{S}_{\ell+1}+1} \geq f_{\bar{S}_{\ell}+1}$, which leads to a contradiction. Therefore, prover j will at most gain utility $\sum_{k=1}^{q} s_{j,k} \cdot (p_{\ell+1} - p_j)$, and will lose utility.

If $j \geq \ell+1$, we show that if with the Sybil attack prover j can gain profit, then $\bar{S}'_{\ell'} > \bar{S}_{\ell}$ and $p_{\ell+1} < p'_{\ell'+1}$. To profit, we must have the first split is allocated, since prover j is originally not allocated, and that $p_j < p'_{\ell'+1}$. Thus, provers 1 to ℓ are still allocated, so $\bar{S}'_{\ell'} \geq \bar{S}_{\ell} + s_{j,1} > \bar{S}_{\ell}$. Note that $p_j \geq p_{\ell+1}$, we have $p_{\ell+1} \leq p_j < p'_{\ell'+1}$. Therefore, welfare will increase. □

A.6 Proof of Proposition 7

Proof. Suppose the coalition comprises a user with valuation f, and a prover with valuation p and capacity s, and that as part of their collusion, the user bids f', and the prover bids a cost p' and capacity s', where $(f', p', s') \neq (f, p, s)$. If $s' > s$ then the prover will get slashed when it is allocated, so let $s' \leq s$. We now split our analysis into different cases.

- If $(f', p') = (f, p)$, then $s' < s$. Given $2s$ tasks with fee $2p$ and another prover with cost $2p$ and capacity s, then the colluders' joint utility is $s \cdot p$, which is lower than they could have obtained honestly.

– If $f' > f$, we give an counterexample as follows. There are another two provers, both with s capacity, and with costs 0 and f' respectively, and there are $3s$ other tasks with fee f'. If $p' = p$, then after the strategic bidding the joint utility will reduce at least $f' - f$. If $p' > p > f'$ or $f' > p' > p$, then after the strategic bidding the joint utility will reduce at least $f' - f$. If $p' > f' > p$, then after the strategic bidding the joint utility will reduce at least $f' - f + s' \cdot (f' - p)$. If $p' < p \leq f'$ or $f' \leq p' < p$, then after the strategic bidding the joint utility will reduce at least $f' - f$. If $p' < f' < p$, then after the strategic bidding the joint utility will reduce at least $f' - f + s' \cdot (p - f')$.
– If $f' < f$, we give an counterexample as follows. There are two other provers, both with s capacity, and with costs 0 and $(f + f')/2$ respectively. There are $3s$ other tasks with fee $(f + f')/2$. If $p' = p$, then after the strategic bidding the joint utility will reduce at least $(f - f')/2$. If $p' > p \geq (f + f')/2$ or $(f + f')/2 \geq p' > p$, then after the strategic bidding the joint utility will reduce at least $(f - f')/2$. If $p' > (f + f')/2 > p$, then after the strategic bidding the joint utility will reduce at least $f' - f + s' \cdot ((f + f')/2 - p)$. If $p' < p \leq (f + f')/2$ or $(f + f')/2 \leq p' < p$, then after the strategic bidding the joint utility will reduce at least $(f' - f)/2$. If $p' < (f + f')/2 < p$, then after the strategic bidding the joint utility will reduce at least $f' - f + s' \cdot (p - (f + f')/2)$.
– If $f' = f$, then $p' \neq p$, and we give a counterexample as follows. There is another prover with s capacity and bid $(p + p')/2$, and there are $2s$ tasks with fee $(p + p')/2$. If $p' > p$, after the strategic bidding the joint utility will reduce at least $s' \cdot (p' - p)/2$. If $p' < p$, after the strategic bidding the joint utility will reduce at least $s' \cdot (p - p')/2$.

□

A.7 Proof of Proposition 8

Proof. Assume that the provers have capacities and costs (s_1, p_1) and (s_2, p_2) respectively with $p_1 < p_2$, and strategically bid (s'_1, p'_1) and (s'_2, p'_2), with $s'_1 \leq s_1$ and $s'_2 \leq s_2$. (Otherwise, if they bid high capacity they will be slashed when allocated.) We discuss the cases as follows:

– If $p'_1 = p_1$ and $p'_2 = p'_2$, we give a counterexample as follows. There is another prover with cost $2p_2$ and capacity 1, and there are $(s_1 + s_2 + 1)$ tasks with fee $2p_2$. Then their joint utility will decrease $p_2 \cdot (s_2 - s'_2) + (2p_2 - p_1) \cdot (s_1 - s'_1)$.
– If $p'_2 > p_2$, we give a counterexample as follows. There is another prover with cost $(p_2 + p'_2)/2$ and capacity s_2, and there are $s_1 + 2s_2$ tasks with fee $(p_2 + p'_2)/2$. In this case, their joint utility will decrease at least $s_2 \cdot (p'_2 - p_2)/2$.
– If $p'_2 < p_2$, we give a counterexample as follows. There is another prover with cost $(p_2 + p'_2)/2$ and capacity s_2, and there are $s_1 + 2s_2$ tasks with fee $(p_2 + p'_2)/2$. In this case, their joint utility will decrease at least $s'_2 \cdot (p_2 - p'_2)/2$.
– If $p'_1 < p_1$, we give a counterexample as follows. There is another prover with cost $(p'_1 + p_1)/2$ and capacity s_1, and there are $2s_1 + s_2$ tasks with fee $(p'_1 + p_1)/2$. In this case, their joint utility will decrease at least $s'_1 \cdot (p_1 - p'_1)/2$.

– If $p'_1 > p_1$, we give a counterexample as follows. There is another prover with cost $(p'_1 + p_1)/2$ and capacity s_1, and there are $2s_1 + s_2$ tasks with fee $(p'_1 + p_1)/2$. In this case, their joint utility will decrease at least $s_1 \cdot (p'_1 - p_1)/2$.

□

B Efficiency of the Core Mechanism

We define efficiency loss as the difference between the social welfare of the outcome of the mechanism and the optimal social welfare. Following the results of [18], if all provers have unit capacity, the core mechanism of *Prooφ* achieves welfare equal to $1/\min(\#\text{ users}, \#\text{ provers})$, which is asymptotically optimal. It is shown in [24] that for any strongly budget-balanced and incentive-compatible mechanism where a seller has M items and a buyer wishes to purchase at most M units, the efficiency asymptotically converges to 0 in M when the per-item trade price is determined exogenously. We suspect that a similar result may apply to the core mechanism of *Prooφ* in a myopic setting, due to pathological cases in which the capacity of the first unallocated prover is large. However, we hypothesize that a non-myopic analysis in which pent-up user demand "spills" over to the next rounds may show that these pathologies have limited impact in the long run, making such an analysis a promising direction for future work. For example, the related work of Nisan [20] shows the monopolistic TFM results in unbounded welfare losses in the myopic case, while in the non-myopic setting, the obtained welfare is at least $\frac{1}{2}$ of the optimal one.