

# Succinct Network: Prove the World’s Software

Uma Roy Succinct uma@succinct.xyz	John Guibas Succinct john@succinct.xyz	Kshitij Kulkarni Succinct k@succinct.xyz
Malleesh Pai Rice University and SMG malleesh.pai@mechanism.org	Dan Robinson Paradigm dan@paradigm.xyz	

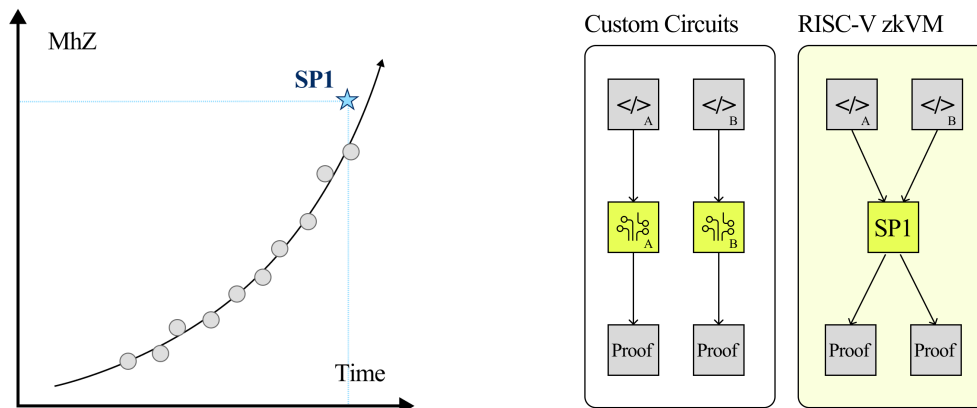
## Abstract

We propose the Succinct Network, a decentralized protocol that proves the world’s software. The network coordinates a distributed set of provers who generate zero-knowledge proofs through a novel incentive mechanism called *proof contests* to create the world’s most efficient, robust proving cluster. Users submit requests in the form of RISC-V programs with inputs and fees that determine proving priority. Provers engage in proof contests, which are all-pay auctions for the right to generate proofs with minimal latency and cost, in order to earn fees. The auctions settle on an application-specific blockchain. Both proof contests and the blockchain are codesigned with SP1, a zkVM that proves the execution of RISC-V bytecode. Proof contests are specifically designed to balance the competing goals of minimizing proving costs while encouraging a decentralized set of provers. The network coordinates the ecosystem of users, hardware teams, and infrastructure operators with cryptoeconomic incentives. Permissionless participation in the protocol catalyzes a global-scale infrastructure buildout that drives down the cost and latency of zero-knowledge proof generation for all.

## 1 Introduction

Decentralized systems such as blockchains operate based on distributed consensus: multiple parties reexecute computation and agree on the result. In exchange, blockchains allow for trustless applications that do not have a centralized mediator. This idea was first applied to decentralized money with Bitcoin and then to arbitrary trustless computation with Ethereum [Nak08; But13]. Redundant computation, however, is costly and imposes overhead on blockchains relative to their centralized counterparts. Due to this reason, blockchains remain bottlenecked.

Succinct zero-knowledge (ZK) proofs promise a path forward. They enable a prover to convince a verifier of the knowledge of a claim (such as the execution of a program



(a) Proof systems have experienced exponential performance improvements.

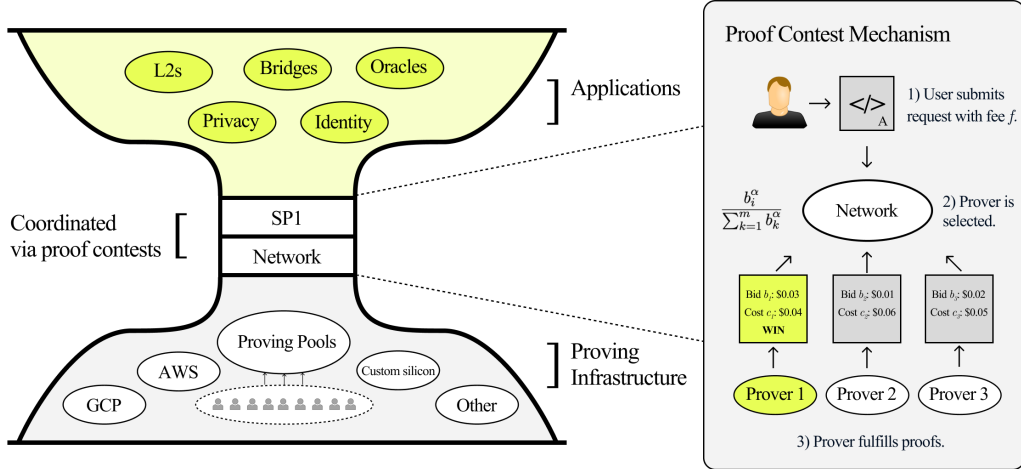
(b) ZK virtual machines convert application-specific circuits to general purpose programs.

**Figure 1:** ZK proving technology has seen rapid progress.

with optionally private inputs) with a proof that is much shorter than the claim itself [BS+13; BS+14]. This allows blockchains to scale their throughput by separating transaction execution, which is expensive, from transaction verification, which can be made cheap through ZK proofs. Only a small fraction of the participants in the system have to do expensive computation, while many nodes can just verify it. In the case of Ethereum, with real-time ZK proofs, “attesters can verify cheap SNARKs instead of naively reexecuting EVM transactions,” allowing Ethereum to arbitrarily scale [Dra24].

Beyond their implications for making blockchains scalable and commercially viable, ZK proofs herald a fundamentally new kind of computing paradigm—*cryptographic computing*—that enables verifiability in all systems. The implications of ubiquitous verifiability are wide-ranging. Today, tasks such as authentication, identity management and transaction verification take up the resources of governments and global companies. With cryptographic computing, it may be possible to vastly reduce the overhead these entities face via protocols that consume ZK proofs and automatically verify correct program execution over private user information. Crucially, ZK proofs differ from earlier cryptographic primitives such as digital signatures, which enabled authentication on the internet, in their programmability. They can allow for verifiable statements about any computation on any private or public inputs, vastly opening the design space for decentralized protocols and business logic involving attestations or identity.

Today, there are two important trends in ZK technology that indicate we are at a significant inflection point. The first is the rapid increase in performance in practical ZK proof systems. While efficient interactive proof systems were introduced in the 1980s [GMR85], they remained difficult to use until modern proof systems made significant advances in proving overhead, cost, and latency [Halo20; Plonky22; Plonky23]. ZK proving performance has been on an exponential curve due to improvements in software and performance engineering (Figure 1(a)). The second is the progress in

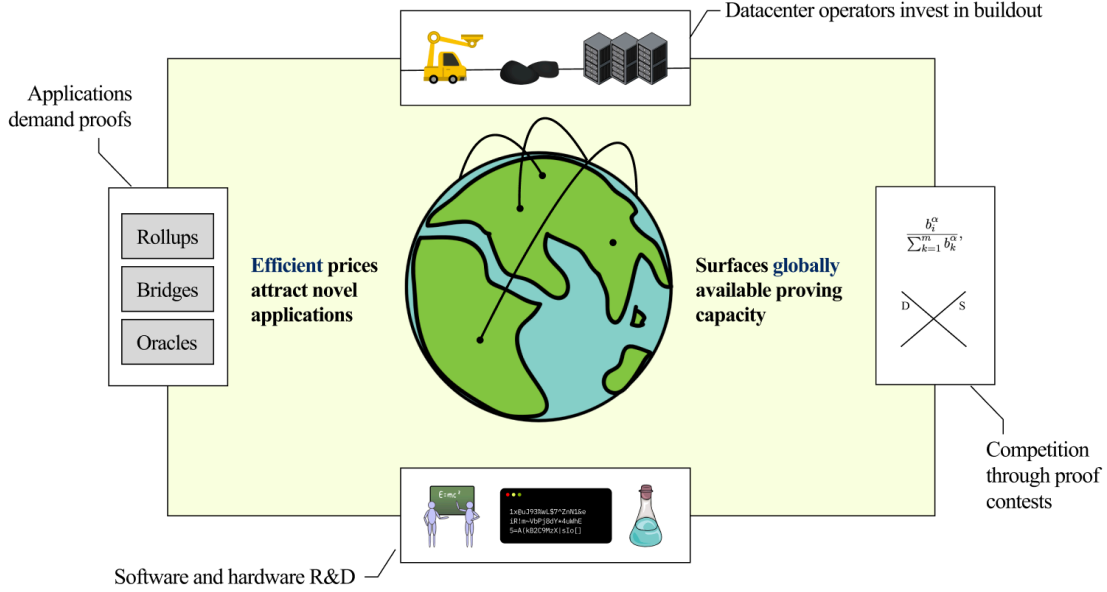


**Figure 2:** The Succinct Network creates a global, distributed proving cluster with a unified experience for users, powered by proof contests.

ZK proving technology from *application-specific* circuit design, which encodes program logic in an arithmetic circuit, to *general purpose* ZK virtual machines (zkVMs) that enable the proving of arbitrary deterministic programs and can utilize normal code and existing software (Figure 1(b)). The flexibility of normal programming languages allows for reuse for existing code. General purpose zkVMs such as SP1 have radically expanded the number of developers who can build applications using ZK and have allowed blockchains to become early consumers of ZK proving [SP124].

The confluence of both exponential performance improvements and general purpose zkVMs suggests a reexamination of the ingredients that will be required to accelerate the adoption of the cryptographic computing paradigm. ZK proving is a computationally intensive workload, on the order of large scale AI systems or Bitcoin mining. Production tasks like proving the execution of a blockchain’s state transition function can require specialized hardware such as GPUs and may require custom ASICs in the future. As the adoption of ZK technology accelerates, proving will necessitate an ecosystem of infrastructure providers, from datacenter buildouts in areas with low cost of electricity to deployment of specialized hardware. For blockchain use-cases, it is important that proving be decentralized, with geographically distributed proving operators in addition to home provers, who can provide reliability and liveness.

Realizing this ecosystem, composed of users, provers, and infrastructure providers, requires global-scale coordination and a *network* that appropriately incentivizes hardware acceleration, prover decentralization, and cost competition on ZK proving. To ensure that advances in proving technology accrue to end users, this network ought to be *codesigned* with a single zkVM. Network codesign ensures that users can write applications on a canonical, open-source zkVM that experiences continual performance improvements, and hardware teams and infrastructure providers can coordinate on a unified platform. The combination of coordination and competition between provers is a catalyst for the creation of a maximally efficient network.



**Figure 3:** The network, by aggregating supply and demand, creates a virtuous flywheel for proving infrastructure and R&D.

## 2 Succinct Network

The Succinct Network introduces the idea of a *global, distributed proving cluster* code-signed with SP1 and powered by a competitive auction mechanism called *proof contests* to dramatically expand the proving capacity of the world (Figure 2).

1. **A global, distributed proving cluster:** The network provides a unified platform for users to submit proof requests and for anyone in the world to provide proving capacity. Provers can join the network permissionlessly and participate by running node software and users can get high-reliability guarantees for their requests.
2. **Proof contests:** The proof contest mechanism is a novel auction that balances proving cost and decentralization, and induces free market competition on proving. It functions as an all-pay auction with collateral that allows for effective price discovery on requests while maintaining a decentralized prover set and solving several related problems, such as proof spamming, contention, and griefing. Proof contests settle on an application-specific blockchain designed for coordinating provers with requests.

### 2.1 Market Structure

There are two primary participants in the network: applications who demand ZK proving and provers who provide proving capacity and infrastructure. The cluster and proof contest mechanism jointly induce a market structure with a virtuous flywheel by aggregating supply and demand (Figure 3). This incentivizes global scale competition

for developing increasingly efficient proving infrastructure. Tight integration with SP1 development and the network also ensures that technological and algorithmic advances in proving can quickly accrue to users. We detail this process and how it affects the market participants.

### 2.1.1 The network kickstarts a virtuous flywheel by aggregating supply and demand

The network is explicitly architected so that any user can permissionlessly send requests in a unified interface and so that any prover can join and provide capacity according to pricing determined by the proof contest mechanism. The existence of this free market aggregates demand across many applications and supply across many provers. This has the following implications:

1. **A unified request experience means more efficiency.** Without the aggregation of supply and demand, it is likely that the market for proving degenerates into applications and provers making bespoke agreements or applications setting up in-house infrastructure. Over the long run, application teams may not be well-positioned to run their own proving infrastructure due to the specialized operational and technical skills involved. Over-the-counter deals are not efficient and lead to frictions in the market. A transparent market with a unified protocol for pricing proofs provided by an aggregated supply chain reduces integration time, development cost, and generates efficient prices.
2. **Transparent pricing accelerates infrastructure.** With transparent pricing and aggregated demand, provers face economics that are far easier to forecast. Provers can build out infrastructure faster and with more certainty because their cash flows are more predictable. The existence of a transparent market for proving means that provers' risks can be priced and financed efficiently.
3. **Home provers with low cost of capital can form *proving pools*.** The protocol allows for anyone in the world with spare proving capacity to join. Often, individuals with home GPUs and hardware have far smaller capital costs due to spare capacity and lower operational costs due to cheap electricity than larger operators. By allowing home provers to participate, proving capacity can be dramatically increased and efficient prices can be maintained. In Section 3.4, we introduce the notion of a *proving pool*, which is a set of provers that pool their capacity together to jointly bid in the proof contest mechanism. This allows home provers to mitigate economic risks.
4. **Permissionless participation encourages prover decentralization.** Because entry into the market from both sides is permissionless, the frictions for joining and beginning to provide proving capacity are minimized. This allows provers to join from anywhere, enabling geographic decentralization. In contrast to a marketplace where only certain actors can participate, this means that downstream users and applications can reliably receive proofs and are not bottlenecked by provers in a particular location.

Importantly, markets that aggregate supply and demand have historically resulted in ever-greater efficiency and better prices for all participants. The virtuous flywheel accelerates both the demand and supply side.

### 2.1.2 Proof contests incentivize global-scale competition while maintaining prover decentralization

The proof contest mechanism induces competition on proving and incentivizes a decentralized prover set, ensuring that applications that require high reliability and liveness are able to receive proofs for minimal cost. Proof contests are all-pay auctions where provers compete to receive fees provided by proof requests sent by users. The all-pay feature induces prover decentralization. We provide more detail about the mechanism in Section 3.

1. **Proof contests enable competition between provers.** Primarily, proof contests are designed to enable provers anywhere in the world to participate in the network while competing on proving costs. This is achieved by asking provers to bid in auctions to be able to receive fees set by users. Proof contests surface price signals by adjusting their fees to find market clearing prices for proofs.
2. **Price discovery allows for reinvestment.** Proof contests pay out a large share of fees set by users to provers that perform their core task well: efficiently generating low-cost ZK proofs within a user-specified deadline. This allows returns from the contests to be reinvested in better proving infrastructure and algorithmic and hardware improvements in proving. Provers are incentivized to continually improve their proving infrastructure. Proving pools can also provide smoothed fees to individual provers, allowing them to participate in the network in a “plug-and-play” fashion.
3. **Gains from competition accrue to applications.** End users are the primary beneficiaries from competition between provers. Over time, provers directly compete with each other on the main feature that users demand: low cost proofs delivered within user-specified deadlines. Importantly, there are applications that require proofs to be delivered at low costs to be viable at scale. The network uses free market competition to serve these applications.
4. **The mechanism enables prover decentralization.** In contrast to naïve mechanisms in which a prover who can generate proofs at a lower cost than any other would be expected to win 100% of auctions, such as a simple reverse auction, the proof contest mechanism incentivizes decentralization even under the existence of such a prover. This is done via an all-pay feature that resembles Tullock contests or contests for fixed prizes. This incentivizes a broad prover set and prevents prover concentration, which helps ensure long-term robustness of the network. This requires sacrificing some efficiency, since it necessarily involves sometimes allocating proofs to higher-cost provers. In exchange, having a robust, decentralized prover set allows applications to rely on the network.

	<b>Permissionless Participation</b>	<b>Incentive</b>	<b>Application</b>
<b>Bitcoin</b>	Setup a miner	Proof-of-work	Decentralized money
<b>Filecoin</b>	Setup storage	Proof-of-replication	Decentralized storage
<b>Aleo</b>	Setup a prover	Proof-of-succinct-work	Private blockchain
<b>Succinct Network</b>	Setup a prover	Proof contests	Decentralized proving

**Table 1:** Comparisons to networks that incentivized infrastructure buildouts.

## 2.2 Related Works

The Succinct Network draws its inspiration from a line of related works that that used cryptoeconomic incentives to coordinate global-scale compute infrastructure buildouts, such as Bitcoin and Filecoin. Today, Bitcoin has a hashrate that exceeds the capacity of any one company or nation. This was catalyzed by the incentives set up by the proof-of-work protocol. We compare and contrast the key features of existing infrastructure buildouts that used incentives with the Succinct Network; Table 1 summarizes this comparison on the axes of permissionless participation, incentives, and the application.

1. **Bitcoin.** Bitcoin bootstrapped a decentralized monetary system by incentivizing miners via the proof-of-work mechanism [Nak08]. The Bitcoin network’s incentives have led to hardware investment and specialized energy management infrastructure. It has also led to the creation of large mining pools that allow individuals to participate in block construction. Bitcoin users, however, do not explicitly demand low cost and low latency block construction from miners.
2. **Filecoin.** Filecoin set up a decentralized storage network via the proof-of-spacetime and proof-of-replication mechanisms [Ben17]. In Filecoin, storage providers can earn rewards for providing capacity to users and users can use the network for long-term, encrypted storage. As the service being provided to users is a commodity (storage), the primary requirement for participating as a storage provider is capital, which is required to purchase hardware. Running a storage node can also require specialized datacenter operation skills. In contrast to proof generation, Filecoin does not require specialized algorithmic knowledge and significant reinvestment in hardware improvements.
3. **Aleo.** Aleo enabled a privacy-first blockchain by incentivizing validators and provers via a mixture of proof-of-stake and proof-of-succinct-work [Wu23]. In Aleo, validators are responsible for proposing batches of transactions to the network, and provers solve coinbase puzzles, akin to proof-of-work mining and generate ZK proofs to earn rewards. The puzzles incentivize provers to invest in hardware acceleration. However, Aleo provers do not face the problem of delivering cost-effective proofs to end users. For this reason, provers can tolerate redundant work and contention caused by mining.

### 3 Proof Contests

We now introduce the proof contest mechanism, which is an auction that allocates requests to provers in the network and allows them to earn fees by balancing the dual forces of cost effectiveness and decentralization. At any time, users have sent a set of outstanding proof requests  $\{r_1, \dots, r_n\}$  and a set of provers  $\{p_1, \dots, p_m\}$  are available to prove them. Each request  $r_j$  comes attached with a fee  $f_j$ , which is the amount the user pays for the proof request, a cycle count  $s_j$ , which is a measure of the amount of work required to generate a proof of the request, and a **deadline**, which is the maximum amount of time the user is willing to wait for the request after assignment. Provers have private capacities for the maximum amount of cycles they can prove and private marginal costs per cycle  $c_j$ . There are several criteria that any mechanism for allocating proof requests to provers should satisfy.

1. **Efficiency.** The mechanism should match the highest value requests to the lowest cost provers to maximize the surplus and correspondingly quote the user low prices. Provers should compete to offer users preferential prices.
2. **User experience.** Users should experience transparent, competitive prices. Requests should be filled within user-specified deadlines with very high reliability. Requesting proofs should be simple.
3. **Prover decentralization.** The mechanism should incentivize a decentralized prover set and allow home provers to join. Applications may not be able to rely on a single, concentrated prover that comes to dominate the market due to economies of scale.
4. **Spam proofs.** Neither provers nor users should be incentivized to send spam proofs to the network to collect fees. The network should maximize the allocation of proving capacity and rewards to proof requests with real economic value.
5. **Throughput.** The mechanism should minimize contention on requests and maximize assignment of requests to available capacity, and correspondingly maximize throughput.

We first cover three candidate mechanisms and evaluate them on the above criteria. In Table 2, we compare these mechanisms with proof contests, which we describe shortly.

**Strawman mechanism: mining.** As a first approach, we can consider a mechanism commonly found in prior work on proof-of-work blockchains. In this approach, multiple provers may work on a proof request simultaneously and submit the resulting proof to earn rewards. One way of assigning rewards is to ask provers to submit proofs that satisfy a certain difficulty level. This approach was pioneered by [KB23] and Aleo [Wu23] in the context of designing a consensus protocol; [KB23] introduced the notion of using excess energy in proof-of-work to generate ZK proofs. Another way of assigning rewards is by the order in which provers submitted proofs: the first prover to submit a valid proof earns the largest share of the rewards, and subsequent provers earn less. An advantage of mining is that it enables a decentralized prover set that



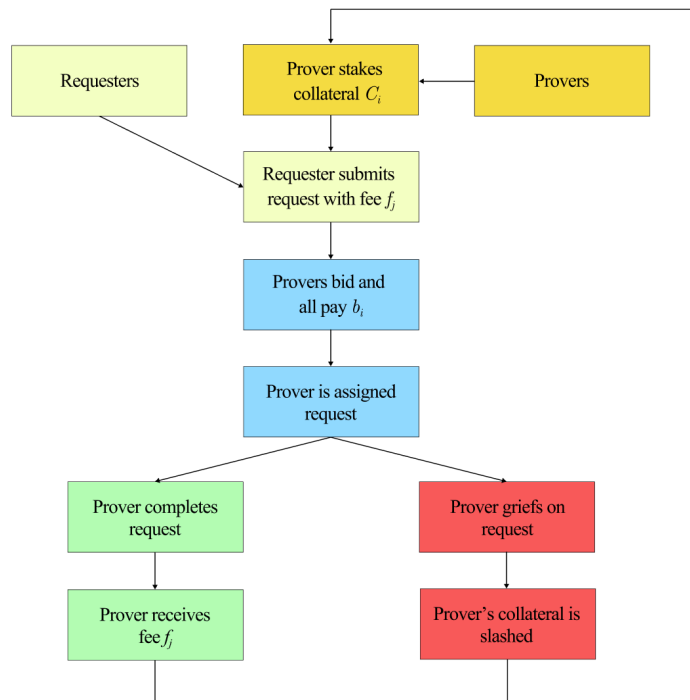
	Mining	Stake-based Selection	Reverse auction	Proof contests
Efficiency	✗	✗	✓	✓
User experience	✗	✗	✓	✓
Prover decentralization	✓	✓	✗	✓
Spam proofs	✗	✓	✓	✓
Throughput	✗	✓	✓	✓

**Table 2:** Evaluating proof contests alongside several candidate mechanisms.

allows for home provers and smaller hardware teams to continue to participate in the network. However, mining leads to significantly higher costs and reduced throughput due to *contention*, as many provers may be working on the same request at the same time, generating unnecessary, redundant proofs. Further, if provers are incentivized based on the difficulty of proofs, there is an incentive to submit proof requests of high difficulty but no economic value to collect rewards.

**Strawman mechanism: stake-based selection.** A second approach involves choosing provers to prove requests via stake-based selection. This can be implemented by asking provers to deposit collateral or stake, and then selecting them to prove a request with probability proportional to their stake. The upside of this mechanism is that it avoids contention by assigning each request to a single prover. However, if implemented in isolation, without a fee mechanism that finds market clearing prices, it does not directly induce competition on cost, since provers are selected purely based on the amount of stake they hold in the system. It separates the incentives of running a prover, which involves algorithmic and hardware improvements, from those of running a staking service, which involves capital management.

**Strawman mechanism: reverse auction.** A third approach involves asking provers to participate in a competitive auction to reduce the price of a proof as much as possible. In this mechanism, provers bid in a reverse or procurement auction for the right to satisfy proof requests. Provers submit bids for each request. The bid is interpreted as the lowest price at which a prover is willing to satisfy the request. The winner is the lowest bidder, and can win at either the lowest or second-lowest price. This mechanism leads to an efficient allocation of requests and minimizes contention due to proof assignment to a single prover. However, it potentially leads to intense prover concentration, as a single prover who is capable of generating proofs at lower cost than any other may come to win every auction.



**Figure 4:** Proof contests ask provers to deposit collateral in order to bid for requests. Winning provers are assigned proofs and have to complete them within specified deadlines, or risk having their collateral slashed.

### 3.1 Mechanism Description

The main novelty of the proof contest mechanism is that it solves for the desired criteria and offers competitive prices to users while encouraging a decentralized prover set, improving upon the mechanisms introduced earlier. Proof contests use a mixture of collateral (resembling stake-based selection) and an all-pay auction to earn market clearing fees (resembling a simple reverse auction) to cause provers to compete on proof generation.

In our design, provers deposit collateral to be able to participate in an all-pay auction or Tullock contest to be assigned the right to generate proofs for requests. Tullock contests model competition between a set of participants for fixed rewards [BTT80]. They have been studied in other contexts in blockchains, most notably in modeling Bitcoin mining [LS20] and Ethereum block building [BGR24], and as part of a proposed mechanism to encourage decentralization in block building [NGR24]. In our design, participation in the contest allows provers to compete for proof generation. The provers' bids in the auction are interpreted as the amount they will pay to receive the right to generate proofs. All provers pay their bids, and the winning prover is selected randomly among the provers with probability that is some function of their bid.

The all-pay auction feature of proof contests induces competition on proving, as in

order to win the contest, provers have to submit competitive bids. The collateral acts to solve the “nothing-at-stake” problem for provers such that they are incentivized to not grief on assigned requests. We first introduce the primitives that proof contests use. A schematic description of the mechanism is shown in Figure 4.

**A censorship resistant bulletin board.** First, we assume that there exists a censorship resistant public bulletin board. In the case of the network implementation, we make use of an application-specific blockchain; we provide further details about the network’s implementation in Section 4. The board provides functions  $\text{READ}(\cdot)$ ,  $\text{WRITE}(\cdot)$ , and  $\text{EXECUTE}(\cdot)$ . The function  $\text{READ}(\cdot)$  allows anyone who calls the function to read all messages written to the bulletin board. The function  $\text{WRITE}(m)$  allows a user to write a message  $m$  to the bulletin board. Finally, the function  $\text{EXECUTE}(\cdot)$  allows the bulletin board to compute the output of an algorithm and write it to the bulletin board; this can be implemented using a smart contract functionality.

**Entry and collateral.** First, each of the provers  $p_i$  deposits a collateral amount  $C_i$  prior to receiving the ability to bid for proof requests. The collateral is set to prevent provers from grieving requests upon assignment; bids are also drawn from collateral. The collateral acts to prevent the provers from having “nothing-at-stake” while participating in proving. The amount of collateral that each prover deposits reflects the number of auctions that they are eligible to participate in.

**The mechanism.** A proof request  $r_j$  is written to the bulletin board via  $\text{WRITE}(r_j)$  by a user, upon which it is publicly viewable to the entire set of provers via  $\text{READ}(\cdot)$ . Provers bid amounts  $b_1, \dots, b_m$  in an auction for the right to prove requests; the bids are drawn from the provers’ collateral. The bids are interpreted as the amount a prover is willing to pay to receive  $f_j$  from requester  $r_j$  and complete the proof. The auction resolves by awarding the proof to prover  $p_i$  with probability

$$\frac{b_i^\alpha}{\sum_{k=1}^m b_k^\alpha},$$

where  $\alpha > 0$  is a parameter to be set by the protocol. All provers pay their bids; these bids can be collected in a contract and then disbursed either to requesters as a rebate or as a refund to bidders over time. The interpretation of this mechanism is that the prover who is willing to pay the most for the right to complete the request wins with the highest probability, but other provers who bid less also win some of the time. The degree of distribution is specified by  $\alpha$ . When  $\alpha = 0$ , the request is assigned to the bidders with uniform probability. When  $\alpha = 1$ , the assignment probability is proportional to each prover’s bid. As  $\alpha \nearrow \infty$ , the proof is increasingly assigned to the highest bidder. The winning prover completes the request and the amount  $f_j$  is transferred from the requester to the winning prover if the proof is completed within the **deadline**. If no proof is submitted prior to the deadline, the request expires and the prover who was assigned to generate the proof has their collateral slashed. The payment and collateral logic can be handled via  $\text{EXECUTE}(\cdot)$ .

**Extension to recurring proofs.** It is possible to extend the proof contest mechanism for a single request to requests that recur over time, as is the case with rollups and applications with repeated transactions. This is a version of the above mechanism that functions without money. In this case, the user’s fee  $f_j$  is interpreted as the amount they would like to pay per proof for recurring requests of cycle count  $s_j$ . Now, instead of bidding money in an all-pay auction, each prover reserves a certain number of slots  $b_i$  ahead of time to fill the proof requests. The total number of slots that provers reserved determines the price. Now, in contrast to the earlier mechanism with money, the per proof price is reduced to  $f_j - \sum_{i=1}^m b_i$ . Prover  $p_i$  is assigned the right to prove a fraction  $\frac{b_i}{\sum_{i=1}^m b_i}$  of the proofs, and provers rotate in this ratio in a round-robin fashion. The advantage of this mechanism is that it achieves a decentralized prover set without asking provers to bid in an auction per request.

We now show how the behavior of provers in this recurring model is related to the earlier setup of an all-pay auction for each proof. Recalling that each prover has a marginal cost  $c_i$  per proof. The total profit of the prover over a single cycle in which everyone rotates is the product of the number of slots the prover is assigned times the profit per slot, or

$$b_i \left( f_j - \sum_{i=1}^m b_i - c_i s_j \right).$$

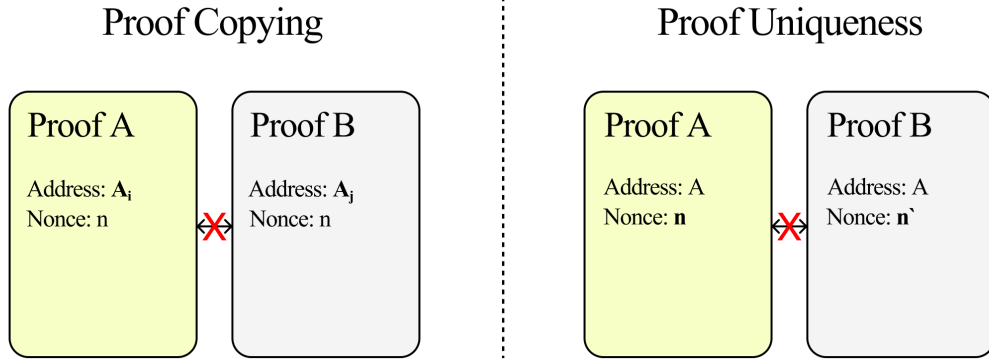
Dividing through by  $\sum_{i=1}^m b_i$ , the per proof profit of the prover is

$$\frac{b_i}{\sum_{i=1}^m b_i} (f_j - c_i s_j) - b_i,$$

which can be interpreted as the prover bidding  $b_i$  in the earlier, nonrecurring mechanism, with per proof profit  $f_j - c_i s_j$  for the proof and with  $\alpha = 1$ .

**Setting collateral.** The collateral  $C_i$  controls the number of auctions that a prover is allowed to participate in. The total sum of all the fees that a prover is currently bidding for should be less than the collateral they have put up at any time. The network automatically locks the number of auctions that a prover can participate in given the collateral they have deposited.

**Setting fees.** We propose a candidate mechanism for setting the fees  $f_j$  that clears proof requests while maintaining prover diversity in the auction. The fee mechanism we consider maintains a posted price, which is a uniform per-cycle fee (that is,  $f_j = f s_j$  for every request, for some fixed  $f$ ). Higher fees mean that the prospect of having multiple bidders in the auction increases. Fees can be set to achieve 100% clearing rate of outstanding proof requests, via a control system that increases fees if requests were left over in the last timestep. To ensure that the fee is set high enough to ensure prover diversity, the proof contest mechanism can be complemented by simple Dutch auctions for a fraction of proof requests. These auctions would start at discounted prices and provide a signal as to the fee that a simple reverse auction would clear at, which could help the proof contest mechanism set a comfortably higher fee in order to



**Figure 5:** The codesign of the network with SP1 prevents proof copying and reused work.

encourage prover diversity. Parameterizing this fee adjustment mechanism to maintain a diverse set of provers that operate without manipulation is a topic for further study. In Appendix C, we derive the minimum transaction fees required to support multiple provers as a function of their costs.

**Setting bounds on bids.** In order to incentivize competitive bidding, the protocol can also impose lower bounds on bids that can be increased over time. This means that the quality of provers, and therefore the costs experienced by requesters, improve. Proving pools and hardware teams jointly have to improve their hardware and proving software to be able to participate in the auction; these entry costs ensure that requesters receive cost-effective proving as performance improves.

### 3.2 Mechanism Implementation

A practical implementation of the mechanism should prevent several failure modes such as copied proofs and reused work (Figure 5). We describe the modifications that are made to fulfilled proofs that are submitted to the network; this is realized in practice by codesigning the network with SP1. Suppose a proof request  $r_j$  is fulfilled by prover  $p_i$ . In the network, proofs are a function of the onchain address  $a_i$  of prover  $p_i$  and a unique nonce  $n$ , which is generated at the time the proof was requested. We succinctly represent this dependence by  $\text{proof}(r_j, a_i, n)$ . Proofs are designed to satisfy the following properties.

**Embedded prover addresses: preventing copying.** First, to prevent a proof generated by a given prover from being copied by another prover, we make use of prover address information. The main property we seek is that for any  $a_k \neq a_i$ , it should be the case that the protocol can distinguish  $\text{proof}(r_j, a_i, n)$  from  $\text{proof}(r_j, a_k, n')$ , for any  $n, n'$ . Therefore, a prover cannot simply reuse a proof posted by another prover

to collect rewards or fees. This can be implemented by embedding the prover address during proof construction in SP1.

**Unique nonce: preventing reuse.** Second, to prevent a prover from reusing the work they did for a given proof, we embed the proof with the unique nonce that is generated at request time. This nonce can be embedded in the challenger of the underlying proof system and can be used by an onchain verifier to check that the proof was constructed using the nonce. The main property we seek now is that for every  $a_i$  and for any  $n, n'$ , the protocol can distinguish  $\text{proof}(r_j, a_i, n)$  from  $\text{proof}(r_j, a_i, n')$  unless  $n = n'$ . Because the nonce is unique and generated at request time, this means that a prover cannot reuse their own work.

### 3.3 Analysis and Incentives

We now describe several important features of the incentives that proof contests set up and how they solve the problems faced by alternate mechanisms. We consider how the all-pay feature trades off cost competition and decentralization, the Sybil resistance of the mechanism, and how proof contests increase throughput by breaking the contention found in mining.

**Prover competition.** The proof contest mechanism induces competition on proving by asking provers to bid for the right to fulfill requests. By asking provers to bid for requests, the mechanism aligns the provers' incentives with a key requirement of the user: finding provers that can competitively generate proofs. We consider a setting where  $m$  provers, each of whom have per cycle marginal proving costs  $c_1 < c_2 < \dots < c_m$ , participate in the proof contest mechanism for a proof request with fee  $f$ , cycle count  $s$ , and  $\alpha = 1$ . In this case, the utility of a prover who bids  $b_i$ , while other provers bid  $b_{-i}$  is given by

$$u_i(b_i, b_{-i}) = \frac{b_i}{\sum_{j=1}^m b_j} (f - c_i s) - b_i.$$

Assuming that the fee is bigger than the proving cost for each prover that participates ( $f - c_i s > 0$ ), we can calculate the equilibrium bids  $b_i^*$  submitted by the provers via the first order conditions  $\frac{\partial u_i}{\partial b_i}(b_i^*, b_{-i}^*) = 0$ , which are

$$\frac{\partial u_i}{\partial b_i}(b_i^*, b_{-i}^*) = (f - c_i s) \frac{\sum_{k \neq i} b_k^*}{(\sum_{k=1}^m b_k^*)^2} - 1 = 0.$$

Rearranging the conditions,

$$\frac{\sum_{k \neq i} b_k^*}{(\sum_{k=1}^m b_k^*)^2} = \frac{1}{(f - c_i s)},$$

and summing over all the provers,

$$\sum_{i=1}^m \frac{\sum_{k \neq i} b_k^*}{(\sum_{k=1}^m b_k^*)^2} = \sum_{i=1}^m \frac{1}{(f - c_i s)}.$$

Now, noting that  $\sum_{i=1}^m \sum_{k \neq i} b_k^* = (m-1) \sum_{k=1}^m b_k^*$ , we see that

$$\sum_{k=1}^m b_k^* = \frac{m-1}{\sum_{i=1}^m \frac{1}{f-c_i s}} \quad (1)$$

$$= \frac{m-1}{m} H(f - c_1 s, \dots, f - c_m s), \quad (2)$$

where  $H$  is the harmonic mean. In the setting where the bids provide a rebate to users over time, we can interpret this as follows: when the costs of the provers are very close to the costs of the best prover, the effective price quoted to the user is nearly the cost of the best prover. When the costs are very disparate, however, the effective price may be higher. We provide a model of how rebates, combined with a preference for decentralization, compare to a simple reverse auction in Appendix B.

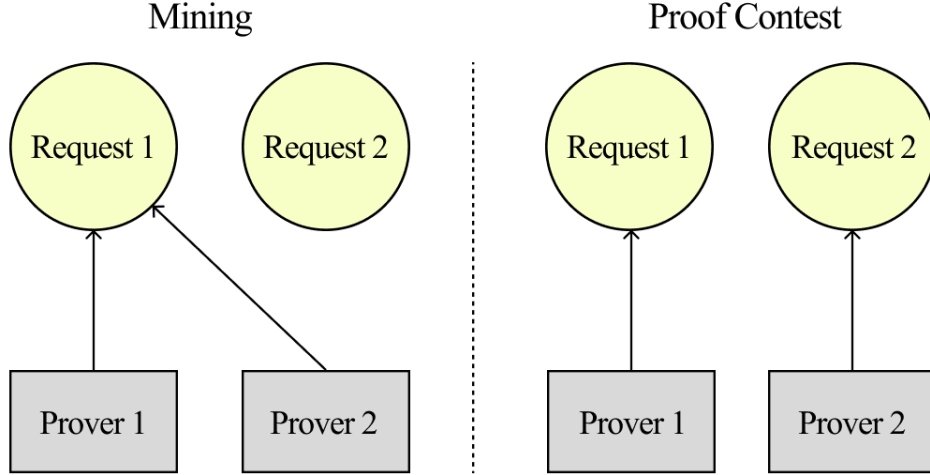
**Prover decentralization.** The allocation mechanism induced by proof contests (and by Tullock contests in general) intrinsically incentivizes a diverse set of provers to participate, and is the basis behind previous attempts in blockchains to achieve a decentralized set of service providers, such as miners in Bitcoin, validators in Ethereum, and block builders in the “execution tickets” mechanism described in [NGR24].

A simple reverse auction would always allocate proofs to the prover who bids the lowest cost. While this should result in an efficient assignment of proofs, it may not result in a robust prover marketplace, since if one prover is consistently more efficient at generating proofs, it may never make sense for another prover to bid.

By contrast, proportional allocation, the rule for proof contests with  $\alpha = 1$ , allocates proofs to provers proportional to their bids. This has the effect that a prover who has already bid a large amount faces diminishing marginal returns to increasing their bid (since it would increase the total amount of bids, which would decrease the expected likelihood of winning for each dollar they’ve already bid). The mechanism makes it difficult for a prover to dominate and earn 100% of fees, because it is often profitable on the margin for small bidders to bid more than zero (and thus have some probability of winning a contest).

While Tullock contests encourage decentralization, they present a tradeoff—since they sometimes allocate proofs to less efficient provers, they will generally result in higher proving costs for users, relative to a simple reverse auction. The mechanism should be parameterized to balance the long-term robustness benefits of prover diversity against the immediate benefits of prover efficiency.

**Sybil resistance.** The proof contest mechanism disincentivizes provers from splitting their identity while bidding in the auction through the all-pay feature and by requiring provers to put up collateral to be able to bid. We consider a prover who attempts to split their bid  $b_i$  into  $M$  bids of size  $\frac{b_i}{M}$  in a proof contest with fee  $f$ , cycle count  $s$ , and arbitrary  $\alpha$ . We can compute the utility to this prover from the modified



**Figure 6:** Proof contests break the contention found in mining by assigning requests to a single prover and correspondingly increase throughput.

bid profile  $\tilde{b}_i$  as

$$u_i(\tilde{b}_i, b_{-i}) = M \left( \frac{\left(\frac{b_i}{M}\right)^\alpha}{M \left(\frac{b_i}{M}\right)^\alpha + S} (f - c_i s) - \frac{b_i}{M} \right),$$

where  $S = \sum_{k \neq i} b_k^\alpha$  is fixed for all other provers. In Appendix D, we demonstrate that it is not in a prover's interest to split their bids into multiple identities (that is,  $u_i(b_i, b_{-i}) \geq u_i(\tilde{b}_i, b_{-i})$ ) in a proof contest for any  $\alpha \geq 1$ .

**Throughput.** The proof contest mechanism breaks the contention found in mining by assigning proof requests to a single prover via bidding. This prevents reductions in throughput. We consider a model in which there are  $n$  requests that arrive every time unit and  $m$  provers are available to prove them. Each prover has enough capacity to fulfill exactly one request. In mechanisms like mining that allow provers to freely choose which requests they will work on, multiple provers can end up choosing to work on one of the requests, leaving some requests unfulfilled. On the other hand, in proof contests, the bidding phase prevents reductions in throughput and matches requests to available proving capacity. We suppose that in mining, each of the provers picks a request to work on uniformly at random. In Appendix E, we show that the reduction in throughput in mining, compared to proof contests, is

$$1 - \frac{n}{m} \left[ 1 - \left( \frac{n-1}{n} \right)^m \right]$$

We depict a simple example of how this can happen with two requests ( $n = 2$ ) and two provers ( $m = 2$ ) in Figure 6. The reduction in throughput in this case is 25%. This



is because the maximum throughput, achieved by proof contests, is 2 proofs per time unit, whereas in mining, the expected number of completed requests is

$$\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1.5,$$

which gives us the reduction in throughput,  $100 \left(1 - \frac{1.5}{2}\right) = 25\%$ .

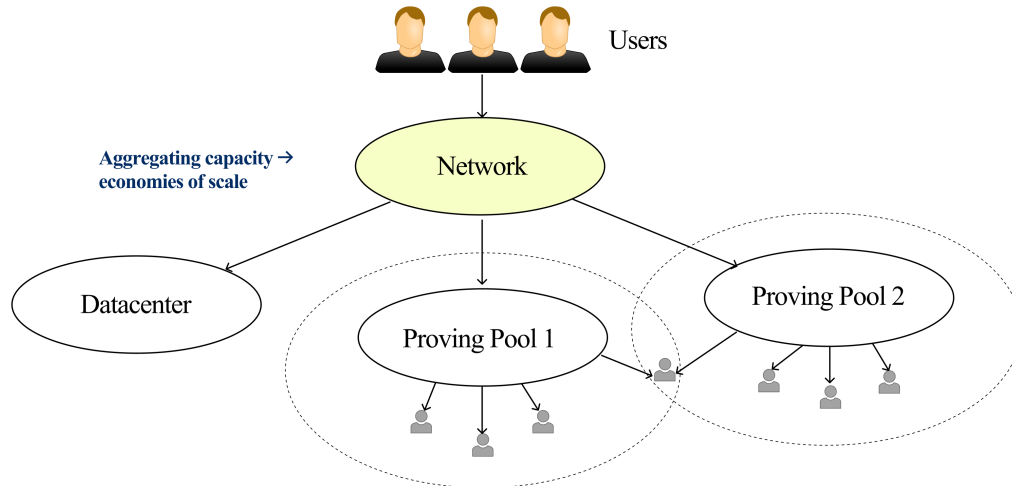
**Spam proofs.** Since in this model, all rewards received by provers are paid by requesters, there is no incentive for provers to submit “spam proofs” for which there is no real economic demand. Further, provers cannot earn bids from other provers by submitting spam requests either, since bids are collected in a contract and uniformly distributed as rebates. In future work, we will show how network rewards can be added to this mechanism to subsidize the cost of proofs without introducing an incentive to submit requests for proofs with zero economic value.

### 3.4 Proving Pools

Individual provers who may not have sufficient sophistication or collateral to participate in the auction outright can form proving pools in which they can collectively bid for request slots, akin to mining pools in Bitcoin. We introduce the notion of *proving pools*, which are composed of many provers who jointly pool their capacity to bid in the proof contest mechanism. Pools allow provers with home setups to start generating proofs without having to learn how to bid in auctions, by allowing for permissionless, bidding-free coordination. This allows the network to bootstrap a global distributed proving cluster that can be joined by anyone. Proving pools can serve as a counterweight to the potentially centralizing effects of specialized hardware and economies of scale. We depict the network with proving pools in Figure 7.

**Permissionless proving pools.** Because members of proving pools do not necessarily have to participate in bidding, pools can themselves be permissionless and require minimal sophistication for home provers to join. Pool operators have an opportunity use free market forces to aggregate proving capacity across the world and earn fees. Pool fees can be paid out in a fashion similar to Bitcoin mining pools, by querying pool members for capacity and paying them a fraction of fees based on the share of the total capacity they provide. The distributed, global nature of proving pools is a key hallmark of the network.

**Pools can be competitive.** Proving pools have the additional benefit that they do not require the large upfront capital costs incurred by hardware teams to get started. Pools can set incentives to cause proving capacity to come online during times of high demand and go offline when there are relatively few proofs. Pools also do not face recurring operational costs such as utilities and rent if pools are primarily composed of home provers. Pools can search for capacity in low cost of electricity areas. Therefore, it is expected that well-run proving pools can remain competitive with hardware teams and optimize their operations, incentivized by the prospect of earning fees. This is similar to how mining pools in proof-of-work cryptocurrencies remain competitive.



**Figure 7:** Proving pools can dramatically expand the network’s capacity and enable home provers to join without bidding.

**Proof delegation.** Pooling also allows for delegation of proving capacity to multiple pools. This means that provers with home setups can express their risk preferences by participating in pools that bid more or less aggressively in the mechanism. Pools can coordinate proving capacity depending on their needs. This model resembles delegation found in proof-of-stake currencies and especially in decentralized staking pools; it is expected that instruments to pool the risk of proving will arise with the emergence of global proving pools.

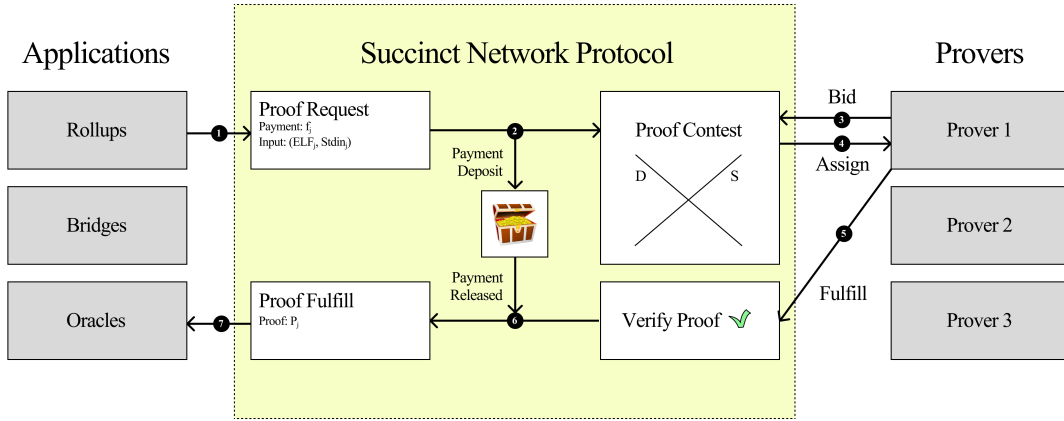
## 4 Network Architecture

Having introduced the proof contest mechanism, we now turn to the question of how to implement the network in practice via an application-specific blockchain. We first illustrate the desired properties of the network and how proof contests can be realized, and demonstrate that the implementation necessitates an application-specific blockchain. Next, we describe the architecture and how users and provers can participate; the architecture is depicted in Figure 8.

### 4.1 Implementation Requirements

Any implementation of the proof contest mechanism for a network that assigns proof requests to provers should satisfy the following:

1. **Low latency:** The implementation needs to coordinate payments, bidding, and proof fulfillment. It is therefore essential that the system be low latency and high throughput.



**Figure 8:** Succinct Network architecture.

- 2. Censorship resistance:** The implementation should afford downstream applications censorship resistance guarantees for their requests.
- 3. Liveness:** For some applications, delayed proofs can cause meaningful economic damage. The implementation should reliably, and with high liveness coordinate proof requests and fulfillment.

For these reasons, we choose to coordinate users and provers via an application-specific blockchain as opposed to a system that runs on trusted intermediaries to coordinate proofs. The blockchain serves to provide censorship resistance and decentralization required for downstream applications. Users submit requests to and provers bid and fulfill proofs on the blockchain. The blockchain is codesigned with SP1 to make fulfilling and verifying proofs seamless and to prevent spamming and other griefing vectors. It can be implemented as a modern layer 1 or rollup and is designed for progressive decentralization; the implementation details are outside the scope of this whitepaper. Permissionless participation and payments mean that anyone can bridge funds directly to the blockchain and begin requesting proofs, making for a desirable user experience.

#### 4.1.1 Performance

The blockchain is designed to be performant and to minimize excess latency in proof fulfillment. With short blocktimes, proof contests can be conducted quickly for each request. Bidding can take place in a short, preassigned window. The blockchain also benefits from ephemeral data storage, where data associated with proof requests can be temporarily stored.

#### 4.1.2 Consensus

The blockchain is designed for progressive decentralization, whereby requests can be given robust censorship resistance and liveness guarantees via nodes that operate the

network and come to consensus on requests, bids, and proof fulfillment. If implemented as a modern layer 1, this is done by having a decentralized validator set for the blockchain. If implemented as a rollup, this means having a decentralized sequencing layer that can come to agreement on the ordering of requests.

## 4.2 Participation

There are two main parties in the network: users, who submit proof requests, and provers, who fulfill proofs. There are third parties called relayers who provide services that enhance the user experience, such as simulating program execution. We describe how users and provers participate (the role of relayers is described in Appendix A).

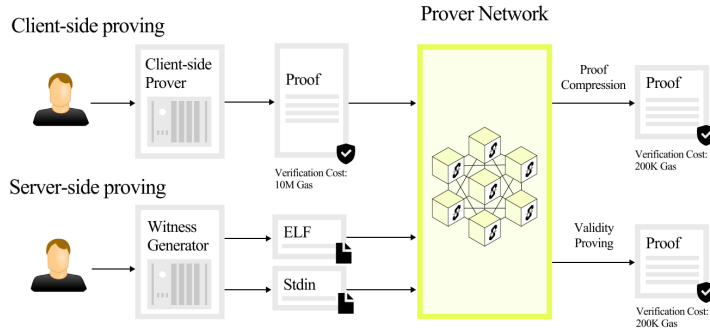
### 4.2.1 Users: Submitting Requests

Users submit proof requests to the network by inputting a `program`, inputs `stdin`, a `max_cycles` parameter, the maximum fee  $f_j$ , the `deadline`, and a `vkey`. The program and inputs are the data required by the prover to generate the proof. We provide additional details about how programs and inputs are stored in Appendix A. The quantity `max_cycles` is the maximum number of RISC-V cycles that the user expects the proof to consume and the fee  $f_j$  is the maximum amount that the user wants to pay for the request. The fee can be independently set by the user or via suggestion from a functionality similar to gas estimation endpoints in Ethereum. Upon sending the request, the user's fee is held in escrow to prevent griefing vectors. Finally, the `vkey` is a key that is used by the verifier to check the correctness of the proof and is generated during an initial setup phase. A unique `proof_id` is created for every submitted request. Proof requests can be sent to the network via standardized RPC endpoints. From the user's point of view, the network is a high liveness, competitive interface for proof requests. Once completed proofs have been submitted by provers, users can relay them to their destination chains directly from the network.

### 4.2.2 Provers: Bidding and Fulfilling Proofs

Provers can permissionlessly participate in the network by simply plugging in their hardware and running node software which allows them to bid for requests. Provers may have capacity limits and operating costs to generate proofs, as shown in Figure 8, but these may not be revealed to the network; the bidding process helps elicit this information. Bidding and fulfilling requests happen on the application-specific blockchain. Provers begin by seeing proof requests that have been submitted and fetching program and input data.

**Bidding.** Provers then bid for the right to complete requests in a proof contest that computes a clearing price and assignment. Proof contests are conducted onchain, after the request is visible. The bidding window is fixed ahead of time for transparency. The blockchain is designed to have short blocktimes to enable bidding at timescales much smaller than proof generation times.



**Figure 9:** The network scales decentralized applications via server-side proving and enables ubiquitous privacy via client-side proving.

**Fulfilling proofs.** Once the assigned prover is finished completing their proof, they can fulfill it on the application-specific blockchain. The prover inputs the `proof_id`, the `program`, `stdin`, and `proof` information. The `vkey` submitted alongside the proof request allows a verifier to check that the proof was generated correctly. SP1 allows for three types of proofs to be generated: `CORE`, `COMPRESSED`, and `GROTH16/PLONK`. `CORE` proofs have size proportional to the program’s execution, whereas `COMPRESSED` proofs have constant size but are more expensive to generate. `GROTH16` and `PLONK` proofs are the most expensive to generate and have the benefit of being highly compressed; they can be verified onchain. Due to the variable and large proof sizes of `CORE` proofs, the network allows requesters to request `COMPRESSED` and `GROTH16/PLONK` proofs.

## 5 Proving the World’s Software

The network has significant implications for accelerating verifiability on the Internet. Verifiability means that applications can outsource expensive computation and business logic without relying on trusted intermediaries, and users can receive guarantees that sensitive information will remain private and that the execution of the computation is correct. The aggregation of proving capacity under one permissionless, global network means that decentralized systems can face shared standards and scale with minimal overhead. This drives down prices and enhances the user experience, creating tailwinds for applications that use ZK.

There are two main modalities through which a user can interact with the network: as a standalone server-side prover, or mixed with client-side proving, as depicted in Figure 9. Server-side proving is useful for applications that require specialized prover nodes to outsource computation, such as public blockchains and rollups. Client-side proving is useful for privacy applications that can hold user information private to generate a large proof that can then be compressed via the network. We now describe several applications that can be built using or enhanced by the network.

1. **Rollups.** The rollup-centric roadmap of Ethereum is increasingly shifting to using ZK technology for transaction verification, compared to the heavy use of

fraud proof technology in the past. As transaction demand for ZK rollups rises due to their low costs and ease of use, the number of proofs demanded will correspondingly increase rapidly. Rollups can use the network for proofs that are delivered with competitive prices and high reliability. This enables all decentralized systems to scale their throughput without sacrificing their verifiability.

2. **Oracles and bridges.** Other components of decentralized systems that can use the network include oracles and bridges. ZK oracles can trustlessly provide blockchains with inputs from external data feeds, and ZK bridges can enable interoperability between all blockchains. Crucially, a decentralized prover set is important for these applications as they cannot rely on a single point of failure.
3. **Identity and authentication.** The zero-knowledge property ensures that private user inputs can be incorporated into proofs without being revealed to anyone else. This is essential for applications that require privacy, such as identity and authentication, in which user inputs might involve sensitive identifying information. To make sure that these inputs remain private during the proving process, client side proving can be used in tandem with the network.
4. **Credit scoring and auditing.** Currently, businesses that perform credit scoring, notarization, or auditing have high operational costs and have to manage complex, legacy databases. Applications that use the network could massively reduce the costs in these operations by leveraging ZK proving to provide proofs as audits.
5. **Verifiable inference.** AI applications can use the network to prove the output of an AI model on a given set of queries to an end user. The network enables applications to receive a large quantity of proofs required for high throughput inference applications.
6. **Coprocessors.** Blockchains can use the network to have coprocessors that offload heavy computation to off-chain actors. This can allow blockchains to access historical data, compute fees with high-fidelity economic models, and allow third parties to query blockchains trustlessly.
7. **Outsourced cloud computing.** The network can be used as a generalized outsourced computing layer. Users can send programs to provers that they would ordinarily run in cloud servers, if not for the trust assumptions and loss of verifiability that these servers bring.

## 6 Conclusion

We propose the Succinct Network, a protocol for coordinating a decentralized set of provers to satisfy user requests for ZK proof generation via a competitive auction called proof contests to create a global-scale distributed proving cluster. The network is co-designed and integrated with SP1, a zkVM that allows for the proving of arbitrary deterministic Rust programs. Proof contests balance cost competition and prover decentralization to allocate requests to provers and are settled on an application-specific blockchain that is specially designed to coordinate users and provers. The network

allows for permissionless entry as a prover and induces free market competition. It enables a broad infrastructure buildout for proof generation akin to other cryptoeconomic systems, and ensures that applications from blockchain scaling to privacy have access to robust, competitive proving. By using incentives to dramatically expand the proving capacity of the world, the network leads to low cost, low latency ZK proof generation for all.

## 7 Acknowledgments

We would like to thank Tarun Chitra, Ciamac Moallemi, Dave White, and Max Resnick for valuable feedback on this paper.

## References

- [Nak08] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: <https://bitcoin.org/bitcoin.pdf>.
- [But13] Vitalik Buterin. *Ethereum whitepaper*. 2013. URL: <https://ethereum.org/en/whitepaper/>.
- [BS+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. “SNARKs for C: Verifying program executions succinctly and in zero knowledge”. In: *Annual Cryptology Conference*. Springer. 2013, pp. 90–108.
- [BS+14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. “Succinct {Non-Interactive} zero knowledge for a von neumann architecture”. In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 781–796.
- [Dra24] Justin Drake. *[AMA] We are EF Research (Pt. 12: 05 September, 2024)*. 2024. URL: [https://www.reddit.com/r/ethereum/comments/1f81ntr/comment/1lmfi28/?utm\\_source=share&utm\\_medium=web3x&utm\\_name=web3xcss&utm\\_term=1&utm\\_content=share\\_button](https://www.reddit.com/r/ethereum/comments/1f81ntr/comment/1lmfi28/?utm_source=share&utm_medium=web3x&utm_name=web3xcss&utm_term=1&utm_content=share_button).
- [GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The knowledge complexity of interactive proof-systems”. In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. 1985, pp. 291–304.
- [Halo20] The Zcash Team. *Halo2*. <https://github.com/zcash/halo2>. 2020.
- [Plonky22] PolygonZero. *Plonky2*. 2022. URL: <https://github.com/0xPolygonZero/plonky2>.

- [Plonky23] Polygon. *Plonky3*. 2023. URL: <https://github.com/Plonky3/Plonky3>.
- [SP124] Succinct Labs. *Introducing SP1: A performant, open-source, contributor-friendly zkVM*. Feb. 2024. URL: <https://blog.succinct.xyz/introducing-sp1/>.
- [Ben17] Juan Benet. *Filecoin: A Decentralized Storage Network*. Protocol Labs. 2017. URL: <https://filecoin.io/filecoin.pdf>.
- [Wu23] Howard Wu. *The Aleo Whitepaper*. <https://hackmd.io/@aleo/HJRwnnhM6>. Oct. 2023.
- [KB23] Assimakis Kattis and Joseph Bonneau. “Proof of necessary work: succinct state verification with fairness guarantees”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2023, pp. 18–35.
- [BTT80] James Buchanan, Robert Tollison, and Gordon Tullock. “Toward a theory of the rent-seeking society”. In: *Texas A&M University economics series* (1980).
- [LS20] Jacob D Leshno and Philipp Strack. “Bitcoin: An axiomatic approach and an impossibility theorem”. In: *American Economic Review: Insights* 2.3 (2020), pp. 269–286.
- [BGR24] Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. “Centralization in block building and proposer-builder separation”. In: *arXiv preprint arXiv:2401.12120* (2024).
- [NGR24] Michael Neuder, Pranav Garimidi, and Tim Roughgarden. *On block-space distribution mechanisms*. <https://ethresear.ch/t/on-block-space-distribution-mechanisms/19764/1>. June 2024.

## A Network Architecture

### A.1 Data Storage

In practice, the program and input data can be very large in size and therefore difficult to store on a blockchain. To account for this, the network makes use of external ephemeral storage where request data can be stored. The artifact ids of the relevant data in ephemeral storage can be used as a proof of data availability and to query the `program` and `stdin` required for proving. With the emergence of large scale, onchain data storage, over time, all data required to generate a proof can be trustlessly transferred from requesters to all the provers.



## A.2 Relayers

Relayers are third parties involved in the network to improve the user experience and to give provers additional information they may need to fulfill proofs.

**Simulating programs.** Relayers simulate programs to compute `public_values`, which are public inputs associated to the request. This simulation step prevents a prover from spending resources on invalid or misconfigured requests. Further, the upper bound on the cycle count, `max_cycles`, inputted by the user may be overly pessimistic. Relayers execute the program to compute exact cycle counts  $s_j$  before provers to ensure that the amount of work taken up by the proof does not exceed the maximum cycle count. The presence of the relayer prevents several attack vectors that cause delays and inefficiencies in proving, such as a prover realizing partway through proving that a given request is not possible to fulfill or that the realized cycle count is higher than expected.

**Relayer payments.** Since relayers provide a commodity service for all the requests that enter the network, they are paid a flat fee for posting the results of simulation and cycle counts to the network. In practice, program simulation is a small fraction of the cost of generating a proof. Therefore, permissionless entry into the relayer role combined with a flat fee ensures that provers have access to the information they need and that the network progresses on proof fulfillment. To ensure that relayers truthfully report the results of the simulation step and the cycle counts, they stake collateral. Since provers also go through the simulation step while proving, it is simple to check that a relayer misreported their outputs.

## B Prover Competition

The proof contest mechanism incentivizes provers to compete on the bids that they submit to the auction. In the setting where these bids provide rebates to users over time, we can compute the effective price the user receives by  $f - \sum_{k=1}^m b_k^*$ . To illustrate how this price deviates from a model in which there is a competitive auction between the provers with no preference for decentralization, we compare proof contests to a second price reverse auction. In this auction, the provers simply compete to lower the price of the user's trade as much as possible; the lowest bidding prover wins at the second lowest bid. Due to the incentive compatibility of the second price auction, in equilibrium, the requester's effective price will be  $f - (f - c_2s) = c_2s$ , the cost of the second best prover. The excess price due to supporting a decentralized prover set, therefore, can be calculated as

$$E(c_1, \dots, c_m) = f - \frac{m-1}{m} H(f - c_1s, \dots, f - c_ms) - c_2s.$$

This quantity is close to zero if the marginal costs of the provers are very similar, but has the potential to be large if there are a few inefficient provers.

## C Setting Fees

We consider a model for setting fees with two provers (Alice and Bob) who bid in a proof contest with  $\alpha = 1$  for a request with cycle count  $s$ , with marginal proving costs  $c_1$  and  $c_2$ , respectively. The utility of a prover bidding  $b_i$  in a proof contest is then given by

$$\frac{b_i}{b_1 + b_2}(f - c_i s) - b_i.$$

As calculated in [NGR24], the equilibrium bids  $b_1^*(f)$  and  $b_2^*(f)$  as a function of the fee are given by

$$b_1^*(f) = \frac{(f - c_1 s)^2 (f - c_2 s)}{(2f - c_1 s - c_2 s)^2}, \quad b_2^*(f) = \frac{(f - c_2 s)^2 (f - c_1 s)}{(2f - c_1 s - c_2 s)^2},$$

and the effective price of the proof is

$$f - b_1^*(f) - b_2^*(f) = f - \frac{(f - c_1 s)^2 (f - c_2 s)}{(2f - c_1 s - c_2 s)^2} - \frac{(f - c_2 s)^2 (f - c_1 s)}{(2f - c_1 s - c_2 s)^2}. \quad (3)$$

Suppose we want to find the smallest fee that the user can pay while maintaining both provers in the auction. We can do this by taking the derivative of (3) with respect to  $f$  and setting it equal to zero:

$$\frac{2(f^2 - f c_1 s - f c_2 s + c_1 c_2 s^2)}{4f^2 - 4f c_1 s - 4f c_2 s + c_1^2 s^2 + 2c_1 c_2 s^2 + c_2^2 s^2} = 0.$$

Setting  $f^2 - f c_1 s - f c_2 s + c_1 c_2 s^2 = 0$ , we get  $(f - c_1 s)(f - c_2 s) = 0$ , which gives us the roots  $f = c_1 s, c_2 s$ . The fee that minimizes the price experienced by the user while preserving both provers in the auction is therefore the cost of the second best prover. Intuitively, this means that the user can minimize the effective price and receive proofs from a decentralized set of provers by incentivizing the second best prover to compete with the best prover.

## D Sybil Resistance

Is it beneficial for a prover to split their bid  $b_i$  into multiple bids? We demonstrate that the prover cannot profit by doing this in a proof contest for any  $\alpha \geq 1$ . Given a prover's bid  $b_i$  and fixing  $S = \sum_{k \neq i} b_k^\alpha$ , the sum of other provers' weighted bids, the expected utility of the prover bidding in a proof contest for fee  $f$  with cycle count  $s$  is

$$u_i(b_i, b_{-i}) = \frac{b_i^\alpha}{\sum_{k=1}^m b_k^\alpha} (f - c_i s) - b_i \quad (4)$$

$$= \frac{b_i^\alpha}{b_i^\alpha + S} (f - c_i s) - b_i. \quad (5)$$

Suppose the prover decides to split the bid  $b_i$  into  $M$  bids of size  $\frac{b_i}{M}$ . The expected utility of the prover for this modified bid profile, which we denote by  $\tilde{b}_i$ , is

$$u_i(\tilde{b}_i, b_{-i}) = M \left( \frac{\left(\frac{b_i}{M}\right)^\alpha}{M \left(\frac{b_i}{M}\right)^\alpha + S} (f - c_i s) - \frac{b_i}{M} \right) \quad (6)$$

$$= \frac{\frac{b_i^\alpha}{M^{\alpha-1}}}{\frac{b_i^\alpha}{M^{\alpha-1}} + S} (f - c_i s) - b_i \quad (7)$$

$$= \frac{b_i^\alpha}{b_i^\alpha + SM^{\alpha-1}} (f - c_i s) - b_i, \quad (8)$$

where the last equality is by multiplying the numerator and denominator with  $M^{\alpha-1}$ . Comparing (5) to (8), we see that for any  $\alpha \geq 1$ ,

$$\begin{aligned} u_i(b_i, b_{-i}) - u_i(\tilde{b}_i, b_{-i}) &= \frac{b_i^\alpha}{b_i^\alpha + S} (f - c_i s) - \frac{b_i^\alpha}{b_i^\alpha + SM^{\alpha-1}} (f - c_i s) \\ &\geq 0. \end{aligned}$$

Therefore, provers are not incentivized to split their bids in proof contests.

## E Throughput

We consider a model with  $n$  requests and  $m$  provers, where  $m \leq n$ . Each prover can fulfill only one request. In mining, each prover randomly selects one of the requests to work on. Multiple provers can select the same request and contention can occur; only one prover fulfills the request, and the others remain idle. In contrast, proof contests mitigate this contention by assigning requests to a single prover; in this case, the maximum throughput is  $m$  proofs per time unit. We derive a formula for the net reduction in throughput due to contention for any given  $n$  and  $m$ . Each prover independently selects a request uniformly at random. The number of provers selecting a particular request  $r_j$  follows a binomial distribution:

$$\mathbb{P}(\text{Exactly } k \text{ provers select } r_j) = \binom{m}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{m-k}.$$

Contention on the request occurs when  $k \geq 2$ . The probability of contention is calculated as

$$\begin{aligned} \mathbb{P}(\text{Contention on } r_j) &= 1 - \mathbb{P}(0 \text{ provers select } r_j) - \mathbb{P}(1 \text{ prover selects } r_j) \\ &= 1 - \left(1 - \frac{1}{n}\right)^m - m \left(\frac{1}{n}\right) \left(1 - \frac{1}{n}\right)^{m-1}. \end{aligned}$$

The expected number of requests fulfilled without contention (those selected by exactly one prover) is

$$\mathbb{E}[\text{Fulfilled without contention}] = n \times \mathbb{P}(\text{1 prover selects } r_j) \quad (9)$$

$$= n \times \left[ m \left( \frac{1}{n} \right) \left( 1 - \frac{1}{n} \right)^{m-1} \right] \quad (10)$$

$$= m \left( 1 - \frac{1}{n} \right)^{m-1}. \quad (11)$$

For requests with contention, only one prover fulfills the request. The expected number of such fulfilled requests in mining is

$$\mathbb{E}[\text{Fulfilled with contention}] = n \times \mathbb{P}(\text{Contention on } r_j) \quad (12)$$

$$= n \left[ 1 - \left( 1 - \frac{1}{n} \right)^m - m \left( \frac{1}{n} \right) \left( 1 - \frac{1}{n} \right)^{m-1} \right]. \quad (13)$$

The expected number of requests fulfilled in mining is the sum of (13) and (11):

$$\begin{aligned} \mathbb{E}[\text{Fulfilled (mining)}] &= m \left( 1 - \frac{1}{n} \right)^{m-1} + n \left[ 1 - \left( 1 - \frac{1}{n} \right)^m - m \left( \frac{1}{n} \right) \left( 1 - \frac{1}{n} \right)^{m-1} \right] \\ &= n \left[ 1 - \left( 1 - \frac{1}{n} \right)^m \right]. \end{aligned}$$

Recall that the maximum throughput, achieved by proof contests, is  $m$ . The reduction in throughput due to contention is therefore

$$\begin{aligned} \text{Throughput reduction} &= 1 - \frac{\mathbb{E}[\text{Fulfilled (mining)}]}{m} \\ &= 1 - \frac{n}{m} \left[ 1 - \left( 1 - \frac{1}{n} \right)^m \right]. \end{aligned}$$