# Brevis ProverNet Whitepaper

Brevis

**v2.0**

**Abstract.**

Brevis ProverNet is a decentralized marketplace that matches diverse zero-knowledge (ZK) workloads—ranging from zkVM programs to ZK coprocessor queries and aggregation proofs—with a heterogeneous supply of provers. Two pillars distinguish Brevis. First, it is designed from day one to support heterogeneous proofs and composite, multistage proving pipelines, allowing applications to mix and match proving systems and workflows without being shoehorned into a single stack.

Second, Brevis introduces TODA, a truthful, budget-balanced, and asymptotically efficient two-sided double auction. TODA matches proving jobs to provers through continuous auction rounds, clearing demand and supply across proof types while preserving permissionless participation by both requesters and provers.

The emphasis on heterogeneous workloads is empirically motivated. The network design is informed by Brevis infrastructure operating in production across applications with divergent proving requirements, demonstrating the need for native support for multiple proof types and multistage pipelines.

To balance performance with openness, Brevis also deploys the auction logic on a dedicated rollup. Running the market contracts on their own rollup isolates auction throughput from L1/L2 congestion while keeping settlement, matching, and slashing transparent and permissionless. This dedicated rollup is designed solely to coordinate bids, assignments, and payments for proofs, while proofs themselves may target arbitrary destination chains.

## 1. Introduction

### 1.1. Current Limitations of Blockchain Computing

Blockchains were conceived as “world computers”, yet their base layers remain constrained. Each transaction must be re-executed by all validating nodes to preserve consensus, security, and determinism. The very property that grants strong integrity (full replication) also imposes a hard performance ceiling: a chain cannot, in steady state, outpace a single machine running the same workload, and its costs scale with the *number of replicas*, not with economies of scale. In practice, this replicated-execution model makes heavy, stateful analytics and personalized logic (such as per-user discount tiers) prohibitively expensive or operationally brittle on-chain.

A concrete example is the *VIP trader* experience on modern decentralized exchanges (DEXes). Hooks for dynamic fees or loyalty programs need to ask questions such as: “What was this addres’s trading volume in the past 30 days on this pool?” or “Does this trader hold sufficient units of the governance token to qualify for a discount?” Storing and recomputing such history on-chain is infeasible; scanning the chain in-contract is cost-prohibitive; and off-chain servers without cryptographic accountability reintroduce trust. Production systems therefore require a different primitive.

**The underlying cause: replicated execution.**

Every honest node must replay the same state transition function on the same inputs and arrive at the same post-state. This guarantees robust consensus but burns compute on duplicate work. Even aggressive L1/L2 scaling does not change the fundamental asymmetry: as long as validators must re-execute, heavy workloads will remain expensive and slow relative to traditional computing environments.

### 1.2. Verifiable Computing with Zero Knowledge

A practical way to decouple execution from validation is to employ *verifiable computing*: one machine (or a small cluster) performs the heavy computation off-chain, then supplies a succinct, cryptographic proof that the result is correct. Instead of re-executing the workload, the blockchain verifies the proof. Modern zero-knowledge (ZK) proof systems make this approach economically compelling as verification costs are small and predictable, while proving can scale up on specialized hardware.

ZK systems work by translating a program into constraints through arithmetization and use a polynomial commitment scheme and interactive oracle proof (IOP) techniques. The verifier checks correctness with often only a few queries and small-size state. Verification time is *sublinear* in the original computation's size (and, for many systems, essentially *constant*). This allows chains to keep consensus simple while outsourcing arbitrarily complex analytics and policy logic to off-chain provers.

In the context of DEX hooks and VIP programs, history scans, tier calculations, and eligibility checks run off-chain while a proof is fed back to the hook contract. The contract then verifies the proof and updates state accordingly, without trusting any opaque server or paying to recompute history on-chain. Uniswap v4 and other ecosystems have showcased prototypes and launches of precisely this pattern.

### 1.3. Brevis Products: Pico zkVM and the ZK Data Coprocessor

Brevis operationalizes verifiable computing through two complementary products:

- **Pico zkVM**—a modular, high-performance zero-knowledge virtual machine designed to efficiently prove general-purpose programs and compose with recursive aggregation. Pico emphasizes a minimal core with high-throughput cryptographic "coprocessors" and acceleration paths, enabling developers to target a stable VM while still capturing hardware gains.
- **ZK Data Coprocessor**—a production-grade pipeline that reads historical on-chain data across supported networks, performs off-chain computations (such as time-weighted balances, trading volume over windows, position analytics), and returns results with validity proofs. Contracts consume these proofs directly, enabling features like VIP trading programs, loyalty rewards, fee discounts, and active-liquidity policies without trusting external servers.

These components are deployed in live partnerships across DeFi, L2 ecosystems, and wallets: Uniswap v4 hook designs for VIP fee tiers, PancakeSwap and QuickSwap dynamic-fee programs, Euler's Incentra-based trustless rewards, and Linea-wide growth programs with millions of proofs and tens of thousands of unique addresses.

## 1.4. Motivation for an Open Marketplace and the Heterogeneity Problem

Production deployments revealed an important reality: *workloads are diverse*. Some jobs are low-latency and per-interaction (such as " before swap" checks for VIP pricing). Others are periodic and large-batch (such as per-epoch incentive distributions across many addresses). Still others are heavyweight VM executions, recursive aggregations or wrap/field conversions for cross-system compatibility. Hardware profiles vary as well: single-GPU rigs, multi-GPU servers, heterogeneous clusters with fast interconnects, or mixed CPU/GPU/FPGA setups. A single centralized prover cannot efficiently cover this spectrum nor provide credible neutrality.

In real deployments, heterogeneity manifests along four axes: (**i**) history reach (how far back in the chain one must scan), (**ii**) aggregation width (addresses/positions), (**iii**) proof cadence (per swap, hourly, daily epochs), and (**iv**) latency SLOs (service-level objectives). Consider the contrast: a DEX hook checking VIP eligibility must generate proofs within 2-3 seconds, scanning 30 days of history for a single address. Meanwhile, a protocol-wide incentive distribution processes 100,000 addresses monthly, prioritizing throughput over latency. The first workload demands low-latency GPUs with fast memory access; the second benefits from CPU clusters optimized for parallel batch processing. No single hardware configuration can efficiently serve both. A monolithic "one-size-fits-all" prover stack inevitably leaves performance on the table. A STARK-first pipeline with excellent FFT throughput might excel at history attestations, whereas a SNARK-first stack with specialized elliptic-curve precompiles might shine in near real-time use cases.

To scale, Brevis advocates an *open, decentralized marketplace* for proof compute. Requesters post typed jobs with explicit constraints (deadline, maximum fee, reliability tier), and provers bid to execute them. Price discovery happens via auctions; correctness is enforced by on-chain verification; and quality-of-service emerges from economic incentives (staking, reputation, and redundancy for systemically important tasks). This structure mirrors cloud marketplaces, but with cryptographic enforcement that eliminates trusted billing ledgers and proprietary dashboards.

## 1.5. Limitations of Existing Approaches

Early proving-as-a-service offerings and cluster managers demonstrate value, yet they typically face two structural limitations:

(1) **Narrow workload support.** Many stacks optimize for one proof system, one VM, or one circuit family. They excel at their niche but cannot gracefully absorb workloads with different arithmetic, memory footprints, or aggregation strategies. When a job type changes (from a small-range history scan to a full zkVM trace with recursion for example), the system either performs poorly or hands the request off to a different silo.

(2) **Homogeneous nodes and monolithic tasks.** Schedulers that assume similar nodes struggle to decompose and route jobs across heterogeneous hardware. In practice, breaking jobs into sub-tasks (such as multi-GPU sharding/pipelining, partial proof generation, or parallel data ingestion) is essential, but requires protocol-level interfaces and incentives so that independent operators can coordinate without mutual trust.

These constraints limit throughput and degrade time-to-proof for real applications. They also impede market efficiency by hiding specialized capacity (a farm that excels at polynomial commitments but not at Keccak-heavy traces, for instance).

## 1.6. A New Market Design: Truthful Online Double Auctions (TODA)

To address these gaps, Brevis proposes and implements a market mechanism with the properties of a *Truthful Online Double Auction* (TODA). A double auction clears a market of many buyers and many sellers simultaneously. "Online" means jobs and capacity arrive over time. TODA matches proving jobs to provers through auction rounds, where requesters bid their maximum willingness to pay and provers bid their costs. "Truthfulness (incentive compatibility)" ensures participants maximize utility by bidding their true valuations (for buyers) or true costs (for sellers).

Desirable properties include:

- **Incentive compatibility (truthfulness):** best strategy is honest bidding; reduces strategic gaming.
- **Individual rationality:** no party is forced into a loss; winners pay at most their

bid (buyers) or receive at least their ask (sellers).

- **Budget balance:** the mechanism does not require continuous subsidies; fees cover payouts.
- **Computational efficiency and low latency:** auctions clear in bounded time to honor SLA (service-level agreement) targets.

Within Brevis, job types (zkVM execution, recursive aggregation, historical data attestations, wrap/field conversions) are first-class citizens. Provers advertise capability profiles (hardware, supported systems, latency percentiles). Matching uses stake-weighted, reputation-aware scoring, with optional redundancy for critical tasks. This combination allows heterogeneous nodes to collaborate on decomposed jobs while competing fairly across categories.

## 1.7. Performance Context: Pico Prism and Real-Time Proving

The viability of a compute marketplace improves as single-job latency drops. Brevis's *Pico Prism* demonstrates a multi-GPU, distributed architecture for zkVM proving that materially advances the state of the art: in tests on current Ethereum L1 with a 45M gas cap, 99.6% of blocks were proven under 12 seconds and 96.8% under 10 seconds (the Ethereum Foundation's real-time bar), with an average of about 6.9 seconds on a 64×RTX 5090 cluster.[1]

## 1.8. Token as Market Plumbing: The Role of BREV

The marketplace requires a settlement medium and collateral. The **BREV** token plays three operational roles (detailed in Section 5). First, it is the *payment medium*: all job fees, verification, and settlement within Brevis ProverNet are denominated in BREV. Second, it functions as *staking collateral*: provers stake BREV to enter auctions and to guarantee service-level objectives; misbehavior or missed deadlines are penalized via slashing. Third, BREV also serves as the *gas token* for on-network transactions (job submissions, bids, proofs, staking updates). Finally, BREV also serves as governance token to adjust certain network parameters. This design links real compute and blockspace demand to sustained demand for BREV, while translating reliability into an economically enforced service property.

## 1.9. Extended Motivation: What Users Actually Want

From the end user's perspective, "blockchain UX" is about *fast, cheap, and fair* interactions. Traders want fee discounts to apply *before* their swap confirms; lenders want rewards that land on time and are computed according to transparent, auditable rules; wallets want to surface account insights without leaking private data. None of these experiences require every validator to replay a multi-second analytics query on-chain. They *do* require a tamper-proof certificate that the query was executed correctly.

**Constant-time verification as a UX primitive.**

A recurring theme in this paper is that constant-time verification becomes a UX building block: once an application expresses its business rule as a verifiable computation, the chain can enforce that rule with small, predictable gas, regardless of the dataset size. This turns previously "prohibitively expensive" ideas (personalized fees, snapshot-based airdrops with fairness guarantees, or multichain reputation) into routine patterns.

## 1.10. Running Example: VIP Trader Programs

To ground the discussion, consider a VIP program on a DEX pool. The goal is to dynamically apply fee tiers to a swap based on the trader's last-30-day volume in that *specific* pool (or across a set of pools), plus optional token-holder discounts for governance token balances. A straightforward implementation would need to read thousands of historical events per address, aggregate volume, check membership tiers, and make the result available within the transaction's execution window. Direct on-chain implementation is infeasible: storing and recomputing such history incurs prohibitively high gas costs, and off-chain servers without cryptographic proofs reintroduce trust assumptions.

Under a ZK coprocessor architecture, the hook contract remains small and deterministic. The coprocessor performs the heavy computation off-chain, produces a proof, and provides the hook with a succinct witness that can be verified in constant time.

## 1.11. From Engineering to Economics

The *engineering* side asks: can we produce proofs fast enough? Results from Pico Prism and Brevis production systems indicate that, for many classes of workloads, we can.

The *economics* side asks: can we allocate the right hardware to the right job, at the right price, under deadlines, without trusting a single operator? The remainder of this paper argues that an online double auction with staking-backed reputation can achieve this allocation efficiently.

### 1.12. Outline

The remainder of this paper proceeds as follows.

Section 2 covers ZK background and recent progress, emphasizing zkVM architectures and recursive composition.

Section 3 surveys Brevis's diverse workloads and launched partnerships.

Section 4 provides a formal explanation of the double auction scheme.

Section 5 formalizes BREV's token utilities and value accrual.

## 2. Background

### 2.1. Recent Progress in ZK (2024–2025)

Progress in zero-knowledge proving has accelerated dramatically across multiple fronts. zkSNARK systems offer short proofs and fast verification, often with strong engineering ecosystems, though sometimes requiring trusted setups. zkSTARKs provide transparency (no trusted setup) and post-quantum security assumptions, with larger proofs but excellent parallelism, leveraging FFT- and hash-heavy computation. Polygon's Plonky3, a modular toolkit for high-performance STARK-based systems, exemplifies recent progress with external audits and increasing adoption. Recursive frameworks like Nova employ folding schemes to enable proof composition and aggregation.

zkVMs have emerged as a critical proving primitive. A zkVM proves the correct execution of a machine model (often RISC-V or WASM). Programs compile to that ISA, and then further translate each instruction step into the zkVM constraints so that the resulting proof certifies the entire execution. zkVMs trade peak performance for flexibility: instead of handwriting circuits for each application, developers target a general VM and rely on the prover to optimize.

| **Metric** | **SP1 Hypercube** | **Pico Prism** | **Improvement** |
|---|---|---|---|
| RTP (<10s) coverage (36M gas limit) | 40.9% | 98.9% | 2.4x higher coverage |
| RTP (<10s) coverage (45M gas limit) | N/A | 96.8% | First to achieve |
| GPU cost (MSRP-based) | $256K | $128K | 50% Reduction |
| Average proving time | 10.3sec | 6.04sec | 71% Faster |

**Figure 1.** Comparison of "real-time proving" results as of October 2025

## 2.2. Real-time Ethereum Proving

The defining challenge of 2024–2025 became real-time Ethereum proving: generating validity proofs for L1 blocks fast enough to enable native rollup security and base-layer verification without sacrificing decentralization. The Ethereum Foundation's benchmark called for proving 99% of blocks in under 10 seconds with hardware under $100K.

Brevis announced *Pico Prism* in October 2025 and reported industry-leading coverage: 99.6% of current 45M-gas L1 Ethereum blocks proven in <12s, with 96.8% proven in <10s (the Ethereum Foundation's "real-time" bar) and an average proving time of ∼6.9s on consumer-grade hardware using 64 RTX 5090 GPUs. The Brevis team further reports a 50% hardware cost reduction vs. the nearest competitor for a given coverage target, yielding a 3.4× performance-per-dollar improvement.[1] Independent media recaps confirm those metrics and frame *real-time proving* as crossing a critical threshold for base-layer verification.

## 2.3. Brevis Pico and Pico Prism: Architectural Context

**Pico.** Pico is Brevis's open-source zkVM that adopts a "glue-and-coprocessor" architecture: a minimal, high-performance core plus pluggable coprocessors for heavy cryptographic operations and domain-specific accelerators.[2] The design aims to preserve programmability while letting jobs opt into specialized acceleration paths when available.

**Pico Prism.** Pico Prism extends Pico with distributed, multi-GPU parallelism and scheduling. Conceptually, Prism shards or pipelines execution traces across GPUs and aggregates partial results into a final proof, balancing memory pressure and throughput. Reported results emphasize *coverage under latency constraints* (<12s / <10s) rather than single-run records. See 1 for comparison with prior art. More details can be found at [1].

**Implications.** If real-time proofs can be produced economically, base layer verification can shift from social consensus about state transitions to cryptographic validity for most blocks. This enables lighter nodes, faster finality for rollups, and new security models for cross-chain validation.

### 2.4. Why a Networked (Market) Approach Still Matters

Even with faster zkVMs, the workload distribution is skewed: some jobs are massive (entire L1 blocks), others are latency-sensitive (bridge updates, auctions), and some are best-effort background tasks (historical aggregation). Hardware availability is similarly heterogeneous: a single workstation with one GPU is not the same as a 64-GPU rig, and specialized coprocessors may only exist in a subset of nodes.

A two-sided market (jobs $\leftrightarrow$ provers) with typed orders solves for matching:

- **Typed jobs:** zkVM execution, recursive aggregation, wrap/compress, field conversions, STARK$\leftrightarrow$SNARK wrapping, etc.
- **Constraints:** max fee, deadline, required coverage percentile (such as prove within 10s for 95% of instances), minimum hardware profile.
- **Signals:** reputation, historical latency distributions, reliability under load.

Brevis's auction design (a truthful, near-optimal double auction with typed orders) aims to clear this market while maintaining incentive compatibility for both sides. The network batches orders each round, computes allocations, and settles on-chain; unfilled users are refunded. (A full formal treatment appears later in the paper.)

## 3. Diverse Workloads

Production deployments across major DeFi protocols, L2 ecosystems, and wallet providers demonstrate the spectrum of proving workloads that Brevis infrastructure handles. This section surveys launched partnerships and the concrete jobs they run, illustrating how workload heterogeneity necessitates a marketplace architecture.

### 3.1. DEX Hooks: Personalized Fees and Data-Driven Trading

**PancakeSwap Infinity (launched May 2025)** rolled out Brevis-powered hooks that make fees *user-aware*, including (**i**) a *Token Holder Discount Hook* that applies up to 45% fee discounts based on time-weighted holdings and (**ii**) a *Trading Volume Discount Hook* that tiers fees by a trader's recent pool-specific volume.These features require scanning historical events, computing statistics off-chain, then verifying results on-chain via ZK proofs, so the router can apply the correct fee tier at swap time. Early telemetry reports show: 6,151 app requests processed, 17,277,767 proofs generated, and 3,678 unique addresses.

**Uniswap v4 Hooks (reference design; shipped examples)** demonstrated "VIP trader" fee tiers that compute last-30-day trading volume per address entirely off-chain and prove it back to the hook contract.[5] The original article details why DEXes cannot economically store/query all historical trades on-chain, and how a ZK coprocessor lets hooks remain simple while still benefiting from rich, verifiable history. The same pattern generalizes to LP loyalty rewards, volatility-aware fees, and ZK-ML assisted active liquidity management.

**QuickSwap Dynamic Fees (launched)** introduced Brevis-powered dynamic-fee hooks on Soneium (Sony's EVM L2), starting with pools such as WBTC/WETH and USDC/WETH. Traders are assigned VIP tiers by last-30-day volume (proved by Brevis) and receive discounted fees, with additional hook variants (holding-based and volatility-based) in the pipeline.

## 3.2. Uniswap v4 Router Gas Rebates (launched)

Uniswap v4 introduced heterogeneous pools where hooks can modify fee logic and execution guarantees. This heterogeneity raises integration costs for aggregators (such as 1inch and ParaSwap), as routers must detect and handle diverse pool behaviors. The Uniswap Foundation partnered with Brevis to launch trustless gas rebates for routers that process swaps through hook-enabled v4 pools, creating direct economic incentives to accelerate adoption.

The rebate system operates entirely trustlessly through Brevis's ZK Data Coprocessor. Routers submit lists of eligible transaction hashes and receive a request ID. The Brevis service fetches receipts, filters swaps by eligible pool IDs, incorporates on-chain claimer data, and generates a ZK proof. Once finalized, the router queries by request ID to retrieve the proof and calldata for on-chain claiming.

On-chain claiming occurs through `claimWithZkProof` in the rebate contract. The function enforces: (**i**) the ZK proof is valid, (**ii**) swaps in the specified block range have

| **Partner** | **Workload** | **Cadence/Scale** |
|---|---|---|
| PancakeSwap | VIP/holding fee discounts via hooks | 17.3M proofs; thousands of addrs |
| Uniswap v4 | VIP tiers; LP loyalty; ZK-ML patterns | per-user periodic updates |
| QuickSwap | Dynamic fees (volume-based) | before-swap VIP checks |
| Euler | Incentra rewards on Arbitrum | 4h epochs; $100k rEUL |
| Linea Ignition | 1B LINEA program (Etherex/Aave/Euler) | 12.1M proofs; 61.9k addrs |
| MetaMask | 2.4% APR via Aave (Linea) | 4h balance proofs |
| Uniswap v4 Rebates | Trustless gas rebates for v4 hooks | router-initiated claims; |

**Table 1.** Representative launched workloads across partners and domains.

not been rebated already, and (**iii**) the caller is the registered claimer address. When checks pass, ETH rebates are released to the specified recipient address. Depending on router policy, rebates can offset operating costs, reduce user fees, or accrue to treasuries, accelerating v4 liquidity and adoption without centralized reward ledgers or opaque calculations.

## 3.3. Trustless Incentives at Scale: Euler, Linea, MetaMask

**Euler (launched on Arbitrum)** was the launch partner for *Incentra*, Brevis's trustless incentive platform. In June 2025 Euler ran four lending reward campaigns distributing $100K in rEUL across USDC, WETH, USDT, and WBTC vaults, with rewards computed every four hours from time-weighted supply balances and verified on-chain.The program improved depth and narrowed borrowing spreads within weeks while requiring *zero* custom contracts or back-office scripts; Incentra handled data ingestion, proving, and on-chain distribution.

**Linea Ignition (launched)** ran a 10-week program distributing 1 billion LINEA tokens across Etherex, Aave, and Euler, with *all* reward computation performed off-chain and proven on-chain. Reported telemetry: 12,147,200 proofs and 61,902 unique addresses.

**MetaMask on Linea (launched)** in partnership with Brevis, powers a program offering MetaMask Card users a 2.4% fixed APR on USDC lending/borrowing via Aave on Linea. Brevis computes time-weighted balances every four hours and proves eligibility; users claim through Incentra.

### 3.4. Ecosystem Expansion: Linea, TAC, and Others

**Linea (strategic):** Beyond Ignition, Linea integrates Brevis as a general verifiable compute layer for dApps (to leverage historic data access, complex position analytics, and trustless rewards) illustrating chain-level adoption.

**TAC and Usual (launched):** Brevis is live on TAC (an EVM L1 for Telegram), with Incentra campaigns by Usual that reward time-weighted USD0++ holdings and liquidity actions. Everything is proven via ZK and verified on-chain.

**PADO (attestations):** Earlier, Brevis enabled PADO users to attest their largest Uniswap trade within a given time range and generate a proof as an on-chain credit credential, a lightweight, user-centric workload that still exercises historical data proofs.

### 3.5. Implications for Network Architecture

These production deployments span radically different computational profiles. DEX hooks demand sub-second latency for per-transaction checks, scanning limited historical ranges. Incentive distributions process hundreds of thousands of addresses over multi-day epochs, prioritizing throughput over latency. Cross-chain attestations require different proof system optimizations than recursive aggregation pipelines. A DEX user's 30-day volume calculation leverages SNARK-friendly elliptic curve operations, while a STARK-first system excels at the FFT-heavy aggregation required for epoch-based reward distributions.

No single prover configuration efficiently serves this spectrum. Hardware optimized for low-latency GPU proving underperforms on CPU-bound batch aggregation tasks. Proof systems optimized for certain arithmetics (Keccak-heavy traces versus polynomial commitment operations) leave performance on the table for other workload types. This empirical reality (validated across 124 million proofs for 94,000+ users) motivates the marketplace architecture detailed in Section 4, where specialized provers compete across typed job categories rather than a monolithic infrastructure attempting to serve all use cases.

## 4. Double Auction for Heterogeneous Prover Network

### 4.1. System Model

We consider a prover network that supports **heterogeneous proof workloads**. For example, a prover network may support the generation of ZK coprocessor proofs, zkVM proofs, aggregation proofs, among others. Without loss of generality, we assume that there are $K$ types of proof workloads that the network can support and denote by $\mathcal{K} = \{1, 2, ..., K\}$ the set of all proof workload types.

In the prover network, there is a set of proof requesters $\mathcal{N} = \{1, 2, ..., N\}$ and a set of provers $\mathcal{M} = \{1, 2, ..., M\}$. Each requester $i$ has a proof request $\boldsymbol{d}_i$ that can be decomposed into a vector of different proof job types $\boldsymbol{d}_i = (d_i^1, d_i^2, ..., d_i^K)$ where $d_i^k$ is the required number of type-$k$ proof in the request $\boldsymbol{d}_i$. We also refer to $\boldsymbol{d}_i$ as the **demand vector** of requester $i$. If the requester $i$ does not require type-$k$ proving, then $d_i^k = 0$. Each request has a deadline by which all proof jobs in the demand vector must be completed. Moreover, each request is *atomic*: either all the proof jobs in the demand vector are 100% fulfilled, or the request is deemed failed. Partial fulfillment of a request has no value.

The above heterogeneous model characterizes a wider range of realistic prover networks than prior literature [4][13], which only assumes homogeneous proving tasks.

Specifically, the heterogeneous model has the following benefits:

- First, the heterogeneous model captures the diversity of ZK applications. With recent advancement of ZK technology and the rapid growth of ZK-based dApps, proving tasks are becoming increasingly heterogeneous. Brevis ProverNet handles proving tasks from applications that use both the Brevis ZK coprocessor [3] and the Brevis Pico zkVM [2]. In the future, a prover network may even support proving tasks from multiple ZK solutions (multiple zkVMs) developed within diverse ZK frameworks.
- Second, the heterogeneous model can characterize any complex proving task that consists of multiple smaller proving sub-tasks (such as to generate an EVM-verifiable zkVM proof). This effectively enables **distributed proof generation** in the prover network and achieves finer-grained proof workload allocation, which encourages the participation of provers with limited proving capability.
- Finally, the heterogeneous model captures the diversity of provers in the net-

work. Due to differences in provers' hardware capabilities and the heterogeneous hardware requirements of proving tasks, provers may have preferences for the types of proving tasks they undertake. Provers with high memory may prefer Plonk-based proving tasks (since a large proving key must be loaded into memory in advance), while provers with limited memory but fast CPUs may prefer STARK-based proving tasks.

**Example: Heterogeneous Proof Workloads for zkVM**

Consider a request to generate an EVM-verifiable proof for a large zkVM program. In the framework of Brevis's Pico zkVM, the request can be decomposed into multiple proving sub-tasks of different types as follows:

(1) First, the VM program is divided into multiple chunks, each assigned a proof (chunk proof). Suppose the VM program is divided into 32 chunks.

(2) Then, the 32 chunk proofs each undergo a recursion step to compress their proof size (compression proof).

(3) Next, the 32 compressed chunk proofs are merged in a binary-tree fashion until a single root proof is generated (merge proof). In this example, the binary tree has 32 leaves and 31 intermediate nodes, so the number of merge proofs is 31.

(4) Then, the merged root proof is wrapped from the field used by the underlying Poseidon2 hash function to BN254, so that it can be natively verified in a Plonk proof (field-wrap proof).

(5) Finally, the wrapped proof recurses through a Plonk system (e.g., Gnark) to generate an EVM-verifiable proof (Plonk-wrap proof).

In this example, the request has five types of proving jobs, and the demand vector is $\boldsymbol{d} = (32, 32, 31, 1, 1)$.

## 4.2. Double Auction Framework

While many auction models exist in the literature (e.g., VCG auctions [14], combinatorial auctions [9, 10], all-pay auctions [6]), we consider a two-sided auction or **double auction** [11] framework that better captures the dynamics of the prover market. In this model, (1) multiple proof requesters bid for prover resources so their requests can be 100% fulfilled at some cost; (2) multiple provers compete for opportunities to serve those requests and earn rewards.

The auction proceeds in a series of rounds. At the beginning of each round, requester $i$ submits a bid $B_i^r = (\boldsymbol{d}_i, v_i)$, where $v_i$ is the requester's reported valuation for completing the job (the maximum fee the requester is willing to pay). Each prover $j$ also submits a bid $B_j^p = (\boldsymbol{w}_j, \boldsymbol{c}_j)$. Here, $\boldsymbol{w}_j = (w_j^1, w_j^2, ..., w_j^K)$ is the prover $j$'s reported supply vector, where each $w_j^k$ denotes the number of type-$k$ proof jobs the prover wishes to process[1]. $\boldsymbol{c}_j = (c_j^1, c_j^2, ..., c_j^K)$ is the reported marginal-cost vector, where each $c_j^k$ represents the cost of processing one type-$k$ proof job. If a prover does not support a particular proof type (say type-$k$ proof), then $w_j^k = 0$ and $c_j^k = \infty$.

The reported value $v_i$ from requester $i$ may differ from its true valuation $\tilde{v}_i$. Similarly, a prover's reported supply vector $\boldsymbol{w}_j$ and marginal cost vector $\boldsymbol{c}_j$ may differ from their true counterparts $\tilde{\boldsymbol{w}}_j$ and $\tilde{\boldsymbol{c}}_j$.

Finally, if a proof request is accepted in a round, the proof must be generated within that same round. In other words, the proof generation deadline coincides with the end of the round.

## 4.3. Problem Formulation

We aim to design a mechanism for the prover market that solves the following two problems.

(1) **(Proof Request Allocation)** The mechanism should decide whether each requester's bid wins the auction and is allocated the required prover resources. Moreover, the mechanism should determine the number of proof jobs (of different types) allocated to each prover. Specifically, let $\boldsymbol{x} = (x_1, x_2, ..., x_N)$ be the allocation vector for requesters, where $x_i \in \{0, 1\}$ indicates whether requester $i$ wins the auction. Let $\boldsymbol{y}_i = (y_j^1, y_j^2, ..., y_j^K)$ be the allocation vector for prover $j$ where $y_j^k$ is an integer indicating the number of type-$k$ proof jobs allocated to prover $j$.

(2) **(Pricing)** The mechanism should determine the payment each requester sends to the platform (i.e., the auctioneer). The payment vector is denoted by $\boldsymbol{p} = (p_1, p_2, ..., p_N)$. Moreover, the mechanism should determine the reward each

[1]To simplify the analysis, we assume that the deadline for each request equals the duration of an auction round so all jobs in the reported supply vector should be completed in the auction round.

prover receives from the platform. The reward vector for prover $j$ is denoted by $\boldsymbol{r}_j = (r_j^1, r_j^2, ..., r_j^K)$, where $r_j^k$ is the reward credited to prover $j$ for fulfilling the allocated type-$k$ jobs. The total reward received by prover $j$ is denoted by $r_j = \sum_{j \in \mathcal{K}} r_j^k$.

The utility function for requester $i$ is

$$U_i^r = \tilde{v}_i x_i - p_i.$$

The utility function for prover $j$ is

$$U_j^p = \sum_{j \in \mathcal{K}} U_{jk}^p = \sum_{j \in \mathcal{K}} (r_j^k - c_j^k y_j^k).$$

Define the social welfare function as

$$SW(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in \mathcal{N}} v_i x_i - \sum_{j \in \mathcal{M}} \sum_{k \in \mathcal{K}} c_j^k y_j^k. \tag{1}$$

Our target is to design a mechanism with the following properties:

- **Truthfulness**: Both proof requesters and provers should not benefit from bidding dishonestly; that is, for any requester $i$, $U_i^r(\tilde{v}_i) \geq U_i^r(v_i),\ \forall v_i$, and for any prover $j$, $U_j^p(\tilde{\boldsymbol{r}}_j, \tilde{\boldsymbol{c}}_j) \geq U_j^p(\boldsymbol{r}_j, \boldsymbol{c}_j),\ \forall \boldsymbol{r}_j, \boldsymbol{c}_j$.
- **Budget Balance**: The total payments made by all proof requesters are no less than the total rewards paid to all provers, i.e., $\sum_{i \in \mathcal{N}} p_i \geq \sum_{j \in \mathcal{M}} \sum_{k \in \mathcal{K}} r_j^k$.
- **Individual Rationality**: Both proof requesters and provers have non-negative utilities when reporting true valuations, i.e., $U_i^r(\tilde{v}_i) \geq 0,\ \forall i \in \mathcal{N}$ and $U_j^p(\tilde{\boldsymbol{r}}_j, \tilde{\boldsymbol{c}}_j) \geq 0,\ \forall j \in \mathcal{M}$.
- **Computational Efficiency**: The mechanism should run in polynomial time.
- **Asymptotic Optimality**: The mechanism should approach the maximum social welfare as the total prover supply becomes increasingly sufficient relative to the requester demands.

## 4.4. TODA: Truthful and Asymptotically Optimal Double Auction Mechanism for a Heterogeneous Prover Network

In this section, we introduce TODA, a **T**ruthful and Asymptotically **O**ptimal **D**ouble **A**uction mechanism for a heterogeneous prover network. The mechanism design is described in Sections 4.4.1 and 4.4.2. We prove that TODA achieves truthfulness, budget balance, individual rationality, computational efficiency, and asymptotic optimality in Section 4.4.3.

### *4.4.1. Proof Allocation Algorithm*

The optimal proof request allocation should maximize the social welfare (1), which can be obtained by solving the following Integer Programming (IP) problem:

$$\begin{aligned} max_{\boldsymbol{x},\boldsymbol{y}} \quad & SW(\boldsymbol{x},\boldsymbol{y}) = \sum_{i\in\mathcal{N}} v_i x_i - \sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}} c_j^k y_j^k \\ s.t. \quad & \sum_{i\in\mathcal{N}} d_i^k x_i = \sum_{j\in\mathcal{M}} y_j^k, \quad \forall k \in \mathcal{K}, \\ & x_i \in \{0,1\}, \quad \forall i \in \mathcal{N}, \\ & y_j^k \in \{0,1,...,w_j^k\}, \quad \forall j \in \mathcal{M}, k \in \mathcal{K}. \end{aligned} \tag{2}$$

**Theorem 4.1.** *The social welfare maximization problem* (2) *is NP-hard.*

***Proof.*** We consider a special case where there is only one type of job ($|\mathcal{K}| = 1$), there is only one prover ($|\mathcal{J}| = 1$), and the marginal cost $c$ of the single prover for this job type is 0. In this case, the problem (2), can be reduced to

$$\begin{aligned} max_{\boldsymbol{x}} \quad & \sum_{i\in\mathcal{N}} v_i x_i \\ s.t. \quad & \sum_{i\in\mathcal{N}} d_i x_i \le w, \\ & x_i \in \{0,1\}, \quad \forall i \in \mathcal{N}, \end{aligned} \tag{3}$$

Here, $d_i$ is requester $i$'s demand for the (only type of) job, and $w$ is the (only) prover's supply for that job. Clearly, the reduced problem (3) is a Knapsack problem, which is NP-hard [12]. As a result, the original problem (2) is also NP-hard. □

A straightforward attempt to solve for the Integer Programming problem (2) is to take its linear relaxation. However, there is no guarantee that the solution to the linear relaxation will be integral. Inspired by the "Padding" method used in [7][8], TODA introduces a "Phantom Proof Requester" (PPR), a virtual powerful requester with an unlimited budget and a demand vector $\boldsymbol{d}_{PPR} = (d^1_{PPR}, ..., d^K_{PPR})$, where $d^k_{PPR} = \max_{j\in\mathcal{M}} w^k_j$ is the PPR's demand for type-$k$ jobs. In other words, the PPR's demand vector can cover the supply vector of any single prover.

With the introduction of PPR, the proof allocation algorithm in TODA is divided into two stages.

**(Stage 1)** In the first stage, we solve a modified linear relaxation of the original Integer Programming problem (2) by introducing the PPR to the auction. Since the PPR has an unlimited budget, its demand will always be fulfilled, and thus the linear relaxation problem becomes:

$$\begin{aligned}
max_{\boldsymbol{x},\boldsymbol{y}} \quad & SW(\boldsymbol{x},\boldsymbol{y}) = \sum_{i\in\mathcal{N}} v_i x_i - \sum_{j\in\mathcal{M}}\sum_{k\in\mathcal{K}} c^k_j y^k_j \\
s.t. \quad & \sum_{i\in\mathcal{N}} d^k_i x_i + d^k_{PPR} = \sum_{j\in\mathcal{M}} y^k_j, \quad \forall k\in\mathcal{K}, \\
& 0 \le x_i \le 1, \quad \forall i \in \mathcal{N}, \\
& 0 \le y^k_j \le w^k_j, \quad \forall j\in\mathcal{M}, k\in\mathcal{K}.
\end{aligned} \tag{4}$$

Denote by $(\boldsymbol{x}', \boldsymbol{y}')$ the optimal solution to the above linear programming problem (4). For each requester $i$, define *critical valuation* $\phi_i$ as the minimum valuation (i.e., price or budget) that requester $i$ must report such that $x'_i = 1$, given other requesters' reported valuations remain unchanged. Let $\mathcal{N}^*$ denote the set of requesters whose reported valuation satisfies $v_i \ge \phi_i$. In other words, $\mathcal{N}^*$ is the set of requesters with $x'_i = 1$ in the above linear programming problem (without any bid changes). Only requesters in $\mathcal{N}^*$ proceed to the next stage.

(**Stage 2**) In the second stage, we solve a new linear programming problem involving only requesters in $\mathcal{N}^*$ and all provers $\mathcal{M}$:

$$\begin{aligned} max_{\boldsymbol{x},\boldsymbol{y}} \quad & SW(\boldsymbol{x},\boldsymbol{y}) = \sum_{i\in\mathcal{N}} v_i x_i - \sum_{j\in\mathcal{M}}\sum_{k\in\mathcal{K}} c_j^k y_j^k \\ s.t. \quad & \sum_{i\in\mathcal{N}^*} d_i^k x_i = \sum_{j\in\mathcal{M}} y_j^k, \quad \forall k\in\mathcal{K}, \\ & 0\le x_i \le 1, \quad \forall i\in\mathcal{N}^*, \\ & 0\le y_j^k \le w_j^k, \quad \forall j\in\mathcal{M}, k\in\mathcal{K}. \end{aligned} \tag{5}$$

Let $(\boldsymbol{x}'', \boldsymbol{y}'')$ be the optimal solution to the above linear programming problem (5). The final allocation by $\mathsf{TODA}$ is

$$\begin{aligned} \boldsymbol{y}^{\mathsf{TODA}} &= \boldsymbol{y}'', \\ x_i^{\mathsf{TODA}} &= \begin{cases} x_i'', & \text{if } i\in\mathcal{N}^* \\ 0, & \text{if } i\notin\mathcal{N}^* \end{cases}. \end{aligned}$$

Following a similar analysis of the padding method [7][8], it can be shown that the solution of $\mathsf{TODA}$ is feasible. That is, the solution $(\boldsymbol{x}^{\mathsf{TODA}}, \boldsymbol{y}^{\mathsf{TODA}})$ is an integer solution.

### *4.4.2.* ***Pricing Policy***

Under $\mathsf{TODA}$, the payment made by requester $i$ to the platform equals the critical price $\phi$ if $i\in\mathcal{N}^*$ and 0 if $i\notin\mathcal{N}^*$:

$$p_i^{\mathsf{TODA}} = \begin{cases} \phi_i, & \text{if } i\in\mathcal{N}^* \\ 0, & \text{if } i\notin\mathcal{N}^* \end{cases}. \tag{6}$$

For provers, a VCG-like pricing scheme is used, where the reward received by prover $j$ is defined as the change of all other bidders' social welfare caused by prover $j$'s participation:

$$r_j^{\mathsf{TODA}} = \sum_{k\in\mathcal{K}} y_j^k c_j^k + SW(\mathcal{N}^*, \mathcal{M}) - SW(\mathcal{N}^*, \mathcal{M}\backslash\{j\}),\ \forall j\in\mathcal{M}. \tag{7}$$

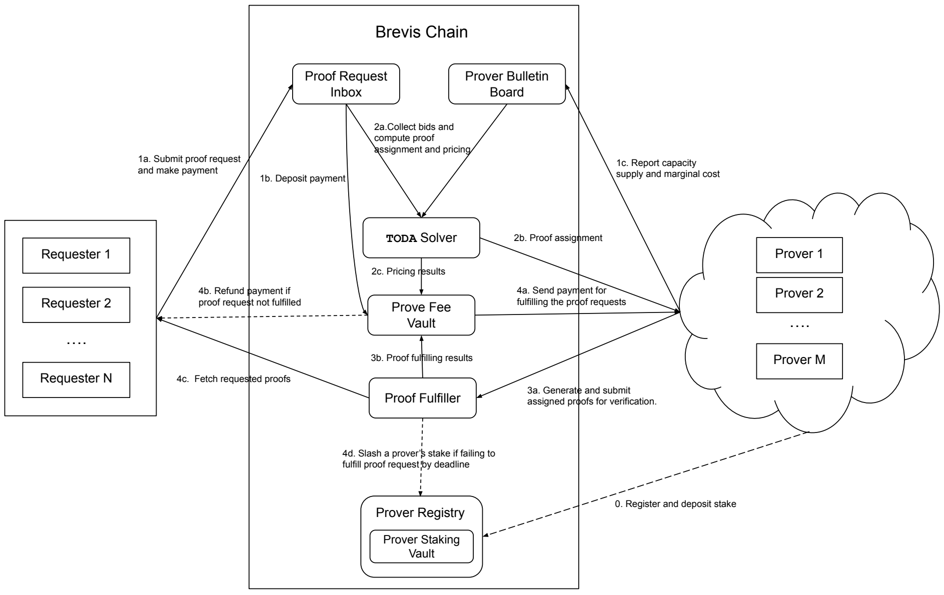
### *4.4.3. Mechanism Analysis of* TODA

We show that TODA achieves the desired economic properties as mentioned in the previous section: (1) Truthfulness, (2) Budget Balance, (3) Individual Rationality, (4) Computational Efficiency, and (5) Asymptotic Optimality. Their proofs follow the same line of arguments as the padding method used in [7][8], and are therefore omitted in this paper for brevity.

**Theorem 4.2.** *TODA is truthful, individually rational, budget-balanced, computationally tractable, and asymptotically efficient as prover supply becomes increasingly sufficient relative requester demand.*

## 4.5. Auction Implementation with Brevis Chain

In this section, we describe the implementation of the above auction with **Brevis Chain** as the **decentralized auctioneer and platform** that coordinates the auction. Figure 2 illustrates the high-level flow.



Brevis Chain
Proof Request Inbox
Prover Bulletin Board
2a.Collect bids and compute proof assignment and pricing
1a. Submit proof request and make payment
1b. Deposit payment
1c. Report capacity supply and marginal cost
Requester 1
Requester 2
....
Requester N
TODA Solver
2b. Proof assignment
2c. Pricing results
4b. Refund payment if proof request not fulfilled
Prove Fee Vault
4a. Send payment for fulfilling the proof requests
Prover 1
Prover 2
....
Prover M
3b. Proof fulfilling results
4c. Fetch requested proofs
Proof Fulfiller
3a. Generate and submit assigned proofs for verification.
4d. Slash a prover's stake if failing to fulfill proof request by deadline
0. Register and deposit stake
Prover Registry
Prover Staking Vault

**Figure 2.** High-level flow for prover network auction with *Brevis Chain*

### *4.5.1. Prover Registration*

Before participating in the proof auction, a prover must first register with the network. Any prover can join permissionlessly. Specifically, each prover should register with the *Prover Registry* contract on the Brevis Chain and deposit a required stake. The stake will be slashed if the prover is assigned a proving job but fails to submit a verified ZK proof before the deadline.

### *4.5.2. Flow for an Auction Round*

(Step 1) At the beginning of an auction round, there is a bid collection window for gathering proof requests and prover capacity reports. Each requester $i$ submits a proof demand vector $\boldsymbol{d}_i$ to the *Proof Request Inbox* contract on the Brevis Chain and places a deposit (equal to the valuation $v_i$) to the *Prove Fee Vault*. During the same window, each prover $j$ must report its supply vector $\boldsymbol{w}_j$, and marginal cost vector $\boldsymbol{c}_j$ to the Prover Bulletin Board.

(Step 2) After the bid collection window, the TODA *Solver* is triggered to compute the proof allocation and pricing for the auction (according to the mechanism described in Sections 4.4.1 and 4.4.2. The pricing results are communicated to the *Prove Fee Vault*, and the proof assignment results are sent to the provers. Note that TODA *Solver* is a **native primitive** supported by the Brevis Chain.

(Step 3) Each prover generates proofs according to the assignments from the TODA *Solver* and submits the proofs to the *Proof Fulfiller* for storage and verification.

(Step 4) If the proof is successfully verified, payment is sent to the prover based on the pricing results produced by the TODA *Solver*, and proof requesters can fetch the required proofs. If a prover fails to submit or verify the requested proofs before the auction round ends, the prover's stake will be slashed. Any requester whose request was not fulfilled will receive a refund after the auction round concludes.

The above flow provides a high-level overview of the auction implementation on the Brevis Chain, omitting details such as proof storage, privacy protection, and related mechanisms.

## 5. Token Utility

### 5.1. Overview

The **BREV** token is the core utility token of the Brevis ecosystem. It serves as the medium of payment for proving and settlement, the collateral that aligns incentives and service–level guarantees among provers, and the governance token for setting global system parameters. This section expands on those utilities and shows how they interlock to produce reliable, market–priced verifiable compute.

### 5.2. Payment Medium for Verifiable Compute

All fees associated with Brevis ProverNet are paid in **BREV**. This includes (**i**) proof generation for zkVM execution, ZK Data Coprocessor, ZKTLS Coprocessor, and recursive aggregation; (**ii**) verification and settlement on the network; and (**iii**) auxiliary services such as result availability and receipt publication. Job prices are discovered via the auction mechanism.

In addition, the Brevis ProverNet can be deployed as a specialized rollup, where BREV serves as the gas token for transactions within the network. This gives BREV additional utility tied to the gas costs payments for the network usage.

### 5.3. Staking for Provers to Receive Order Flow

Provers in the distributed network must **stake BREV** or receive delegated stake from BREV token holders to receive work.

Staking performs three roles:

(1) **Sybil resistance & admission control:** minimum effective stake gates access to sensitive or high-value jobs.
(2) **Economic alignment:** staked value is placed at risk for missed SLAs, incorrect proofs, or equivocation.
(3) **Capacity signaling:** larger effective stake (with good historical performance) pushes a prover higher in the matching queue for larger or time-critical jobs.

Applications requesting proofs will submit their proof request along with the amount of stake they require the prover to lock up to accept this request. It reflects the opportunity costs/loss if the request is not fulfilled in time with predefined verification parameters. The corresponding stake is unlocked only when a proof is successfully delivered.

Token holders may delegate BREV to professional provers. The process of delegation is a form of active participation similar to proof of stake delegation.

With these delegated tokens, a prover can therefore receive more proving workload by locking them up and earn additional revenue. In return, the prover will share a portion of the proving fees with the delegators. At the same time, if the delegated prover violates its committed SLAs, the delegated stake will also be slashed. Therefore, it is critical for delegators to carefully evaluate different proving providers and adjust their delegation from time to time.

## 5.4. Slashing for Breaking SLA

The entire Brevis system is rooted in ZK proof. There is no way for provers to generate an invalid proof that can be successfully verified by the task settlement contract.

The worst thing that can happen to disrupt the normal operation of the network is previously committed provers breaking their Service-Level Agreement, such as missing deadlines, requesting more payment and/or generating proofs with lower security level or larger size than previously committed.

If a prover broke the SLA, the proof requester may trigger the staking contract to slash the locked stake from the prover. The slashing percentage is a system parameter and is initially set as 1%. It will gradually increase according to the protocol governance process.

## 5.5. Governance for Protocol Parameters

The BREV token will also serve as the governance token for setting important system parameters through on-chain governance processes. Initially, the following parameters are governed:

(1) **Acceptable Proof Size:** Brevis ProverNet accepts smaller than 1MB proof.
(2) **Level of Security:** Brevis ProverNet accepts larger than 100-bit security for all proofs.
(3) **Slashing Percentage:** 1% of the locked stake will be slashed for provers breaking their SLA.
(4) **Auction Market Fee:** 3% of the fees paid by requesters will be allocated to

Brevis ProverNet.

## 6. Tokenomics

The BREV token serves as the core utility and value-accrual asset of the Brevis ecosystem, designed to sustain network growth, incentivize participation, and ensure economic alignment across provers, developers, and users. The total supply of BREV is fixed at 1,000,000,000 tokens, with allocations structured to balance long-term ecosystem health, decentralization, and operational sustainability.

The distribution of BREV tokens is designed to align incentives among contributors, users, and stakeholders:

- **Ecosystem Development (37%)**: Funds ecosystem growth, research and development, strategic partners, initial market making, and long-term protocol expansion.
- **Community Incentive (28.7%)**:Covers rewards for provers, stakers, and community contributors and developer integrations.
- **Team (20%)**: Allocated to current and future core developers and contributors of Brevis.
- **Investors (10.8%)**: Seed investors supporting development and launch of Brevis.
- **Airdrop (3.5%)**: Initial airdrops allocations of different categories to qualifying contributors and community members.

Ecosystem Development and Community Incentives vest linearly over 24 months after an initial unlock at TGE, with 14.50% and 7.50% circulating at launch respectively. 3% of the airdrops will be unlocked at TGE with the remaining 0.5% released at the 6th month after TGE. Team and Investor allocations are fully locked for the first year with no initial unlock, followed by 24-month linear vesting. At TGE, the circulating supply is 25% of the total supply.

## 7. Business Model

Brevis's business model is built on a sustainable and usage-driven economy centered around verifiable computation. Applications built on Brevis, including those using the

ZK Data Coprocessor, Pico zkVM, and, in the future, Ethereum full block proving, continuously generate workloads that require zero-knowledge proof computation. These workloads form the order flow that powers the Brevis Prover Network, a decentralized marketplace for compute.

Within this marketplace, independent provers and prover pools compete to fulfill computation requests submitted by applications. Each proving task is matched through a transparent auction process governed by the Service Level Agreement (SLA) mechanism and the TODA double auction system. Performance, reliability, and stake size determine how tasks and rewards are allocated. Provers must stake BREV tokens to participate, signaling commitment and ensuring service quality. The more BREV a prover stakes and the better their SLA record, the greater their share of proving jobs and corresponding revenue.

This model creates a self-reinforcing value loop: as more applications integrate Brevis and demand verifiable computation, the demand for BREV increases to pay for network usage and staking. In turn, staking strengthens network reliability and capacity, attracting more applications and computation workloads.

As the proving workload increases, the Brevis Prover Network can transition to a specialized rollup, ensuring scalability and efficient coordination of auctions and settlements. All transactions within this rollup, including job payments, staking, and prover rewards, are denominated in BREV tokens, embedding intrinsic utility directly into the network's economic flow. Additionally, each successful ProverNet auction contributes a small protocol fee to the Brevis protocol treasury, creating a recurring and transparent source of revenue that scales with network activity.

Through this architecture, Brevis transforms computation demand into economic value, aligning token incentives with real, measurable network usage and long-term ecosystem growth.

## 8. Roadmap

### Q4 2025

- Launch **Brevis Prover Network** testnet, onboarding third-party provers.
- Complete **Proving Grounds** user activation campaign and pre-TGE ecosystem rollout.

- Target **TGE** early December 2025.

**H1 2026**

- Achieve full **real-time Ethereum block proving** on 16 consumer-grade GPUs.
- Integrate **ZK-TLS Coprocessors** for verifiable web data and AI inference.
- Deploy **TODA auction mechanism** for decentralized prover task allocation.
- Expand adoption in **Intelligent DeFi**, **stablecoin/RWA incentive rails**, and **ZK-powered prediction oracles**.

**H2 2026**

- Explore **custom hardware acceleration (FPGA / ASIC)** for Pico zkVM.
- Scale **Brevis Prover Network** into a decentralized marketplace for global verifiable computing.
- Enshrine **Pico zkVM** into Ethereum L1 as part of the real-time proving stack.
- Launch **Brevis Ecosystem Grants** to support builders integrating verifiable computing into DeFi, AI, and cross-chain systems.

## 9. Conclusion

This white paper advances a simple thesis: the way to break through the performance and cost limits of replicated execution is to make *verifiable computing* the default interface between applications and consensus, and to supply that computing through an *open, heterogeneous marketplace.* The Brevis ProverNet operationalizes this idea by treating proof generation as a first-class market in which demand arrives as *typed jobs* and supply is a diverse set of provers with different hardware profiles and proof-system specializations. Matching occurs through an on-chain auction with clear service-level objectives and cryptographic settlement, so correctness and payment do not depend on trust in any single operator.

## References

[1] *Brevis pico prism: Real time proving*, `https://blog.brevis.network/2025/10/15/pico-prism-99-6-real-time-proving-for-45m-gas-ethereum-blocks-on-consumer-hardware/`.

[2] *Brevis pico zkvm*, `https://pico-docs.brevis.network/`.

[3] *Brevis zkcoprocessor*, `https://coprocessor-docs.brevis.network/`.

[4] *Succinct network: Prove the world's software*, `https://www.provewith.us/`.

[5] *Uniswap v4 hook: Brevis zk*, `https://blog.brevis.network/2023/11/01/uniswap-v4-hook-brevis-zk-coprocessor-data-driven-dex-experiences/`.

[6] M.R. Baye, D. Kovenock, and C.G. De Vries, *The all-pay auction with complete information*, Economic Theory 8 (1996), pp. 291–305.

[7] S. Chen, M. Liu, and X. Chen, *A truthful double auction for two-sided heterogeneous mobile crowdsensing markets*, Computer Communications 81 (2016), pp. 31–42.

[8] L.Y. Chu, *Truthful bundle/multiunit double auctions*, Management Science 55 (2009), pp. 1184–1198.

[9] S. De Vries and R.V. Vohra, *Combinatorial auctions: A survey*, INFORMS Journal on computing 15 (2003), pp. 284–309.

[10] M. Dong, G. Sun, X. Wang, and Q. Zhang, *Combinatorial auction with time-frequency flexibility in cognitive radio networks*, in *2012 Proceedings IEEE INFOCOM.* IEEE, 2012, pp. 2282–2290.

[11] D. Friedman, *The double auction market institution: A survey*, in *The double auction market*, Routledge, 2018, pp. 3–26.

[12] R.M. Karp, *Reducibility among combinatorial problems*, in *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, Springer, 2009, pp. 219–241.

[13] A. Lazzaretti, C. Papamanthou, and I. Hishon-Rezaizadeh, *Robust double auctions for resource allocation*, Cryptology ePrint Archive (2024).

[14] H.R. Varian and C. Harris, *The vcg auction in theory and practice*, American Economic Review 104 (2014), pp. 442–445.