ValueMine: A Blockchain-based System for Permissionless and Trustless Computation

Canhui Chen, Zerui Cheng, Xinze Liu, Shutong Qu, Zhixuan Fang

***Abstract*—Addressing the underutilization of decentralized computational resources, we introduce *ValueMine*, an innovative, energy-efficient blockchain system. *ValueMine* leverages practical computation to facilitate decentralized consensus, enabling permissionless and trustless computation. In *ValueMine*, computational tasks proposed by users are disseminated among miners via an on-chain marketplace. Through the novel combination of Reward-Burning Scheme, Commit-Reveal Solution Releasing Procedure, and Timeout Mechanism, we guarantee high-quality computation results for users and fair rewards for miners. The computational resources dedicated to these tasks are concurrently employed in our meticulously designed blockchain consensus protocol, termed "Proof of Valuable Work" (PoVW), which ensures equitable transaction sequencing and secure data storage within a permissionless, trustless, and decentralized network. In terms of security, our system exhibits resilience against typical attacks on similar blockchains, including the problem-constructing attack and the solution-stealing attack. We have implemented *ValueMine* by initiating a fork of Ethereum official client go-ethereum and introducing a new consensus engine "PoVW". We have operationalized this system across several distributed nodes to validate its performance and stability.**

***Index Terms*—blockchains, mechanism design, proof-of-useful-work, decentralized computing.**

I. Introduction

The increasing demand for computing resources has solidified the dominance of centralized cloud service providers like Amazon Web Services (AWS), Google Cloud, and Microsoft Intelligent Cloud (Azure) [2]. These platforms offer scalable and reliable services, but they also pose significant concerns, particularly regarding transparency, censorship, and fairness [3]. As these providers maintain control over the data and applications hosted on their servers, users are vulnerable to the risks of platform-imposed restrictions and potential data manipulation [4], [5]. Additionally, the reliability of these services entirely depends on the centralized platforms, making them susceptible to outages and policy changes that can disrupt access and operations [6].

In contrast, a vast number of underutilized personal computers (PCs) owned by individuals and organizations around the world offer a decentralized alternative [7], [8]. These decentralized computation resources not only mitigate the risks associated with centralization but also present a more cost-effective solution. For instance, platforms like *io.net* [9], which offer decentralized GPU provisioning, highlight substantial savings of up to 90% on compute costs [10].

One promising solution for building an open platform that can effectively leverage decentralized resources in a permissionless and trustless manner is through blockchain systems (e.g., [11], [12]). Blockchain, exemplified by Bitcoin [13] and Ethereum [14], operates as a decentralized ledger that establishes consensus and trust among distributed participants [15]. To coordinate the generation of decentralized records, blockchain systems employ consensus protocols such as Proof of Work (PoW) [13] and Proof of Stake (PoS) [16].

While PoW guarantees reassuring security for blockchain, it comes at a significant cost: the computation involved in PoW is energy-intensive and, beyond securing the network, largely unproductive. The tremendous energy consumption associated with PoW mining is widely recognized as a fundamental drawback of blockchain technology, posing a key challenge for its future development [17]–[19]. Rauchs et al. [20] report that Bitcoin mining consumes approximately 119.87 TWh of electricity annually, exceeding the consumption of a medium-sized country. Moreover, this energy usage continues to rise rapidly [17], [21], [22].

In an effort to address the inefficiencies of PoW, Proof of Stake (PoS) [16] has been proposed as an energy-efficient alternative. However, concerns have been raised regarding the potential issues of centralization, fairness, and reliability in PoS systems (e.g., [23] [24] [25]). Beyond PoS, various other consensus mechanisms have been proposed to improve efficiency, such as Delegated Proof of Stake (DPoS) [26], Ripple Protocol Consensus Algorithm (RPCA) [27], AlgoRand [28], and reputation-based models like Proof of Trust [29] and Proof of Prestige [30]. While these methods reduce energy consumption, they often weaken decentralization and security [31]–[33]. Another direction, Proof of Useful Work (PoUW), seeks to make mining computationally meaningful [34]–[37], yet most PoUW schemes still depend on trusted parties or support only limited problem types.

In response to the pressing challenge of energy inefficiency in blockchain technology and the underutilized potential of

A preliminary version of this submission has appeared in *Proceedings of the 7th Asia-Pacific Workshop on Networking (APNet) 2023* [1].

Canhui Chen is with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China, and Shanghai Qi Zhi Institute, Shanghai, China. E-mail: chen-ch24@mails.tsinghua.edu.cn

Zerui Cheng is with Princeton University, USA. E-mail: zerui.cheng@princeton.edu

Xinze Liu is with Shanghai Qi Zhi Institute, Shanghai, China E-mail: liuxz@sqz.ac.cn

Shutong Qu is with the School of Engineering and Technology, University of Washington Tacoma, USA. E-mail: shutongq@uw.edu

Zhixuan Fang is with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China, and Shanghai Qi Zhi Institute, Shanghai, China.

This work is supported by Tsinghua University Dushi Program and Shanghai Qi Zhi Institute Innovation Program.

Corresponding Author: Zhixuan Fang (zfang@mail.tsinghua.edu.cn).

decentralized computation resources, this paper presents a novel decentralized computing system named *ValueMine*. This system is designed to offer a superior, faster, and more cost-effective decentralized computation service. Notably, *ValueMine* operates in a decentralized and permissionless manner, ensuring accessibility and inclusivity across its network. The key idea of our paradigm is that we utilize miners' computation resources to solve actual computing tasks proposed by distributed problem proposers (i.e., users), instead of solving the meaningless hash puzzle in PoW. More importantly, we also utilize this useful computation to secure the transactions in the blockchain. We further show that *ValueMine* can efficiently defend against the double-spending attack, the problem-constructing attack, and the solution-stealing attack, maintaining security in the long term. We summarize the design principles of the system as follows.

- A miner is eligible to create a new block if and only if the miner successfully finishes a computational task (i.e., solves a "problem"), either proposed by users or the system. Thus, we utilize miners' resources for useful computation.
- We measure the miner's computational contribution by the reward of the task, and propose the *Maximum Aggregated Value protocol* (MAV) to resolve forking branches.
- Eligible miners publish blocks according to the proposed *Proof of Valuable Work* (PoVW) protocol. PoVW chooses miners with probabilities in proportion to their computational contribution to users' proposed tasks.

ValueMine is novel and unique in two aspects compared with existing systems. First, there are existing studies on Proof-of-Useful-Work (PoUW) that tried to replace PoW with useful computation, but all these works require at least one of the following strong assumptions, including a single problem proposer [38] [39], a trusted third party or hardware [40] [41], or a centralized organizer [36]. Second, in previous work about on-chain crowdsourcing (e.g., [42] [43]), the computation effort on solving user problems usually does not contribute to the security of the system. The system security relies on some existing blockchains, and the crowdsourcing work is mostly an application beyond the underlying blockchain. As a result, these systems still adopt other consensus protocols for security, which do not fully exploit the value of computation work.

To summarize, our main contributions are listed as follows.

- We propose a novel permissionless and decentralized blockchain system, *ValueMine*, that allows users to post various computation tasks and utilizes the computation in the corresponding task-solving process to establish a novel consensus mechanism in our system. Essentially, *ValueMine* is a blockchain that offers fair transaction ordering and reliable on-chain storage which is secured by useful computation and is thus environmentally friendly.
- From the perspective of users in need of computation service, *ValueMine* is a decentralized, secure, and reliable platform for matching their computation requests with appropriate computation resource holders without the interference of any trusted third party. In *ValueMine*, any computation task whose outcome is easy to validate can be initiated by users along with a quotation and get solved by the miner who claims it, and the value of each problem is determined by the computation power marketplace affiliated with our system. Compared with existing literature on Proof-of-Useful-Work, *ValueMine* offers the maximum ever flexibility to users in the sense of both problem category and pricing, whereas security, liveness, and decentralization are still guaranteed.
- We investigate the security properties of *ValueMine*. We show that the system can defend against the problem-constructing attack, and the solution-stealing attack. We also investigate the safety and liveness of PoVW protocol.
- We implement our system *ValueMine* by forking the go-ethereum repository, ensuring seamless forward compatibility with Ethereum. We also introduce the Standard User Problem (SUP) contract, serving as a standardized template for user problems. This initiative promotes consistency and user-friendliness across the platform. Furthermore, to enrich user engagement, we offer a comprehensive blockchain explorer, simplifying demonstrations and enhancing user convenience. Finally, we conduct experiments to stress the system and measure its performance in different aspects. The results validate the robustness and efficiency of our system. The core code of *ValueMine* and the SUP contract have been open-sourced.[1]

The rest of this paper is organized as follows. In Section II, we review related work. In Section III, we introduce the system architecture. In Section IV, we elaborate on the details of miners' workflow in our paradigm. In Section V, we investigate typical attacks against our designed paradigm and show that our paradigm is resilient to all of them. In Section VI, we describe the implementation of our system. In Section VII, we conduct several experiments to demonstrate the performance of our proposed paradigm. In Section VIII, we discuss several potential improvements to the system. Section IX concludes the paper.

II. Related Work

The exploration of decentralized computing resources as an alternative to traditional centralized cloud services has gained significant attention in recent years. Cai et al. [44] explore decentralized control of distributed cloud networks using generalized network flows. Ramaswamy et al. [45] propose a method for node clustering in decentralized peer-to-peer networks, enhancing network efficiency without central coordination. Singh and Chopra [46] discuss integrating the Internet of Things (IoT) with multi-agent systems for decentralized intelligence in distributed environments. Muhammed et al. [47] provide a review of distributed and mobile cloud computing, while Nguyen et al. [48] examine a decentralized service deployment platform within mobile edge computing for IoT devices. However, without the integration of blockchain systems, these decentralized cloud computing platforms still require a centralized controller to manage the dispatch of computational tasks. The presence of a centralized controller

[1] https://github.com/valuemine/valuemine-geth

introduces the risk of censorship and imposes additional trust and reliability assumptions on the central server, undermining the full potential of decentralization [49].

One of the key challenges in building decentralized computing platforms on blockchain is the inefficiency of the Proof of Work (PoW) mechanism, which consumes vast amounts of computing resources, often without contributing to meaningful computational tasks. This inefficiency poses a significant barrier to the adoption of blockchain-based decentralized computing solutions. Blockchain researchers have proposed many miner selection protocols without massive computation. Typical proposals include Proof of Stake (PoS) [16], Delegated Proof of Stake (DPoS) [26], Ripple Protocol Consensus Algorithm (RPCA) [27], AlgoRand [28], Proof of Activity [50], Proof of Prestige [30], Proof of Trust [29], Proof of User Similarity [51], Proof of Karma [52] and more. However, abandoning computational work introduces distinct security trade-offs, requiring complex economic mechanisms to mitigate vulnerabilities like long-range attack or 'nothing-at-stake' attacks [31] [32] [33].

Proof of Useful Work (PoUW) arises as another way to resolve the energy inefficiency problem, where miners' computation power can be of practical use, while still being a security assurance for consensus. One way for PoUW is to convert practical problems into hash puzzles. In theory, it can be shown that hard problems such as OV, 3SUM, APSP ([38], [39]), TSP [53] and Discrete Logarithm [54] can be solved by constructing hash puzzles in traditional PoW. However, the range of applicable problems is still rather limited, and a centralized trusted server is required to construct the hash puzzles and collect the results. Another way is to replace the hash puzzle calculation with other kinds of math problems. Its representatives include PrimeCoin (number theoretical PoW) [34] and Cuckoo cycle (graph theoretical PoW) [35]. However, their utility is restricted to specific mathematical domains, limiting their applicability for general user-centric tasks. In [55], the authors propose "Proof of Necessary Work" (PoNW), in which proof generation is an integral part of the proof-of-work used in the Nakamoto consensus. But with PoNW, we still can not leverage the energy expended in PoW towards meaningful user-centric tasks. Moreover, GridCoin [36] incentivizes miners to contribute their computational power to BOINC [56], where correctness is verified through BOINC's job replication and server-side validation. However, the verification process still relies on certain off-chain trust assumptions, specifically, the integrity of BOINC project servers and reporting oracles, rather than being entirely verified on-chain in a fully trustless manner. CoinAI [37] incorporates the mining process with neural network training, but it is vulnerable to possible attacks from dishonest miners. With the intervention of a trusted third-party (such as Intel SGX [57]), REM [40], and PoET [41] can make the miners' computation useful, but finding a trusted third-party could be difficult in a decentralized system with permissionless participation.

Researchers have also proposed many blockchain-enabled crowdsourcing systems to enable decentralized computing. Separ [58] is a multi-platform crowdsourcing system that enforces regulations in a privacy-preserving manner. SingularityNET [42] is a distributed AI service system. CrowdBC [43] is a decentralized crowdsourcing platform, and [59] further puts forward an open platform strategy that combines a blockchain-endowed token economy for finding trust in a decentralized setting. But most of the proposed systems are built on top of existing blockchains (e.g., through smart contracts or other methods) to organize the crowdsourcing [60], relying on the underlying blockchains to guarantee the system security, which are still PoW-based in many cases.

In contrast to these existing works, ValueMine addresses PoUW for general problem-solving and decentralized problem posting, aiming to utilize computational expenditure for practical tasks while maintaining robust security guarantees comparable to traditional PoW. Compared with blockchain-enabled crowdsourcing [42] [43], our paradigm is different in that we incorporate problem solving into the mining process without relying on an underlying PoW hash computation. Compared with previous works on PoUW (e.g., [40] [41] [36]), our system does not require a trusted third party, including the task allocation and the proof verification. Compared with systems in [38] [39] [53] [54] [34] [35], our system can solve a wide range of problems as long as verification is simpler than computation. In summary, while existing consensus mechanisms such as PoS [16], DPoS [26], and PoUW [34]–[37] aim to improve efficiency or usefulness, they often rely on trusted components or limited application domains. Different from these approaches, our proposed system enables fully on-chain, trustless verification of computational work through smart-contract-based scoring functions, ensuring both decentralization and practical utility. This design achieves an efficient balance between security, resource utilization, and transparency, representing a novel step beyond existing useful-work paradigms.

III. System Architecture

Our system is designed to utilize distributed computational resources to solve user-proposed computation tasks and maintain a decentralized ledger at the same time. To achieve decentralized task solving and record consensus, we propose Proof of Valuable Work (PoVW) as the mining protocol. The aggregated computing power of miners, and the reward of computing tasks paid by the users, can altogether provide a security guarantee for the system. The system architecture is shown in Figure 1.

A. Basic Setup

The proposed system operates in a permissionless and anonymous blockchain environment, where nodes communicate through a peer-to-peer (P2P) network. We assume a network model with a bounded message delay denoted by Δ. The protocol adopts the majority-honest assumption, meaning that the system remains secure as long as most of the computational power is controlled by honest nodes, thereby tolerating a limited fraction of malicious participants. Each node employs standard cryptographic primitives, including a collision-resistant hash function that ensures pre-image resistance, such that revealing only the hash value of a message preserves the

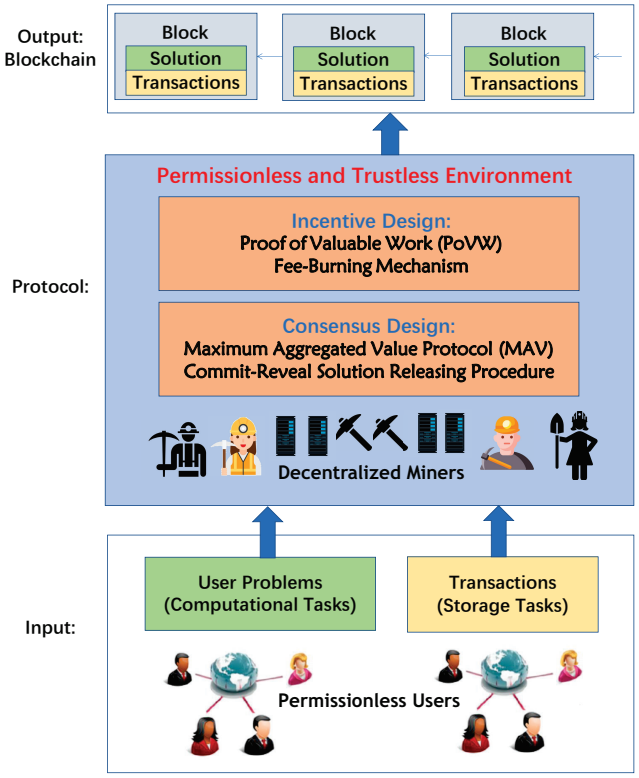


Fig. 1: System architecture of ValueMine.

integrity and confidentiality of its original content. Under these assumptions, the system guarantees fundamental security and liveness properties against common blockchain attacks.

B. Participants

The system is permissionless and anonymous. Similar to many account-based blockchains like Ethereum, each participant in the system is represented by an address associated with its asset account. There are two different roles of participants in the system, i.e., miners and users. Users are participants that need to get their computational tasks solved or to get some transactions recorded in the system. They announce the tasks to be solved or the transactions to be recorded, as well as the associated reward. Miners are the block creators in the system. They earn the reward by solving computational tasks or recording transactions requested by users.

C. Blockchain System

1) Transactions: Transactions are cryptographically signed instructions from accounts. An account will initiate a transaction to update the state of the blockchain network. The simplest transaction is transferring cryptocurrency from one account to another. In addition, the sender proposes the transaction fee to reward the miner who includes the transaction in the newly-mined block on chain.

2) Smart Contracts: Smart contracts are self-executing contracts with the terms of the agreement directly written into code, deployed and stored on the blockchain. Once deployed, a smart contract resides at a specific address and can be interacted with by sending transactions to that address. These interactions may include invoking contract functions, modifying contract state variables, or transferring assets managed by the contract.

In our system, smart contracts play a pivotal role in enabling programmable logic and decentralized applications. For example, user problems and their associated rules are formalized as smart contracts, providing an autonomous and tamper-proof framework for problem submission, solution verification, and reward distribution. The transparency and immutability of smart contracts ensure that all participants adhere to predefined protocols without the need for centralized control or third-party enforcement.

Moreover, every invocation of a smart contract function results in a transaction, and the execution incurs a transaction fee to incentivize miners and prevent abuse. The integration of smart contracts greatly expands the expressiveness and versatility of our blockchain system, allowing users to encode complex application logic directly on chain.

3) Blocks: Only when a miner successfully solves a problem, a new block can be created and appended to the end of blockchain. The miner can claim the problem reward in the block, and include transactions in the block to obtain transaction fees. Thus, a block is associated specifically with one solved problem, and contains multiple transactions. The detailed block generation process and chain rule will be introduced in Section IV.

D. Problem

In this system, a *problem* refers to a computational task amenable to solution and subsequent verification. We distinguish between two categories of computational tasks: *user problems* and *system problems*.

- *User problems*: These are proposed by users, who specify a detailed problem statement, the associated verification procedure, and the reward for successful solutions. User problems diversify the range of computations supported by the platform, reflecting the varied and potentially application-specific requirements of users.
- *System problems*: These are analogous to hash puzzles in traditional proof-of-work (PoW) protocols, where miners must identify a nonce that satisfies a predetermined, publicly known cryptographic constraint [13]. System problems are always available to miners, thereby ensuring the timely processing of transactions even in the absence of user problems. Both the problem specification and the associated reward for system problems are fixed and established by the protocol.

The formal definition of a problem is presented below.

Definition 1 (Problem)**.** *A* problem $\mathcal{P}$ *is defined as the tuple*

$$\mathcal{P} = (Id,\ S,\ \mathcal{A},\ \mathsf{score},\ R)$$

where:

- ***Identifier*** *(*Id*): A unique label distinguishing the problem instance within the system. This may be instantiated*

using, for example, the transaction hash responsible for publishing the problem, or via a monotonic counter managed by the smart contract.

- ***Specification*** *(S): A description encapsulating the computational task and all public information necessary to interpret the problem and to facilitate verification of candidate answers.*
  - *If S is sufficiently short to remain within the blockchain's transaction size and gas constraints, it may be stored directly on-chain.*
  - *For large specifications that exceed feasible on-chain storage limits, S can be stored off-chain in decentralized storage (e.g., IPFS [61]). In this case, an on-chain cryptographic commitment $\mathsf{commit}(S)$ (such as a hash or Merkle root) is maintained.*
- ***Answer Space*** *($\mathcal{A}$): The set of admissible candidate solutions to the problem. This set dictates the expected format and type of an answer, i.e., $\mathcal{A} \subseteq \Sigma^*$, where Σ is a suitable alphabet, typically $\Sigma = \{0,1\}$. The structure of $\mathcal{A}$ is determined by the problem type, e.g., for zero-knowledge proofs, $\mathcal{A}$ comprises potential proofs; for optimization, it encompasses feasible solutions.*
- ***Score Function*** *(score): A deterministic function $\mathsf{score} : \mathcal{A} \to [0,1]$, implemented as a smart contract. Given the problem specification S and a submitted answer $a \in \mathcal{A}$, $\mathsf{score}(a)$ quantifies, within the interval $[0,1]$, the quality, validity, or objective value of a with respect to S. Notably:*
  - Binary Scoring: *For problems requiring strict correctness (e.g., mathematical proofs or zero-knowledge verifications), $\mathsf{score}(a) \in \{0,1\}, \forall a \in \mathcal{A}$, where $\mathsf{score}(a) = 1$ if and only if a is a correct solution.*
  - Graded Scoring: *For problems admitting solutions of varying quality or optimality (such as in optimization problems, constraint satisfaction with soft constraints, or machine learning tasks), $\mathsf{score}(a) \in [0,1]$ reflects the performance, utility, or objective score of a.*

  Since score is executed on-chain, its computational cost must be bounded for practical deployment. Accordingly, we introduce a ***Score Evaluation Cost Constraint*** *parameter C, which sets an upper bound on the computational resources (e.g., gas consumption) permitted for any score evaluation. Formally, for all $a \in \mathcal{A}$,*

$$\mathrm{cost}(\mathsf{score}(a)) < C,$$

  where $\mathrm{cost}(\cdot)$ denotes the on-chain resource usage. This constraint ensures that score evaluations remain efficient and economically feasible, regardless of the underlying problem complexity.
- ***Reward*** *(R): The incentive offered by the problem proposer, formally $R \in \mathbb{R}_{\geq 0}$, typically denominated in tokens or cryptocurrency, to motivate the submission of valid or high-quality solutions.*

As an extension of the problem definition above, if the native score function score is computationally intensive to evaluate on-chain, or when the submitter wishes to keep certain details of a or intermediate computation private, we can convert score into a *zero-knowledge score function* $\mathsf{score}^{\mathsf{zk}}$. In this setting, the prover submits not only the solution a but also a non-interactive zero-knowledge proof π attesting to the correctness or quality of the score w assigned to a with respect to S, without revealing sensitive information. Formally, let R_S be a relation that encodes the score function, i.e.,

$$R_S = \{(a,\, w) \mid \mathsf{score}(a) = w\},$$

where $w \in [0,1]$ represents the score output. The prover generates a proof π such that

$$\mathsf{ZKProofGen}(S,\, a) \to \pi,$$

and the smart contract checks:

$$\mathsf{ZKScoreVerify}(S,\, a,\, w,\, \pi) = \begin{cases} 1, & \text{if } \pi \text{ is a valid ZK proof} \\ 0, & \text{otherwise} \end{cases}$$

The public contract then accepts w as the score, i.e.,

$$\mathsf{score}^{\mathsf{zk}}(a, \pi) = \begin{cases} w, & \text{if } \mathsf{ZKScoreVerify}(S,\, a,\, w,\, \pi) = 1 \\ 0, & \text{otherwise} \end{cases}$$

In the binary setting ($w \in \{0,1\}$), π certifies a's validity; in the graded setting ($w \in [0,1]$), π certifies the quantitative score. This approach, enabled by state-of-the-art succinct non-interactive zero-knowledge proofs (SNARKs, STARKs, etc.) [62], ensures that both privacy and efficiency can be achieved for a broad class of score functions, greatly expanding the applicability of the general problem framework.

The above definition of a problem is sufficiently flexible to accommodate a wide range of real-world computational tasks, including both user-specified computations and traditional system-level problems such as proof-of-work (PoW). Here, we provide some case studies of the problem.

a) Case I: System Problem (PoW Puzzle): The system problem (PoW puzzle) for a blockchain protocol at block height h can be formally instantiated as a tuple $\mathscr{P}_{\text{PoW}} = (Id,\, S,\, \mathcal{A},\, \mathsf{score},\, R)$, where:

- $Id = h$, representing the unique identifier associated with the current block height.
- S denotes the specification of the mining challenge. It typically includes the block header B (comprising the Merkle root, timestamp, previous block hash, etc.) and the target difficulty T.
- $\mathcal{A}$ is the solution space, here taken as the set of all possible nonces, i.e., $\mathcal{A} = \{0,1\}^n$ for a fixed length n (e.g., $n = 32$).
- score is the score function defined as:

$$\mathsf{score}(B,\, T,\, \text{nonce}) = \begin{cases} 1, & \text{if } \mathsf{Hash}(B \| \text{nonce}) < T \\ 0, & \text{otherwise} \end{cases}$$

  where $\mathsf{Hash}(\cdot)$ denotes a cryptographically secure hash function and $\|$ denotes concatenation.
- R is the protocol-determined reward for successfully solving the PoW puzzle, typically comprising a block subsidy and transaction fees.

This shows that the standard Proof-of-Work puzzle is an instance of a general problem, highlighting the model's expressive power.

b) Case II: Constraint Satisfaction and Optimization Problem: Many real-world optimization problems, such as travel itinerary planning under temporal and ordering constraints, can be framed as a CSP and mapped to the $\mathcal{P}$ framework. For example, considering a travel planning problem [63], [64]: given cities $\mathcal{V} = \{v_1, ..., v_n\}$, travel costs $c_{i,j}$, time windows $[t_i^{(start)}, t_i^{(end)}]$ for each city, an initial city v_{start}, a destination v_{end}, and a partial order constraint $\mathcal{O}$, the task is to find a route of minimal total cost visiting each city at most once and respecting all constraints. This can be specified as $\mathcal{P}_{\text{travel}} = (Id, S, \mathcal{A}, \mathsf{score}, R)$ where:

- Id identifies the instance,
- S describes nodes, costs, valid time windows, and order constraints,
- $\mathcal{A}$ is the set of candidate itineraries,
- $\mathsf{score}(a) = 1 - \frac{\text{cost}(a) - \text{cost}^*}{M}$ if a is feasible, and 0 otherwise, where $\text{cost}(a)$ is the total travel cost, cost^* is a known lower bound, M is normalization,
- R is the problem reward.

This demonstrates that our framework can also accurately model this class of constrained optimization problems with graded solution quality assessment.

c) Case III: Zero-Knowledge (ZK) Scored Machine Learning Accuracy: A representative class of problems that naturally benefit from zero-knowledge scoring is private model evaluation, such as verifying the prediction accuracy of a classifier without disclosing either the evaluation dataset or the model parameters. We denote this as $\mathcal{P}_{\text{ML}} = (Id, S, \mathcal{A}, \mathsf{score}^{\mathsf{zk}}, R)$ where:

- Id identifies the instance;
- S describes the evaluation setting, including a commitment to the hidden dataset (e.g., a Merkle root R of labeled samples (x_i, y_i)) and the deterministic rule for selecting an index subset I used for evaluation;
- $\mathcal{A}$ is the set of candidate models or classifiers submitted by miners;
- $\mathsf{score}^{\mathsf{zk}}(a, \pi)$ is the zero-knowledge score function: a solver submits a claimed accuracy $w \in [0, 1]$ together with a succinct proof π attesting that when the committed model a is evaluated on the committed samples $\{(x_i, y_i)\}_{i \in I}$, it predicts $w|I|$ samples correctly. The smart contract verifies π on-chain using $\mathsf{ZKScoreVerify}(S, a, w, \pi)$, and if the proof is valid, the score is accepted as $\mathsf{score}^{\mathsf{zk}}(a, \pi) = w$; otherwise, $\mathsf{score}^{\mathsf{zk}}(a, \pi) = 0$.
- R denotes the problem reward.

This case illustrates how the problem framework can be extended to privacy-preserving tasks. Our approach verifies the claimed result entirely on-chain through zero-knowledge proofs. Consequently, both correctness and confidentiality are ensured without redundant computation, making the verification process efficient, transparent, and economically feasible.

IV. Workflow

In this section, we introduce the mining process in *ValueMine*. Figure 2 demonstrates the workflow for users and miners.

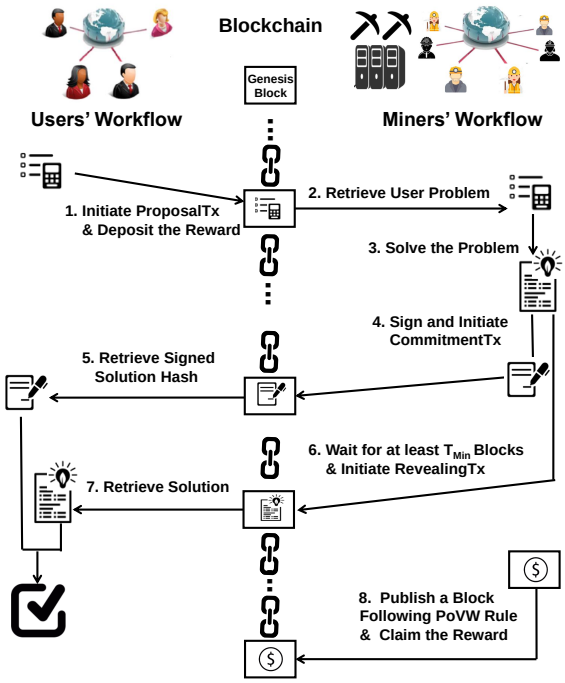


Fig. 2: Problem handling process.

A. Workflow of User Problem Proposal

To submit a computational task, a user initiates a transaction termed the *Problem Proposal Transaction* (denoted `ProposalTx`). As part of this transaction, the user must deposit the associated problem reward R into the system. In accordance with Definition 1, each proposed problem is accompanied by a well-defined score function, which is used to evaluate the quality of submitted solutions.

To incentivize the submission of high-quality solutions, we incorporate a timeout mechanism into our protocol. Specifically, each problem is assigned a search duration, denoted T_{search}, measured in the number of blocks following the proposal. During this interval, any miner may propose a candidate solution on-chain. Upon expiration of T_{search}, the solution with the highest score, as determined by the score function, is selected as the winning submission. In the case that multiple solutions achieve the same highest score, the system applies a deterministic tie-breaking rule, where the earliest on-chain submission is selected as the winner. Only the winning miner receives the problem reward, while unselected submitters, even those with identical scores, do not receive any partial reward.

It is noteworthy that, even if all submitted solutions receive a score of zero, the miner who submits the earliest such solution will still be recognized as the winner and is eligible to claim the problem reward R. If no solutions are submitted within the designated period, the problem reward remains permanently locked in the system, effectively resulting in its burn.[2]

[2]We note that alternative reward mechanisms are possible, for example, distributing rewards in proportion to the score (i.e., the winning miner receives $\mathsf{score} \times R$). However, for simplicity and clarity, this manuscript adopts the straightforward policy that the winner receives the full reward R, even in the case where the best score is zero.

This design prevents potential denial-of-service (DoS) attacks in which a malicious user could post numerous unsolvable problems with high rewards to waste miners' computational resources and discourage honest participation.

B. Reward-Burning Scheme

For each confirmed block, the miner is awarded a block reward, which comprises both the problem reward and the transaction fees. The underlying problem can be either a *system problem* or a *user problem*. We denote the problem reward as R, consistent with Definition 1. The user problem reward is specified by the user and, therefore, may vary across different problems. In contrast, the system problem reward is set by the system and can be regarded as a constant during a certain period. For notational simplicity, we denote the system problem reward as $\tilde{R}$.

Reward-Burning Mechanism. We incorporate a reward-burning mechanism into the allocation of the problem reward. Specifically, when a block is generated with an associated problem reward R (regardless of whether the problem is user-defined or system-defined), only a portion $(1-k)R$ is transferred to the miner, while the remaining portion kR is considered "burnt" by the system. Here, $k \in [0, 1)$ represents the reward-burning ratio, which is determined by the system.

The reward-burning mechanism is similar to the EIP-1559 protocol in Ethereum [65]. Such a mechanism not only helps stabilize the circulating tokens (see Section VII-C for detailed discussions), but also addresses security concerns that will be elaborated in Section V.

C. Workflow of Block Generation

Each block is associated with one problem, either the system problem or the user problem. To generate a block, the miner needs to solve a problem, either the system problem or the user problem.

1) Block Generation with System Problem: If a miner elects to solve a system problem, the task reduces to a conventional PoW (Proof of Work) mining puzzle. Specifically, the miner must identify a nonce such that

$$\text{Hash}(\texttt{nonce}, \text{candidate block}) < D_{\text{PoW}},$$

where D_{PoW} denotes the mining difficulty. Once a nonce satisfying the above condition is discovered, the miner is entitled to immediately generate a new block and claim the system problem reward $\tilde{R}$.

2) Block Generation with User Problem: If the miner solves a user problem, to prevent the solution from being stolen, the miner should follow the **Commit-Reveal Solution Releasing Procedure** to safely broadcast the solution, if the miner is the winner of the corresponding user problem, the miner can claim the reward and generate a block following the Proof of Valuable Work (PoVW) protocol.

Commit-Reveal Solution Releasing Procedure. First, the miner should initiate a *Solution Commitment Transaction* (`CommitmentTx` for brevity), which is a transaction that stores the hash digest of the miner's account address and the solution. Note that the `CommitmentTx` should be sent before the problem timeout T_{search}. Note that, with the cryptographic guarantee, other miners cannot obtain the solution by observing `CommitmentTx`. After the confirmation of `CommitmentTx`, the miner should further initiate a *Solution Revealing Transaction* (`RevealingTx` for brevity) which records the solution in consistency with the hash digest. `RevealingTx` is allowed to be published on chain only within a certain time window after the block containing the commitment transaction. Specifically, `RevealingTx` should be published $t \in [T_{\min}, T_{\max}]$ blocks after the corresponding `CommitmentTx`, where both $T_{\min}, T_{\max}$ are the system configurations. After $T_{\max}$ blocks, if no corresponding `RevealingTx` appears on chain, the `CommitmentTx` is expired, and will not be considered as a valid solution. After the problem timeout T_{search}, the miner who proposes the solution with the highest score is the winner of the corresponding problem. Particularly, when there are multiple solutions with the highest score, the system can specify a tie-breaking rule to decide the only winner, e.g., by the order of the corresponding `CommitmentTx` on chain. And the winner can generate a block following the Proof of Valuable Work (PoVW) as follows.

Proof of Valuable Work (PoVW). The winning miner is eligible to pack transactions and publish a candidate block following the Proof of Valuable Work (PoVW) protocol. Specifically, the miner can create a valid block if the candidate block satisfies the condition of PoVW:

$$\text{Hash}(\texttt{time}, \text{candidate block}) < D_{\text{PoVW}} \cdot R, \tag{1}$$

where `time` denotes the timestamp at which the candidate block is generated, R represents the reward associated with the solved problem, and D_{PoVW} is a pre-determined difficulty parameter specific to the PoVW protocol.

Unlike traditional proof-of-work (PoW) protocols in which miners must repeatedly search for an appropriate nonce, the PoVW protocol does not require such computationally intensive operations. Instead, miners await an appropriate timestamp at which they become eligible to generate a block, thereby significantly reducing energy consumption relative to conventional PoW schemes. More precisely, at each timestamp, the probability that a miner can successfully generate a valid block is given by $P_{\text{PoVW}} = D_{\text{PoVW}} \cdot R/2^{256}$, assuming a hash output length of 256 bits. Consequently, the expected waiting time for a miner to produce a valid block in the PoVW protocol can be expressed as $T_{\text{PoVW}} = 1/P_{\text{PoVW}} = 2^{256}/(D_{\text{PoVW}} \cdot R)$.

The core innovation of the PoVW protocol lies in its direct linkage between block generation probability and the reward R for solving a user-submitted problem. Unlike proof-of-stake (PoS) protocols, which rely on historical accumulated stake, or PoW protocols, which are based on computational effort, PoVW ties a miner's block production rights to the societal value of their recent contributions, as evidenced by R. Specifically, a larger R, indicating a more valuable or difficult problem solved, results in a higher probability of block generation and, therefore, a reduced expected waiting time T_{PoVW}. As the problem reward R is determined by user demand in the market, PoVW meaningfully transforms the

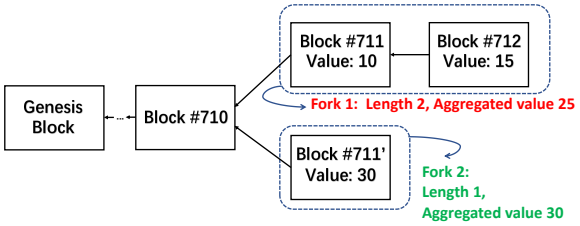


Fig. 3: Fork 2 has a larger aggregated block value.

value of a miner's useful work into the right to generate blocks, thus creating a fair and market-driven block allocation process.

D. Chain Rule: Maximum Aggregated Value

In our system, we maintain a linear and sequential chain of blocks, i.e., the "main chain", for consensus. For orphaned blocks that are not recognized in the main chain, the miners will not receive any reward, and the included transactions won't be admitted, either.

When miners see forking branches of chains, **"Maximum Aggregated Value Protocol" (MAV)** is applied as the rule for the main chain selection. Define the "value" of a block to be the reward of the associated solved problem, regardless of whether the problem is a user-proposed one or a system-generated one. According to the MAV protocol, the branch with the maximum aggregated block value should be selected as the main chain. If multiple forks have the same aggregated value, a miner can arbitrarily select one as the main chain to continue mining, and the tie is broken once another block is appended to one of the forks.

Figure 3 shows an example of two fork branches, where the branch with blocks #711, #712 has an aggregated value of 25 and the branch with block #711' has an aggregated value of 30. In this case, the branch with block #711' should be the main chain.

E. Transaction Confirmation Rule

In blockchain systems, transaction confirmation rules serve as a crucial security measure against double-spending and chain reorganization attacks. Classic examples include Bitcoin's well-known "6 blocks confirmation" rule [13], where a transaction is considered secure only after six additional blocks have been appended after the block containing the transaction. In practice, the confirmation depth can be tailored to the security requirements and value of each transaction: the greater the number of confirmations (i.e., the longer the wait), the higher the security assurance and the lower the probability that the transaction will be reverted. Inspired by these principles, we propose a flexible and value-aware transaction confirmation rule in our system. Specifically, we suggest the following criterion:

Transaction Confirmation Rule: *A transaction with value v included in a block at height h is considered confirmed only when $k \cdot \sum_{j \geq h} R_j > v$, where R_j denotes the problem reward of the block at height j, and k is a system security parameter.*

This rule requires that the cumulative weighted rewards of all blocks succeeding (and including) the transaction's block surpass a threshold determined by the transaction's value. As a result, high-value transactions must wait for proportionally more computational work in subsequent blocks, thereby enhancing their security. Conversely, low-value transactions can be confirmed after fewer blocks, improving efficiency for routine or low-risk payments.

Our confirmation rule dynamically adapts to the economic risk inherent in each transaction, thus closely aligning with real-world security needs. The approach is analogous to confirmation practices in traditional blockchain systems, where large-value transactions are typically subject to longer confirmation periods to mitigate the risk of attacks such as double spending [13].

V. Security Analysis

In this section, we discuss the system security and show resilience against the problem-constructing attack and the solution-stealing attack.

A. Defend Against Problem-Constructing Attack

Compared with classical blockchain systems where users can only initiate transactions, a unique feature in our system is that users can propose useful computation problems. Thus, it is important to address possible malicious user behaviors during problem proposal. Since a user can create multiple identities/addresses in a decentralized system, a malicious user can exploit the protocol by constructing problems as follows. The attacker can propose a problem with a known solution. Then the attacker can solve the problem with another identity to win the permission to produce a block and conduct the double-spending attack by mining a new branch.

In the following, we demonstrate that the transaction confirmation rule introduced in the previous section can effectively prevent such double-spending attacks. Even in the presence of permissionless problem proposals and adversarial attempts to revert transactions without performing genuine computations, our confirmation mechanism provides robust security guarantees.

Double-spending Attack. Double spending occurs when the miner can revert an on-chain transaction, by forcing other miners to switch to a deliberately forked chain with a conflicting transaction. In this case, the attacker can spend the money twice. We consider an attacker who tries to revert a transaction tx, by building a private chain from solving the attacker's constructed problems. Let $u_{\text{double-spending}}(tx)$ denote the attacker's payoff of double spending tx by forking a new branch of the constructed attacker problems. We have the following result.

Theorem 1. *Under the transaction confirmation rule, double spending a transaction tx by constructing attacker problems is not profitable, i.e., $u_{\text{double-spending}}(tx) < 0$.*

Proof. We first consider the case where the target transaction tx is packed in the block at height h. Let R_h denote the problem reward of the block at height h, and v_{tx} denote the value of transaction tx. Suppose that transaction tx is confirmed when the height of the latest block on the main chain is h'. Then according to the transaction confirmation rule, we have that $k \cdot \sum_{j=h}^{h'} R_j > v_{\text{tx}}$.

Assume that the attacker wants to construct user problems to mine on his private chain to revert transaction tx. Denote the aggregated problem reward in the attacker's private chain as $R_{\text{attacker}} = \sum_{j \geq h} R'_j$, where R'_j is the problem reward of the block at height j on the attacker's private chain.

According to the MAV chain rule introduced in Section IV-D, if the attacker wants to fork the main chain and revert the transaction, the aggregated value of the attacker's new branch should be larger than the current main chain, i.e., $R_{\text{attacker}} \geq \sum_{j=h}^{h'} R_j$.

Due to the reward-burning mechanism, it costs the attacker at least kR_{attacker} to solve the constructed problem, and the gained value will be v_{tx} if the attacker succeeds. Thus, the attacker's profit $u_{\text{double-spending}}$ by successfully conducting the double-spending attack is at most

$$\begin{aligned} u_{\text{double-spending}} =& v_{\text{tx}} - kR_{\text{attacker}} \\ \leq& v_{\text{tx}} - k \sum_{j=h}^{h'} R_j < 0. \end{aligned}$$

Thus, double spending by constructing attacker problems is not profitable. □

The result in Theorem 1 shows that the system can naturally prevent malicious users from gaining extra profit through double-spending in the block associated with the constructed problem.

B. Defend Against Solution-Stealing Attack

Next, we investigate the malicious behaviors of miners. The major difference between our proposed system and classic PoW systems is the proof for a miner to generate a block. In PoW, the proof is a number (nonce) that is valid only on the corresponding miner address. But in *ValueMine*, any miner that provides a solution to the user problem is eligible for block generation. Thus, a natural concern is: What happens if a malicious miner copies the published solution and generates blocks based on the stolen solution?

Note that once a `RevealingTx` is broadcast, the attacker can observe the solution and steal it. The attacker can copy and claim the solution on a deliberately created fork after the solution is revealed. If the malicious fork turns out to be the main chain, the attacker steals the solution.

Figure 4 demonstrates the solution-stealing attack with $T_{\min} = 10$. On the main chain, an honest miner solves a user problem and initiates the corresponding `CommitmentTx` which is packed in Block #953. After waiting for $T_{\min} = 10$ blocks, the miner initiates the corresponding `RevealingTx` which is packed in Block #964. Afterwards, the attacker observes the solution from the corresponding `RevealingTx` and tries to steal it as follows. The attacker deliberately creates

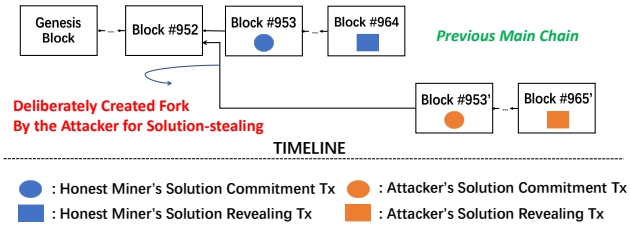


Fig. 4: Demonstration of the solution-stealing attack.

a fork starting from Block #953', following Block #952. Block #953' contains the `CommitmentTx` associated with the address of the attacker. Then, after managing to append $T_{\min}+1 = 11$ blocks on the attacker's branch, the attacker can reveal the solution in a transaction contained in Block #965'. In this case, the deliberately created fork becomes the main chain and the solution is successfully stolen by the attacker.

However, thanks to the **Commit-Reveal Solution Releasing Procedure**, our system is resistant to the attack. Note that the attacker can propose a `CommitmentTx` on the fork only after observing the `RevealingTx` on the main chain. We know that `RevealingTx` is at least T_{min} blocks after the corresponding `CommitmentTx`. Thus, when the attacker tries to steal the solution, the `CommitmentTx` on the fork is at least $T_{\min}$ blocks behind the main chain. Then the attack can succeed only if the attacker can generate a fork that is at least $T_{\min} + 1$ blocks longer than the main chain in a very short period.

To generate these blocks on the attacker's private chain, the attacker needs to solve $T_{\min} + 1$ problems, each being either a system problem or a user problem. If the attacker generates blocks by solving system problems, the security threat is the same as launching a 51% attack in the system. On the other hand, if the attacker generates blocks by solving user problems or even constructing user problems, the block generation process will be delayed by the PoVW protocol, which traps the attacker at a disadvantage in the race between the main chain and the attacker's chain. In this way, the probability that the attacker's chain catches up with the current main chain with $T_{\min}$ blocks in the system drastically decays with the value of $T_{\min}$, which means the attack hardly succeeds. This is similar to the six-block-confirmation mechanism of Bitcoin. Furthermore, note that the solutions of system problems on the main chain can never be stolen because they are associated with the solver's address. Since these solutions also contribute to the aggregated value of the main chain, they also enhance the system's capability against the solution-stealing attack.

A larger value of $T_{\min}$ provides a higher level of security, but affects the users' quality of experience, as it also indicates an increase in the latency of the solution-revealing procedure. Therefore, depending on the required security level, there is a subtle trade-off between security and users' satisfaction in the design of the system parameter $T_{\min}$.

VI. System Implementation

A. Implementation Overview

To implement our system, we initiated a fork of the Ethereum official client go-ethereum [66] repository (version 1.10.21) and introduced a new consensus engine, denoted as `PoVW` (implemented in 1500 lines of Golang). We distinguish our blockchain as PoVW-Ethereum with a unique chainID. Ensuring minimal modifications to the official client, we prioritize forward compatibility with Ethereum. Consequently, contracts and transactions executed on Ethereum can seamlessly operate within our system. Furthermore, to standardize user problems, we devised a specialized abstract contract named Standard User Problem (SUP), serving as a template for user problems. This component, implemented with 1500 lines of Solidity, ensures uniformity in the presentation of user problems. Our system incorporates various user problems and their respective solvers. To enhance user interaction, we provide a comprehensive blockchain explorer, facilitating demonstrations and user convenience.

B. Smart Contract Design

In order to standardize the user problem, we designed a special abstract contract called standard user problem (SUP) as a template for a user problem. When a user needs to propose a user problem, he needs to deploy a new contract inherited from SUP and implement the corresponding abstract function. SUP incorporates several crucial functions, each contributing to the structured execution of the problem-solving process:

- `launchProblem`: after the user launches the problem, the problem is available for miners to solve it.
- `commitAnswer`: when the miner finds an answer, he should first commit the hash of the answer in the smart contract to prevent the malicious node from stealing the answer.
- `revealAnswer`: after committing the hash of the answer, the miner can now reveal his real answer on chain. Subsequently, the miner can invoke the `calScore` function to calculate the score of his answer.
- `calScore`: is an abstract function for calculating the score of a given answer in SUP. Users are required to implement this function in their respective smart contracts.
- `Settle`: after the problem-solving period, invoking `Settle` identifies the highest-scoring answer and the corresponding miner. The function then distributes the relevant problem reward accordingly.

To initiate a user problem, the user needs to provide a detailed problem description in his contract, and implement the abstract function `calScore(uint256[] calldata answer)`. In `calScore`, the user can evaluate the quality of the provided answer. Only the highest-quality answer can obtain the problem reward.

C. Block Header Design

We add 4 attributes in the block header:

- `type`: type=0 denotes that the current block is generated by PoW (system problem). type=1 implies that the current block is generated by PoVW (user problem)

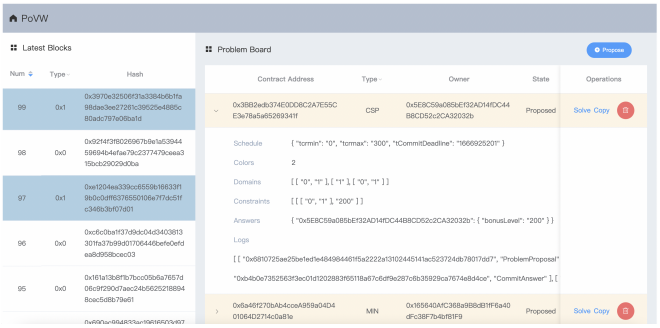


Fig. 5: System portal.

- `ProblemReward`: denotes the problem reward (system problem or user problem)
- `ProblemAddress`: if type=0, this field should be `null`, otherwise, this field denotes the address of the user problem contract.
- `SealTime`: is the timestamp when the block is generated.

D. Consensus Engine Design

Within the go-ethereum project [66], three consensus engines exist, namely `beacon`, `clique`, `ethash`, representing the PoS protocol, PoA protocol in the testnet, and the PoW protocol, respectively. To introduce a compatible design, we augment this repertoire with a new engine named `PoVW`.

1) Separation of Solver and Host: In the PoVW consensus engine, a clear distinction is made between the solver and the mining host. The mining host perseveres in mining a new block using the conventional PoW unless prompted by a PoVW block generation request or a miner-initiated halt.

On the solver side, it constantly monitors the user problem contract events. When a valuable user problem is added to the user problem pool, the miner can interrupt ongoing PoW hashing operations and switch to solving the user problem. Given the potential variability in solver implementations, we do not integrate the solver directly into Geth. Instead, we treat it as an external library. This separation not only streamlines the Geth implementation but also enhances the scalability of the solver library.

Once the solver successfully resolves the user problem, it uploads the solutions to the user problem contract and awaits the outcome. If the miner emerges as the winner of the user problem, they can initiate the generation of a new block using the PoVW protocol. At this juncture, the solver notifies the mining host to prepare for generating the PoVW block via the RPC API `povw_addUserProblemBlockReq`. This API is exclusively accessible to the miner and remains shielded from external exposure. The miner utilizes this API to inform the mining host of the successful resolution of a user problem. Upon receiving this RPC request, the mining host incorporates the block generation request into the local buffer and endeavors to generate a new block under the PoVW protocol.

2) Mining Block: The mining host consistently engages in mining new blocks using the traditional PoW, unless a PoVW block generation request is received or the miner voluntarily

ceases mining. Upon receiving a PoVW block generation request, the mining host initiates a new go-routine to handle the request. To generate a PoVW block, adherence to PoVW constraints in the block header is essential:

$$\text{Hash}(\text{sealTime}, \cdots) < D_{\text{PoVW}} \cdot \text{ProblemReward}$$

For each timestamp, the mining host updates the `sealTime` to the current timestamp and verifies whether it satisfies the PoVW constraint. If the constraint is met, the mining host generates a new block and disseminates it across the P2P network.

3) Verify Block Header: Two types of block headers exist in the system: Type-0 and Type-1. A Type-0 block header signifies a block mined under the PoW protocol, while Type-1 block headers indicate blocks mined under the PoVW protocol.

- Verifying a Type-0 block header is identical to verifying a PoW block header. The miner must confirm that the hash value of the block header is smaller than the target mining difficulty.
- Verifying a Type-1 block header involves checking:
  - The miner is the winner of the corresponding user problem, with its address specified in the block header.
  - The user problem has not been previously mined.
  - The user should verify that the `sealTime` is approximately synchronized with the local time.
  - The block header satisfies PoVW constraints, i.e.,

    $$\text{Hash}(\text{sealTime}, \cdots) < D_{\text{PoVW}} \cdot \text{ProblemReward}$$

    where D_{PoVW} is the PoVW mining difficulty.

E. User Problem and Solver

The designed system exhibits versatility in addressing a range of problems, particularly those characterized by a low verification cost. For the sake of experimental convenience, we illustrate the system's capabilities by exploring three distinct problem paradigms, serving as demonstrations of user problems. The first is the classic Constraint Satisfaction Problem (CSP), a general optimization framework that captures many practical scenarios [67]. CSP allows solutions of different qualities, by satisfying different numbers of constraints. Three different types of user problems are implemented in our experiments: graph coloring [68], Sudoku [69], and zero-one programming [70]. These problems are classic representatives of CSPs. We adopt the exhaustive random search, a general technique [71], to solve these problems. The second paradigm addresses classic optimization problems, such as identifying the minimum or maximum of a given function. To tackle these problems, we employ Newton's iteration method [72] as a solver. The third paradigm is a Zero-Knowledge-Proof (ZKP) problem that tasks the solver with demonstrating knowledge of a secret witness without revealing it. The solver translates the statement into an arithmetic circuit, computes the witness, and produces a succinct zero-knowledge proof using the Groth16 zk-SNARK construction [73]. The smart contract then verifies the proof on-chain in constant time, learning nothing about the witness while being convinced of its existence. This paradigm supports a wide range of practical problems expressible as arithmetic circuits.

F. Blockchain Explorer

As depicted in Figure 5, we have implemented a dedicated blockchain explorer for our system. This explorer serves to monitor the real-time blockchain status, meticulously recording all transactions, user problems, and blocks within the blockchain system. Additionally, the explorer provides a visual representation of the user problem pool's current status, simplifying the process for miners to select and engage with specific user problems.

VII. Experiment Results

A. Experiment Setting

The experimental evaluation was conducted on a single physical server equipped with dual-socket Intel(R) Xeon(R) Gold 5220R CPUs, each providing 24 cores, totaling 48 physical cores and 96 logical threads. The system was provisioned with 503.5 GiB of RAM and utilized a dual-storage setup: the operating system was installed on an 894 GB SAMSUNG MZ7LH960 SSD, and experimental data was stored on a 3.7 TB TOSHIBA MG04ACA4 hard disk drive. The server ran Ubuntu 20.04.6 LTS with Linux kernel version 5.4.0-174-generic, and network communication was facilitated through a 1 Gbps Ethernet interface.

The blockchain infrastructure was built using the go-ethereum (`geth`) client, version 1.10.21-stable. A containerized environment was managed via Docker Engine - Community (v24.0.7), where each blockchain node (miner) operated within an isolated Docker container. Each container was constrained to a single logical CPU thread to emulate realistic mining behavior. In total, the system hosted 25 independent miners.

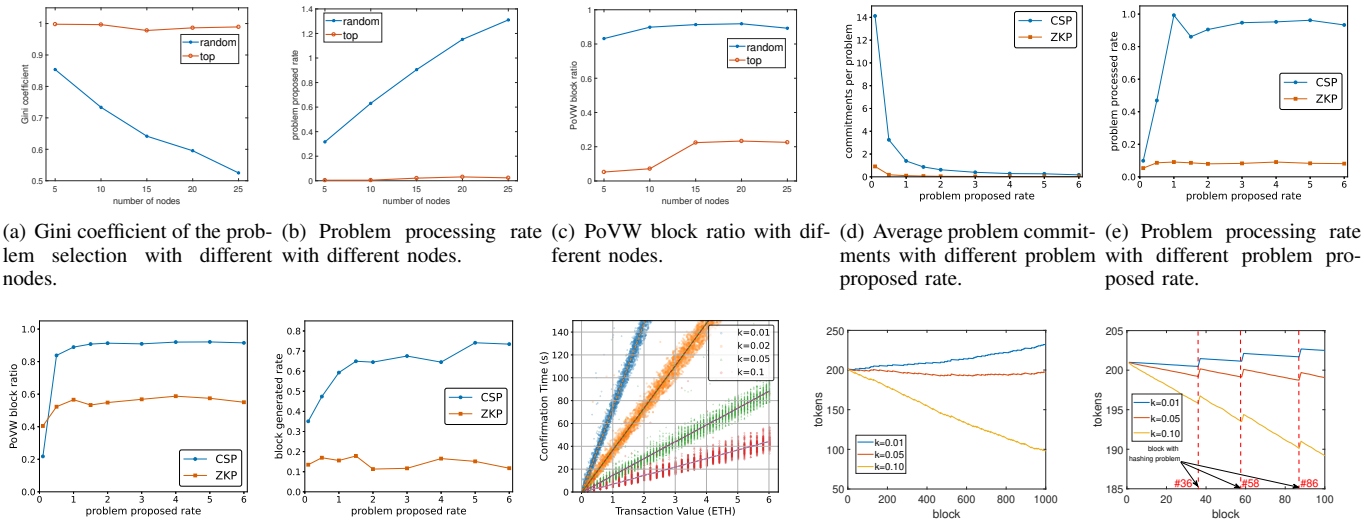
Each miner was assigned exclusive computing resources and ran independently within its container. The miners formed a fully connected mesh network topology, with node discovery disabled to maintain controlled and predictable inter-node connectivity. Communication among miners occurred over a peer-to-peer (P2P) network.

In our setup, all miners behaved honestly and strictly followed the chain rule. Each miner independently selected user problems. To simulate heterogeneous user demand, user problems were randomly generated and broadcast across the network. Additionally, transactions were also generated and broadcast to simulate transaction circulation within the blockchain network.

B. Performance Results

We initiate our experiments by first studying the influence of various problem selection strategies on system performance. Specifically, we explore two distinct problem selection strategies:

- **random selection**: Miners employ a random selection strategy, choosing a problem randomly from the user problem pool to solve.
- **top selection**: Miners adopt a top selection strategy, consistently opting to solve the problem with the highest associated reward from the user problem pool.



(a) Gini coefficient of the problem selection with different nodes.

(b) Problem processing rate with different nodes.

(c) PoVW block ratio with different nodes.

(d) Average problem commitments with different problem proposed rate.

(e) Problem processing rate with different problem proposed rate.

(f) PoVW block ratio with different problem proposed rate.

(g) Block generation rate with different problem proposed rate.

(h) Confirmation time with different transaction amount for different k.

(i) Circulating supply of token.

(j) Zoom-in view of (i) on the first 100 blocks.

Fig. 6: Results of our experiment.

Figures 6(a), 6(b), and 6(c) provide a comprehensive analysis of system performance under varying numbers of nodes and different problem selection strategies.

In Figure 6(a), the "Gini Coefficient" is employed to assess the concentration of miner selection for problem solving. A higher Gini coefficient indicates a more concentrated problem selection. It is observed that the "top selection" strategy results in all miners attempting to solve the same problem, leading to a high degree of concentration. Conversely, the "random selection" strategy demonstrates less centralization. As the number of nodes increases with the "random selection" strategy, problem selection becomes more decentralized.

Figure 6(b) depicts the problem processed rate, defined as the number of user problems solved per second. The problem processing rate exhibits an upward trend with an increase in the number of nodes, signifying a stronger system capability to solve user problems. Notably, the "random selection" strategy outperforms the "top selection" strategy. The latter leads to problem selection collisions, resulting in the wasteful consumption of computing power, a scenario effectively mitigated by the "random selection" strategy.

In Figure 6(c), the PoVW block ratio is presented, reflecting the proportion of blocks generated according to the PoVW rule on the main chain. This ratio serves as an indicator of computing resource utilization, with the ratio positively correlating with the number of nodes. As the number of nodes increases, the system's capacity to solve user problems improves, thereby enhancing its overall utilization. Additionally, the "random selection" strategy achieves around 90% utilization, while the "top selection" strategy can only attain approximately 20% utilization due to the waste of computing power on problem selection collisions.

Figures 6(d), 6(e), 6(f), and 6(g) delve into system performance under different problem proposed rates, assuming all miners adopt the "random selection" strategy in these experiments.

Figure 6(d) presents the average number of miners' commitments per user problem, indicating how often multiple miners successfully solve the same instance. The results demonstrate that when the problem rate is low, miners' selections tend to overlap, leading to intensified competition and more frequent duplicate solutions. In our experiments, the average solving time for a CSP problem is approximately five seconds, whereas a ZKP problem requires roughly two minutes. Consequently, CSP problems attract a higher number of commitments per problem, whereas ZKP problems yield significantly fewer duplicates. As the proposal rate increases, miners are afforded more distinct tasks, resulting in a decreased number of commitments per problem and thus reducing wasted duplicate work for both types of problems.

Figure 6(e) illustrates the rate at which user problems are processed. When the rate of problem proposals is low, overall system throughput is limited by this submission rate. As a result, the number of user problems processed per second increases proportionally with the problem generation rate. However, since solving ZKP proofs requires significantly more time than solving CSP problems, the processing rate for ZKP problems is correspondingly lower than that for CSP problems.

In Figure 6(f), the PoVW block ratio with different problem proposed rate is presented. We can find that the PoVW block ratio correlates positively with the problem proposed rate. As the rate of problem proposed rises, miners are better positioned to solve user problems instead of the meaningless system problems.

Figure 6(g) indicates that as the proposal rate rises, CSP miners stay busy and lift their block generation rate from roughly 0.35 to 0.74. ZKP miners gain only modestly and peak below 0.18 because the long proof time caps how many blocks they can generate.

Figure 6(h) illustrates the confirmation latency for transac-

tions of varying values. According to the transaction confirmation rule, a transaction with a larger value typically requires a longer confirmation period. Specifically, a transaction of value v_{tx} is considered confirmed once the blockchain has accumulated at least v_{tx}/k in subsequent block rewards. In our experiment, the average problem reward per block is 2 ETH. The results indicate that confirmation latency increases approximately linearly with transaction value, with higher values of k reducing the slope of this relationship. Thus, increasing k leads to shorter confirmation times, whereas larger transaction values necessitate longer confirmation periods. At the safest configuration examined in this figure, where $k = 0.05$, a payment of 1 ETH settles in approximately 15 seconds.

C. Circulating Supply of Token

In Section V, we have shown that the reward-burning mechanism is crucial for the system security. Here we further investigate the reward-burning mechanism and analyze its impact on the circulating supply of the cryptocurrency tokens in the system.

We record and plot the total tokens in the system during the generation of the first 1000 blocks in our experiment with 40 nodes and a problem generation rate of 0.2. The results are shown in Figure 6(i) and Figure 6(j).

As shown in Figure 6(i), we can tell that, when k is smaller than 0.05, the supply of tokens in the system increases. When k is larger than 0.05, the token supply decreases. Specifically, at the tipping point $k = 0.05$, tokens remain relatively stable. Figure 6(j) shows the zoom-in view of the first 100 blocks in Figure 6(i).

From Figure 6(j), we can find out that, after several blocks with user problems, there will be a block with a system problem that includes some solution commitment transactions. The block with a system problem occurs when there are no unsolved user problems, which would occasionally happen according to queueing theory results based on the configuration of our implementation.

As mentioned above, the total cryptocurrency in circulation decreases when miners mine a block with a user problem, and the total cryptocurrency in circulation increases when miners mine a block with a system problem. Therefore, from Figure 6(j), at the blocks with system problems, i.e., block #36, block #58 and block # 86, we can observe an increase in total number of circulating tokens, while the remaining blocks with user problems lead to a decrease.

Remark (Economic observation). The experiment on the different token changes after blocks with user problems and blocks with system problems sheds light on self-adaptive elasticity on token supply. When there are insufficient user problems in the system, the cryptocurrency tokens in the system will gradually increase, since miners are solving system problems. Such issuance strategy mimics the inflation of currency, which devalues the tokens and tends to lower the cost for users to propose a problem. On the contrary, when there are too many user problems in the system, the token supply decreases, which decreases the token liquidity and suppresses user demand. However, the total token supply of the system will not decrease to 0 because the system has its self-balancing mechanism as described below. When the total tokens in the system decrease, users will reduce their problem fee reward due to the token tightening. However, the reward of system problems remains stable. When the reward for user problems is no longer attractive, miners will prefer to solve the system problems. As analyzed above, the circulating supply of token increases when miners mine a block with a system problem. Thus, the token supply can fluctuate within a feasible range. In summary, the reward-burning mechanism can help regulate the token supply in the system and keep the ecosystem healthy in the long term.

VIII. Further Discussions

In this section, we discuss some possible improvements on the system implementation.

A. Problem Decomposition

A problem with high computational complexity that takes a long time to solve has several concerns. From the user's perspective, the user who proposes a very difficult problem may need to wait a long time for the solution, which leads to a bad user experience. From the miner's perspective, it is risky to solve a "big" problem, though the reward is usually high for these kinds of problems, only one miner can eventually claim the reward.

A practical approach to mitigate these challenges is problem decomposition, wherein a complex problem is divided into multiple sub-problems of lower computational complexity. This technique is both common and effective. For example, when tackling a challenging CSP (Constraint Satisfaction Problem), a user can partition the domain of the problem into several sub-domains; each sub-domain then becomes the basis for a sub-problem. These sub-problems may be addressed in parallel by independent miners across the network, allowing for distributed and more expedient problem-solving. After all sub-problems have been successfully resolved, the user can aggregate their results to obtain the final solution to the original problem. To further enhance reliability and service quality, users may also introduce redundancy by decomposing the problem into overlapping sub-problems, as seen in coded machine learning [74].

In addition to user-driven decomposition, problem-solving efficiency can be further improved through cooperative mining strategies, such as the formation of mining pools [75]. In a mining pool, multiple miners collaborate to collectively solve a complex problem or a set of sub-problems. Similar to pooled mining in cryptocurrency networks [75], participants combine their computational resources and share rewards according to their individual contributions to the pool's overall effort. Mining pools not only reduce variance in rewards but also lower individual risk and speed up the resolution of high-complexity tasks, making them an attractive option for both small and large miners. Such collaborative mechanisms enhance system scalability and ensure more consistent performance when addressing difficult computational problems.

B. DAG-based Paradigm

Directed Acyclic Graph (DAG) allows multiple blocks to be appended simultaneously, which is suitable for our system. DAG structure supports high concurrency of solution appending, and can greatly boost the system performance. As discussed in Section IV, when the temporary fork happens on the blocks that solve different user problems, the orphan blocks can be appended in the main chain, which is similar to the Inclusive protocol [76], where the transactions in the orphan blocks can be included in the main chain.

C. Problem Selection Strategy

Due to the high concurrency and the network delay, the miners usually do not have complete information of the updated problem pool and the other miners' problem selection. Thus, miners may try to solve the same problem concurrently, generating redundant blocks in the system. The collision in problem selection wastes the miner's computing power and may degrade the system performance. Similar problems also occur in the current DAG-based blockchain systems, where the miners may include the same transactions in the concurrent blocks [76]. The technologies that help to resolve the transaction inclusion collision in DAG-based blockchain such as [77] can also be adopted here to resolve the problem selection collision in the system. Besides, the collision will degrade the miner's revenue. Therefore, the rational miners are motivated to take the equilibrium selection strategy to avoid the collision [76]. This is similar to the transaction inclusion collision in the DAG-based blockchain systems [76].

D. Privacy Considerations

A potential privacy concern arises in the commit-reveal solution publishing procedure, as the solution is eventually revealed on-chain. For some applications, the public disclosure of the solution may compromise sensitive information or intellectual property. To address this, our framework supports a privacy-preserving mechanism based on zero-knowledge (zk) score functions. Specifically, if a user wishes to protect the privacy of their submitted solution, they can adopt a *zero-knowledge score function* $\mathsf{score}^{\mathsf{zk}}$. In this case, rather than requiring the miner to reveal the detailed content of the answer a on-chain, the miner instead generates and submits a non-interactive zero-knowledge proof π that attests to either the correctness or quality (score) of the solution a with respect to the problem instance S. The smart contract verifies this proof using a verifier function (e.g., $\mathsf{ZKScoreVerify}$), which checks the validity of the score w for a without learning any information about the solution itself.

After the winner is determined, the miner can securely transfer the actual solution to the problem proposer using cryptographic techniques. For example, the miner can encrypt the solution a with the public key of the user and send the ciphertext via the blockchain or any secure off-chain channel. This ensures that only the intended user can decrypt and access the solution, preserving privacy even against all other on-chain participants.

E. Theoretical Model of Token-Supply Dynamics

We consider a minimal model in which the total token supply $S(t)$ evolves according to a block production rate λ, a fixed system-puzzle reward R_{sys}, and a user-puzzle reward $R_{\text{user}}(t)$ subject to a burn ratio k. When a miner includes a user problem, they earn net reward $(1-k)R_{\text{user}}(t)$ and burn $k\,R_{\text{user}}(t)$. At each block, miners compare the minting reward R_{sys} against the net reward $(1-k)R_{\text{user}}(t)$ and choose whichever is greater. Hence the supply dynamics follow

$$\frac{dS}{dt} \;=\; \lambda \begin{cases} R_{\text{sys}}, & R_{\text{sys}} \;\geq\; (1-k)\,R_{\text{user}}(t), \\ -\,k\,R_{\text{user}}(t), & R_{\text{sys}} \;<\; (1-k)\,R_{\text{user}}(t). \end{cases}$$

According to the quantity theory of money [78], which is mathematically represented as $MV = PY$, and under the simplifying assumptions that both the velocity of money (V) and real output (Q) are constant, an increase in the money supply (M) leads directly to a proportional increase in the price level (P). Under these conditions, the relationship between the money supply and the price level is linear. By analogy, we postulate a linear relationship between user bids and supply, expressed as $R_{\text{user}}(t) = \alpha S(t)$, where $\alpha > 0$ is a sensitivity parameter. Substituting yields

$$\frac{dS}{dt} = \lambda \begin{cases} R_{\text{sys}}, & R_{\text{sys}} \;\geq\; (1-k)\,\alpha\,S(t), \\ -\,k\,\alpha\,S(t), & R_{\text{sys}} \;<\; (1-k)\,\alpha\,S(t). \end{cases}$$

The switching threshold S^* is determined by

$$(1-k)\,\alpha\,S^* = R_{\text{sys}} \quad\Longrightarrow\quad S^* = \frac{R_{\text{sys}}}{(1-k)\,\alpha}.$$

One verifies that for $S < S^*$, $dS/dt = +\lambda R_{\text{sys}} > 0$ (inflationary regime), while for $S > S^*$, $dS/dt = -\lambda k \alpha S < 0$ (deflationary regime). Thus $S(t)$ converges stably to S^*, which scales inversely with $(1-k)$. Besides, increasing the burn ratio k increases S^*, since a larger k reduces miners' net incentive from user problems and delays the transition to the deflationary regime. Conversely, decreasing k lowers the threshold S^*, causing deflation to activate earlier and capping long-run supply at a smaller value.

To target a steady-state supply band $[S_{\min}, S_{\max}]$, one requires

$$S_{\min} \;\leq\; \frac{R_{\text{sys}}}{(1-k)\,\alpha} \;\leq\; S_{\max}.$$

Rearranging gives

$$1 - \frac{R_{\text{sys}}}{\alpha\,S_{\min}} \;\leq\; k \;\leq\; 1 - \frac{R_{\text{sys}}}{\alpha\,S_{\max}}.$$

Thus the burn ratio k should be chosen within this interval to maintain supply within the desired range.

IX. Conclusion

In this paper, we propose *ValueMine*, a blockchain framework that enables decentralized Proof-of-Useful-Work with general computation tasks proposed by users in a totally permissionless and trustless environment. Then we further analyze its security, implement its prototype, and evaluate its performance, showing the practicality, feasibility, scalability, and robustness from both theory and practice.

REFERENCES

[1] C. Chen, Z. Cheng, S. Qu, and Z. Fang, "Crowdsourcing work as mining: A decentralized computation and storage paradigm," in *Proceedings of the 7th Asia-Pacific Workshop on Networking*, 2023, pp. 174–175.

[2] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet computing*, vol. 13, no. 5, pp. 10–13, 2009.

[3] S. He, Q. Tang, C. Q. Wu, and X. Shen, "Decentralizing iot management systems using blockchain for censorship resistance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 715–727, 2019.

[4] B. De Bruin and L. Floridi, "The ethics of cloud computing," *Science and engineering ethics*, vol. 23, pp. 21–39, 2017.

[5] W. K. Hon, C. Millard, and I. Walden, "The problem of 'personal data'in cloud computing: what information is regulated?—the cloud of unknowing," *International Data Privacy Law*, vol. 1, no. 4, pp. 211–228, 2011.

[6] J. Timmermans, B. C. Stahl, V. Ikonen, and E. Bozdag, "The ethics of cloud computing: A conceptual review," in *2010 IEEE second international conference on cloud computing technology and science*. IEEE, 2010, pp. 614–620.

[7] I. Mekawy, A. Alburaikan, and I. Atighi, "Exploring the feasibility of a data center-free cloud computing framework utilizing underutilized personal computers," *Big Data and Computing Visions*, vol. 3, no. 3, pp. 84–90, 2023.

[8] H. Sun, Q. Huang, P. P. Lee, W. Bai, F. Zhu, and Y. Bao, "Distributed network telemetry with resource efficiency and full accuracy," *IEEE/ACM Transactions on Networking*, 2024.

[9] "IO.NET, The Internet of GPUs," https://io.net/, 2024.

[10] "MIIX Capital: io.net Research and Analysis Report," https://medium.com/@miixcapitalen/miix-capital-io-net-research-analysis-report-3500fd6ad2b6, 2024.

[11] G. J. Mendis, Y. Wu, J. Wei, M. Sabounchi, and R. Roche, "A blockchain-powered decentralized and secure computing paradigm," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 2201–2222, 2020.

[12] D. Ayepah-Mensah, G. Sun, G. O. Boateng, S. Anokye, and G. Liu, "Blockchain-enabled federated learning-based resource allocation and trading for network slicing in 5g," *IEEE/ACM Transactions on Networking*, 2023.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[14] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[15] J. Leng, M. Zhou, J. L. Zhao, Y. Huang, and Y. Bian, "Blockchain security: A survey of techniques and research directions," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2490–2510, 2020.

[16] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, no. 1, 2012.

[17] A. De Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801–805, 2018.

[18] R. Beck, "Beyond bitcoin: The rise of blockchain world," *Computer*, vol. 51, no. 2, pp. 54–58, 2018.

[19] A. Gemajli, S. Patel, and P. G. Bradford, "Towards a low carbon proof-of-work blockchain," *arXiv preprint arXiv:2404.04729*, 2024.

[20] M. Rauchs, A. Blandin, and A. Dek, "Cambridge bitcoin electricity consumption index (cbeci)," *URL: https://cbeci. org/mining_map (accessed 2021-09-10)*, 2020.

[21] U. Gallersdörfer, L. Klaaßen, and C. Stoll, "Energy consumption of cryptocurrencies beyond bitcoin," *Joule*, vol. 4, no. 9, pp. 1843–1846, 2020.

[22] J. Truby, "Decarbonizing bitcoin: Law and policy choices for reducing the energy consumption of blockchain technologies and digital currencies," *Energy research & social science*, vol. 44, pp. 399–410, 2018.

[23] M. Haouari, M. Mhiri, M. El-Masri, and K. Al-Yafi, "A novel proof of useful work for a blockchain storing transportation transactions," *Information Processing & Management*, vol. 59, no. 1, p. 102749, 2022.

[24] P. R. Nair and D. R. Dorai, "Evaluation of performance and security of proof of work and proof of stake using blockchain," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. IEEE, 2021, pp. 279–283.

[25] M. Saad, Z. Qin, K. Ren, D. Nyang, and D. Mohaisen, "e-pos: Making proof-of-stake decentralized and fair," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 1961–1973, 2021.

[26] D. Larimer, "Delegated proof-of-stake (dpos)," *Bitshare whitepaper*, vol. 81, p. 85, 2014.

[27] D. Schwartz, N. Youngs, A. Britto *et al.*, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, p. 151, 2014.

[28] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[29] L. Bahri and S. Girdzijauskas, "When trust saves energy: a reference framework for proof of trust (pot) blockchains," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 1165–1169.

[30] M. Król, A. Sonnino, M. Al-Bassam, A. Tasiopoulos, and I. Psaras, "Proof-of-prestige: A useful work reward system for unverifiable tasks," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 293–301.

[31] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 297–315.

[32] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[33] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," in *Proceedings of the 2019 ACM Conference on Economics and Computation*, 2019, pp. 459–473.

[34] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," *July 7th*, vol. 1, no. 6, 2013.

[35] J. Tromp, "Cuckoo cycle: a memory bound graph-theoretic proof-of-work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 49–62.

[36] R. Halford, "Gridcoin: Crypto-currency using berkeley open infrastructure network computing grid as a proof of work," 2014.

[37] A. Baldominos and Y. Saez, "Coin. ai: A proof-of-useful-work scheme for blockchain-based distributed deep learning," *Entropy*, vol. 21, no. 8, p. 723, 2019.

[38] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work." *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 203, 2017.

[39] M. Ball, A. Rosen, M. Sabin, and P.-N. Vasudevan, "Proofs of work from worst-case assumptions," in *Annual International Cryptology Conference*. Springer, 2018, pp. 789–819.

[40] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. Van Renesse, "{REM}: Resource-efficient mining for blockchains," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1427–1444.

[41] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 282–297.

[42] B. Goertzel, S. Giacomelli, D. Hanson, C. Pennachin, and M. Argentieri, "Singularitynet: A decentralized, open market and inter-network for ais," *Thoughts, Theories Stud. Artif. Intell. Res.*, 2017.

[43] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "Crowdbc: A blockchain-based decentralized framework for crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251–1266, 2018.

[44] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Decentralized control of distributed cloud networks with generalized network flows," *IEEE Transactions on Communications*, vol. 71, no. 1, pp. 256–268, 2022.

[45] L. Ramaswamy, B. Gedik, and L. Liu, "A distributed approach to node clustering in decentralized peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 9, pp. 814–829, 2005.

[46] M. P. Singh and A. K. Chopra, "The internet of things and multiagent systems: Decentralized intelligence in distributed computing," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1738–1747.

[47] N. T. Muhammed, S. R. Zeebaree, and Z. N. Rashid, "Distributed cloud computing and mobile cloud computing: A review," *Qalaai Zanist Journal*, vol. 7, no. 2, pp. 1183–1201, 2022.

[48] T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4162–4175, 2018.

[49] J. Zarrin, H. Wen Phang, L. Babu Saheer, and B. Zarrin, "Blockchain for decentralization of internet: prospects, trends, and challenges," *Cluster Computing*, vol. 24, no. 4, pp. 2841–2866, 2021.

[50] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[51] S. Wang, L. Shi, H. Shi, Y. Zhang, Q. Hu, and X. Cheng, "Proof of user similarity: the spatial measurer of blockchain," *IEEE Transactions on Services Computing*, 2023.

[52] A. Biswas, R. Yadav, G. Baranwal, and A. K. Tripathi, "Proof of karma (pok): A novel consensus mechanism for consortium blockchain," *IEEE Transactions on Services Computing*, 2022.

[53] A. F. Loe and E. A. Quaglia, "Conquering generals: an np-hard proof of useful work," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, 2018, pp. 54–59.

[54] M. Hastings, N. Heninger, and E. Wustrow, "Short paper: The proof is in the pudding," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 396–404.

[55] A. Kattis and J. 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.

[56] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Fifth IEEE/ACM international workshop on grid computing*. IEEE, 2004, pp. 4–10.

[57] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[58] M. J. Amiri, J. Duguépéroux, T. Allard, D. Agrawal, and A. El Abbadi, "Separ: Towards regulating future of work multi-platform crowdworking environments with privacy guarantees," in *Proceedings of the Web Conference 2021*, 2021, pp. 1891–1903.

[59] J. Ducrée, M. Gravitt, R. Walshe, S. Bartling, M. Etzrodt, and T. Harrington, "Open platform concept for blockchain-enabled crowdsourcing of technology development and supply chains," *Frontiers in Blockchain*, vol. 3, 2020.

[60] N. Szabo, "Formalizing and securing relationships on public networks," *First monday*, 1997.

[61] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[62] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

[63] M. C. Cooper, M. de Roquemaurel, and P. Régnier, "A weighted csp approach to cost-optimal planning," *Ai Communications*, vol. 24, no. 1, pp. 1–29, 2011.

[64] J. Xie, K. Zhang, J. Chen, T. Zhu, R. Lou, Y. Tian, Y. Xiao, and Y. Su, "Travelplanner: A benchmark for real-world planning with language agents," *arXiv preprint arXiv:2402.01622*, 2024.

[65] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta, "Eip-1559: Fee market change for eth 1.0 chain," *Published online*, 2019.

[66] "ethereum/go-ethereum," https://github.com/ethereum/go-ethereum, 2023.

[67] P. Prosser, "Hybrid algorithms for the constraint satisfaction problem," *Computational intelligence*, vol. 9, no. 3, pp. 268–299, 1993.

[68] T. R. Jensen and B. Toft, *Graph coloring problems*. John Wiley & Sons, 2011.

[69] H. Simonis, "Sudoku as a constraint problem," in *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, vol. 12. Citeseer, 2005, pp. 13–27.

[70] M. W. Padberg, "A note on zero-one programming," *Operations Research*, vol. 23, no. 4, pp. 833–837, 1975.

[71] M. J. Heule and O. Kullmann, "The science of brute force," *Communications of the ACM*, vol. 60, no. 8, pp. 70–79, 2017.

[72] C. T. Kelley, *Iterative methods for optimization*. SIAM, 1999.

[73] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*. Springer, 2016, pp. 305–326.

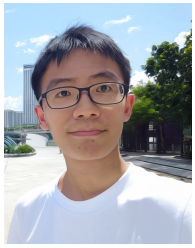
[74] N. Ding, Z. Fang, L. Duan, and J. Huang, "Optimal incentive and load design for distributed coded machine learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2090–2104, 2021.

[75] L. W. Cong, Z. He, and J. Li, "Decentralized mining in centralized pools," *The Review of Financial Studies*, vol. 34, no. 3, pp. 1191–1235, 2021.

[76] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.

[77] C. Chen, X. Chen, and Z. Fang, "Tips: Transaction inclusion protocol with signaling in dag-based blockchain," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3685–3701, 2022.

[78] M. Friedman, "Quantity theory of money," in *Money*. London: Palgrave Macmillan UK, 1989, pp. 1–40.



Canhui Chen received the B.Eng. degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2021. Now he is currently pursuing a PhD degree in computer science at the Institute for Interdisciplinary Information Sciences (IIIS) of Tsinghua University. His current research interests include blockchain system and game theory.

Zerui Cheng received the B.S. degree in computer science from the Institute for Interdisciplinary Information Sciences (IIIS) of Tsinghua University, Beijing, China. Now he is currently pursuing a PhD degree at Princeton University. He is experienced in competitive programming. His current research interests include AI and blockchains.

Xinze Liu received the B.S. degree in Automation from Zhejiang University, Hangzhou, China, in 2017, and the M.A. degree in Statistics from Columbia University, New York, in 2020. She is currently a Research Assistant at the Shanghai Qi Zhi Institute. She mainly focuses on blockchain platform design, including consensus mechanisms, decentralized swarm intelligence, and on-chain computing.

Shutong Qu received the B.Eng. degree in Computer Science from the Institute for Interdisciplinary Information Science, Tsinghua University, China in 2023, and the M.Phil. degree in Computer and Information Engineering from The Chinese University of Hong Kong, Shenzhen, China in 2025. He is currently a Ph.D. Student in Computer Science and Systems at the School of Engineering and Technology, University of Washington, Tacoma, WA, USA. His current research interests include distributed systems and mechanism design.

Zhixuan Fang is a tenure-track assistant professor at the Institute for Interdisciplinary Information Sciences (IIIS) at Tsinghua University, Beijing, China. He mainly focuses on the design and analysis of multi-agent systems and networked systems. He received his Ph.D. degree in computer science from Tsinghua University, China, in 2018, and his B.S. degree in physics from Peking University, China, in 2013.

APPENDIX

A. Safety and Liveness Analysis

In this section, we investigate the safety and liveness of ValueMine.

Definition 2. ***Safety:*** *A ledger achieves safety when all honest miners have views of the ledger that are consistent with one another. More precisely, any transaction included in one miner's ledger at a specific time is also included at the same position in all other miners at all later times.*

Definition 3. ***Liveness:*** *A ledger achieves liveness if whenever honest miners attempt to add a transaction to the ledger, it gets added to all miners' ledgers within a finite time.*

In our analysis, we consider the following assumptions: we assume that the adversarial fraction of hash power is $\beta \in (0, 1)$, the block generation rate in ValueMine follows the Poisson distribution with the total mining rate denoted as λ, and the maximum network delay is denoted as Δ. We also assume that the attacker would not construct a user problem to generate a new block, because we have proved that constructing a user problem for double-spending is not profitable in Section V. With this assumption, we will show that ValueMine can achieve safety and liveness under the following condition.

Theorem 2. *ValueMine can achieve safety and liveness, exactly under the following condition:*

$$\frac{(1-\beta)}{1+(1-\beta)\lambda\Delta} > \beta.$$

Proof. We first investigate the safety and introduce the definition of chain growth as follows.

Definition 4. ***Chain Growth:*** *For a blockchain protocol, we define the chain growth of the main chain as the average number of blocks per unit of time, denoted as CG.*

We now study the growth rate of the honest chain under the network delay. Suppose that a certain honest block B is mined by a miner P at time T and B is the first block at level l. In the worst case, B is not received by the remaining honest miners until time $t + \Delta$. Since the network has a very large number of honest miners, it is very unlikely that P will mine another block before time $t + \Delta$. Since other miners have not seen B, their block will still be mined at the same level as B (with a parent block at level $l-1$). We assume that the honest mining process is a Poisson process with rate $(1-\beta)\lambda$, then on average $(1-\beta)\lambda\Delta$ honest blocks are mined in the time interval $[t, t+\Delta]$. These blocks will not increase the length of the longest chain as they are all at level l. The first block mined after $t+\Delta$ will increase the length of the longest chain by one. Thus on average, only 1 out of $1+(1-\beta)\lambda\Delta$ blocks will increase the length of the longest chain. Thus, the growth rate of the longest chain is $CG = \frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta}$.

For safety, we need the growth rate of the honest chain to be larger than that of the adversary, i.e.,

$$\frac{(1-\beta)\lambda}{1+(1-\beta)\lambda\Delta} > \beta\lambda,$$

which leads to the condition

$$\frac{(1-\beta)}{1+(1-\beta)\lambda\Delta} > \beta.$$

For the liveness analysis, we now derive a lower bound on the chain quality. If the chain growth is CG, and the adversary can mine blocks at a rate of at most $\beta\lambda$, by the definition of the chain quality, we have a lower bound:

$$CQ \geq \frac{CG - \beta\lambda}{CG}$$

Then $CQ > 0$ if and only if $\frac{(1-\beta)}{1+(1-\beta)\lambda\Delta} > \beta$, which is exactly the same threshold as the safety guarantee.

Therefore, the longest chain protocol (PoVW) is safe and live, exactly under the following condition

$$\frac{(1-\beta)}{1+(1-\beta)\lambda\Delta} > \beta.$$

The proof is thus completed. □

B. Zero-Knowledge Scoring Protocol for ML Accuracy

Setting. Let S denote the public problem specification containing: (i) a commitment to the evaluation dataset, e.g., a Merkle root R over labeled samples (x_i, y_i); (ii) a deterministic rule for deriving the evaluation index subset I (e.g., from $(Id, \text{blockhash})$). A solver's submission a commits to a model (parameters, architecture tag, etc.). The goal is to provide a succinct zero-knowledge proof π that the claimed accuracy $w \in [0, 1]$ is correct on the committed subset $\{(x_i, y_i)\}_{i \in I}$, without revealing the dataset or the model.

Relation (Circuit) R_S. The circuit receives public inputs (R, I, w) and private witnesses consisting of:

- the model parameters (committed inside a),
- membership proofs for each (x_i, y_i) in the Merkle tree rooted at R,
- intermediate values for model evaluation.

It enforces:

$$\sum_{i \in I} \mathbf{1}\{\text{Model}(a; x_i) = y_i\} \;=\; w \cdot |I|.$$

Additionally, it verifies each Merkle-path proof against R and the model-commitment consistency inside a.

Score Function. As in the main text,

$$\mathsf{score}^{\mathsf{zk}}(a, \pi) = \begin{cases} w, & \text{if } \mathsf{ZKScoreVerify}(S, a, w, \pi) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Algorithm 1: Circuit Definition C_S (encoded as R_S)

Input: Public (R, I, w); Private model parameters in a, samples $\{(x_i, y_i)\}_{i \in I}$, and Merkle paths $\{\mathsf{path}_i\}_{i \in I}$
for each $i \in I$ **do**
  assert $\text{MerkleVerify}(R, (x_i, y_i), \mathsf{path}_i) = 1$
  $\hat{y}_i \leftarrow \text{Model}(a; x_i)$ // Compute prediction in-circuit
  $c_i \leftarrow \mathbf{1}\{\hat{y}_i = y_i\}$ // Boolean equality constraint
$W \leftarrow \sum_{i \in I} c_i$ // Constraint-friendly sum (e.g., via PLONK gates)
assert $W = w \cdot |I|$ // Bind claimed w to counted correct predictions
Output: 1 // Circuit accepts if and only if all constraints hold

Algorithm 2: Prover: $\mathsf{ZKProofGen}(S, a) \to \pi$

Input: $S = (R, I, \text{vk}, \dots)$, model commitment a
Output: succinct proof π
derive I from S
for each $i \in I$, obtain (x_i, y_i) and Merkle path path_i such that $\text{MerkleVerify}(R, (x_i, y_i), \mathsf{path}_i) = 1$
evaluate model off-chain to get predictions $\hat{y}_i \leftarrow \text{Model}(a; x_i)$
compute $W \leftarrow |\{i \in I : \hat{y}_i = y_i\}|$, set $w \leftarrow W/|I|$
construct witness: model parameters, $\{(x_i, y_i)\}_{i \in I}$, $\{\mathsf{path}_i\}_{i \in I}$, and any model-commitment openings
$\pi \leftarrow \text{ZK.Prove}(\mathcal{C}_S, \text{witness}; \text{public } (R, I, w))$ // e.g., Groth16/PLONK/STARK
return π

Algorithm 3: Verifier: $\mathsf{ZKScoreVerify}(S, a, w, \pi)$

Input: $S = (R, I, \text{vk}, \dots)$, a, w, proof π
Output: 1 if valid, 0 otherwise
return $\text{ZK.Verify}(\text{vk}, \pi; \text{public } (R, I, w))$ // Accept if proof valid

Security and Efficiency Notes.

- *Privacy:* The circuit reveals only (R, I, w); dataset samples and model parameters remain hidden as witnesses.
- *Soundness:* A prover cannot increase w without breaking SNARK/STARK soundness or Merkle-binding.
- *Determinism/anti-overfitting:* The rule deriving I from $(Id, \text{blockhash})$ is public and deterministic; miners cannot adaptively cherry-pick I after observing others' proofs.
- *On-chain cost:* Verifier gas is constant (SNARK) or quasilinear in $|I|$ (STARK), while heavy work stays off-chain.
- *Composability:* The same pattern supports graded scores beyond accuracy, e.g., F1, MSE (via bounded-range fixed-point encodings), or multi-task weighted sums.