[redacted]

Anonymous Author(s)
Submission Id: 184

Abstract

Existing blockchains struggle to meet enterprise demands for high throughput while remaining decentralized and accessible for lightweight clients. We present [redacted], a novel blockchain that achieves high performance and allows lightweight nodes to participate without incentives. [redacted] introduces [redacted] ([redacted]), utilizing threshold cryptography to enhance efficiency and integrity through signature aggregation, ensuring collective security and verifiability. This method significantly reduces storage requirements and resource usage in high-participant systems, as it compacts multiple signature shares into a single representation, maintaining cryptographic robustness and the authenticity of participant consensus.

[redacted] delegates intensive computations such as signature verification and block merging to untrusted servers while minimizing additional trust and voting-based checks conducted by honest lightweight voting nodes. [redacted] enables compute scaling through untrusted servers without sacrificing decentralization. Evaluations show [redacted] with [redacted] provides up to 4x throughput compared to state-of-the-art blockchain systems.

1 Introduction

Blockchain technology, pioneered by Bitcoin [34], has emerged as a disruptive decentralized system for secure transaction processing and record keeping. Blockchains maintain a distributed tamper-proof ledger of transactions through a decentralized consensus protocol run by the nodes in a peer-to-peer network. The decentralized nature of blockchains provides transparency, auditability, and fault tolerance without requiring a trusted third party [15, 25].

However, simultaneously achieving decentralization and high performance has remained a challenging goal for blockchain systems [14, 27]. A key component in blockchains is the consensus protocol, which allows distributed nodes to agree upon the state of the ledger. Practical Byzantine Fault Tolerance (PBFT) [21] and other Byzantine fault tolerant (BFT) algorithms are widely adopted due to their ability to tolerate Byzantine failures with relatively low overhead.

Unfortunately, traditional BFT protocols [4, 10, 41] have inherent trade-offs between performance and decentralization. Each consensus member must perform a number of cryptographic operations proportional to the committee size to validate each block, incurring computation overhead that grows linearly with the number of nodes. Furthermore, every node must send messages to every other node during the consensus procedure, resulting in communication overhead that also scales linearly. Consequently, the performance of BFT degrades rapidly as the consensus group expands, limiting BFT-based systems to at most a few dozens of active consensus nodes, especially if high consensus throughput is a prerequisite.

To achieve high performance and decentralization simultaneously, our key insight is that by introducing a new set of "delegated" nodes to offload the intensive cryptographic work, we can scale BFT performance without reducing decentralization or sacrificing safety/security guarantees. However, securely delegating cryptographic operations (such as cryptographic signatures) to potentially malicious nodes poses a major challenge.

In this paper, we propose [redacted] ([redacted]), a novel approach that replaces digital signatures with efficiently verifiable signatures based on Threshold Signature Schemes (TSS) [12, 18]. TSS provides collective security (or aggregation integrity of signatures). It signifies that the final, aggregated signature is whole, collective, untampered, and a true representation of the individual components that constitute it. The signatures remain verifiable before and after the aggregation process.

More importantly, in traditional digital signature systems, each signature is a standalone entity, often requiring substantial storage space. When dealing with multiple signatures, the size becomes a significant concern, particularly in systems with numerous participants or high-frequency transactions. Each signature share, while necessary for validating individual participant approval, contributes to the bulk that the system must manage, often leading to increased costs and resource usage. TSS takes a different approach. Instead of each participant's signature share bulking up the transaction or message, these shares are aggregated into a single, compact signature representing the entire group's consensus. This aggregated signature maintains the same cryptographic security levels as the individual signature shares, ensuring the process's integrity and trustworthiness.

As a result, in the context of BFT consensus, TSS allows consensus nodes to delegate computing-intensive cryptography duties to untrusted nodes while still being able to validate the correctness of results with minimal overhead. Crucially, our design allows independent scaling of delegated nodes. Adding more delegated nodes increases the achievable transaction throughput. Thus, our system can scale BFT consensus to thousands of consensus nodes [27, 38, 39] (and substantially millions of participants [26]) while ensuring safety through verification and retaining decentralization.

We implement and evaluate [redacted], which decouples consensus participation from computation and communication

costs. Experimental results demonstrate [redacted] achieves substantially higher throughput and scalability compared to previous decentralized BFT systems.

In summary, our key contributions are:

- We introduce [redacted], a new technique to scale the computation of blockchain signatures using untrusted nodes.
- We present [redacted], a novel blockchain that achieves up to 4x throughput compared with state-of-the-art blockchain systems.

2 Background

This section provides a background synthesis of the available blockchain systems and points out their limitations. [redacted] is inspired by these works, and it is specifically improved for public blockchains.

2.1 Threshold Signature Schemes

Threshold signature schemes (TSS) are cryptographic protocols that allow a group of participants to jointly generate a digital signature, without any single party having access to the full private key. TSS split the private key in a digital signature system into n parts, with each participant holding 1 part. To generate a valid digital signature, t or more of the n participants need to collaborate by contributing their partial signature. This is also known as a (t, n) threshold signature scheme.

To generate a collective public key, each participant i generates a secret key share sk_i and the corresponding public key pk_i through Distributed Key Generation (DKG). The collective public key can be obtained by summing the public polynomials of all participants.

To sign a message M, each participant i computes a partial signature $\sigma_i = sk_i * H(M)$, where H is a cryptographic hash function. Denote w as a pre-computed weight vector calculated through Lagrange interpolation. The partial signatures are then aggregated into $\sigma = \Sigma w_i * \sigma_i$, which can be verified against the message and the public key.

The security of threshold signatures rests on the secrecy of the key shares and the difficulty of solving discrete logarithms. By distributing trust across multiple parties, malicious individual nodes does not compromise the overall private key. Threshold signatures thus provide improved resilience and fault tolerance compared to single-key schemes.

Overall, threshold signatures minimize additional trust assumptions and offer security advantages for distributed systems like blockchains.

2.2 Consensus

Blockchains rely on consensus algorithms to achieve a common agreement on the state of the network among distributed processes or multi-agent systems. These algorithms are essential for ensuring the security and integrity of data in the system [42].

Proof-of-Work (PoW). This consensus algorithm is dominant in the blockchain space and is utilized by major chains like Bitcoin [34], Dogecoin [22], Litecoin [17], and Monero [32]. The core premise of PoW is for participants to solve complex cryptographic puzzles, where the first to solve broadcasts the solution to others. It is straightforward in concept but its significant energy consumption has sparked debates about its environmental sustainability.

Proof-of-Stake (PoS). In search of more energy-efficient alternatives, the blockchain community introduced proof-of-stake (PoS). Unlike PoW, which rewards miners based on their computational effort, PoS factors in the number of tokens held by a user (their "stake") when determining their mining rewards. Ethereum [4], Polkadot [20], and Celo [30] are examples of chains that have either adopted or are transitioning to PoS. While PoS is seen as a more environmentally friendly alternative, it derives its security largely from economic incentives.

Byzantine Agreement (BA). Byzantine agreement (BA) offers a mechanism to achieve consensus in blockchain systems without the presence of an economic incentive model. One prominent algorithm following the BA principle is Practical Byzantine Fault Tolerance (PBFT) [21]. In essence, PBFT works by having nodes in a distributed system reach an agreement on the system's state, even in the presence of malicious nodes. The process involves intricate steps, including the multicasting of requests, execution of operations by backup nodes, collection of replies, and multicasting of ordered operations. This elaborate communication mechanism allows PBFT to ensure that consensus is reached despite the presence of a certain number of Byzantine nodes.

PBFT, and other BA algorithms, can be utilized in conjunction with PoS mechanisms for validator selection. For example, Algorand [26] employs PoS to select validators, while utilizing a BA-based consensus algorithm to ensure agreement among these validators regarding the network's state.

Algorand's innovation spurred further exploration, leading to systems like Blockene which aimed to make BA feasible for mobile clients. To make this work, Blockene [39] proposed a two-tier model: one with lightweight mobile clients and the other with resource-intensive servers. This model allowed mobile devices to engage in the consensus process, though performance challenges, such as signature computation, remained.

Subsequent developments like SBFT [27] showcased that threshold signatures could be efficiently employed in public blockchains. By merging BFT with threshold secret sharing, SBFT reduced both communication overheads and complexities, paving the way for more scalable and efficient consensus

methods. Notably, SBFT still requires its participants to remain resource-intensive.

3 Trust-minimized Computing Delegation

In this section, we provide an in-depth understanding of [redacted]'s approach to delegate computation workloads with minimized trust through TSS. We present the delegation model and delve into the security implications of [redacted].

3.1 Delegation Model

In traditional Byzantine Fault Tolerance (BFT) consensus mechanisms, participating nodes endorse messages by disseminating their signatures to every other node in the network. While optimizations employed by linear PBFT [38] or Blockene [39] can mitigate network performance issues by incorporating collectors, the computational burden on each participant remains unchanged. This is due to the fact that each participant must still receive and authenticate the same volume of signatures corresponding to the committee size. [redacted]'s primary objective is to mitigate this challenge.

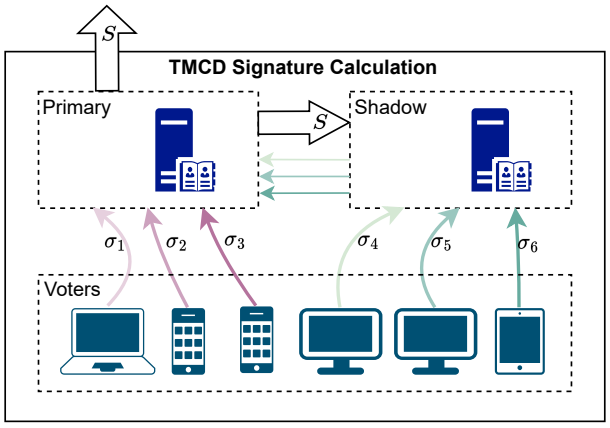

Figure 1. The Signature Calculation Process of [redacted]

Figure 1 illustrates the signature calculation process of [redacted], where each server communicates with a set of consensus nodes. The architecture of the proposed system is divided into two layers, referred to as the delegation and consensus tiers, respectively. This dual-structure approach is instrumental in optimizing the system's efficiency and inclusivity, allowing for a diverse range of devices to contribute to the network, thereby enhancing its practicality and ease of deployment.

Delegation Tier and Its Functionality. The delegation tier is comprised of delegation nodes, which are essentially robust servers with substantial processing capabilities. These nodes play a pivotal role in maintaining the network's integrity and functionality. However, they are specifically designed to facilitate computational tasks associated with signature or 'voting' operations, rather than directly engaging in consensus decisions. This strategic division of responsibilities ensures that the fundamental security premises of blockchain technology remain intact while streamlining the efficiency of the overall system.

Consensus Tier and Participant Dynamics. In contrast, the consensus tier features consensus nodes, also known as 'voters'. These are typically lightweight, implying they do not require extensive computational resources, which makes the system accessible and feasible for various devices. The consensus nodes, organized into committees, are selected from a pool of volunteers, reflecting a democratic and inclusive approach reminiscent of systems like Algorand [26] and Blockene [39]. This structure not only promotes participation but also reinforces the decentralized nature of the blockchain.

Operational Framework and Server Distribution. Within this system, assuming a presence of m delegated nodes and n consensus nodes, the operational framework mandates that each consensus node, denoted by c_i for the i-th index, aligns with a primary server identified by $s_{(i \bmod m)}$. This arrangement, whether uniform or sequenced in a round-robin fashion, is crucial as it underpins the protocol's performance and correctness without influencing them directly. Similarly, each block within the chain, labeled b_j for the j-th index, is assigned a primary server $s_{(j \bmod m)}$ responsible for handling the majority of computational tasks associated with that particular block.

Redundancies and System Resilience. The system acknowledges the potential for server failures and incorporates measures to counteract disruptions. In instances of malfunction, tasks are promptly reassigned to backup or 'shadow' servers, ensuring uninterrupted service. This immediate redirection of business logic and operational tasks is a testament to the system's resilience and reliability.

Transaction Processing and Consensus Protocol. The protocol dictates that consensus nodes intermittently retrieve transaction data, encapsulated within proposed blocks, from their respective primary servers. During the voting phase, each consensus node, indexed i, initiates the process by computing a hash of all transactions, followed by calculating its signature share σ_i utilizing Threshold Signature Schemes (TSS). We leverages TSS since conventional digital signatures are not ***aggregatable*** and therefore remains linear complexity in the real world.

[redacted]trades off the loads on the client and server sides. Compared to ECDSA [28], BLS signatures are faster in signing, though there are slower in key generation and signature verification [35]. Consequently, we select BLS signature as it reduces the signing time consumption for the light-weight consensus nodes. Notably, the system employs the BLS12-381 curve [12, 18], owing to its expedited signing process and widespread acceptance in threshold cryptography. For

the pairing-based threshold cryptography in our referenced implementation [8], signature shares and aggregated signatures share the same data structure. Each of them is stored in a 96-byte space.

First, assign a secret key share to each participant. This can be achieved centralized or in a distributed norm through DKG by picking a random and unpredictable secret key value from a finite group. Then, the public key share of each participant and the overall public key can be obtained by adding up all key shares.

Second, participants may generate their votes by calculating the product of their secret key shares and the hash of message. Afterwards, signature shares shall be accepted by receivers if the validation process passes. Next, signature aggregation can be performed on delegated nodes by computing weighted sum of signature shares, where the weights are pre-calculated through Lagrange interpolation. Finally, the aggregated signature can be validated through plain BLS validation by checking whether the aggregated signature still satisfies the bi-linear properties of the original finite group.

Signature Aggregation and Network Decision. Nodes forward their signature shares to their primary servers post-calculation. These shares are essential, collectively forming the basis of consensus decisions. They converge at the block's primary server via inter-server transmissions, where they undergo a verification process. Upon successful authentication, these signature shares are amalgamated, culminating in a singular, aggregated signature. This consolidated signature, representing the network's collective decision, is then broadcast across the network, marking the completion of the consensus process.

3.2 Security Review

Staleness Attacks on Delegated Nodes. A potential vulnerability could arise if the delegated nodes deceive consensus nodes by transmitting spurious blocks or transactions. In [redacted], the integrity of transactions is safeguarded by TSS, making the proclamation of unsigned messages cryptographically infeasible. Consequently, delegated nodes can only mislead consensus nodes by dispatching outdated blocks, termed 'staleness attacks'. Two strategies counteract this: (i) as proposed in Blockene [39], clients can be configured to concurrently retrieve blocks from multiple servers; (ii) [redacted] introduces a probabilistic gossip technique amongst consensus nodes, detailed in § 4.4.

Attacks on consensus node. Malevolent actors may attempt to corrupt the committee by offering bribes, with the intent to influence transactional decisions. Nevertheless, within the TSS framework, the scope of potential malicious activities is constrained. Given that [redacted] is Byzantine fault-tolerant, compromising its integrity would necessitate the bribery of at least one-third of all consensus nodes.

Split-view Attacks. Given that blocks are obtained from a variety of consensus nodes, there's a heightened risk of split-view attacks. It is essential to understand that a split-view attack occurs when a delegated node sends differing states to different consensus nodes. Conversely, a staleness attack appears when a delegated node sends outdated blocks to its consensus nodes (delegators).

A significant mitigation strategy against split-view attacks is for consensus nodes to read data from various peer consensus nodes in a randomized manner. As long as data from a sufficient number of consensus nodes is accessed, it is highly probable that security is maintained. Within this setup, nodes prioritize the most recent messages while disregarding the outdated ones.

Denial of Service (DoS) Attacks. The cryptographic security of [redacted] is ensured through the implementation of TSS, which ensures that delegated nodes cannot provide misleading information to consensus nodes. However, a challenge arises if a delegated node sends numerous fabricated blocks to consensus nodes, which might disrupt the voting mechanism. The creation of a block is computationally demanding due to the requirements of generating both its payload and signatures. As a result, excessive requests to a delegated node could reduce its operational speed.

To protect the system against DoS attacks, consensus nodes can be pre-allocated to one or more delegated nodes using hash-based random functions. In environments where participants are permissioned, it becomes straightforward to identify malicious consensus nodes. Additionally, this configuration provides a defense against Sybil attacks on delegated nodes, as detailed in [24].

4 Overview of [redacted]

[redacted] in general presents a split-trust model [39] that allows servers (or clusters) to accelerate the signing (or voting) process, and network transmission in a blockchain system. By relieving workloads on regular nodes, TMCD allows lightweight devices to participate without loosening the security guarantees.

4.1 System Configuration

Elaborating further on the system's architecture, the delegated nodes are interfaced through a high-speed network. A Delegated node i out of m is represented as S_i. On the other hand, consensus nodes are geographically dispersed

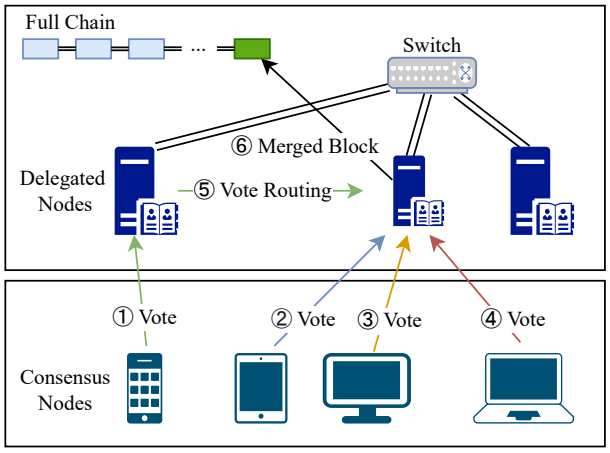

Figure 2. The [redacted] architecture

and communicate predominantly over the Internet. Consensus node i out of n is denoted as C_i. The threshold parameter for TSS is determined by t, such that $t \leq n$.

Consensus nodes can be various devices, such as smartphones or Raspberry Pis, typically characterized by constrained computational capabilities. Although a consensus node connects to multiple delegated nodes, only one serves a principal role, with the rest serving backup functions.

To maintain protocol simplicity, interactions between the two layers are crafted to be stateless, and operations—including proposing, voting, and committing—are designed for idempotency. Initiating an operation necessitates a consensus node transmitting a request to its primary delegated node.

4.2 Protocol

Figure 3 shows the workflow to commit a block, which contains the following steps.

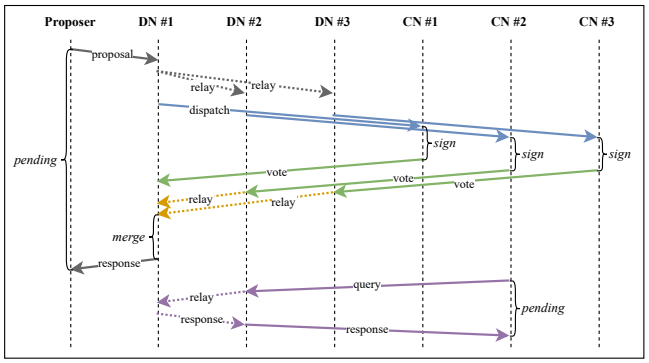

Figure 3. [redacted] workflow, where m=t=n=3. DN=delegated node, CN=consensus node. The lifecycle of a proposed block is labeled chronologically.

Initialize. During initialization, consensus nodes establish connections to delegated nodes and execute the Threshold Key Generation (TKG) process via a relay mechanism. A designated proposer is authorized to introduce a new block (or transactions) to the system network.

Propose. Consensus nodes propose a block by uploading it to any available delegated node, rendering this block as a pending block. Upon receipt of this new proposal, the delegated node dispatches the information across the network, thereby updating its pending block list.

Dispatch. When a delegated node receives a pending block from the proposer, it broadcasts this new block to all other delegated nodes. Consensus nodes periodically pull blocks from delegated nodes to update their pending block lists. Afterwards, this proposed block will be distributed to all consensus nodes. Upon receipt of the proposed block, each consensus node casts its vote (either affirmative or negative) by dispatching its cryptographically signed vote share to the respective delegated nodes.

Algorithm 1 Voting for a block

Require: $t, S_m, C_n, b(v)$ ▷ $count(S) = m, count(C) = n$
Ensure: $m \geq 0, n \geq v \geq 0, n \geq t \geq 0$
  $s \leftarrow db(b)$
  $c \leftarrow 1$
  while $c \leq n$ **do** ▷ loops are executed in parallel
    if $dc(c) = s$ **then**
      S_s transmits b to C_c
      $v \leftarrow sign_c([hash(b); vote])$ ▷ TSS sig-shares
      C_c sends v to S_s
    else if $dc(c) \neq s$ **then**
      S_s relays b via $dc(c)$ to C_c
      $v \leftarrow sign_c([hash(b); vote])$ ▷ TSS sig-shares
      C_c sends v via $dc(c)$ to S_s
    end if
    $c \leftarrow c + 1$
  end while

Vote. In the [redacted] protocol, every consensus node possesses an equivalent voting weight. As detailed in Algorithm 1, a specific consensus node C_i begins its procedure upon the receipt of a previously unencountered proposed block b. The node then undertakes a verification process for the block to decide its approval status. Subsequent to this verification, the consensus node attaches a threshold signature share to its vote and dispatches it to a delegated node where all votes will be merged if applicable.

Merge. Upon receipt of a sufficient number of block signatures by a delegated node, it can be inferred that the block has received approval from the consensus node. Consequently, the delegated node initiates a procedure to aggregate the

threshold signatures associated with the block. These signature shares are then aggregated into a unified threshold signature, which facilitates future verification by the consensus nodes.

Algorithm 2 Committing a block

Require: $t, S_m, C_n, b(v)$ ▷ $count(S) = m, count(C) = n$
Ensure: $m \geq 0, n \geq v \geq 0, n \geq t \geq 0$
  $s \leftarrow db(b)$
  $vc \leftarrow 0$
  $vl \leftarrow []$
  while $vc \leq t$ **do**
    if s receives vote v **then**
      $vc \leftarrow vc + 1$
      $vl \leftarrow vl + v$
    end if
  end while
  $mb \leftarrow merge(b, vc)$ ▷ TSS sig-aggregation
  s broadcasts mb to $S_m[]$ ▷ Intra-cluster communication

Commit. Once a block is merged, it undergoes a persistence process, subsequently transitioning to a committed state. To facilitate future inquiries, the committed block is replicated throughout the entire cluster. As shown in Algorithm 2, a delegated node notifies its peer delegated nodes to persist the block.

4.3 Transaction Dependency

Blockchains fundamentally function as distributed ledgers maintaining consistency, with transactions as their pivotal components. Inherent dependencies exist among these transactions, predominantly due to the interplay with the balance of specific accounts. Thus, handling transactional dependencies becomes imperative for blockchains.

To address this issue, [redacted] utilizes a timed global snapshot approach. Rather than memorizing the whole transaction history, consensus nodes maintain a record of the balance status for all accounts. At stipulated intervals, delegated nodes capture global snapshots of the status of all accounts. These snapshots undergo validation and are subsequently signed by the consensus nodes. Thereafter, these are disseminated incrementally, prompting consensus nodes to update their local versions. Each snapshot is distinctly identified by a snapshot ID. To ensure the integrity of transactions, an account is restricted to initiating a singular transaction between two consecutive snapshots.

In terms of storage requirements for snapshot data, assuming a blockchain encompassing 2M accounts, if each account reserves 4 KB for its status, the cumulative space demanded for a snapshot would approximate 8 GiB, calculated as 2M multiplied by 4 KB max.

4.4 Probabilistic Gossip

Problem of Split-View Attacks. While the Threshold Signature Scheme (TSS) ensures that malicious delegated nodes cannot violate the integrity of voting results or undermine the accuracy of storage, it does not address the risk of split-view attacks. In such attacks, malicious delegated nodes present different information to distinct consensus nodes, leading to split-view attacks.

Solution. An effective countermeasure entails consensus nodes conducting periodic inter-node queries based on a predetermined probability. To verify the reliability of delegate node responses, consensus nodes employ a gossip protocol. If a pattern of divergent responses is detected between two consensus nodes, the associated delegated node is marked as malicious and subsequently removed from the system. This approach effectively mitigates the risk of split-view attacks with minimal gossip overhead. The probabilistic gossip mechanism can be described in three stages:

1. Peer selection. Any consensus node intending to engage in gossip should identify a group of target nodes. To prevent the inadvertent selection of compromised or malicious consensus nodes, the selection procedure should incorporate randomization. This can be seamlessly implemented using random or hash functions. Even if some responses prove to be inaccurate or inadmissible, the system's integrity remains intact as long as more than 2/3 of the consensus nodes operate honestly.
2. Query. Having established a suitable target group, consensus nodes can initiate peer queries. The gossip protocol requires a response from a delegated node, accompanied by corresponding signatures. Upon receipt of the signed responses, consensus nodes compare them to their local records. To optimize resource utilization, this operation should be executed at a low frequency.
3. Report. In the event that a consensus node identifies malicious activities - evidenced by inconsistent messages from delegated nodes - a two-step action is undertaken. The consensus node first halts all communications with the perceived malicious delegated node. Subsequently, the node reports the issue, providing relevant evidence, across the gossip network. Servers will be regarded malicious if a threshold number of reports have been received.

Furthermore, it's worth noting that malicious delegated nodes might attempt to disseminate erroneous messages within the network, targeting other peers. However, the robust security framework provided by TSS ensures that transactions and votes remain resistant to tampering, even if delegated nodes conspires with consensus nodes.

5 Implementation

This paper presents a preliminary implementation of [redacted]. The [redacted] system is implemented in both in Rust and C++,

whereas the experiments in § 6 only reports the performance of the C++ version. Part of the code is shared by both the consensus and delegated nodes.

The insights gleaned thus far from the C++ implementation have paved the way for comprehensive future studies, promising further optimization and refinements that will contribute to the evolution of distributed systems. The shared code strategy, meanwhile, stands as a testament to the system's sophisticated design philosophy, prioritizing coherence, maintainability, and high performance.

5.1 Delegated Nodes

The delegated nodes are implemented headlessly in C++ with 4,920 LoC. Each delegated node is connected to the (optionally) RDMA-enabled network and listens to eRPC messages from peer nodes. On receiving a new proposed block, the delegated node stores the block in its local in-memory cache and broadcasts the block to all peer nodes. Consensus nodes get this block through polling. On receiving a vote v for block b, the delegated node routes this vote to node $db(b)$. Once a delegated node collects enough votes for a block, it merges the signatures of this block and posts the result attached with an aggregated threshold signature in the network.

Given the tendency for delegated nodes to be structured in clustered formations, the system embraces the eRPC framework [2, 29] as a high-performance alternative. This choice is due to its compatibility with advanced networking paradigms like IB verbs, DPDK, and raw UDP, ensuring that the system remains adaptable and capable of handling high-throughput demands. For the efficient local caching of transactions and votes, which is paramount for maintaining system speed and integrity, the technology adopted is concurrent hash maps from Intel's OneTBB [7]. These are utilized as the in-memory caching solution, chosen for their proven performance and reliability.

In addressing the communication demands between the two critical tiers within this infrastructure, the system incorporates the gRPC communication framework [6]. This framework is renowned for its performance and reliability in handling protocol communications. Complementing this, the FlexBuffers serializer [5, 37] is employed, known for its flexibility and ease of implementation, which is crucial for maintaining a robust, adaptable, and efficient communication protocol within this complex system.

5.2 Consensus Nodes

The consensus nodes are implemented headlessly in C++ with 2,841 LoC. Each consensus node c periodically polls its engaged delegated node $dc(c)$ to download new blocks. On receiving a new proposed block b, the consensus node looks up its latest global state cache and determines whether the block should be approved. The voting message is calculated by $m = sign([hash(b); vote])$ and then sent to delegated node $dc(c)$.

However, the responsibilities of the consensus nodes extend beyond these individual interactions. They are guardians of the network's integrity, tasked with mitigating the risk of corruption or manipulation within the ranks of the delegated nodes. To combat potential misinformation or anomalous behavior, consensus nodes employ a gossip protocol, a probabilistic communication mechanism that ensures they cross-verify the information disseminated by the delegated nodes. Through this, they establish a consensus on the veracity of the data received, promoting transparency and trust in the network.

6 Evaluation

[redacted] is examined and evaluated from multiple dimensions. This section answers the following questions with experiments:

- What throughput does [redacted] achieve?
  [redacted] achieves ~8000 TPS in the experiment network. This will be reported in § 6.2.
- How good is the transaction latency?
  The median latency of [redacted] is in the scale of seconds. We prove that [redacted] latency (§ 6.3) aligns with the state-of-the-art solutions.
- How long does each step takes in the full life cycle of a transaction (or a block)?
  We prove in § 6.4 to have minimized the time costs of cryptography operations during consensus. This helps to improve both throughput and latency comparing to previous BFT-based blockchains.
- To what extent does [redacted] tolerate malicious behaviors?
  We justify the robustness levels from both security and fault tolerance perspectives whiling maintaining scalability. These will be presented in § 6.5 and § 6.7.

6.1 Experiment Configurations

The research presented utilizes the [redacted] protocol, tested within a controlled environment to simulate its operational capabilities and efficiency in a real-world scenario. The testing environment was meticulously set up on a cluster consisting of nine high-performance servers, all housed within a single rack, ensuring uniformity in connectivity and access conditions. Each server is designated as a delegated node, equipped with an Intel Xeon E5-2643 v4 processor, clocking at 3.40GHz, ensuring rapid data processing and computational efficiency. Furthermore, these nodes were equipped with a substantial 128 GB DDR4 DRAM. The infrastructure was designed to mimic real-world operating conditions, with a robust 56 Gbps RDMA inter-connection network facilitating high-speed internal communication and a separate 1

Gbps NIC for internet connectivity. This setup ensured a balance between internal data transfer efficiency and external communication requirements.

For the simulation of consensus nodes, the experiment utilized a single physical server, significantly enhanced with dual 64-core processors, simulating 128 consensus nodes. The server maintained a standard 1 Gbps full-duplex connection, ensuring consistent bandwidth during interactions with other nodes, even with Internet-scale latency, thereby closely mimicking real-world internet communication delays. Critical to the experiment's success was the use of NVMe SSDs for persistent storage, ensuring rapid data retrieval and efficient storage management. These devices stored various essential data, including committed blocks, block headers, and global states, critical for the integrity and continuity of the blockchain operations.

The experiment was benchmarked against multiple existing solutions [4, 26, 27, 39]. Notably, the hardware settings, particularly the ratio between delegated and consensus nodes, were aligned similarly to the Blockene [39] protocol. However, a critical distinction was observed in the efficiency of the delegators in the [redacted], which required significantly fewer computing resources compared to Blockene, while effectively servicing an equivalent number of voting participants. For authenticity and to maintain a focus on critical operational phases, the experimental setup deliberately excluded the node discovery phase. Instead, it relied on fixed dialing addresses, acknowledging the potential variability in IP addresses and domain names that could occur in a dynamic real-world scenario. The experiment strictly utilized TCP for message transmissions, ensuring reliability in the delivery of data packets.

Performance metrics, particularly average and throughput measurements, were diligently recorded at four-second intervals, a time frame exceeding the maximum latency conceivable for block transmissions and timeout configurations, ensuring a comprehensive capture of performance data. The [redacted] protocol's ability to execute blocks in parallel was a focal testing point, with stages in the consensus process being pipelined during the experiments to test operational efficiency. To maintain the experiment's integrity and ensure the results reflected practical constraints, the parallelization factor was deliberately capped at 8. This restriction meant that no more than eight blocks were pending at any given stage of the consensus life cycle, thereby providing a realistic assessment of the protocol's performance under practical workload conditions. This careful capping also prevented system overload or skewed results due to excessive simultaneous processes, ensuring that the findings were both reliable and indicative of real-world performance.

6.2 Throughput

In Figure 4, we delve into a detailed analysis of the performance metrics of [redacted], focusing primarily on its throughput capabilities. The throughput, a critical indicator of performance, is measured in transactions per second (TPS) and reflects the efficiency and speed at which data processing is executed within the blockchain network. This metric is particularly crucial in understanding the practical implications of deploying [redacted] in real-world scenarios, where high transaction volumes are commonplace.

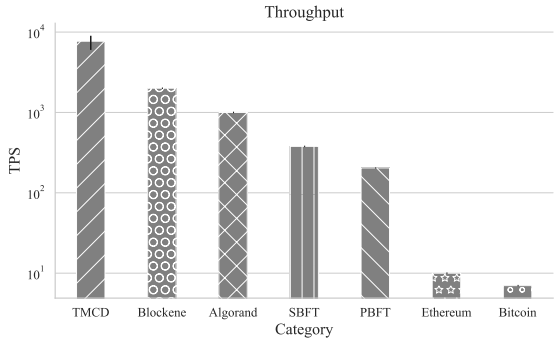

Figure 4. Maximum throughput. The throughput-latency balance is modulated by batch size of transactions.

Our observations are anchored on the relationship between the throughput and the volume of data committed to the chain. In an impressive display of efficiency, [redacted] consistently achieves a throughput of approximately 8,000 to 10,000 TPS. This remarkable performance is attributed to its utilization of Byzantine Fault Tolerance (BFT) consensus mechanisms, a model known for its resilience and high processing capacity, coupled with an optimized block size containing 2,000 transactions.

For a comparative analysis, we reference studies [39] and [26], which showcase a range of similar blockchain solutions. Table 2 synthesizes these comparisons, clearly indicating that [redacted] stands at the forefront in terms of throughput efficiency. This superior performance underscores its potential as a market leader and reinforces its suitability for high-demand applications.

It's important to delineate that our discussion does not extend to Layer-2 scaling solutions. While these frameworks offer acceleration tools that undoubtedly enhance blockchain transaction processing speeds, they operate on principles distinct from those underpinning [redacted]'s core architecture. However, this does not preclude their relevance or utility. In fact, Layer-2 solutions present a complementary relationship with [redacted], offering potential avenues for integration and the subsequent bolstering of [redacted]'s performance metrics. We anticipate that the synergistic combination of [redacted]'s robust architecture with Layer-2 scaling solutions could unlock

unprecedented throughput levels, a topic we will explore comprehensively in § 7 of this paper.

6.3 Latency

In the realm of distributed ledger technology, the efficiency of systems is often bottlenecked by issues such as latency and throughput. Our research delves into these critical aspects, focusing on the performance of [redacted], particularly through the lens of its latency characteristics. The experiments conducted were meticulously designed to quantify and validate the latency level of [redacted], ensuring a comprehensive evaluation of its operational efficiency.

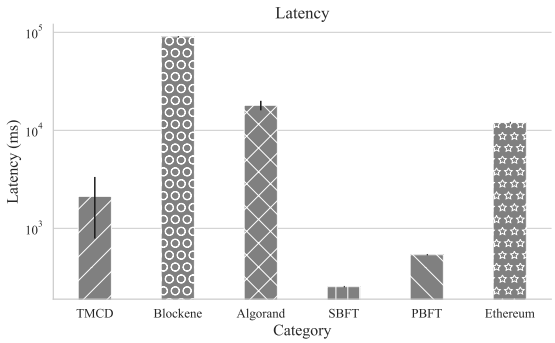

Figure 5. Latency on maximum throughput. We configure [redacted] to reach its maximum throughput and measure the latency.

One of the critical observations is the measurement of end-to-end latency, an essential metric in the assessment of blockchain performance. This latency is clocked from the moment the proposer initiates a transaction by posting a block, encapsulating the transaction's journey until its final commit to the chain. Experiments are conducted to prove the latency level of [redacted]. Figure 5 presents the latency metrics of [redacted]. End-to-end latency starts when the proposer posts a block and ends when the block is committed to the chain. [redacted] achieves a competent average latency among the competitors while remaining a high batch size.

We illustrate the trade-off between throughput (in TPS) vs. latency in Figure 6. A comparative analysis with existing protocol SBFT [27] reveals [redacted]'s pronounced efficiency. Particularly, when juxtaposed with systems employing SBFT with fast paths - a method designed for speed - [redacted] emerges superior. It not only excels by registering higher TPS, indicative of better transactional throughput, but it also maintains commendable latency metrics. This balance is crucial, as it underscores [redacted]'s capability to sustain a high-performance threshold without undermining transaction processing speed or system responsiveness.

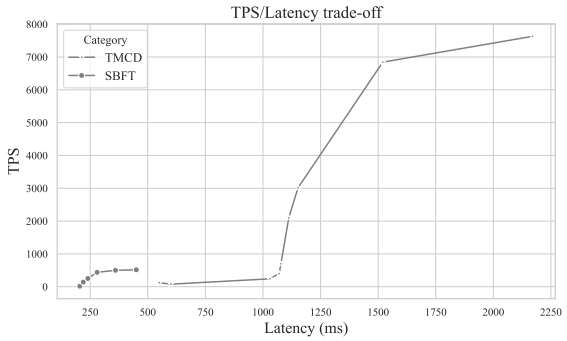

Figure 6. Throughput vs. Latency trade-off

6.4 Time Break-up

In the study presented, Figure 7 meticulously details the temporal aspects of each phase within the [redacted] process. This analysis is crucial as it sheds light on the efficiency and bottlenecks of the system. The procedure is dissected into three pivotal stages: distribution, voting, and committing, each marked by specific time stamps that reflect various actions within the [redacted] protocol.

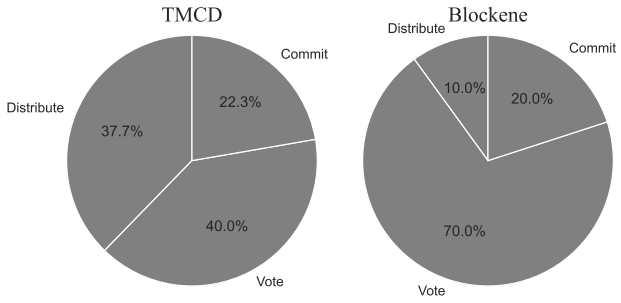

Figure 7. Time break-up. Time consumption of each stages in the two protocols are compared.

Distribution Phase. This initial stage is crucial as it marks the onset of the consensus process. The time stamp here, referred to as the distribution time, signifies the moment the proposal has been finalized and initiates transmission to the consensus nodes. This phase's efficiency is paramount as it sets the pace for the subsequent stages, necessitating a robust system to mitigate any delays.

Voting Phase. Following distribution, the process enters the voting phase, a critical juncture where the consensus nodes begin to cast their votes. The 'voting time' specifically denotes the period when the inaugural consensus nodes cast votes for the block in question. This stage's agility is vital, and the system's responsiveness is tested as it needs to handle multiple votes, possibly arriving in quick succession or simultaneously.

Committing Phase. The final stage in this tripartite process is marked by the 'commit time'. This time stamp is pivotal,

signifying the juncture at which sufficient votes have been accrued, consolidated, and a conclusive decision has been rendered regarding the block's fate. This phase's culmination is the fruit of the preceding stages' labors, underscoring the interconnectedness of the [redacted] process.

An insightful deduction from the study highlights that certain protocols are beleaguered with inefficiencies, particularly those that require consensus nodes to synchronize via gossip communications. This synchronization is a double-edged sword; while it aims to maintain consistency and fairness, it inadvertently becomes a consensus bottleneck. The real-world implication of this is pronounced, given the generally high network latencies experienced, thereby throttling the throughput and efficiency of the consensus process.

[redacted] innovatively circumvents this bottleneck through strategic 'computing delegation'. This technique significantly truncates the waiting time, previously rendered necessary by the synchronization requirement. The primary objective here is the resolution of the signing and voting overheads that plague consensus nodes. This is particularly evident in the current implementation [8], where a security measure has been instituted, capping each signing operation to a constant 10 milliseconds to thwart timing attacks.

6.5 Robustness and Stability

Since the most prone attack manner of delegated nodes is DoS, we measure the performance of [redacted] given different failure scenarios of delegation nodes to prove the liveness even under attacks. Figure 8 presents the robustness evaluations of [redacted], where four delegation nodes failed designated (labeled Failure A-D with vertical lines). Failure A and B are intentionally seperate apart by a long gap time, where C and D are more sticked together. We prove that [redacted] appears resilient to network fluctuation or DoS attacks, even with consecutive and concurrent failures. On each failure, the TPS first drops as consensus nodes are configured to wait for a timeout until they find that the delegated node is unavailable. Afterwards, the consensus nodes will change their bound delegated nodes and redirect votes to backups. Then, the TPS returns to its normal state. Robustness test also reports that our system has a minimum starting time, where it takes merely one observation point to reach a normal working state.

In terms of stability, error bars illustrated in Figure 4 and 5 show that [redacted] remains of high performance on network fluctuations. The height of each bar is calculated by the mean value within each time window. Each error bar records the maximum and minimum values of TPS and latency over all observations.

6.6 System Loads

The data presented in Table 1 and Figure 9 provides a comprehensive analysis of system loads, specifically focusing on the metrics of CPU utilization patterns (%user/%system)

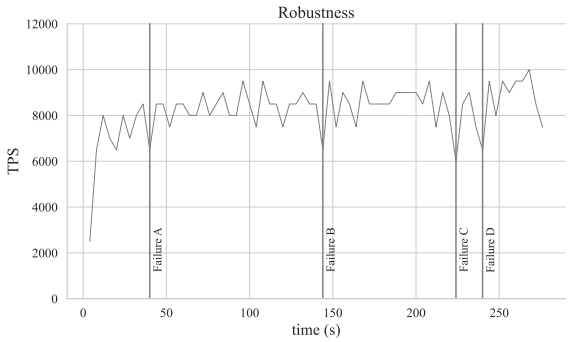

Figure 8. Robustness test. Nine delegated nodes are deployed in the start, and four failures are intentionally set to test failure resilience.

and network usage parameters. These critical observations play a significant role in understanding the operational efficiency and resource allocation within the distributed systems in question, particularly concerning consensus nodes in a blockchain or similar decentralized setup.

A critical review of the CPU time allocation reveals that the consensus nodes expend the majority of their processing power on tasks integral to the system's integrity and operational continuity. These tasks predominantly include processing cryptographic signatures, which are paramount for securing transactions and ensuring their authenticity, and executing transaction logics, which form the backbone of interaction within the network. Such activities are computationally intensive, underscoring the significance of the resources dedicated to these processes.

Furthermore, network usage analysis indicates an efficient communication protocol among the consensus nodes. On average, each node registers a network usage below 10 Mbps, a benchmark that underscores the protocol's efficiency in minimizing bandwidth consumption without compromising transactional throughput. This efficiency is primarily achieved through the implementation of Threshold Signature Schemes (TSS), which streamline the communication process. TSS plays a pivotal role in maintaining a low-level communication footprint, essential for the system's scalability and operational sustainability.

In contrast, when we shift our focus to delegators—entities responsible for processing delegation logic, an essential component of stakeholder representation in decision-making - we observe a different resource utilization pattern. Delegators exhibit a surplus in CPU resources, indicating that the current system configuration and workload do not fully engage the available computational power. This redundancy underscores potential areas for optimization, possibly hinting at the capability to handle more substantial or more complex workloads.

Network latency could overshadow other aspects of system performance, primarily due to the increased distances involved in data transmission and the potential inconsistencies in global internet service quality. As such, future system enhancements need to prioritize the minimization of latency to ensure consistent performance and overall system reliability. This focus will help in maintaining the delicate balance between operational efficiency, resource utilization, and system scalability. These are crucial for the long-term success of any large-scale decentralized systems [14].

Role	%CPU	pps (rx/tx)	kBps (rx/tx)
Consensus nodes	12/65	41203/22854	36187/2143
Delegated nodes	14/2	3795/7948	824/9718

Table 1. System loads

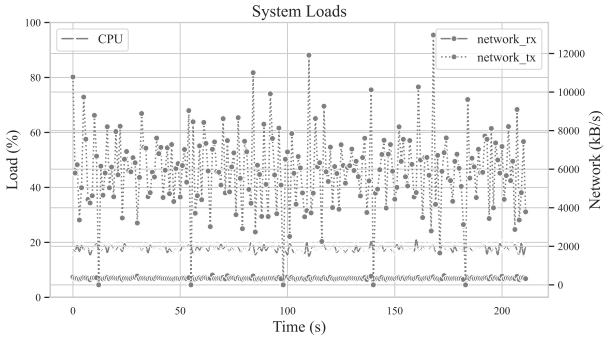

Figure 9. System loads on delegated nodes. CPU consumption percentages and network usage are observed simultaneously.

6.7 Scalability

Figure 10 proves that [redacted] scales from the perspectives of both consensus and delegation nodes. Our experiments were somewhat constrained by the hardware capabilities at our disposal, limiting us to the use of 128 consensus nodes. It's important to note that within the context of our optimized implementation, these nodes don't come close to maxing out the capacity of the delegated nodes.

To rigorously test the scalability potential of [redacted], we imposed a restriction on the CPU usage of the processes running on the delegated nodes. This was achieved by capping the CPU consumption at 400% (equivalent to 4 cores) using the 'cpulimit' command. This artificial limitation was set to simulate a real-world scenario where resources are finite and to analyze how the system performs and whether the system remains robust under such constraints.

The first pivotal strategy in ensuring efficient scalability lies in the election of committees. By electing committees,

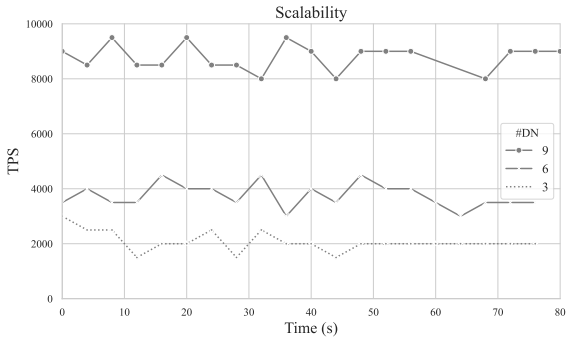

Figure 10. Scalability test. DN=delegated nodes. The curve becomes less stable because we are restricting the peak performance of each physical servers.

we can effectively keep the consensus node group within a manageable size. Our experiments, which meticulously evaluated system loads and latency parameters, revealed that the consensus nodes do not create a bottleneck within the protocol. This finding is significant as it dispels the notion that increasing the number of consensus nodes would proportionally strain the system. Instead, it suggests that other factors could be optimized for better performance. This indicates room for further optimization but also scalability potentials of delegated nodes.

Secondly, we observed minimal heavy communication traffic among the delegation nodes, especially along the critical path of consensus. This lack of communication congestion is vital. It means that as the number of delegated nodes expands, the overall system performance and throughput can increase correspondingly, without the conventionally expected linear increase in communication overhead. This aspect underscores a design optimized for performance, where delegation nodes can efficiently process and handle increased workloads without unnecessary inter-node communication, thereby bolstering the overall throughput as the system scales.

7 Related Work

This section delves into an exhaustive review of existing literature and previous studies pertinent to this project in Table 2, providing a critical analysis of their contributions within this domain. We meticulously evaluate how [redacted] evolves to address the constraints and limitations inherent in these prior works, offering an innovative approach that heralds advancements in the field.

One notable strategy is employed by Solana [41], which capitalizes on the Proof of History (PoH) concept to enhance blockchain consensus performance. This method, while innovative, necessitates substantial computational power from

Category	Name	TPS	Members	Incentives	Energy consump.	Safety
PoW/PoS	BitCoin [34]	4-10	~220M	Yes	Extreme	50%
	Ethereum [4] [3]	~10	~250M	Yes	High	50%
	Algorand [26]	~1000	Millions	No	Fair	Depends on crypto-sortition
	Blockene [39]	~2000	Millions	No	Low	Depends on multiple RW
BFT	linear-PBFT [27] [38]	~1000	~200	No	Fair	33%, depends on scale
	SBFT [27]	~2000	~200	No	Fair	33%, limited by scale
(Ours)	[redacted]	~8000	Millions	No	Low	33%, scalable

Table 2. Comparison of existing scalable solutions

participants, complicating the validation process. The reliance on resource-intensive nodes also hampers practical deployment, presenting a logistical challenge.

In contrast, Algorand [26] employs cryptographic sortition to streamline participant scalability, a feature further augmented by Blockene's [39] introduction of auxiliary servers, enabling nodes with limited resources to join the Algorand consensus. However, this system's efficiency is undermined by its intensive multiple read/write schemes, imposing an excessive burden on both client and server infrastructures.

SBFT [27] takes a different approach by integrating TSS and linear optimizations to facilitate the geographical scaling of BFT participants. Despite its ingenuity, this system presupposes constant online presence and substantial computational capacity among participants, as replicas are selected sequentially in a round-robin configuration.

The advent of zk-Rollup technology marks a significant stride forward, enabling the efficient and succinct verification of off-chain transactions on-chain. The emergence of EVM-compatible zkEVMs is particularly noteworthy, as they endorse Turing-complete primitives through ZKP. Platforms such as zkSync [13, 16], Polygon [10], Taiko [11], and Consensys [19], while exhibiting commendable performance and security standards, are inherently constrained by the EVM and Ethereum consensus protocols. We note that the field of zkEVM and ZK-based scaling research is orthogonal to [redacted], as these systems generally rely on Layer-1 consensus protocols. This underscores the potential of overlay ZKP-based networks when integrated with [redacted].

Sharding techniques enhance performance by segmenting the network into more manageable shards, interconnected through cross-shard transactions. Scaling solutions like Monoxide [40], RapidChain [43] and OmniLedger [31] introduce innovative consensus that optimizes transaction efficiency based on sharding. However, these methods, while maintaining intra-shard performance, significantly decelerate cross-shard transactions and often sacrifice blockchain's fundamental security protocols by reducing participant numbers in each sub-group.

DAG-based blockchain architectures [36] offer a unique solution, inherently safeguarding against forks and thereby preserving the sanctity of historical transactions. Despite their complexity, systems such as Nxt [9], IOTA [33], DagCoin [1], Byteball [23] are pioneering this architectural paradigm. A salient advantage of DAG systems is the diminished reliance on miners, allowing transactions to proceed independently. Nonetheless, these structures are susceptible to double-spending hazards, necessitating intricate design strategies to circumvent these vulnerabilities.

8 Conclusion

In this study, we introduced a groundbreaking framework known as [redacted], an advanced permissioned blockchain technology specifically designed to alleviate the computational burden traditionally placed on participants. This innovative approach is anchored in the utilization of TSS, coupled with a pioneering split-trust architecture.

The essence of [redacted] is not merely its novelty; it is in its revolutionary approach to reconciling three often conflicting blockchain properties: scalability, integrity, and the facilitation of light-weight nodes. In the realm of blockchain technology, these attributes are frequently considered as trade-offs, where the enhancement of one property comes at the expense of others. However, with the [redacted] framework, we demonstrate that it is possible to integrate all three, thereby disrupting the conventional paradigm.

Furthermore, our framework is designed to support light-weight nodes, allowing users with minimal computational resources to join the blockchain. This inclusivity broadens the user base, ensuring that the benefits of the blockchains are not just reserved for those with substantial computational power. The light-weight design also enhances overall system efficiency, as it requires less power consumption, thereby making it environmentally friendly and cost-effective.

Our empirical tests underscore the efficacy of our [redacted] and the [redacted] framework. One of the most striking results from our experiments is the achievement of a throughput measured up to ~8000 TPS, which is 4x as high as state-of-the-art blockchain systems. This level of performance, particularly in a system designed to accommodate thousands of consensus nodes and millions of users, is testament to the system's optimized efficiency and its capability to handle large-scale operations without degradation in performance.

References

[1] Dagcoin whitepaper. https://prismic-io.s3.amazonaws.com/dagcoin/f4e531e1-a5db-43b6-930c-14bf705e65ee_Dagcoin_White_Paper.pdf. Accessed: 2023-10-19.

[2] erpc. https://github.com/erpc-io/eRPC. Accessed: 2023-10-19.

[3] The ethereum blockchain explorer. https://ww7.etherscan.io/. Accessed: 2023-10-19.

[4] Ethereum whitepaper. https://ethereum.org/en/whitepaper/. Accessed: 2023-10-19.

[5] Flatbuffers whitepaper. https://flatbuffers.dev/flatbuffers_white_paper.html. Accessed: 2023-10-19.

[6] grpc documentation. https://grpc.io/docs/. Accessed: 2023-10-19.

[7] Intel oneapi threading building blocks. https://www.intel.com/content/www/us/en/developer/tools/oneapi/onetbb.html. Accessed: 2023-10-19.

[8] libbls. https://github.com/skalenetwork/libBLS. Accessed: 2023-10-19.

[9] Nxt whitepaper. https://nxtdocs.jelurida.com/Nxt_Whitepaper. Accessed: 2023-10-19.

[10] Pol: One token for all polygon chains. https://polygon.technology/papers/pol-whitepaper. Accessed: 2023-10-19.

[11] Taiko roadmap. https://taiko.mirror.xyz/NfYQFzzkcEIy3jU9PTBonem2HlNiZre-3WwLnbGnwQ. Accessed: 2023-10-19.

[12] Threshold signature schemes. https://medium.com/nethermind-eth/threshold-signature-schemes-36f40bc42aca. Accessed: 2023-10-19.

[13] zksync overview. https://docs.zksync.io/userdocs/intro/. Accessed: 2023-10-19.

[14] Ananda Badari and Archie Chaudhury. An overview of bitcoin and ethereum white-papers, forks, and prices. *Forks, and Prices (April 26, 2021)*, 2021.

[15] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, 2020.

[16] Léo Besançon, Catarina Ferreira Da Silva, Parisa Ghodous, and Jean-Patrick Gelas. A blockchain ontology for dapps development. *IEEE Access*, 10:49905–49933, 2022.

[17] Jaysing Bhosale and Sushil Mavale. Volatility of select cryptocurrencies: A comparison of bitcoin, ethereum and litecoin. *Annu. Res. J. SCMS, Pune*, 6, 2018.

[18] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.

[19] Matthieu Bouchaud, Tom Lyons, Matthieu Saint Olive, Ken Timsit, Shailee Adinolfi, Benjamin Calmejane, Guillaume Dechaux, Faustine Fleuret, Vanessa Grellet, Joyce Lai, et al. Central banks and the future of digital money. *ConsenSys AG Whitepaper*, pages 01–20, 2020.

[20] Jeff Burdges, Alfonso Cevallos, Peter Czaban, Rob Habermeier, Syed Hosseini, Fabio Lama, Handan Kilinc Alper, Ximin Luo, Fatemeh Shirazi, Alistair Stewart, et al. Overview of polkadot and its design considerations. *arXiv preprint arXiv:2005.13456*, 2020.

[21] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[22] Usman W Chohan. A history of dogecoin. *Discussion Series: Notes on the 21st Century*, 2021.

[23] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value. https://obyte.org/Byteball.pdf. Accessed: 2023-10-19.

[24] John R Douceur. The sybil attack. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1*, pages 251–260. Springer, 2002.

[25] Caixiang Fan, Sara Ghaemi, Hamzeh Khazaei, and Petr Musilek. Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, 8:126927–126950, 2020.

[26] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.

[27] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. Sbft: a scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 568–580. IEEE, 2019.

[28] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[29] Anuj Kalia, Michael Kaminsky, and David Andersen. Datacenter {RPCs} can be general and fast. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 1–16, 2019.

[30] Sep Kamvar, Marek Olszewski, and Rene Reinsberg. Celo: A multi-asset cryptographic protocol for decentralized social payments. *DRAFT version 0.24 https://storage. googleapis. com/celo whitepapers/Celo A Multi Asset Cryptographic Protocol for Decentralized Social Payments. pdf*, 2019.

[31] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE symposium on security and privacy (SP)*, pages 583–598. IEEE, 2018.

[32] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *arXiv preprint arXiv:1704.04299*, 2017.

[33] Sebastian Müller, Andreas Penzkofer, Nikita Polyanskii, Jonas Theis, William Sanders, and Hans Moog. Tangle 2.0 leaderless nakamoto consensus on the heaviest dag. *IEEE Access*, 10:105807–105842, 2022.

[34] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[35] D Page, DJ Bernstein, and T Lange. Report on ebats performance benchmarks. *European Network of Excellence in Cryptology, Tech. Rep. IST-2002-507932-D. VAM*, 9, 2007.

[36] Huma Pervez, Muhammad Muneeb, Muhammad Usama Irfan, and Irfan Ul Haq. A comparative analysis of dag-based blockchain architectures. In *2018 12th International conference on open source systems and technologies (ICOSST)*, pages 27–34. IEEE, 2018.

[37] Muhammad Adna Pradana, Andrian Rakhmatsyah, and Aulia Arif Wardana. Flatbuffers implementation on mqtt publish/subscribe communication as data delivery format. In *2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 142–146. IEEE, 2019.

[38] Xiaodong Qi, Yin Yang, Zhao Zhang, Cheqing Jin, and Aoying Zhou. Linsbft: Linear-communication one-step bft protocol for public blockchains. *arXiv preprint arXiv:2007.07642*, 2020.

[39] Sambhav Satija, Apurv Mehra, Sudheesh Singanamalla, Karan Grover, Muthian Sivathanu, Nishanth Chandran, Divya Gupta, and Satya Lokam. Blockene: A high-throughput blockchain over mobile devices. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 567–582, 2020.

[40] Jiaping Wang and Hao Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, pages 95–112, 2019.

[41] Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*, 2018.

[42] Di Yang, Chengnian Long, Han Xu, and Shaoliang Peng. A review on scalability of blockchain. In *Proceedings of the 2020 the 2nd International Conference on Blockchain Technology*, pages 1–6, 2020.

[43] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.