DChain: Scaling Blockchains through Trust-Minimized Computing Delegation

Abstract—**Existing blockchains struggle to meet enterprise demands for high throughput while remaining decentralized and accessible for lightweight clients. We present DChain, a novel blockchain that achieves high performance and allows lightweight nodes to participate without incentives. DChain introduces Trust-Minimized Computing Delegation (TMCD), 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 consensus.**

TMCD 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. TMCD enables compute scaling through untrusted servers without sacrificing decentralization. Evaluations show DChain with TMCD provides up to 4x throughput compared to state-of-the-art blockchain systems.

Index Terms—**blockchain, scalability, BFT, TSS**

I. Introduction

Blockchain technology, pioneered by Bitcoin [36], 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 remains 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.

Blockchain protocols tend to elect committees (consensus groups) as a selected group of nodes (participants) within the network that is responsible for performing specific tasks related to achieving consensus on the state of the blockchain. These tasks might include validating transactions, proposing blocks, or voting on various matters. The concept of a committee is often used in blockchain systems to improve efficiency, scalability, and security. Unfortunately, traditional BFT protocols [4], [10], [43] 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 (and quadratically if considering the network as a whole) with the number of nodes. Furthermore, every node must send messages to every other node during the consensus, resulting in linear communication overhead. Consequently, the performance of BFT degrades rapidly as the consensus group expands, limiting BFT-based systems to tens of active consensus nodes, especially when high consensus throughput is a prerequisite.

To achieve high performance and decentralization, our key insight is that by introducing a new set of ‘delegated’ nodes to offload the intensive voting and validation work, we can scale consensus 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. Existing split-trust architectures [41] also involve delegated nodes. However, they tend to introduce additional trust requirements for the delegated tier. Comparing with these models, DChain achieves scalability from both storage and computational perspectives, while only minimized additional trust issues are required.

In this paper, we propose Trust-Minimized Computing Delegation (TMCD), a novel approach that allows racked servers to undertake some of computational burden in consensus and accelerate the consensus process. To achieve this novel architecture, a modification from the cryptographic perspective is to replace digital signatures with efficiently verifiable signatures based on Threshold Signature Scheme (TSS) [12], [18]. It provides collective security (or aggregation integrity) of signatures, signifying that the aggregated signature is whole, collective, untampered, and a true representation of the individual components that constitute it. The signatures remain verifiable after the aggregation process.

By contrast, in traditional digital signature systems, each signature in a consensus is a standalone entity, often requiring substantial storage space and network communication if considering all of them as a whole. 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. TMCD takes a different approach. Instead of each participant’s signature share bulking up the messages, these shares are aggregated into a single, compact signature representing the entire group’s decision. This aggregated signature will maintain the same cryptographic security level as the individual signature shares, ensuring the process's trustworthiness.

As a result, in the context of BFT consensus, TMCD allows consensus nodes to delegate both storage and computing-intensive duties to untrusted nodes while still being scalable and able to validate the correctness of results with minimal overhead. Keeping most of other configurations the same as existing systems, our system can still scale BFT consensus to thousands of consensus nodes [27], [40], [41] (and potentially millions of participants [26]) while ensuring safety through verification and retaining decentralization.

The application scenario of DChain can be a company pioneered to provide ledger (or blockchain) infrastructure upon its original web services to enable new security or trust features. In this case, the company is self-incentivized, and the clients are the original customers of the original service.

We implement and evaluate DChain, which decouples consensus participation from computation and communication costs. Experimental results demonstrate DChain achieves substantially higher throughput and scalability compared to previous decentralized BFT systems.

In summary, our key contributions are:

- We introduce TMCD, a new technique to scale the computation of blockchain signatures using untrusted nodes.
- We present DChain, a novel blockchain that achieves up to 4x throughput compared with state-of-the-art blockchain systems [41].

II. Background

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

A. 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. For a TSS-oriented BFT consensus, t should be set at least $\frac{2n}{3}$ to ensure security.

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 (public key) of all participants. View changes lead to regeneration of key pairs, which can be computationally intensive. In DChain, we adopt the DKG and view change (days) configurations as our referred works [26] [41], as the DKG part is out of this paper's scope.

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 TSS rests on the secrecy of the key shares and the difficulty of solving discrete logarithms. By distributing trust across multiple parties, malicious individual nodes do not compromise the overall private key. Threshold signatures thus provide improved resilience and fault tolerance compared to single-key schemes. Furthermore, it enables the potential of accountability (and therefore disincentivizes malicious attempts) as the collective signature is smaller comparing to thousands of regular digital signatures. Overall, threshold signatures minimize additional trust and offer security advantages for distributed systems like blockchains.

B. Consensus

Blockchains rely on consensus algorithms to achieve a common agreement on the state of the network among distributed processes. These algorithms are essential for ensuring the security and integrity of data in the system [44].

a) Proof-of-Work (PoW): This consensus algorithm is dominant in the blockchain space and is utilized by major chains like Bitcoin [36], Dogecoin [22], Litecoin [17], and Monero [34]. 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.

b) 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.

c) 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 BFT 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 [41] 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, 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.

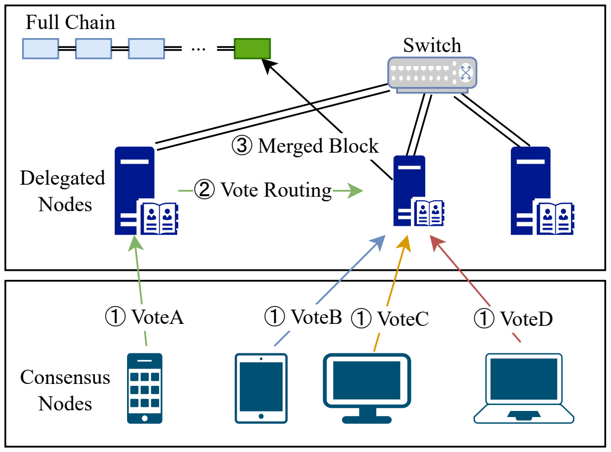

Fig. 1. The DChain architecture. The voting operations happen concurrently, and ultimately the votes are routed to the primary delegated node of the proposed block for aggregation.

III. Design Overview of DChain

DChain in general divides the blockchain trust model into two tiers, the delegated dier and the consensus tier. DChain is a split-trust model [41] 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, DChain allows light-weight devices to participate without loosening the security guarantees.

A. Architecture

Figure 1 illustrates the architecture of DChain, where the votes of blocks are routed and merged by delegated nodes. 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 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, 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.

B. 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 [40] or Blockene [41] 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 verify the same volume of signatures corresponding to the committee size. DChain's primary objective is to mitigate this challenge. It aims to solve the scalability of consensus, instead of the application level.

a) Delegated Tier and Its Functionality: The delegated tier is comprised of delegated nodes, which are essentially robust servers with substantial processing capabilities. 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.

b) 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 [41]. This structure not only promotes participation but also reinforces the decentralized nature of the blockchain.

c) 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 a signature share σ_i utilizing Threshold Signature Schemes (TSS). We leverage TSS since conventional digital signatures are not **aggregatable** and therefore remains linear complexity.

First, secret key shares are assigned to each node. This can be achieved in centralized or distributed norms through DKG by picking a random, unpredictable secret key from a finite group. Then, the public key shares of each participant and the global public key can be obtained by adding up all shares. Second, participants may generate their votes by calculating the product of their secret key shares and the message hash. Afterwards, signature shares shall be accepted by receivers if the validation process passes. Next, signature aggregation is performed on delegated nodes by computing the weighted sum of signature shares, where the weights are pre-calculated through Lagrange interpolation for arbitration. Finally, the aggregated signature is validated through plain BLS validation by checking whether the aggregated signature still satisfies the bi-linear properties of the original finite group.

d) 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.

C. Summary of DChain Consensus

The DChain design adopts a committee-based BFT consensus architecture. It involves committee election, voting, vote validation, network decision rules (committing). Additionally, the design also incorporates the resolution for malicious behaviours, divergent messages and persistent process.

In general, the election process is kept the same with existing BFT protocols such as [26]. The voting and vote validation process follows a TSS approach, where the vote shares are submitted and merged ultimately to represent the collective identity (aggregated signature). Following this, to commit a block, the network decision is made according to the collective identity endorsing the signature. This paper will also discuss the detailed design of DChain in Section IV, introducing Trust-Minimized Computing Delegation (TMCD).

IV. Trust-minimized Computing Delegation

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

Figure 2 illustrates the signature calculation process of TMCD, where each server communicates with a set of consensus nodes. The architecture of the proposed system is divided into two tiers, referred to as the delegated and consensus tiers.

A. 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 $dc(c_i) = s_{(i \bmod m)}$. 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 $db(b_j) = s_{(j \bmod m)}$ responsible for handling the majority of computational tasks associated with that particular block.

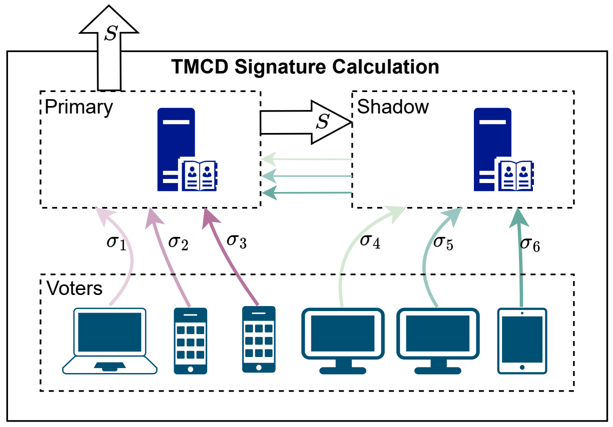

Fig. 2. The signature calculation process of TMCD. Consensus nodes are assigned (say, according to hash functions) to servers. The assignment can be designed as spontaneous to prevent predictable operations. Only two servers are shown here for simplification, but it could be far more than this number.

B. Consensus Protocol

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

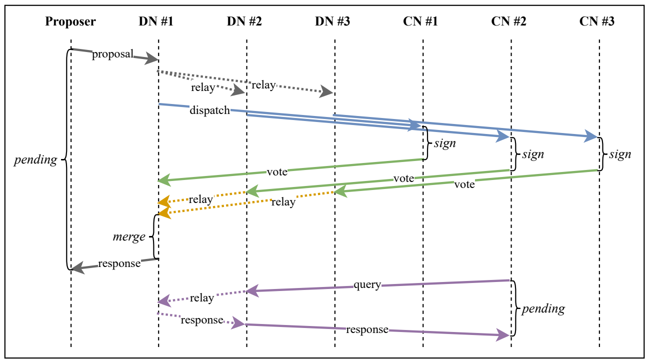

Fig. 3. TMCD workflow, where m=t=n=3. DN=delegated node, CN=consensus node. The lifecycle of block is labeled chronologically.

a) 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. Proposers can be any entity as long as it holds the secret key of an account. In our tests, the consensus nodes are responsible of generating transactions and proposals following a 80% Zipfian distribution.

b) 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.

c) 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: $m \geq 0$, $n \geq v \geq 0$, $n \geq t \geq 0$
 1: $s \leftarrow db(b)$, $c \leftarrow 1$
 2: while $c \leq n$ do    ▷ loops can break at ($c == t$)
 3:     if $dc(c) = s$ then
 4:         $S_s$ transmits $b$ to $C_c$
 5:         $v \leftarrow sign_c([hash(b); vote])$    ▷ TSS sig-shares
 6:         $C_c$ sends $v$ to $S_s$
 7:     else if $dc(c) \neq s$ then
 8:         $S_s$ relays $b$ via $dc(c)$ to $C_c$
 9:         $v \leftarrow sign_c([hash(b); vote])$    ▷ TSS sig-shares
10:         $C_c$ sends $v$ via $dc(c)$ to $S_s$
11:     end if
12:     $c \leftarrow c + 1$
13: end while
```

d) Vote: In DChain protocol, each consensus node possesses a pre-determined voting weight. If using trusted hardware (such as TEEs), the weights will be equalized; alternatively, the weight allocation can also follow a PoS manner. As detailed in Algorithm 1, a primary delegated node of a block will wait until collecting enough number of votes before validation and commit. Each consensus node C_i begins its voting procedure upon the receipt of a previously unencountered proposed block b. After making a decision of the transactions, this consensus node will sign the block hash and its decision with its secret key share, and send the vote v to its delegated node. On the delegated tier side, if validated, all the votes will ultimately be relayed to one primary delegated node of the block for final aggregation.

e) 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.

f) Commit: Once a block is merged, it undergoes a persistence process, subsequently transitioning to the committed state. To facilitate future inquiries, the committed block is replicated throughout the entire delegated tiers. The trust root of each block lies in its attached aggregated signature. Votes transmitted to a delegated node can come from consensus nodes or peer delegated nodes by vote relay. As shown in Algorithm 2, a delegated node merges all votes after it receives and validates enough votes ($\geq t$). It then attaches

Algorithm 2 Committing a block

```
Require: $t, S_m, C_n, b(v)$    ▷ $count(S) = m$, $count(C) = n$
Ensure: $m \geq 0$, $n \geq v \geq 0$, $n \geq t \geq 0$
 1: $s \leftarrow db(b)$, $vc \leftarrow 0$, $vl \leftarrow []$
 2: while $vc \leq t$ do
 3:     if $s$ receives vote $v$ then
 4:         $vc \leftarrow vc + 1$
 5:         $vl \leftarrow vl + v$
 6:     end if
 7: end while
 8: $mb \leftarrow merge(b, vc)$    ▷ TSS sig-aggregation
 9: $s$ broadcasts $mb$ to $S_m[]$
```

the aggregated signature to the block, and notifies its peer delegated nodes to persist the block.

C. Security Review

a) Staleness Attacks on Delegated Nodes: A potential vulnerability could arise if the delegated nodes deceive consensus nodes by transmitting spurious blocks or transactions. In TMCD, 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 [41], clients can be configured to concurrently retrieve blocks from multiple servers; (ii) DChain introduces a probabilistic gossip technique amongst consensus nodes, detailed in § IV-E.

b) 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 DChain is Byzantine fault-tolerant, compromising its integrity would necessitate the bribery of at least one-third of all consensus nodes.

c) 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.

d) Sybil Attacks: The configuration of TMCD provides a defense against Sybil attacks [24] of both delegated and consensus nodes. For consensus nodes, sybil nodes can be easily prevented to join the consensus by Proof-of-X (can mostly and straightforwardly be of stakes) or secure hardware such as TEEs. This is a common setting especially when targeting at modern and mobile devices. Regarding delegated nodes, sybils attacks can be initiated by the adversary (say, services providers) by deploying multiple malicious delegated servers and tampering the initial status of newly joined consensus nodes, presumably. However, this will not be the case due to two reasons. First, the security of block signatures is ensured by the cryptography primitives. The collective signature is protected by a global secret, which can be obtained by nobody. Second, the hash value of each committee and committee members will be stored in each snapshot and be stored in both delegated and consensus nodes.

D. 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, DChain 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 (1K blocks in our implementation), 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. 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.

Resolving transaction dependencies also enables parallel processing of blocks. If any two transactions in a block have no dependencies, these two blocks can potentially be voted, executed, and committed in parallel. However, the parallelization factor should be calibrated to avoid vote fragmentation slowing down the system progress.

E. Probabilistic Gossip

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.

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.

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 $\frac{2}{3}$ of the consensus nodes operate honestly.

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. The frequency of peer-queries can be regarded as a security parameter. The implementation strategy in our test framework is that consensus nodes check for inconsistencies orthogonally to the original consensus pipeline and only when the system resources is not run out. However, we note that with the cryptography assumptions, this checking does not affect the correctness of ledgers, but is used to identify malicious delegated nodes preventing liveness.

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.

V. Implementation

This paper presents a preliminary implementation of DChain. The DChain system is implemented in both in Rust and C++, whereas the experiments in § VI only reports the performance of the C++ version.

DChain trades off loads on the two tiers. Compared to ECDSA [28], BLS signatures are faster in multi-signing, though they are slower in key generation and signature verification [37]. Consequently, it reduces the signing time for light-weight consensus nodes. 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.

A. Delegated Nodes

The delegated nodes are implemented in C++ with 4,920 LoC. 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 its primary delegated server $db(b)$. Once a delegated node collects enough votes for a block, it merges the signature shares and posts the result attached with an aggregated 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 networking paradigms like IB verbs and DPDK, ensuring that the system remains adaptable of handling HPC demands. For the efficient local caching of transactions and votes, the technology adopted is concurrent hashmaps from OneTBB [7].

In addressing the communication demands between the two tiers within this infrastructure, the system incorporates the gRPC communication framework [6]. Complementing this, the FlexBuffers serializer [5], [39] is employed, known for its flexibility and ease of implementation.

B. Consensus Nodes

The consensus nodes are implemented 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 its primary delegated node $dc(c)$. In our experiments, consensus nodes periodically poll the delegated nodes for new blocks in a RESTful manner. We note that, to further improve timeliness, a pushing approach of delegated nodes can be adopted. However, this part does not affect the protocol correctness.

VI. Evaluation

TMCD is examined and evaluated from multiple dimensions. This section answers the following questions:

- What throughput does TMCD achieve?
- How good is the transaction latency?
- How long does each step takes in the consensus lifecycle?
- To what extent does TMCD tolerate malicious behaviors?

A. Experiment Configurations

The research presented utilizes the TMCD 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 machines, all housed within a single rack. Each server is designated as a delegated node, equipped with an Intel Xeon E5-2643 v4 processor @ 3.40GHz, ensuring rapid data processing and computational efficiency. Furthermore, these nodes were equipped with 128 GB DRAM and NVMe SSD.

To simulate consensus nodes, the experiment utilized a single physical server with two 64-core processors, simulating 128 consensus nodes. This server maintained a standard 1 Gbps full-duplex connection. These devices stored various essential data, including committed blocks, block headers, and global states, critical for the integrity and continuity of the blockchain operations. Overall, to simulate the real world environment, the network between each two nodes are capped at 10 Mbps and a latency of 20 ms minimum.

The experiment was benchmarked against multiple existing solutions [4], [26], [27], [41]. Notably, the hardware settings, particularly the ratio between delegated and consensus nodes, were aligned similarly to the Blockene [41] protocol. However, a critical distinction was observed in the efficiency of the delegators in the TMCD, which required significantly fewer computing resources compared to Blockene, while effectively servicing an equivalent number of voting participants.

To avoid vote fragmentation over too many pending blocks, the parallelization factor was deliberately capped at 8. This restriction meant that no more than 8 blocks were pending at any given stage of the consensus life cycle.

B. Throughput

In Figure 4, we delve into a detailed analysis of the performance metrics of TMCD, 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 TMCD in real-world scenarios, where high transaction volumes are commonplace.

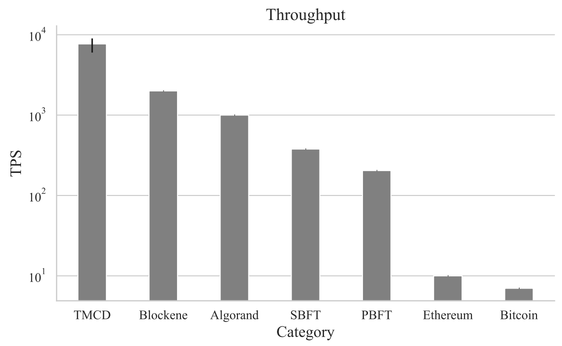

Fig. 4. Maximum throughput. The throughput-latency balance is modulated by batch size of transactions.

Our observations are anchored on the relationship between the throughput and the volume of data committed to the chain. TMCD consistently achieves a throughput of 8-10 kTPS, coupling with an block size containing 2k transactions.

For a comparative analysis, we reference studies [41] and [26], which showcase a range of similar blockchain solutions. Table II synthesizes these comparisons, clearly indicating that TMCD stands at the forefront in terms of throughput efficiency. 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 TMCD's core architecture. However, this does not preclude their relevance or utility. In fact, Layer-2 solutions present a complementary relationship with TMCD, offering potential avenues for integration and the subsequent bolstering of TMCD's performance metrics. We anticipate that the synergistic combination of TMCD's robust architecture with Layer-2 scaling solutions could unlock unprecedented throughput levels, a topic we will explore comprehensively in § VII of this paper.

C. 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 TMCD, particularly through the lens of its latency characteristics. The experiments conducted were meticulously designed to quantify and validate the latency level of TMCD, ensuring a comprehensive evaluation of its operational efficiency.

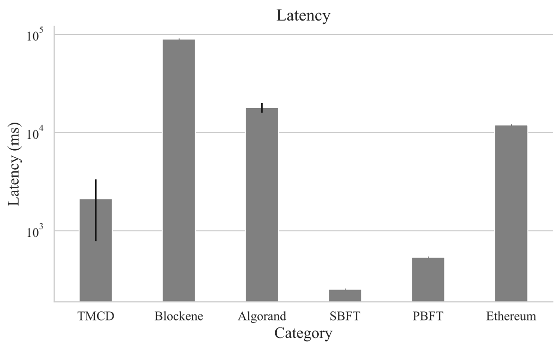

Fig. 5. Latency on maximum throughput. We configure DChain to reach its maximum throughput and measure the latency.

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 TMCD. Figure 5 presents the latency metrics of TMCD. End-to-end latency starts when the proposer posts a block and ends when the block is committed to the chain. TMCD 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 TMCD's pronounced efficiency. Particularly, when juxtaposed with systems employing SBFT with fast paths - a method designed for speed - TMCD emerges superior. It not only excels by registering higher TPS, indicative of better transactional throughput, but it also maintains commendable latency metrics.

D. Time Break-up

Figure 7 details the temporal aspects of each phase within the TMCD 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 TMCD protocol.

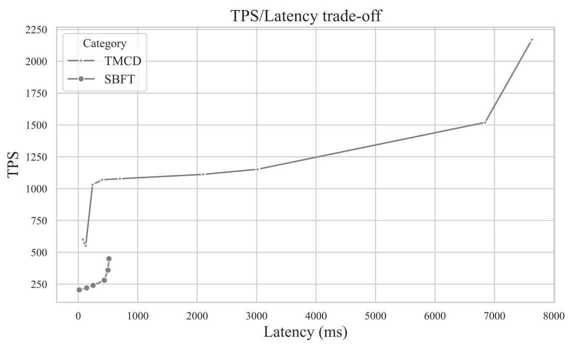

Fig. 6. Throughput vs. Latency trade-off

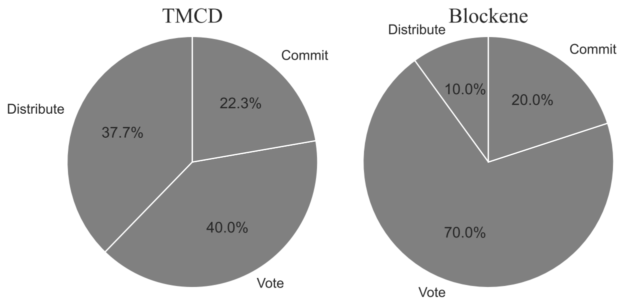

Fig. 7. Time break-up. Time consumption of each stages in the two protocols are compared.

Distribution Phase. This initial stage is crucial as it marks the onset of the consensus process. The time stamp, referred to as the distribution time, signifies the moment the proposal has been finalized and initiates transmission to consensus nodes.

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 TMCD process.

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 ms (set to ensure each operation can be finished even on a slow machine) to thwart timing attacks.

E. Robustness and Stability

Since the most prone attack manner of delegated nodes is DoS, we measure the performance of TMCD given different failure scenarios of delegation nodes to prove the liveness even under attacks. Figure 8 presents the robustness evaluations of TMCD, 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 TMCD 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.

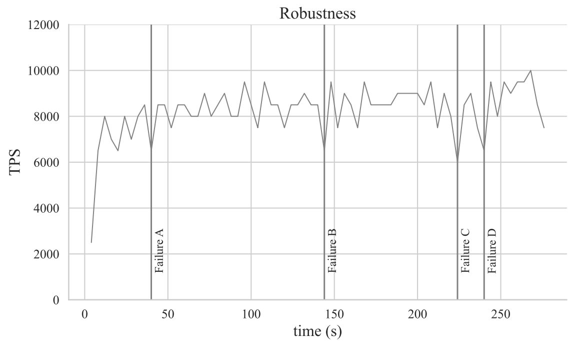

Fig. 8. Robustness test. Nine delegated nodes are deployed in the start, and four failures are intentionally set to test failure resilience.

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

F. System Loads

The data presented in Table I and Figure 9 provides a comprehensive analysis of system loads, specifically focusing on the metrics of CPU utilization patterns (%user/%system) and network usage parameters in packets per second (pps). 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.

The focus on network latency 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].

G. Scalability

Figure 10 proves that TMCD 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.

TABLE I
SYSTEM LOADS

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

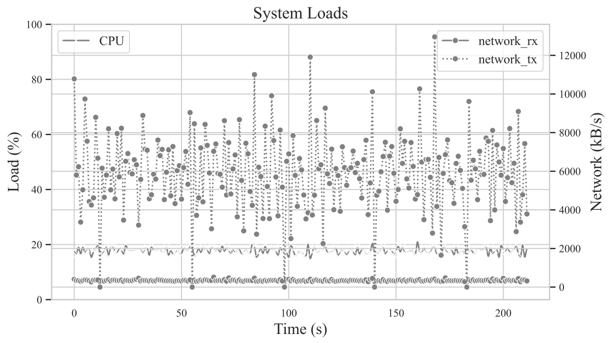

Fig. 9. System loads on delegated nodes. CPU consumption percentages and network usage are observed simultaneously.

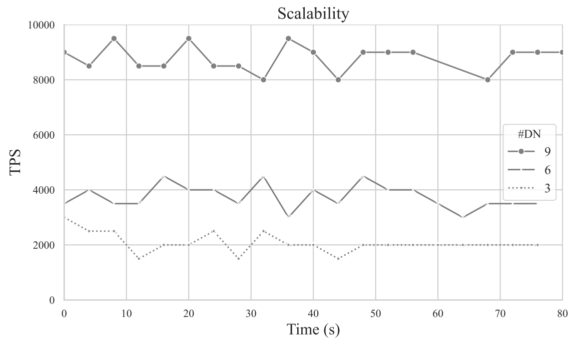

Fig. 10. Scalability test. DN=delegated nodes. The curve becomes less stable because we are restricting the peak performance of each physical servers.

To test the scalability potential of DChain, 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' tool. This artificial limitation was set to simulate a real-world scenario where resources are finite and whether the system remains robust under such constraints.

VII. RELATED WORK

This section delves into an review of existing literature and previous studies pertinent to this project in Table II. We evaluate how TMCD evolves to address the constraints

TABLE II
COMPARISON OF EXISTING SCALABLE SOLUTIONS

Category	Name	TPS	#Members	Incentives	Energy consump.	Safety
PoW/PoS	BitCoin [36]	4-10	∼400M	Yes	Extreme	50%
	Ethereum [4] [3]	∼10	∼250M	Yes	High	50%
	Algorand [26]	∼1000	Millions	No	Fair	Depends on crypto-sortition
	Blockene [41]	∼2000	Millions	No	Low	Depends on multiple RW
BFT	linear-PBFT [27] [40]	∼200	∼200	No	Fair	33%, depends on scale
	SBFT [27]	∼400	∼200	No	Fair	33%, limited by scale
(Ours)	DChain	∼8000	Millions	No	Low	33%, scalable

and limitations inherent in these prior works[1], offering an innovative approach that heralds advancements in the field.

One notable strategy is employed by Solana [43], which capitalizes on the Proof of History (PoH) concept to enhance blockchain consensus performance. This method, while innovative, necessitates substantial computational power from 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 [41] 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. Other solutions leveraging TSS include ByzCoin [31] and Motor [32]. These two advancements utilizes TSS as a primitive of fusing votes to improve efficiency and performance. However, they chose a different approach from DChain as they adopt a single-layer architecture. DChain splits the trust into two layers to provide even better performance while remaining security.

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 DChain, as these systems generally rely on Layer-1 consensus protocols. This underscores the potential of overlay ZKP-based networks when integrated with TMCD.

[1]Some #members and TPS data are obtained from technical reports (such as techopedia and theblock). Although they may be not accurate, the magnitudes can still be regarded comparable.

Sharding techniques enhance performance by segmenting the network into more manageable shards, interconnected through cross-shard transactions. Scaling solutions like Monoxide [42], RapidChain [45] and OmniLedger [33] 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 fundamental security protocols by reducing participant numbers in each sub-group.

DAG-based blockchain architectures [38] 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 [35], 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.

VIII. CONCLUSION

In this study, we introduced a groundbreaking framework known as TMCD, 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 TMCD is not merely its novelty; it is in its revolutionary approach to reconciling three 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. We demonstrate that TMCD 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 DChain and the TMCD 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/NfYQFzzkcEIy3jU9PTBo-nem2HlNiZre-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 crypto-currencies: 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, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. pages 279–296, 2016.

[32] Eleftherios Kokoris-Kogias. Robust and Scalable Consensus for Sharded Distributed Ledgers, 2019. Publication info: Preprint. MINOR revision.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.