Public Blockchain via Adaptive State Sharding with Zero-Knowledge Proofs and Secure Proof of Stake

The emergence of secure public blockchains, starting with Bitcoin and later Ethereum, has sparked widespread interest and substantial capital investment, laying the groundwork for a global surge in permissionless innovation. However, the ambitious goal of creating a decentralized, secure, and scalable public blockchain has proven to be a formidable challenge.

This paper presents AI8, a groundbreaking architecture that surpasses current state-of-the-art solutions by introducing a genuine state-sharding scheme for practical scalability. AI8## addresses issues of energy and computational waste while ensuring distributed fairness through a Secure Proof of Stake (SPoS) consensus mechanism and Zero-Knowledge Proofs . With a paramount emphasis
on security, AI8 network is meticulously designed to resist known
vulnerabilities such as Sybil attacks and Nothing at Stake attacks.

In an ecosystem that prioritizes interconnectivity, our smart contract solution incorporates an EVM-compliant engine, facilitating interoperability by design. Preliminary simulations and testnet results demonstrate that AI8AI8 not only surpasses Visa's average throughput but achieves an improvement exceeding three orders of magnitude or 1000x compared to existing viable approaches. Moreover, AI8 drastically reduces the costs associated with bootstrapping and storage, ensuring long-term sustainability.

In summary, AI8 represents a quantum leap in blockchain technology, offering a unique combination of scalability, security, and cost-effectiveness that positions it at the forefront of the evolving landscape of decentralized and permissionless innovation.

Overview:

The emergence of cryptocurrencies and smart contract platforms, exemplified by Bitcoin and Ethereum, has generated widespread interest, presenting promising solutions for electronic payments, decentralized applications, and potential digital stores of value. Despite their potential, a critical examination reveals that current public blockchain implementations face substantial challenges, especially in scalability, when compared to their centralized counterparts. These limitations hinder mainstream adoption and delay widespread public use.

In fact, the existing state of affairs highlights the formidable engineering constraints imposed by the trade-offs in the blockchain trilemma paradigm – the delicate balance between decentralization, security, and scalability. Addressing these challenges necessitates a thorough reconsideration of public blockchain infrastructures, as current iterations struggle to overcome the inherent limitations and trade-offs.

Various solutions have been proposed to tackle the scalability problem, yet only a handful have demonstrated significant and practical results. Therefore, resolving the scalability challenge requires not just incremental improvements but a fundamental reevaluation of the architecture and design principles governing public blockchains. This imperative shift aims to foster innovation that transcends existing engineering boundaries and provides a robust foundation for the mainstream adoption of decentralized technologies.

Numerous challenges must be meticulously addressed in the development of an innovative public blockchain solution designed for scalability. AI8# proposes a comprehensive solution to these challenges, introducing key notions such as users and nodes. Users, identified by unique account addresses, are external actors, while nodes are the computers/devices within the AI8 network that execute the protocol.

AI8 approach to addressing these challenges includes:

1. Shard-based Account Address Space Division: AI8 employs a binary tree to divide the account address space into shards, minimizing accumulated latency and enhancing network liveness. The predetermined hierarchy ensures no split overhead, and the state redundancy mechanism reduces latency during shard merges.
2. Node Balancing Technique: The introduction of a technique to balance nodes within each shard ensures overall architectural equilibrium. This approach distributes workload and rewards evenly among network nodes.
3. Automatic Transaction Routing Mechanism: AI8 incorporates a built-in mechanism for automatic transaction routing within corresponding shards, significantly reducing latency. The routing algorithm is detailed in chapter 4 - AI8 sharding approach.

4. Shard Pruning Mechanism for Bootstrapping and Storage Enhancement: AI8 implements a shard pruning mechanism to achieve substantial improvements in bootstrapping and storage. This ensures the sustainability of the architecture even with a high throughput of tens of thousands of transactions per second (TPS).

Addressing the overarching challenges of full decentralization, robust security, high scalability, efficiency, bootstrapping, storage enhancement, and cross-chain interoperability, AI8 represents a paradigm shift in public blockchain infrastructure. Its unique contributions are anchored in two fundamental building blocks:

1. Genuine State Sharding Approach: Effectively partitioning the blockchain and account state into multiple shards, managed in parallel by different participating validators.
2. Secure Proof of Stake Consensus Mechanism: An advanced iteration of Proof of Stake (PoS) ensuring long-term security and distributed fairness without relying on energy-intensive Proof of Work (PoW) algorithms.

This holistic approach positions AI8 as a revolutionary solution, purposefully designed for security, efficiency, scalability, and interoperability in the ever-evolving landscape of public blockchain technology.

AI8 introduces an innovatively dynamic and adaptive sharding mechanism, paving the way for shard computation and reorganization based on the evolving needs and the fluctuating number of active network nodes. The progressive and nondeterministic reassignment of nodes within shards at the onset of each epoch is executed seamlessly, causing no transient liveness penalties. While the adaptive state sharding proposed by AI8# presents promising advantages, it also brings forth unique challenges in comparison to static models.

A crucial aspect lies in how AI8 manages shard-splitting and shard-merging to mitigate potential latency penalties stemming from synchronization and communication requirements during changes in the shard count. In this context, latency refers to the communication overhead incurred by nodes as they retrieve the new state following modifications to their shard address space assignment. This intricate process is carefully orchestrated within the AI8 framework to ensure optimal performance without compromising the overall network efficiency.

We present a Secure Proof of Stake (SPoS) consensus mechanism that builds upon the concept of a random selection mechanism, as seen in Algorand [3], with distinctive features that set it apart:

1. Enhanced Latency Reduction: AI8 introduces a refinement that minimizes latency by enabling each node in the shard to ascertain the consensus group members (block proposer and validators) at the onset of a round. This improvement is achieved through the inclusion of a randomization factor 'r,' stored in every block and generated by the block proposer using a BLS signature [4] on the previous 'r.'
2. Efficient Block Proposer Selection: The block proposer in AI8 consensus group is determined by the smallest hash of the public key and randomization factor. In contrast to Algorand's approach, where the committee selection can take up to 12 seconds, AI8## significantly reduces the time required for random selection (estimated under 100 ms), excluding network latency. This streamlined process allows AI8## to have a freshly and randomly selected group capable of committing a new block to the ledger in each round. The tradeoff is based on the assumption that an adversary cannot adapt faster than the round's timeframe, retaining the option to refrain from proposing a block. To bolster the security of the randomness source, a further enhancement involves considering verifiable delay functions (VDFs) to prevent tampering, although current VDF research is ongoing.
3. Innovative Consensus Weighting with Rating: AI8 refines its consensus mechanism by introducing an additional weight factor called "rating" alongside the commonly used stake factor in PoS architectures. The node's probability of being selected in the consensus group is determined by both stake and rating. Rating is recalculated at the end of each epoch, except in instances of slashing, where the decrease occurs instantly, promoting an additional layer of security through a meritocratic approach.
4. Modified BLS Multisignature Scheme: The consensus group utilizes a modified BLS multisignature scheme [5] with two communication rounds for block signing.
5. Formal Verification for Critical Protocols: AI8 emphasizes formal verification for critical protocol implementations, such as the SPoS consensus mechanism, to validate the correctness of algorithms, ensuring a robust foundation for the blockchain network.

Architecture Overview: Entities

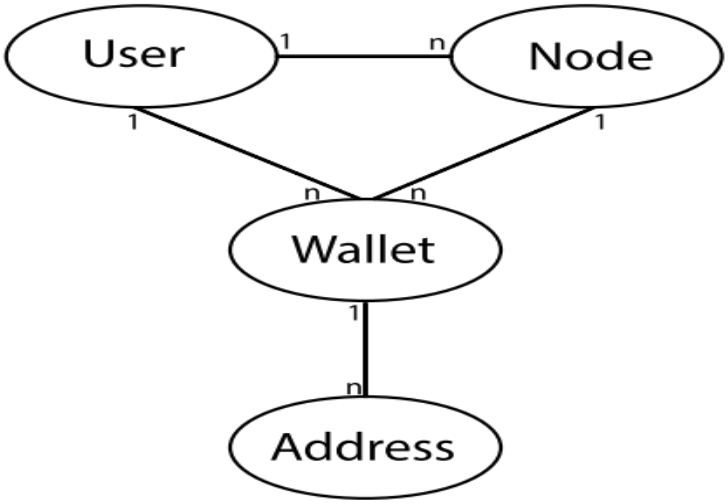
Within the AI8 ecosystem, two primary entities coexist: users and nodes. Users, each possessing a finite number of public/private key pairs (Pk/sk) stored in various wallet applications, leverage the AI8 network to deploy signed transactions, facilitating value transfers or the execution of smart contracts. Users are identifiable by one of their account addresses, derived from their public key. On the other hand, nodes represent the devices constituting the AI8 network, serving in either passive or active capacities for processing tasks.

Eligible validators, a subset of nodes, play a crucial role as active participants in AI8 network. Their responsibilities encompass running consensus, adding blocks, maintaining the state, and earning rewards for their contributions. Each eligible validator is uniquely identified by a public key, constructed through the derivation of the address that staked the requisite amount and the corresponding node ID.

The relationships between these entities are illustrated in Figure 1, highlighting the interconnected nature of users, nodes, and eligible validators within the AI8 protocol.

Moreover, the network architecture involves the subdivision of the entire network into smaller units known as shards. An algorithm governs the assignment of eligible validators to specific shards, ensuring an even distribution of nodes across shards based on the tree level. Within each shard, a consensus group is randomly selected. The block proposer in this group is tasked with aggregating transactions into a new block. Validators within the group then assume the responsibility of either approving or rejecting the proposed block, thereby validating and committing it to the blockchain.

This intricate system of entities and their interplay forms the foundation of the AI8 architecture, fostering a decentralized and efficient environment for transactions and smart contract executions.



Native Token: Intrinsic Value

The AI8 network provides access to its functionalities through its native utility tokens, referred to as AI8 or simply AI8-s. These intrinsic tokens serve as the primary unit for covering all costs associated with processing transactions, executing smart contracts, and rewarding contributors to the network. Any mentions of fees, payments, or balances within the AI8 ecosystem are inherently denominated in AI8-s, underscoring the integral role these tokens play in facilitating the seamless operation and sustainability of the network.

Security Framework

AI8 operates under the assumption of a Byzantine adversarial model, where a minimum of 2n+1 eligible nodes in a shard are considered honest to achieve consensus. The protocol accommodates adversaries with stake, good ratings, potential delays, conflicting messages, compromised nodes, bugs, or collusion. As long as 2n+1 eligible validators in a shard remain honest, the protocol maintains the ability to achieve consensus.

The protocol addresses highly adaptive adversaries that cannot outpace a round's timeframe. Adversaries are computationally bounded, ensuring the cryptographic assumptions align with the security level of chosen primitives within the polynomial time complexity class of problems solvable by a Turing machine. Honest nodes form a well-connected graph, facilitating the timely propagation of messages.

Prevention of Attack Vectors

1. Sybil Attacks: Alleviated through stake locking during network entry, imposing a cost equivalent to the minimum stake for generating new identities.
2. Nothing at Stake: Mitigated through the necessity of multiple signatures, not limited to the proposer, and stake slashing. The reward per block compared to the locked stake discourages such behavior.
3. Long-Range Attacks: Mitigated by the pruning mechanism, a randomly selected consensus group each round, stake locking, and the pBFT consensus algorithm ensuring finality.
4. DDoS Attacks: The consensus group undergoes random sampling every few seconds, rendering DDoS attempts almost impossible due to the small timeframe.

Considered Attack Vectors

Beyond the mentioned vectors, AI8 also addresses potential threats such as shard takeover attacks, transaction censorship, double spending, bribery attacks, among others. The multifaceted security framework demonstrates AI8's commitment to robustness against a wide spectrum of adversarial scenarios

Temporal Framework

Within AI8's network, the temporal structure is delineated into epochs and rounds. Epochs, with a current fixed duration set at one day (subject to modification with evolving architecture), conclude with the reorganization and pruning of shards. Each epoch is further subdivided into rounds, each persisting for a predetermined timeframe. In every round, a new consensus group is randomly selected for each shard, empowered to commit a maximum of one block to the shard's ledger.

Prospective validators can integrate into the network by staking, following the mechanism outlined in Chapter V.2 - Secure Proof of Stake. Upon joining, they enter the unassigned node pool during the current epoch (e). Subsequently, they are allocated to the waiting list of a shard at the onset of the next epoch (e + 1). However, these new validators attain eligibility to participate in consensus and receive rewards only in the subsequent epoch (e + 2).

Detailed elucidation of the temporal intricacies is available in Section IX.1, providing a comprehensive understanding of AI8's timeline dynamics. This structured approach contributes to the systematic organization and execution of network activities, promoting efficiency and reliability in AI8's operational chronology.

Evolutionary Foundations

AI8 draws inspiration and builds upon the ideas of several blockchain projects, including Ethereum [6], Omniledger [7], Zilliqa [8], Algorand [3], and Chainspace [9]. Rooted in this collective wisdom, AI8's architecture advances the state of the art, enhancing performance while seeking an optimal Nash equilibrium among security, scalability, and decentralization.

1. Ethereum

Ethereum's success lies in decentralized applications facilitated by EVM, Solidity, and Web3j. While Ethereum's scalability faces challenges, ongoing research, including initiatives like Casper and Plasma-based side-chains, aims to address these limitations. AI8 differentiates itself by eliminating energy and computational waste with SPoS consensus and introducing transaction processing parallelism through sharding.

2. Omniledger

Omniledger proposes a scale-out distributed ledger ensuring long-term security. AI8 adapts by introducing an adaptive state sharding approach, faster consensus group selection, and enhanced security with frequent validator set replacements.

3. Zilliqa

Zilliqa pioneers transaction sharding for high throughput, utilizing pBFT for consensus and PoW for identity establishment. AI8 goes beyond by incorporating both transaction and state sharding, replacing PoW with SPoS for consensus, and prioritizing EVM compliance for smart contracts.

4. Algorand

Algorand aims for decentralized efficiency without the weaknesses of existing implementations. AI8 distinguishes itself through sharding for increased throughput, a more rapid consensus group selection, and an assumption that adversaries cannot adapt within a round.

5. Chainspace

Chainspace focuses on high-integrity transaction processing with language-agnostic smart contracts and privacy features. AI8, while addressing blockchain size concerns with an efficient pruning mechanism, ensures a fixed-size consensus group for scalability, resilience to node population changes, and resistance to malicious shard takeovers.

In essence, AI8's architectural evolution is a synthesis of the innovative ideas from its predecessors, pushing boundaries to achieve a harmonious balance between security, scalability, and decentralization in the ever-evolving blockchain landscape

Scalability through Dynamic State Sharding

The Rationale for Sharding

Originally employed in databases, sharding is a technique for distributing data across multiple machines. In the context of blockchains, sharding involves partitioning states and transaction processing, enabling each node to handle a subset of transactions concurrently. By ensuring a sufficient number of nodes verify each transaction, maintaining high reliability and security, sharding facilitates parallel transaction processing. This approach, known as horizontal scaling, holds the promise of significantly improving transaction throughput and efficiency as the validator network expands.

Sharding Types

An in-depth exploration [16] highlights three primary types of sharding: network sharding, transaction sharding, and state sharding. Network sharding optimizes node grouping into shards, enhancing communication efficiency within a shard. The mapping of nodes to shards must consider potential attacks from attackers gaining control over specific shards. Transaction sharding manages how transactions are assigned to the shards for processing, often based on the sender's address in account-based systems. State sharding, the most challenging type, involves maintaining only a portion of the state within each shard. Transactions involving accounts in different shards necessitate message exchange and state updates in multiple shards. To enhance resilience to attacks, nodes in shards need periodic reshuffling. However, redistributing nodes introduces synchronization overhead, requiring careful subset redistribution to avoid downtime during synchronization.

Sharding Approaches

Some sharding proposals focus on sharding transactions or sharding state individually, potentially burdening nodes with excessive state data or computational demands. Recent claims suggest success in performing both transaction and state sharding

simultaneously without compromising storage or processing power. Despite the benefits, sharding introduces new challenges, including single-shard takeover attacks, cross-shard communication, data availability, and the need for an abstraction layer to conceal shards. Properly addressed, these challenges can lead to significant overall improvements: increased transaction throughput and reduced transaction fees, two critical factors poised to drive mainstream adoption of blockchain technology.

AI8 sharding approach

While dealing with the complexity of combining network, transaction and state sharding, AI8's approach was designed with the following goals in mind:

1) Scalability without affecting availability: Increasing or decreasing the number of shards should affect a negligibly small vicinity of nodes without causing down-times, or minimizing them while updating states;
2) Dispatching and instant traceability: Finding out the destination shard of a transaction should be determinis-tic, trivial to calculate, eliminating the need for commu-nication rounds;
3) Efficiency and adaptability: The shards should be as balanced as possible at any given time.

Method Description

To calculate an optimum number of shards N_{sh} in epoch e_{i+1} ($N_{sh;i+1}$), we have defined one threshold coefficient for the number of transactions in a block, TX. Variable optN represents the optimal number of nodes in a shard, sh is a positive number and represents the number of nodes a shard can vary by. totalNi is the total number of nodes (eligible validators, nodes in the waiting lists and newly added nodes in the node pool) on all shards in epoch e_i, while $NTXB;i$ is the average number of transactions in a block on all shards in epoch e_i. $Nsh;0$ will be considered as 1. The total number of shards $Nsh;i+1$ will change if the number of nodes totalNi in the network changes and if the blockchain utilization needs it: if the number of nodes increases above a threshold nSplit from one epoch to another and the average number of transactions per block is greater than the threshold

number of transactions per block $NTXB;i > TX$ or if the number of nodes decreases below a threshold nMerge as shown in function ComputeShardsN.

```
1: function COMPUTESHARDSN(totalNi + 1 ;Nsh ; i ) 2:
       nSplit (Nsh ; i  + 1) (optN + sh )
3:     nMerge (Nsh ; i       1) a
4:     N_sh;i+1          N_sh;i
5:     if (totalNi + 1  > nSplit and NTXB; i > TX ) then 6:
Nsh ; i + 1                    totalNi + 1 =(optN + sh )
7:     else if totalNi + 1  < nMerge then
8:                Nsh ; i + 1   totalNi + 1 =(optN)
9:      return Nsh ; i + 1
```

From one epoch to another, there is a probability that the number of active nodes changes. If this aspect influences the number of shards, anyone can calculate the two masks m1 and m2, used in transaction dispatching.

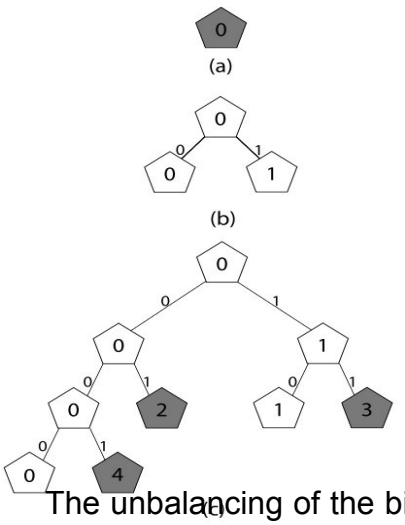
```
1: function COMPUTEM1ANDM2(Nsh )
2:     n    math.ceil(log2 Nsh )
3:     m1   (1 < < n)     1
```

4: m_2 (1 < < (n 1)) 1 5: return $m_1;m_2$

As the main goal is to increase the throughput beyond thousands of transactions per second and to diminish the cross-shard communication, AI8## proposes a dispatching mecha-nism which determines automatically the shards involved in the current transaction and routes the transaction accordingly. The dispatcher will take into consideration the account address (addr) of the transaction sender/receiver. The result is the number of the shard (shard) the transaction will be dispatched to.

1: function COMPUTESHARD(Ns h ;addr;m1 ;m2) 2:
shard (addr and m1)
3: if shard > Ns h then
4: shard (addr and m2) 5:
return shard

The entire sharding scheme is based on a binary tree structure that distributes the account addresses, favors the scalability and deals with the state transitions. A representation of the tree can be seen in Fig. 2.



The presented tree structure is merely a logical represen-tation of the account address space used for a deterministic mapping; e.g. shard allocation, sibling computation etc. The leaves of the binary tree represent the shards with their ID number. Starting from root (node/shard 0), if there is only one shard/leaf (a), all account addresses are mapped to this one and all transactions will be executed here. Further on, if the formula for Ns h dictates the necessity of 2 shards (b), the address space will be split in equal parts, according to the last bits in the address.

Sometimes, the tree can also become unbalanced (c) if Ns h is not a power of 2. This case only affects the leaves on the last level. The structure will become balanced again when the number of shards reaches a power of 2.

The unbalancing of the binary tree causes the shards located in the lowest level to have half the address space of nodes of a shard located one level higher, so it can be argued that the active nodes allocated to these shards will have a lower fee income - block rewards are not affected. However, this problem is solved by having a third of each shard nodes redistributed randomly each epoch (detailed in the Chronology section) and having a balanced distribution of nodes according to the tree level.

Looking at the tree, starting from any leaf and going through branches towards the root, the encoding from branches represents the last n bits of the account addresses that will have their associated originating transactions processed by that leaf/shard. Going the other way around, from root to leaf, the information is related to the evolution of the structure, sibling shards, the parent shard from where they split. Using this hierarchy, the shard that will split when Ns h increases or the shards that will merge when Ns h decreases can easily be calculated. The entire state sharding mechanism benefits from this structure by always keeping the address and the associated state within the same shard.

Knowing Ns h , any node can follow the redistribution pro-cess without the need of

communication. The allocation of ID's for the new shards is incremental and reducing the number of shards involves that the higher numbered shards will be removed. For example, when going from N_{shard} to $N_{shard}-1$, two shards will be merged, the shard to be removed is the highest numbered shard ($shard_{merge}=N_{shard}-1$). Finding the shard number that $shard_{merge}$ will be merged with is trivial. According to the tree structure, the resulting shard has the sibling's number:

```
1: function COMPUTESIBLING(shard_merge; n)
2:     sibling ← (shard_merge xor (1 << (n − 1)))
3:     return sibling
```

For shard redundancy, traceability of the state transitions and fast scaling, it is important to determine the sibling and parent of a generic shard with number p:

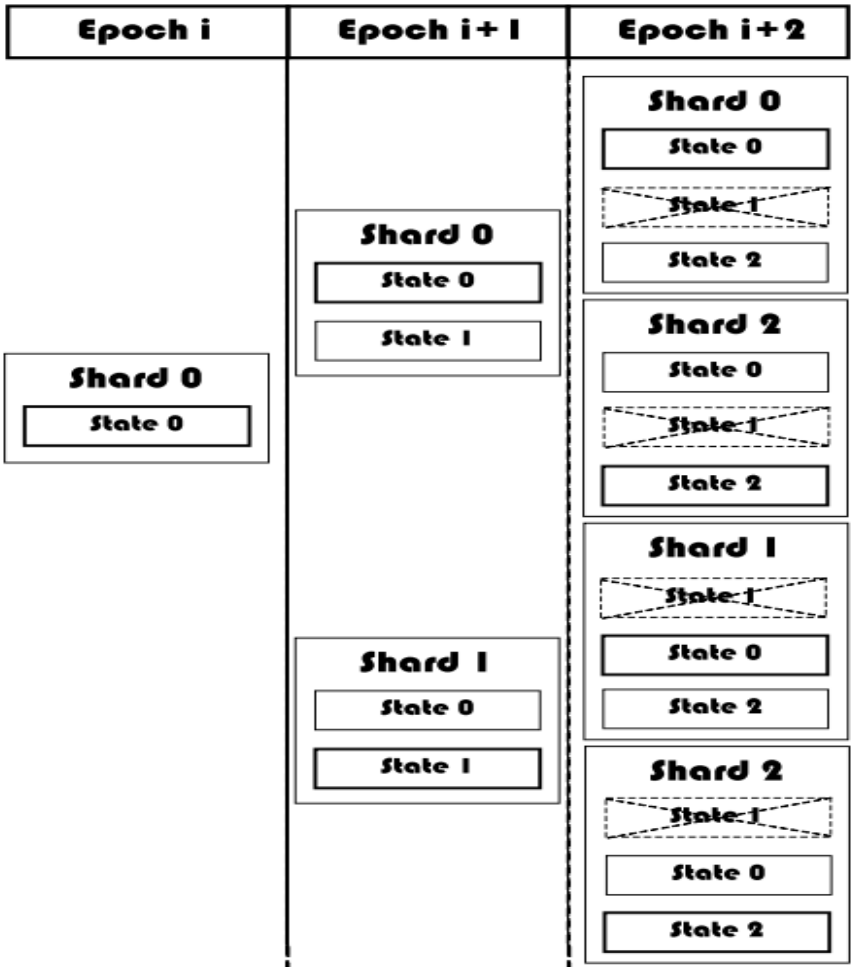
```
1: function COMPUTEPARENTSIBLINGS(n; p; N_shard)
2:     mask1 ← 1 << (n − 1)
3:     mask2 ← 1 << (n − 2)
4:     sibling ← (p xor mask1)
5:     parent ← min(p; sibling)
6:     if sibling < N_shard then
7:         sibling ← (p xor mask2)
8:         sibling2 ← (sibling xor mask1)
9:         parent ← min(p; sibling)
10:        if sibling2 < N_shard then . sibling is a shard
11:            return parent; sibling; NULL
12:        else
13:            . sibling is a subtree with
14:            . shards (sibling; sibling2)
15:            return parent; sibling; sibling2
16:    else . sibling is a shard
17:        return parent; sibling; NULL
```

Shard Redundancy

In the realm of blockchain, the vulnerability of state sharding to shard failure arises when a shard lacks an adequate number of online nodes or experiences geographical concentration. In the rare event of shard failure, where either the shard is unreachable (all nodes offline) or consensus is unattainable (more than one node unresponsive), there's a potential reliance on super-full nodes—nodes that fully download and verify every block of every shard. Illustrated in Fig. 3, our protocol incorporates a protective mechanism introducing a tradeoff in the state-holding structure. It mandates shards from the last tree level to also maintain the state from their siblings, mitigating communication and eliminating the need for bootstrapping during the merging of sibling shards.

Context Switching

In the context of sharded public blockchains, ensuring security necessitates the implementation of context switching [7]. This involves the periodic reallocation of active nodes among shards based on some random criteria. AI8##'s approach considers context switching not only as a security enhancement but also as a factor that complicates the maintenance of consistency across multiple states. The state transition significantly impacts performance as the movement of active nodes requires resynchronization of state, blockchain, and transactions alongside the eligible nodes in the new shard. At the onset of each epoch, to sustain liveness, fewer than 1% of these nodes are uniformly redistributed across shards. This mechanism proves highly effective in preventing the formation of malicious groups.



All network and global data operations, encompassing node additions and departures, computation of eligible validator lists, nodes' assignment to shard waiting lists, and the resolution of challenges related to invalid blocks, will undergo notarization in the metachain. The metachain's consensus is orchestrated by a distinct shard responsible for communicating with all other shards and facilitating cross-shard operations. In each round of every epoch, the metachain receives block headers from other shards and, if necessary, proofs pertaining to challenges associated with invalid blocks. This information is then consolidated into metachain blocks, subject to consensus. After

validation by the consensus group, shards can solicit information about blocks, miniblocks (refer to Chapter VII for details), eligible validators, nodes in waiting lists, etc. This exchange of information ensures the secure processing of cross-shard transactions. For a more in-depth understanding of the cross-shard transaction execution, communication protocols between shards and the metachain, please refer to Chapter VII on Cross-shard Transaction Processing.

V. Consensus through Secure Proof of Stake

In the genesis of blockchain consensus, Proof of Work (PoW) emerged, harnessed by Bitcoin, Ethereum, and other platforms. PoW mandates nodes to solve intricate mathematical puzzles, with the first successful node reaping the reward. While effective in preventing double-spending and certain attacks, PoW comes at the expense of high energy consumption.

Introducing Proof of Stake (PoS) as a more efficient alternative, it factors in stake (wealth), randomness, and/or age to select the node proposing the next block. PoS mitigates energy concerns but introduces challenges like the Nothing at Stake attack and increased centralization risk. Constellation's Proof of Meme builds on historical node participation, addressing Sybil attacks through the NetFlow algorithm.

Delegated Proof of Stake (DPoS), featured in Bitshares, Steemit, and EOS, combines Proof of Authority and Proof of Stake. Elected nodes deploy new blocks, yet it faces human-related issues like bribery and corruption, along with susceptibility to DDoS attacks.

Secure Proof of Stake (SPoS)

AI8's consensus strategy combines random validator selection, eligibility through stake and rating, with an optimal consensus group size. The algorithm unfolds as follows:

1. Each node (ni) is defined by a tuple of public key (Pk), rating (default at 0), and locked stake. To partake in the consensus, ni registers through a smart contract, sending a transaction with an amount matching the minimum required stake and additional information.
2. Node ni joins the node pool, awaiting shard assignment at the epoch's end (epoch e). The shard assignment mechanism creates a new set of nodes comprising those joining in epoch e and those requiring reshuffling (less than 1% of each shard). This set is then reassigned to waiting lists of shards (Wj, where j represents the shard, and Nsh is the number of shards). Each node possesses a secret key (sk) kept confidential by nature.

$$n_i = (Pk_i ; rating_i ; stake_i)$$

$$n_i \in W_{j;0} \; j < N_{sh}$$

3) At the end of the epoch in which it has joined, the node will be moved to the list of eligible nodes (E_j) of a shard j, where e is the current epoch.

$$n_i \in W_{j;e+1} \Leftrightarrow n_i \in W_{j;e} \wedge n_i \in E_{j;e}$$

4) Each node from the list E_j can be selected as part of an optimally dimensioned consensus group (in terms of se-curity and communication), by a deterministic function, based on the randomness source added to the previous block, the round r and a set of variation parameters. The random number, known to all shard nodes through gossip, cannot be predicted before the block is actually signed by the previous consensus group. This property makes it a good source of randomness and prevents highly adaptive malicious attacks. We define a selection function to return the set of chosen nodes (consensus group) N_{chosen} with the first being the block proposer, that takes following parameters: E, r and sig_{r-1} - the previous block signature.

$$N_{chosen} = f(E;r;sig_{r-1}); \text{where } N_{chosen} \subset E$$

5) The block will be created by the block proposer and the validators will co-sign it based on a modified practical Byzantine Fault Tolerance (pBFT).
6) If, for any reason, the block proposer did not create a block during its allocated time slot (malicious, offline, etc.), round r will be used together with the randomness source from the last block to select a new consensus group.

If the current block proposer acts in a malicious way, the rest of the group members apply a negative feedback to change its rating, decreasing or even cancelling out the chances that this particular node will be selected again. The feedback function for the block proposer (n_i) in round number r, with parameter ratingModifier $\in Z$ is computed as:

$$feedbackfunction = ff(n_i;ratingModifier;r)$$

When ratingModifier < 0, slashing occurs so the node n_i loses its stake.

The consensus protocol remains safe in the face of DDoS attacks by having a high number of possible validators from the list E (hundreds of nodes) and no way to predict the order of the validators before they are selected.

To reduce the communication overhead that comes with an increased number of shards, a consensus will be run on a composite block. This composite block is formed by:

Ledger block: the block to be added into the shard's ledger, having all intra shard transactions and cross shard transactions for which confirmation proof was received;
Multiple mini-blocks: each of them holding cross shard transactions for a different shard;

The consensus will be run only once, on the composite block containing both intra-and cross-shard transactions. After consensus is reached, the block header of each shard is sent to the metachain for notarization

VI Cryptographic Layer Signature Analysis

Digital signatures are cryptographic primitives used to achieve information security by providing several properties like message authentication, data integrity and non-repudiation [24].

Most of the schemes used for existing blockchain platforms rely on the discrete

logarithm (DL) problem: one-way expo-nentiation function $y ! {}^{y} \bmod p$. It is scientifically proven that calculating the discrete logarithm with base is hard [25]. Elliptic curve cryptography (ECC) uses a cyclic group of points instead of a cyclic group of integers. The scheme reduces the computational effort, such that for key lengths of only 160 - 256 bits, ECC provides same security level that RSA, Elgamal, DSA and others provide for key lengths of
1024 - 3072 bits (see Table 1 [24]).
The reason why ECC provides a similar security level for much smaller parameter lengths is because existing attacks on elliptic curve groups are weaker than the existing integer DL attacks, the complexity of such algorithms require on average p steps to solve. This means that an elliptic curve using a prime p of 256 bit length provides on

average a security of
2^{128} steps needed to break it [24].

Both Ethereum and Bitcoin use curve cryptography, with the ECDSA signing algorithm. The security of the algorithm is very dependent on the random number generator, because if the generator does not produce a different number on each query, the private key can be leaked [26].

Another digital signature scheme is EdDSA, a Schnorr variant based on twisted Edwards curves that support fast arithmetic [27]. In contrast to ECDSA, it is provably non-malleable, meaning that starting from a simple signature, it is impossible to find another set of parameters that defines the same point on the elliptic curve [28], [29]. Additionally, EdDSA doesn't need a random number generator because it uses a nonce, calculated as the hash of the private key and the message, so the attack vector of a broken random number generator that can reveal the private key is avoided.

Schnorr signature variants are gaining more attention [8], [30] due to a native multi-signature capability and being provably secure in the random oracle model [31]. A multi-signature scheme is a combination of a signing and verification algorithms, where multiple signers, each with their own private and public keys, can sign the same message, producing a single signature [32], [33]. This signature can then be checked by a verifier which has access to the message and the public keys of the signers. A sub-optimal method would be to have each node calculate his own signature and then concatenate all results in a single string. However, such an approach is unfeasible as the generated string size grows with the number of signers. A practical solution would be to aggregate the output into a single fixed size signature, independent of the number of participants. There have been multiple proposals of such schemes, most of them are susceptible to rogue-key (cancellation) attacks. One solution for this problem would be to introduce a step where each signer needs to prove possession of the private key associated with its public key [34].
Bellare and Neven [35] (BN) proposed a secure multi-signature scheme without a proof of possession, in the plain public key model, under the discrete logarithm assumption [31]. The participants commit first to their share Ri by prop-agating its hash to all other signers so they cannot calculate a function of it. Each signer computes a different challenge for their partial signature. However, this scheme sacrifices the public key aggregation. In this case, the verification of the aggregated signature, requires the public key from each signer.
A recent paper by Gregory Maxwell et al. [29] proposes another multi-signature scheme in the plain public key model [36], under the 'one more discrete logarithm' assumption (OMDL). This approach improves the previous scheme [35] by reducing the communication rounds from 3 to 2, reintroducing
the key aggregation with a higher complexity cost.

BLS [4] is another interesting signature scheme, from the Weil pairing, which bases its security on the Computational Diffie-Hellman assumption on certain elliptic curves and gen-erates short signatures. It has several useful properties like batch verification, signature aggregation, public key aggrega-tion, making BLS a good candidate for threshold and multi-signature schemes.

Dan Boneh, Manu Drijvers and Gregory Neven recently proposed a BLS multi-signature scheme [5], using ideas from the previous work of [35], [30] to provide the scheme with defenses against rogue key attacks. The scheme supports efficient verification with only two pairings needed to verify a multi-signature and without any proof of knowledge of the secret key (works in the plain public key model). Another advantage is that the multi-signature can be created in only two communication rounds.

For traceability and security reasons, a consensus based on a reduced set of validators requires the public key from each signer. In this context, our analysis concludes that the most appropriate multi-signature scheme for block signing in AI8## is BLS multi-signature [5], which is faster overall than the other options due to only two communication rounds.

Algorithm Family	Crypto systems	Security level (bit)			
		80	128	192	256
Integer factorization	RSA	1024	3072	7680	15360
Discrete logarithm	DH, DSA, Elgamal	1024	3072	7680	15360
Elliptic curves	ECDH, ECDSA	160	256	384	512
Symmetric key	AES, 3DES	80	128	192	256

Block signing in AI8

For block signing, AI8 uses curve cryptography based on the BLS multi-signature scheme over the bn256 bilinear group, which implements the Optimal Ate pairing over a 256-bit Barreto Naehrig curve. The bilinear pairing is defined as:

$$e : g0\ g1\ !\ gt \qquad (1)$$

where g0 , g1 and gt are elliptic curves of prime order p defined by bn256, and e is a bilinear map (i.e. pairing function). Let G0 and G1 be generators for g0 and g1 . Also, let H0 be a hashing function that produces points on the curve g0 :

$$H0 : M\ !\ g0 \qquad (2)$$

where M is the set of all possible binary messages of any length. The signing scheme used by AI8 employs a second hasing function as well, with parameters known by all signers:

$$H1 : M\ !\ Zp \qquad (3)$$

cE a c h si g n er i h a s it s o w n pr iv a te / p ub l ic k e y pa ir h o s en f r om Z p · F or e ac h k ey p a ir , th e p r op er t y P k i (s k ¡; P k ¡), w h e r e sk$_i$ is randomly
= s k i G 1 h ol d s .

Let $L = Pk_1;::::;Pk_n$ be the set of public keys of all possible signers during a specific round which, in the case of AI8, is the set of public keys of all the nodes in the consensus group. Below, the two stages of block signing process is presented: signing and verification.

Practical signing - Round 1

The leader of the consensus group creates a block with transactions, then signs and broadcasts this block to the consensus group members.

Practical signing - Round 2

Each member of the consensus group (including the leader) who receives the block must validate it, and if found valid, it signs it with BLS and then sends the signature to the leader:

$$Sigi = ski \; H0(m) \qquad (4)$$ where $Sigi$ is a point on $g0$.

Practical signing - Round 3

The leader waits to receive the signatures for a specific timeframe. If it does not receive at least $\frac{2}{3}$ n + 1 signatures in that timeframe, the consensus round is aborted. But if the leader does receive $\frac{2}{3}$ n + 1 or more valid signatures, it uses them to generate the aggregated signature:

$$SigAgg = \sum_i H1(P ki) Sigi \; B[i] \qquad (5)$$

where SigAgg is a point on $g0$.

The leader then adds the aggregated signature to the block together with the selected signers bitmap B, where a 1 indicates that the corresponding signer in the list L had its signature added to the aggregated signature SigAgg.

Practical verification

Given the list of public keys L, the bitmap for the signers B, the aggregated signature SigAgg, and a message m (block), the verifier computes the aggregated public key:

$$PkAgg = \sum_i H1(P ki) Pki \; Bi \qquad (6)$$

The result, PkAgg, is a point on $g1$. The final verification is $e(G1;SigAgg) = =$

$$e(PkAgg;H0(m)) \qquad (7)$$

where e is the pairing function.

Cross-shard Execution

For an in depth example of how the cross-shard transactions are being executed and how the communication between shards and the metachain occurs, we are simplifying the entire process to just two shards and the metachain. Assuming that a user generates a transaction from his wallet, which has an address in shard 0 and wants to send ERDs to another user that has a wallet with an address in shard 1, the steps depicted in Fig. 4 are required for processing the cross-shard transaction. As mentioned in chapter V -Consensus via Secure Proof of Stake, the blocks structure is represented by a block Header that contains information about the block (block nonce, round, proposer, validators timestamp etc), and a list of miniblocks for each shard that contain the actual transactions inside. Every miniblock contains all transactions that have either the sender in the current

shard and the receiver in another shard or the sender in a different shard and the destination in the current shard. In our case, for a block in shard 0, there will normally be 3 miniblocks:

miniblock 0: containing the intrashard transactions for shard 0
miniblock 1: containing cross-shard transactions with the sender in shard 0 and destination in shard 1
miniblock 2: containing cross-shard transactions with sender in shard 1 and destination in shard 0. Thesetransactions were already processed in the sender shard 1 and will be finalized after the processing also in the current shard.

There is no limitation on the number of miniblocks with the same sender and receiver in one block. Meaning multiple miniblocks with the same sender and receiver can appear in the same block.

Processing

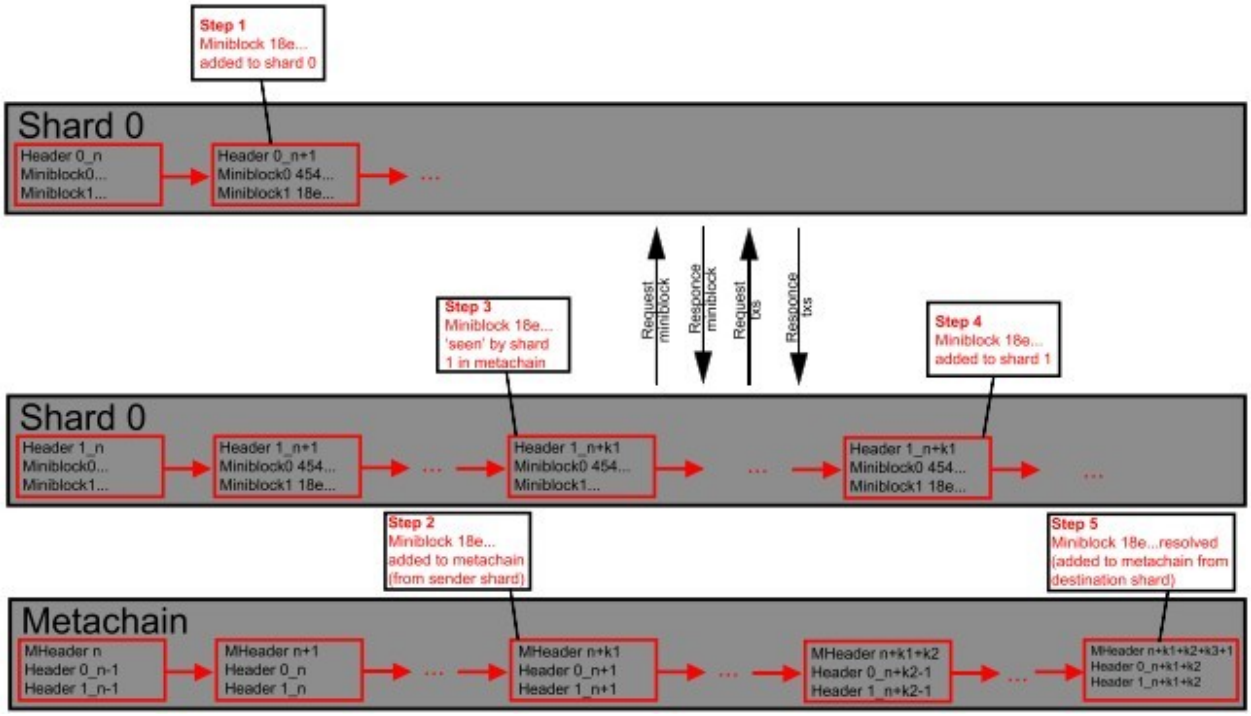
Currently the atomic unit of processing in cross-shard execution is a miniblock: either all the transactions of the miniblock are processed at once or none and the miniblock's execution will be retried in the next round.

Our cross-shard transaction strategy uses an asynchronous model. Validation and processing is done first in sender's shard and then in receivers' shard. Transactions are first dispatched in the sender's shard, as it can fully validate any transaction initiated from the account in this shard – mainly the current balance. Afterwards, in the receivers' shard, the nodes only need proof of execution offered by metachain, do signature verification and check for replay attack and finally update the balance for the receiver, adding the amount from the transaction.

Shard 0 processes both intra-shard transactions in miniblock 0 and a set of cross-shard transactions that have addresses from shard 1 as a receiver in miniblock 1. The block header and miniblocks are sent to the metachain. The metachain notarizes the block from shard 0, by creating a new metachain block (metablock) that contains the following information about each miniblock: sender shard ID, receiver shard ID, miniblock hash.

Shard 1 fetches the hash of miniblock 1 from metablock, requests the miniblock from shard 0, parses the transaction list, requests missing transactions (if any), executes the same miniblock 1 in shard 1 and sends to the metachain resulting block. After notarization the cross transaction set can be considered finalized.

The next diagram shows the number of rounds required for a transaction to be finalized. The rounds are considered between the first inclusion in a miniblock until the last miniblock is notarised.



Smart Contracts

The execution of smart contracts is a key element in all future blockchain architectures. Most of the existing solutions avoid to properly explain the transactions and data dependency. This context leads to the following two scenarios:

1) When there is no direct correlation between smart con-tract transactions, as displayed in Fig. 5, any architecture can use out of order scheduling. This means there are no additional constraints on the time and place (shard) where a smart contract is executed.
2) The second scenario refers to the parallelism induced by the transactions that involve correlated smart contracts [37]. This case, reflected in Fig. 6, adds additional pressure on the performance and considerably increases the complexity. Basically there must be a mechanism to ensure that contracts are executed in the right order and on the right place (shard). To cover this aspect, AI8## protocol proposes a solution that assigns and moves the smart contract to the same shard where their static dependencies reside. This way most, if not all SC calls will have dependencies in the same shard and no cross-shard locking/unlocking will be needed.

AI8## focuses on the implementation of the AI8## Virtual Machine, an EVM compliant engine.

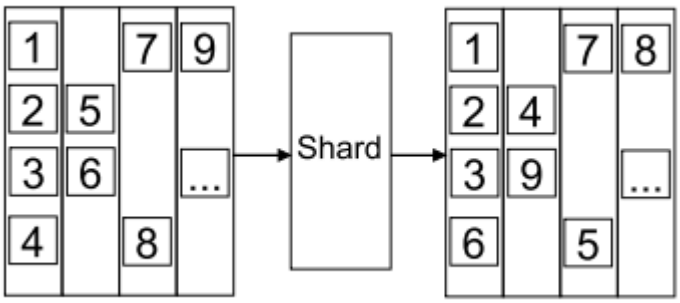


Fig. 5: Independent transaction processing under simple smart contracts that can be executed out of order

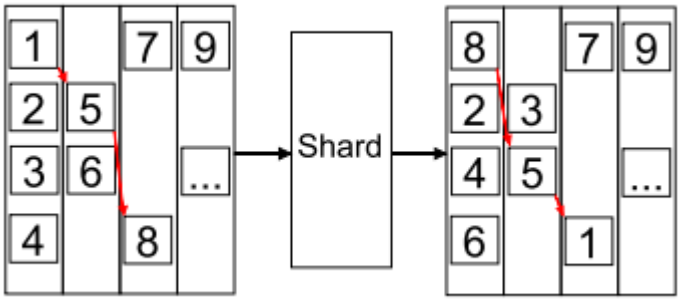


Fig. 6: Mechanism for correlated smart contracts that can be executed only sequentially

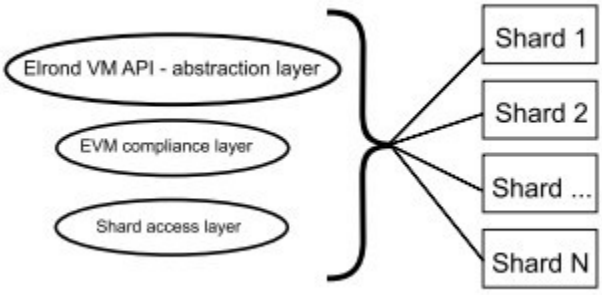


Fig. 7: Abstraction Layer for Smart Contracts

The EVM compliance is extremely important for adoption purposes, due to the large number of smart contracts built on Ethereum's platform.

The AI8 Virtual Machine's implementation will hide the underlying architecture isolating the smart contract developers from system internals ensuring a proper abstraction layer, as displayed in Fig. 7.

In AI8, cross chain interoperability can be implemented by using an adapter

mechanism at the Virtual Machine level as proposed by Cosmos [38]. This approach requires specialized adapters and an external medium for communication between adapter SC for each chain that will interoperate with AI8. The value exchange will be operated using some specialized smart contracts acting as asset custodians, capable of taking custody of adapted chain native tokens and issuing AI8 native tokens.

VM Infrastructure: Unleashing AI8's Power

AI8's VM infrastructure is meticulously crafted atop the K Framework, a groundbreaking executable semantic framework. This framework serves as a versatile platform where programming languages, calculi, type systems, and formal analysis tools can be precisely defined [39].

The pivotal strength of embracing the K Framework lies in its ability to unambiguously define smart contract languages. By eradicating the potential for unspecified behavior and elusive bugs, AI8 ensures a robust and bug-resistant foundation for its smart contract execution.

What sets the K Framework apart is its executability—semantic specifications of languages can seamlessly function as working interpreters for the designated languages. This execution capability manifests in two ways: programs can either be run against the specifications using the K Framework's core implementation, or interpreters can be generated in various programming languages, often termed as "backends."

AI8, prioritizing execution speed and interoperability, has developed its bespoke K Framework backend. This strategic decision not only enhances performance but also ensures seamless interoperability, cementing AI8's commitment to pushing the boundaries of innovation in the realm of blockchain VM infrastructure.

Smart Contract Languages: A Symphony of K Framework Innovation

The K Framework unfolds its brilliance by offering a unique advantage—generating interpreters for any language defined within K without the need for additional programming. This distinctive feature ensures that interpreters produced in this manner are inherently "correct-by-construction."

Within the expansive landscape of the K Framework, several smart contract languages have found their home or are undergoing meticulous development. AI8 Network is poised to support three low-level languages, each serving a specific purpose: IELE VM, KEVM, and WASM.

IELE VM stands out as an intermediate-level language, akin to LLVM but tailored for the blockchain. Uniquely, it was constructed directly within the K Framework, setting it apart with no external specification or implementation [40]. IELE VM boasts human readability, speed, and addresses certain limitations of EVM. While developers have the option to program directly in IELE, the majority may opt for coding in Solidity and leveraging a Solidity to IELE compiler, as depicted in Fig. 8.

KEVM represents a rendition of the Ethereum Virtual Machine (EVM), skillfully scripted in K [41]. This iteration not only rectifies certain vulnerabilities present in EVM but also omits or refines susceptible features, enhancing overall security.

Web Assembly (WASM) introduces a binary instruction format for a stack-based virtual machine, facilitating the execution of smart contracts. This infrastructure empowers developers to write contracts in diverse languages such as C/C++, Rust, C#, and more.

While having a language specification and generating interpreters is a pivotal achievement, the integration with the AI8 network presents the other half of the challenge. AI8 has successfully constructed a universal VM interface, enabling the seamless incorporation of any VM into an AI8 node, as illustrated in Fig. 9. Each VM is equipped with an adapter implementing this interface, ensuring that every contract, saved as bytecode, aligns with the VM it was compiled for, harmonizing the smart contract ecosystem on AI8.

Support for formal modelling and verification

Because the smart contract languages are formally defined in K Framework, it is possible to perform formal verification of smart contracts written in these languages. To do this, it is necessary to also formally model their requirements, which can also be performed using the K Framework [42].

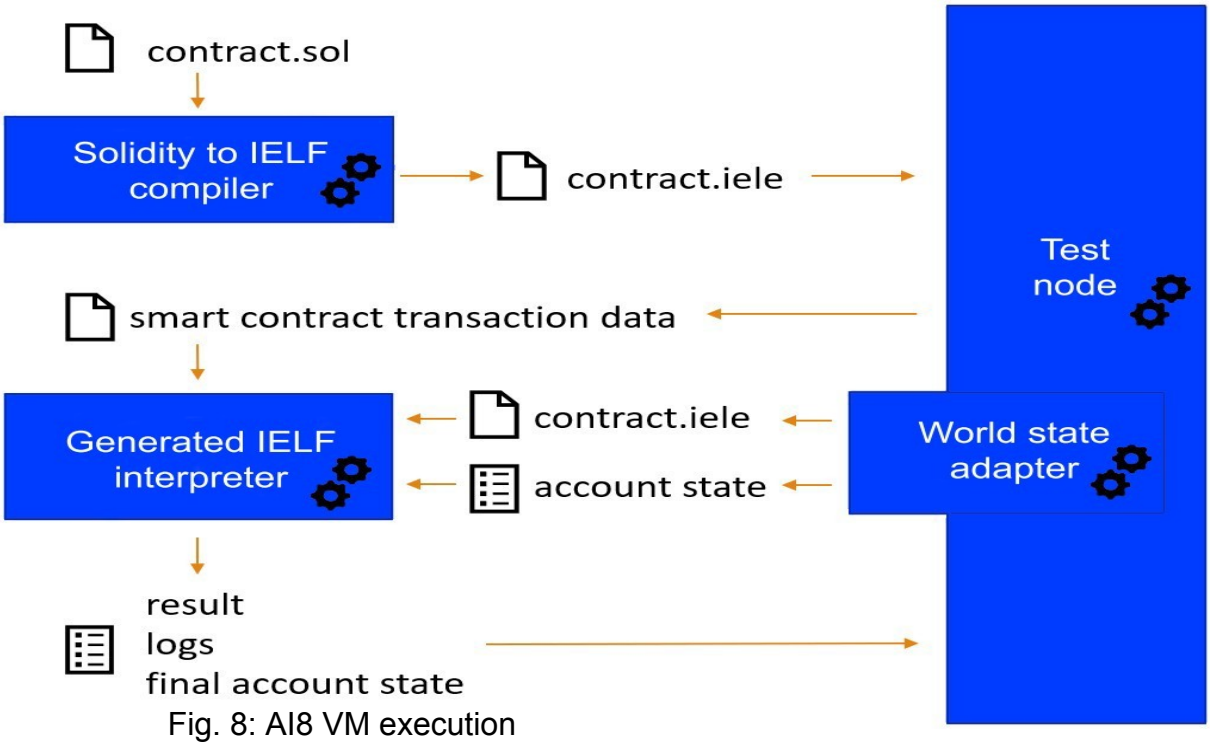


Fig. 8: AI8 VM execution

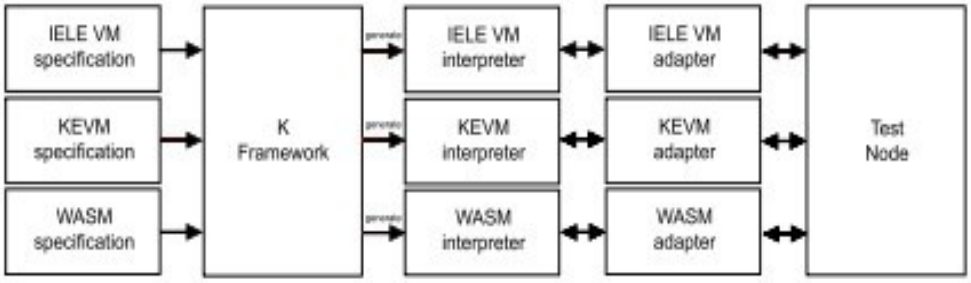


Fig. 9: AI8 VM components

Navigating the Challenges of Smart Contracts in Sharded Architectures

The realm of smart contracts within sharded architectures is a frontier still under the banner of research and development, laden with formidable challenges. Protocols like Atomix [7] and S-BAC [9] serve as foundational points, yet they merely mark the beginning. The intricate web of dynamic smart contract dependencies proves elusive, especially when seeking resolution by co-locating contracts within the same shard. Deployment time complexities emerge as not all dependencies can be computed.

Current research in this space explores potential solutions:

1. Locking Mechanism for Atomic Execution: This mechanism allows the atomic execution of smart contracts from different shards, ensuring that either all involved contracts execute simultaneously or none at all. However, it demands multiple interaction messages and synchronization among the consensuses of diverse shards [9].
2. Cross-Shard Contract Yanking Proposal: Envisioned for Ethereum 2.0, this proposal involves moving smart contract code and data into the caller shard during execution. While atomic execution isn't mandatory, a crucial locking mechanism is required for the relocated smart contract, potentially blocking other transactions. Although simpler, this approach mandates the transfer of the entire internal state of the smart contract [43].

In alignment with Ethereum's model, AI8 classifies transactions into the following types:

1. Smart Contract Construction and Deployment: Transactions with an empty receiver address and the smart contract code embedded as a byte array in the data field.
2. Smart Contract Method Invoking: Transactions where the receiver address is non-empty, and the associated address has code.

3. Payment Transactions: Transactions with a non-empty receiver address, but the address lacks code.

AI8 tackles this challenge by adopting an asynchronous cross-shard execution model for smart contracts. When a user initiates a smart contract execution transaction, and the smart contract resides in a different shard, the transaction is treated as a payment transaction. The transaction value is subtracted from the sender account, added to the block in the sender shard, and included in a miniblock destined for the shard where the receiver account resides. The metachain notarizes this transaction before it undergoes processing in the destination shard.

In the destination shard, the transaction is treated as a smart contract method invocation, creating a temporary account shadowing the sender account. This account holds the balance from the transaction value, and the smart contract is invoked. Post-execution, the smart contract may yield results affecting accounts in different shards. In-shard accounts are processed in the same round, while for out-of-shard accounts, Smart Contract Results miniblocks are created. These miniblocks, notarized by the metachain, are then processed in their respective shards. In cases where one smart contract dynamically calls another from a different shard, this call is logged as an intermediate result and handled similarly to accounts.

While this solution involves multiple steps and requires a minimum of 5 rounds for the finalization of a cross-shard smart contract call, it eliminates the need for locking and state movement across shards. This intricate dance of transactions and execution ensures the seamless interaction of smart contracts within AI8's sharded architecture.

Temporal Structure in Proof of Stake Systems: Epochs and Rounds

Proof of Stake (PoS) systems universally adopt a temporal framework, breaking down time into epochs, each further segmented into rounds [19]. While the specifics of the timeline and terminology may vary across architectures, the underlying approach remains largely consistent.

Epochs: In the AI8 Protocol, each epoch boasts a fixed duration, initially set at 24 hours (subject to updates following rigorous testnet confirmation stages). Throughout this time span, the configuration of shards remains static. To address scalability needs between epochs, the system adjusts the number of shards. Collusion prevention mandates a post-epoch reshuffling of each shard's configuration. While a complete reshuffling for maximal security is conceivable, it introduces latency due to bootstrapping, impacting system liveness. Consequently, at the epoch's end, less than 1% of eligible validators from a shard are randomly and uniformly redistributed to other shards' waiting lists. Validator distribution to shards is determined just before a new

epoch commences, requiring no additional communication, as depicted in Fig. 10. The node shuffling unfolds in several steps:

1. New nodes registered in the current epoch (ei) land in the unassigned node pool until the epoch's end.
2. Less than 1% of nodes in each shard are randomly selected for reshuffling and added to the assigned node pool.
3. The new number of shards (Nsh;i+1) is computed based on the network's node count (ki) and usage.
4. Nodes previously in all shard waiting lists, currently synchronized, are added to eligible validators' lists.
5. Nodes from the unassigned node pool are uniformly and randomly distributed across all shards' waiting lists during epoch (ei+1).
6. Reshuffled nodes from the assigned node pool are redistributed with higher ratios to shards' waiting lists anticipating a split in the next epoch (ei+2).

Rounds: Each round maintains a fixed time duration of 5 seconds (subject to updates after rigorous testnet confirmation stages). Within every round, a randomly selected set of block validators (including a block proposer) can produce a new block within each shard. The set changes from one round to another, utilizing the eligible nodes list as detailed in Chapter IV. This combination of shard reconfiguration within epochs and arbitrary validator selection across rounds serves to discourage unfair coalitions, mitigates the risk of Distributed Denial of Service (DDoS) and bribery attacks, all while upholding decentralization and ensuring a high transaction throughput.

Pruning for Efficient Blockchain Growth

In the realm of high-throughput blockchain networks, the challenge of managing a swiftly expanding ledger is paramount, entailing increased bootstrapping costs in terms of time and storage, as elucidated in section XI.1.

To counteract this cost surge, AI8 protocol employs a sophisticated pruning algorithm [7], outlined below. Assuming the current epoch is denoted as e and the current shard as a:

1. Merkle Tree Balances:
    - Shard nodes meticulously maintain account balances of epoch e in a Merkle tree structure [44].
2. State Block Creation:
    - As each epoch concludes, the block proposer crafts a state block, denoted as sb(a;e).

- ¬ This state block encapsulates the Merkle tree's root hash in the block's header and the balances in the block's body.

3. Consensus Verification:

- ¬ Validators scrutinize and engage in consensus regarding sb(a;e).

4. Genesis Block Storage:

- ¬ Upon consensus achievement, the block proposer archives sb(a;e) in the shard's ledger, designating it as the genesis block for the ensuing epoch e + 1.

5. Pruning at Epoch's End:

- ¬ At the termination of epoch e + 1, nodes discard the body of sb(a;e) along with all blocks antedating sb(a;e).

By implementing this dynamic pruning mechanism, the onboarding of new nodes becomes markedly efficient. New nodes initiate their journey from the latest valid state block, computing only the subsequent blocks instead of grappling with the entire historical ledger. This nuanced approach ensures that the network remains nimble, facilitating a streamlined onboarding process for participants without compromising the integrity of the blockchain's historical record.

Security Evaluation

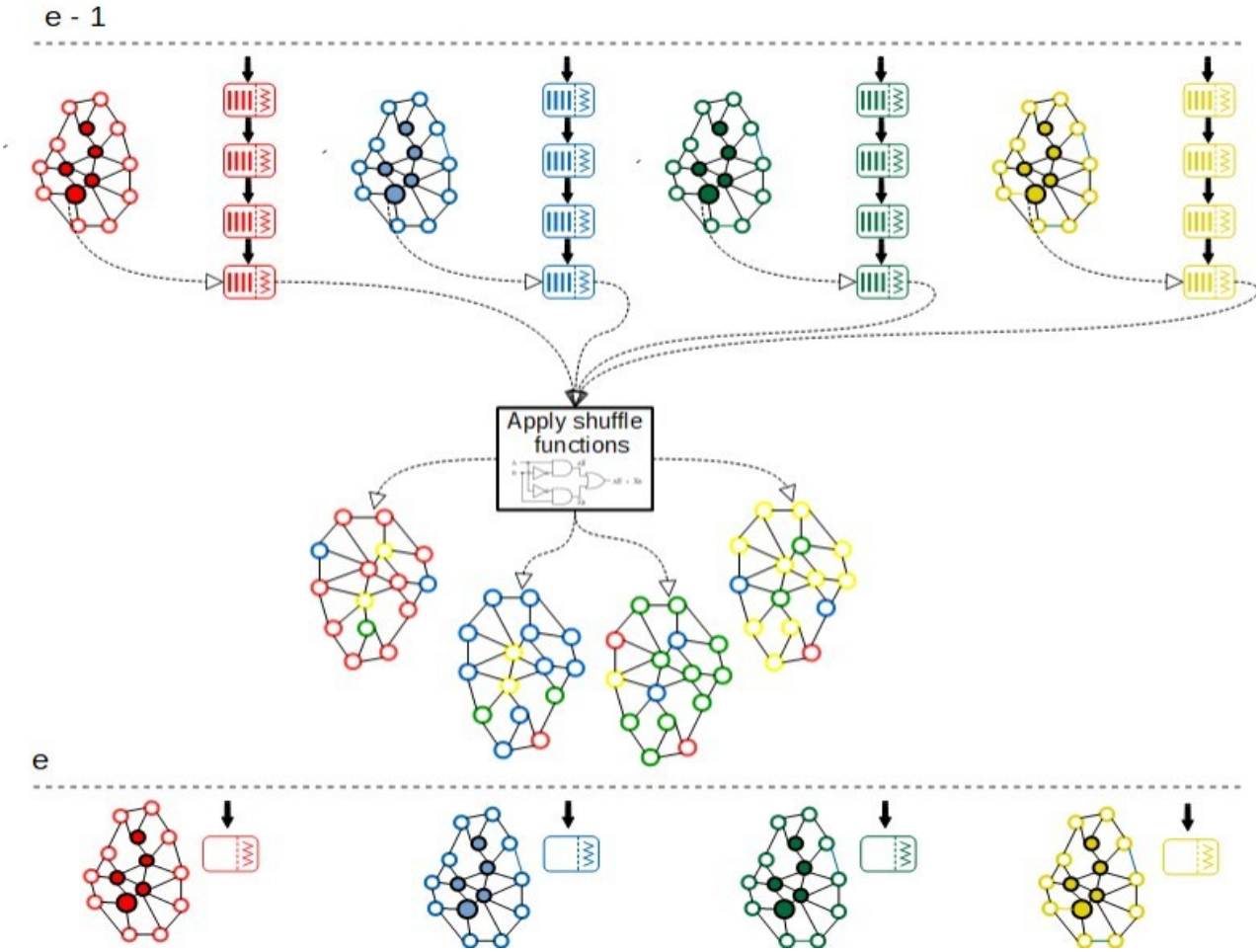
Randomness source

AI8 makes use of random numbers in its operation e.g. for the random sampling of block proposer and validators into consensus groups and the shuffling of nodes between shards at the end of an epoch. Because these features contribute to AI8##'s security guarantees, it is therefore important to make use of random numbers that are provably unbiasable and unpredictable. In addition to these properties, the generation of random numbers also needs to be efficient so that it can be used in a scalable and high throughput blockchain architecture. These properties can be found in some asymmetric cryptog-raphy schemes, like the BLS signing scheme. One important property of BLS is that using the same private key to sign the same message always produces the same results. This is similar to what is achieved using ECDSA with deterministic k generation and is due to the scheme not using any random parameters:

$$sig = sk\ H(m) \qquad (8)$$

where H is a hashing function that hashes to points on the used curve and sk is the private key.

Fig. 10: Shuffling the nodes at the end of each epoch



Fisherman Challenge Protocol: Safeguarding Against Malicious Blocks

In the event of a malevolent majority proposing an invalid block, an insidious act that tampers with the shard state root, the Fisherman Challenge Protocol becomes the bastion of defense. In this robust mechanism, an honest node can mount a challenge by furnishing a combined Merkle proof for a set of accounts, elucidating the fraudulent alterations to the state tree. The challenge entails a comprehensive package comprising the block of transactions, the antecedent reduced Merkle tree featuring all affected accounts before the contested block's application, and the smart contract states. This submission effectively exposes the invalid transaction or state.

To maintain the integrity of the system, a predefined timeframe is set for the submission of challenges with proof. Failure to meet this deadline results in the block being deemed

valid. The cost of initiating an invalid challenge carries a steep penalty—the complete stake of the challenging node.

The metachain assumes the role of sentinel, detecting inconsistencies triggered by either invalid transactions or spurious state roots through the challenges and proofs presented. The entire process is traceable, empowering the consensus group to slash the malevolent actors. Simultaneously, the challenger may be rewarded with a portion of the forfeited stake.

A lurking concern involves a malicious group attempting to conceal the invalid block from non-malicious nodes. However, this subterfuge is thwarted by mandating the incumbent consensus to disseminate the produced block to the sibling shard and observer nodes. The communication overhead is further minimized by transmitting only the intrashard miniblock to the sibling shard. Cross-shard miniblocks are unfailingly dispatched on distinct topics accessible to interested nodes.

Multiple honest nodes have the prerogative to raise challenges, enhancing security layers. The setup of peer-to-peer topics also fortifies the system. Communication from one shard to the metachain adheres strictly to predefined topics or channels, accessible to any honest validator. The metachain unequivocally rejects messages from alternative channels, ensuring a controlled communication environment. This nuanced solution introduces a minimal delay in the metachain, primarily during challenges—an infrequent occurrence with a negligible probability. The nodes, cognizant of the high risk associated with detection, face the jeopardy of forfeiting their entire stake, thereby reinforcing the resilience of the Fisherman Challenge Protocol.

Shard reorganization

After each epoch, less than 1 / 3 n of the nodes from each shard are redistributed uniformly and non-deterministicallyacross the other shards, to prevent collusion. This method adds bootstrapping overhead for the nodes that were redistributed, but doesn't affect liveness as shuffled nodes do not participate in the consensus in the epoch they have been redistributed. The pruning mechanism will decrease this time to a feasible amount, as explained in section IX.2.

6 Consensus group selection

After each round a new set of validators are selected using the random seed of the last commited block, current round and the eligible nodes list. In case of network desynchronization due to the delays in message propagation, the protocol has a recovery mechanism, and takes into consideration both the round r and the randomness seed from the last committed block in order to select new consensus groups every round. This avoids forking and allows synchronization on last block. The small time window (round time) in which the validators group is known, minimizes the attack vectors.

7 Node rating

Beside stake, the eligible validator's rating influences the chances to be selected as part of the consensus group. If the block proposer is honest and its block gets committed to the blockchain, it will have its rating increased, otherwise, it's rating will be decreased. This way, each possible validator is incentivized to be honest, run the most up-to-date client software version, increase its service availability and thus

ensuring the network functions as designed.

8 Shard redundancy

The nodes that were distributed in sibling shards on the tree's lowest level (see section IV.4) keep track of each other's blockchain data and application state. By introducing the concept of shard redundancy, when the number of nodes in the network decreases, some of the sibling shards will need to be merged. The targeted nodes will instantly initiate the process of shard merging.

Understanding the real problems 1 Centralized vs Decentralized

Blockchain was initially instantiated as an alternative to the centralized financial system of systems [45]. Even if the freedom and anonymity of distributed architectures remains an undisputed advantage, the performance has to be analyzed at a global scale in a real-world environment.

The most relevant metric measuring performance is transac-tions per second (TPS), as seen in Table 2. A TPS comparison of traditional centralized systems with decentralized novel architectures that were validated as trusted and efficient on a large scale, reflects an objective yet unsettling reality [46], [47], [48], [49].

The scalability of blockchain architectures is a critical but still unsolved problem. Take, for instance, the example determining the data storage and bootstrapping implications of current blockchain architectures suddenly functioning at Visa level throughput. By performing such exercises, the magnitude of multiple secondary problems becomes obvious (see Fig. 11).

Architecture	Type	Dispersion	TPS (average)	TPS (max limit)
VISA	Distributed virtualization	Centralized	3500	55000
Paypal	Distributed virtualization	Centralized	200	450
Ripple	Private Blockchain	Permissioned	1500	55000
NEO	Private Blockchain	Mixed	1000	10000
Ethereum	Public Blockchain	Decentralized	15	25
Bitcoin	Public Blockchain	Decentralized	2	7

The blockchain performance paradigm

The process of designing distributed architectures on blockchain faces several challenges, perhaps one of the most challenging being the struggle to maintain operability under contextual pressure conditions. The main components that determine the performance pressure are:

complexity system size
transaction volume

Complexity

The first element that limits the system performance, is the consensus protocol. A more complicated protocol determines a bigger hotspot. In PoW consensus architectures a big perfor-mance penalty is induced by the mining complexity that aims to keep the system decentralized and ASIC resilient [50]. To overrun this problem PoS makes a trade-off, simplifies the network management by concentrating the computing power to a subset of the network, but yields more complexity on the control mechanism.

System size

Expanding the number of nodes in existing validated archi-tectures forces a serious performance degradation and induces a higher computational price that must be paid. Sharding seems to be a good approach, but the shard size plays a major role. Smaller shards are agile but more likely to be affected by malicious groups, bigger shards are safer, but their reconfiguration affects the system liveness.

Transaction volume

With a higher relevance compared to the others, the last item on the list represents the transaction processing performance. In order to correctly measure the impact of this criteria, this must be analyzed considering the following two standpoints:

C1 transaction throughput - how many transactions a system can process per time unit, known as TPS, an output of a system [51];

C2 transaction finality - how fast one particular trans-action is processed, referring to the interval between its launch and its finalization - an input to output path.

C1: Transactionthroughput in single chain architectures is very low and can be increased by using workarounds such as sidechain [52]. In a sharded architecture like ours, the transaction throughput is influenced by the number of shards, the computing capabilities of the validators/block proposers and the messaging infrastructure [8]. In general, as displayed in Fig. 13, this goes well to the public, but despite the importance of the metric, it provides only a fragmented view. C2: Transaction finality - A more delicate aspect that emphasizes that even if the system may have a throughput of 1000 TPS, it may take a while to process a particular transac-tion. Beside the computing capabilities of the validators/block proposers and the messaging infrastructure, the transaction finality is mainly affected by the dispatching algorithm (when the decision is made) and the routing protocol (where should the transaction be executed). Most of the existing state of the art architectures refuse to mention this aspect but from a user standpoint this is extremely important. This is displayed inFig. 14, where the total time required to execute a certain transaction from start to end is considered.

In AI8##, the dispatching mechanism (detailed in section V) allows an improved time to finality by routing the transactions directly to the right shard, mitigating the overall delays.

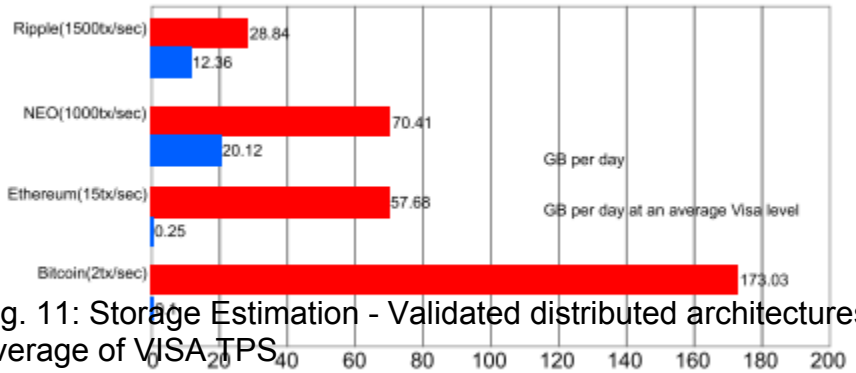
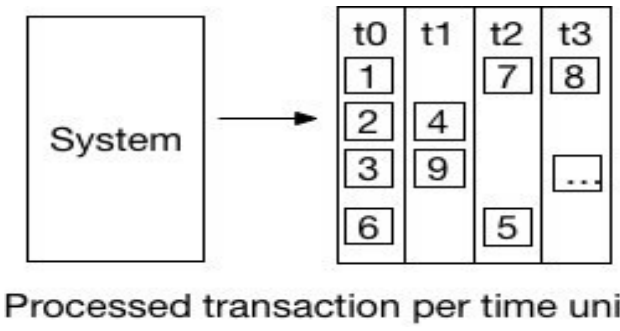
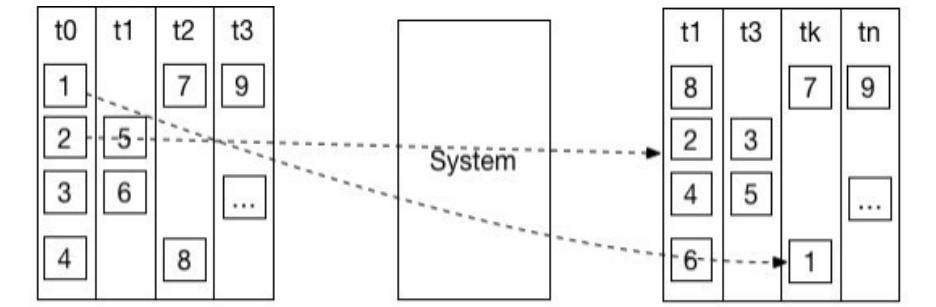


Fig. 11: Storage Estimation - Validated distributed architectures working at an average of VISA TPS



Processed transaction per time unit



Starting time of transaction

Ending time of transaction

Conclusion: Unleashing Scalability and Performance Prowess

In the crucible of performance evaluations and simulations, vividly portrayed in Figure 12, the solution emerges as a paragon of efficiency—a distributed ledger endowed with unparalleled scalability. The trajectory of its throughput graph delineates a linear ascent, an ode to its sharding approach, as an increasing cohort of nodes seamlessly integrates into the network fabric.

The chosen consensus model, marked by multiple communication rounds, introduces a caveat—its efficacy is symbiotically entwined with the quality of the network. Parameters such as speed, latency, and availability wield considerable influence over the outcome. In simulations conducted on our expansive testnet, leveraging global network speed averages, AI8## unfurls its wings, surpassing the VISA benchmark with a mere two shards. With each incremental addition to the shard count, the system inexorably marches toward the zenith of VISA-level performance.

The nexus of theoretical limits and real-world performance underscores AI8##'s mettle, projecting a trajectory that promises to rival and potentially surpass the zenith of VISA-level throughput. As the network continues to burgeon with nodes, AI8## stands as a testament to the harmonious fusion of cutting-edge technology and a vision for unparalleled scalability.

Ongoing Exploration and Future Frontiers in Research

The crucible of innovation never cools for AI8's dedicated team, persistently fine-tuning and elevating the architecture. The blueprint, a synthesis of adaptive state sharding, secure Proof of Stake consensus, and heightened energy efficiency, stands poised for continuous refinement. The trajectory of improvement unfurls across the following domains:

1. Reinforcement Learning Prowess: Charting a course toward enhanced sharding efficiency, AI8 sets its sights on leveraging reinforcement learning. The objective: orchestrating the placement of frequently trading clients within the same shard, strategically curbing the overall operational cost.
2. Sentinel AI Vigilance: An ambitious venture into the realm of artificial intelligence supervision emerges. The goal—development of an AI supervisor attuned to the detection of malicious behavioral patterns. Yet, the challenge lies in seamlessly integrating this surveillance without disrupting the delicate balance of decentralization.
3. Reliability Integration: An evolution in the consensus calculus beckons as AI8## contemplates the addition of reliability as a pivotal metric. This metric, derived in a distributed manner, unfolds post-consensus protocol application on recent block submissions, injecting an additional layer of resilience.
4. Interwoven Blockchain Tapestry: Pioneering strides in cross-chain interoperability become a focal point. AI8 commits to implementing and contributing to emerging standards championed by the Decentralized Identity Foundation and the Blockchain Interoperability Alliance.
5. Cloak of Privacy: AI8's vision extends to the realm of privacy-preserving transactions. Envisaging the utilization of Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, the goal is clear: shield participant identities, furnish audit capabilities, and preserve the sanctity of privacy.

In this dynamic landscape of exploration, AI8 propels itself forward, each iteration an augmentation of its commitment to pioneering blockchain excellence. The roadmap is rich with promise, underscoring a dedication to innovation that is unwavering.

Comprehensive Insights and Pathbreaking Innovations

AI8 stands as the pioneering vanguard, charting unexplored territories in the realm of public blockchains. The adoption of the groundbreaking Secure Proof of Stake algorithm, seamlessly intertwined with a genuine state-sharded architecture, catapults AI8 into the echelons of unprecedented scalability, boasting VISA-level throughput and confirmation times mere seconds apart.

AI8 ingenuity surpasses even the heralded Omniledger, fortifying security and throughput through adaptive state sharding. The intrinsic mechanisms of automatic transaction routing and state redundancy don the mantle of latency reduction, a pivotal stride in the evolution of blockchain frameworks. The introduction of shard pruning not only slashes bootstrapping and storage costs but sets AI8 apart in its efficiency compared to contemporaneous approaches.

At the heart of AI8 resilience lies the Secure Proof of Stake consensus algorithm, a beacon of distributed fairness. In a realm where every second counts, AI8 redefines the paradigm of random selection, compressing the time needed for consensus group curation from 12 seconds to a mere 100 milliseconds, an unprecedented leap that leaves predecessors, like Algorand, in the rearview.

In a symphony of innovation, the fusion of state sharding and the astoundingly efficient Secure Proof of Stake consensus algorithm unfolds as a beacon of promise. Preliminary estimations blossom into tangible results, validated through rigorous testnet trials. AI8## narrative unfolds not just as a blockchain but as a transformative force, etching new chapters in the evolving saga of decentralized technologies

References

G. Hileman and M. Rauchs, “2017 Global Cryptocurrency Benchmarking Study,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2965436, Apr. 2017. [Online]. Available: https://papers.ssrn.com/abstract=2965436

[2] “The Ethereum Wiki - Sharding FAQ,” 2018, original-date: 2014-02-14T23:05:17Z. [Online]. Available: https://github.com/ethereum/wiki/ wiki/Sharding-FAQ

[3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling Byzantine Agreements for Cryptocurrencies,” in Proceedings of the 26th Symposium on Operating Systems Principles, ser. SOSP ’17. New York, NY, USA: ACM, 2017, pp. 51–68. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132757

[4] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in Advances in Cryptology – ASIACRYPT ’01, LNCS. Springer, 2001, pp. 514–532.

[5] D. Boneh, M. Drijvers, and G. Neven, “Compact multi-signatures for smaller blockchains,” in Advances in Cryptology – ASIACRYPT 2018, ser. Lecture Notes in Computer Science, vol. 11273. Springer, 2018, pp. 435–464.

[6] V. Buterin, “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,” 2013. [Online]. Available: https: //www.ethereum.org/pdfs/EthereumWhitePaper.pdf

[7] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding,” Tech. Rep. 406, 2017. [Online]. Available: https://eprint.iacr.org/2017/406

[8] “The ZILLIQA Technical Whitepaper,” 2017. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf

[9] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, “Chainspace: A Sharded Smart Contracts Platform,” arXiv:1708.03778 [cs], Aug. 2017, arXiv: 1708.03778. [Online]. Available: http: //arxiv.org/abs/1708.03778

[10] G. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger,” 2017. [Online]. Available: https://ethereum.

github.io/yellowpaper/paper.pdf
[11] “Solidity — Solidity 0.4.21 documentation.” [Online]. Available: https://solidity.readthedocs.io/en/v0.4.21/
[12] “web3j,” 2018. [Online]. Available: https://github.com/web3j [13] “Casper,” 2018. [Online]. Available: http://ethresear.ch/c/casper
[14] “The State of Ethereum Scaling, March 2018 – Highlights from EthCC on Plasma Cash, Minimum Viable Plasma, and More... – Medium,” 2018. [Online]. Available: https://medium.com/loom-network/ the-state-of-ethereum-scaling-march-2018-74ac08198a36
[15] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in Proceedings of the Third Symposium on Operating Systems Design and Implementation, ser. OSDI ’99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=296806.296824
[16] Y. Jia, “Op Ed: The Many Faces of Sharding for Blockchain Scalability,” 2018. [Online]. Available: https://bitcoinmagazine.com/ articles/op-ed-many-faces-sharding-blockchain-scalability/
[17] “Using Merklix tree to shard block validation j Deadalnix’s den,” 2016. [Online]. Available: https://www.deadalnix.me/2016/11/06/ using-merklix-tree-to-shard-block-validation/
[18] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” p. 9, 2008.
[19] “Why we are building Cardano - Introduction.” [Online]. Available: https://whycardano.com/
[20] “Constellation - a blockchain microservice operating system - White Paper,” 2017, original-date: 2018-01-05T20:42:05Z. [Online]. Available: https://github.com/Constellation-Labs/Whitepaper
[21] “Bitshares - Delegated Proof-of-Stake Consensus,” 2014. [Online]. Available: https://bitshares.org/technology/ delegated-proof-of-stake-consensus/
[22] dantheman, “DPOS Consensus Algorithm - The Missing White Paper,” May 2017. [Online]. Available: https://steemit.com/dpos/@dantheman/ dpos-consensus-algorithm-this-missing-white-paper
[23] “EOS.IO Technical White Paper v2,” 2018, original-date: 2017-06-06T07:55:17Z. [Online]. Available: https://github.com/EOSIO/ Documentation/blob/master/TechnicalWhitePaper.md
[24] C. Paar and J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners. Berlin Heidelberg: Springer-Verlag, 2010. [Online]. Available: //www.springer.com/gp/book/9783642041006
[25] C. Schnorr, “Efficient signature generation by smart cards,” Journal of Cryptology, vol. 4, pp. 161–174, Jan. 1991.
[26] K. Michaelis, C. Meyer, and J. Schwenk, “Randomly Failed! The State of Randomness in Current Java Implementations,” in Topics in Cryptology – CT-RSA 2013, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Feb. 2013, pp. 129–144. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-36095-4_9
[27] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, “Twisted Edwards Curves,” in Progress in Cryptology – AFRICACRYPT 2008, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jun. 2008, pp. 389–405. [Online]. Available: https: //link.springer.com/chapter/10.1007/978-3-540-68164-9_26
[28] A. Poelstra, “Schnorr Signatures are Non-Malleable in the Random Oracle Model,” 2014. [Online]. Available: https://download.wpsoftware. net/bitcoin/wizardry/schnorr-mall.pdf
[29] C. Decker and R. Wattenhofer, “Bitcoin Transaction Malleability and MtGox,” arXiv:1403.6676 [cs], vol. 8713, pp. 313–326, 2014, arXiv: 1403.6676. [Online]. Available: http://arxiv.org/abs/1403.6676

[30] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, “Simple Schnorr Multi-Signatures with Applications to Bitcoin,” Tech. Rep. 068, 2018. [Online]. Available: https://eprint.iacr.org/2018/068
[31] Y. Seurin, “On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model,” in Advances in Cryptology – EUROCRYPT 2012, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2012, pp. 554–571. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-29011-4_33
[32] K. Itakura and K. Nakamura, “A public-key cryptosystem suitable for digital multisignatures,” 1983.
[33] S. Micali, K. Ohta, and L. Reyzin, “Accountable-subgroup Multisignatures: Extended Abstract,” in Proceedings of the 8th ACM Conference on Computer and Communications Security, ser. CCS ’01. New York, NY, USA: ACM, 2001, pp. 245–254. [Online]. Available: http://doi.acm.org/10.1145/501983.502017
[34] T. Ristenpart and S. Yilek, “The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks,” in Advances in Cryptology -EUROCRYPT 2007, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 2007, pp. 228–245. [Online]. Available: https://link.springer.com/chapter/10.1 007/ 978-3-540-72540-4_13

[35] M. Bellare and G. Neven, “Multi-signatures in the Plain public-Key Model and a General Forking Lemma,” in Proceedings of the 13th ACM Conference on Computer and Communications Security, ser. CCS ’06. New York, NY, USA: ACM, 2006, pp. 390–399. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180453
[36] D.-P. Le, A. Bonnecaze, and A. Gabillon, “Multisignatures as Secure as the Diffie-Hellman Problem in the Plain Public-Key Model,” in Pairing-Based Cryptography – Pairing 2009, ser. Lecture Notes in Computer Science.Springer, Berlin, Heidelberg, Aug. 2009, pp. 35–51. [Online]. Available: https://link.springer.com/chapter/10.1007/ 978-3-642-03298-1 3
[37] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, “Adding Concurrency to Smart Contracts,” in Proceedings of the ACM Symposium on Principles of Distributed Computing, ser. PODC ’17. New York, NY, USA: ACM, 2017, pp. 303–312. [Online]. Available: http://doi.acm.org/10.1145/3087801.3087835
[38] J. Kwon and E. Buchman, “Cosmos Network - Internet of Blockchains,” 2017. [Online]. Available: https://cosmos.network/whitepaper
[39] G. Rosu and T. F. Serbanuta, “An overview of the k semantic frame-work,” The Journal of Logic and Algebraic Programming, vol. 79, no. 6, pp. 397–434, 2010.
[40] T. Kasampalis, D. Guth, B. Moore, T. Serbanuta, V. Serbanuta, D. Fi-laretti, G. Rosu, and R. Johnson, “Iele: An intermediate-level blockchain language designed and implemented using formal semantics,” Tech. Rep., 2018.
[41] E. Hildenbrandt, M. Saxena, X. Zhu, N. Rodrigues, P. Daian, D. Guth, and G. Rosu, “Kevm: A complete semantics of the ethereum virtual machine,” Tech. Rep., 2017. [42] “How Formal Verification of SmartContracts Worksj RV Blog.” [Online]. Available: https://runtimeverification.com/blog/ how-formal-verification-of-smart-contracts-works/
[43] “Cross-shard contract yanking.” [Online]. Available: https://ethresear. ch/t/cross-shard-contract-yanking/1450
[44] R. C. Merkle, “A Certified Digital Signature,” in Advances in Cryptology — CRYPTO’ 89 Proceedings, ser. Lecture Notes in Computer Science. Springer, New York, NY, Aug. 1989, pp. 218–238. [Online]. Available: https://link.springer.com/chapter/10.1007/0-387-34805-0

[45] A. Veysov and M. Stolbov, “Financial System Classification: From Conventional Dichotomy to a More Modern View,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2114842, Jul. 2012. [Online]. Available: https://papers.ssrn.com/abstract=2114842
[46] “XRP ‐ The Digital Asset for Payments.” [Online]. Available: https://ripple.com/xrp/
[47] “Visa - Annual Report 2017,” 2018. [Online]. Avail-able: https://s1.q4cdn.com/050606653/files/doc financials/annual/2017/ Visa-2017-Annual-Report.pdf
[48] “PayPal Reports Fourth Quarter and Full Year 2017 Results (NASDAQ:PYPL),” 2018. [Online]. Available: https://investor. paypal-corp.com/releasedetail.cfm?releaseid=1055924
[49] M. Schwarz, “Crypto Transaction Speeds 2018 - All the Major Cryptocurrencies,” 2018. [Online]. Available: https://www.abitgreedy. com/transaction-speed/
[50] “The Ethereum Wiki - Mining,” 2018, original-date: 2014-02-14T23:05:17Z. [Online]. Available: https://github.com/ethereum/wiki/ wiki/Mininghttps://github.com/ethereum/wiki
[51] “Transaction throughput.” [Online]. Available: https://docs.oracle.com/ cd/E17276 01/html/programmer reference/transapp throughput.html
[52] W. Martino, M. Quaintance, and S. Popejoy, “Chainweb: A Proof-of-Work Parallel-Chain Architecture for Massive Throughput,” 2018. [Online]. Available: http://kadena.io/docs/chainweb-v15.pd
[53] “DIF ‐ Decentralized Identity Foundation.” [Online]. Available: http://identity.foundation/
[54] H.I. World,“Blockchain Interoperability Alliance: ICON x Aion x Wanchain,” Dec. 2017. [Online]. Available: https://medium.com/helloiconworld/ blockchain-interoperability-alliance-icon-x-aion-x-wanchain-8aeaafb3ebdd
[55] S. Goldwasser, S. Micali, and C. Rackoff, “The Knowledge Complexity of Interactive Proof-systems,” in Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, ser. STOC ’85. New York, NY, USA: ACM, 1985, pp. 291–304. [Online]. Available: http://doi.acm.org/10.1145/22145.22178