ZENONE

A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake

[Technical whitepaper — release 1 — revision 8]
November 21, 2018 - The Zenone Team
https://www.zenone.io/

Abstract—**The advent of secure public blockchains through Bitcoin and later Ethereum, has brought forth a notable degree of interest and capital influx, providing the premise for a global wave of permissionless innovation. Despite lofty promises, creating a decentralized, secure and scalable public blockchain has proved to be a strenuous task.**

This paper proposes Zenone, a novel architecture which goes beyond state of the art by introducing a genuine state sharding scheme for practical scalability, eliminating energy and com-putational waste while ensuring distributed fairness through a Secure Proof of Stake (SPoS) consensus. Having a strong focus on security, Zenone network is built to ensure resistance to known security problems like Sybil attack, Nothing at Stake attack and others. In an ecosystem that strives for interconnectivity, our solution for smart contracts offers an EVM compliant engine to ensure interoperability by design.

Preliminary imulations eflect hat Zenone xceeds isa's average throughput and achieves an improvement beyond three orders of magnitude or 1000x compared to the existing viable approaches, while drastically reducing the costs of bootstrapping and storage to ensure long term sustainability.

I Introduction

1 General aspects

Cryptocurrency and smart contract platforms such as Bitcoin and Ethereum have sparked considerable interest and have become promising solutions for electronic payments, decentralized applications and potential digital stores of value. However, when compared to their centralized counterparts in key metrics[1], the current state of affairs suggests that present public blockchain iterations exhibit severe limitations, particularly with respect to scalability, hindering their mainstream adoption and delaying public use. In fact, it has proved extremely challenging to deal with the current engineering boundaries imposed by the trade-offs in the blockchain trilemma paradigm[2]. Several solutions have been proposed, but few of them have shown significant and viable results. Thus, in order to solve the scalability problem, a complete rethinking of public blockchain infrastructures was required.

2 Defining the challenges

Several challenges must be addressed properly in the process of creating an innovative public blockchain solution designed to scale:

- **Full decentralization** - Eliminating the need for any trusted third party, hence removing any single point of failure;
- **Robust security** - Allowing secure transactions and preventing any attacks based on known attack vectors;
- **High scalability** - Enabling the network to achieve a performance at least equal to the centralized counterpart, as measured in TPS;
- **Efficiency** - Performing all network services with minimal energy and computational requirements;
- **Bootstrapping and storage enhancement** - Ensuring a competitive cost for data storage and synchronization;
- **Cross-chain interoperability** - Enforced by design, permitting unlimited communication between external services.

Starting from the above challenges, we've created Zenone as a complete rethinking of public blockchain infrastructure, specifically esigned o e ecure, fficient, calable nd inter-operable. Zenone main ontribution ests n wo cornerstone uilding locks:

1) **A genuine State Sharding approach:** effectively partitioning the blockchain and account state into multiple shards, handled in parallel by different participating validators;
2) **Secure Proof of Stake consensus mechanism:** an improved variation of Proof of Stake (PoS) that ensures long term security and distributed fairness, while eliminating the need for energy intensive PoW algorithms.

3 Adaptive State Sharding

Zenone proposes a dynamically adaptive sharding mechanism that enables shard computation and reorganizing based on necessity and the number of active network nodes. The reassignment of nodes in the shards at the beginning of each epoch is progressive and nondeterministic, inducing no temporary liveness penalties. Adaptive state sharding comes with additional challenges compared to the static model. One of the key-points resides in how shard-splitting and shard-merging is done to prevent overall latency penalties introduced by the synchronization/communication needs when the shard number changes. Latency, in this case, is the communication overhead required by nodes, in order to retrieve the new state, once their shard address space assignment has been modified.

Zenone proposes a solution for this problem below, but first some notions have to be defined: users and nodes. Users are external actors and can be identified by an unique account address; nodes are computers/devices in the Zenone network that run our protocol. Notions like users, nodes, addresses will be further described in chapter II.1 - Entities.

Zenone solves this challenge by:

1) Dividing the account address space in shards, using a binary tree which can be built with the sole requirement of knowing the exact number of shards in a certain epoch. Using this method, the accumulated latency is reduced and the network liveness is improved in two ways. First, thanks to the designed model, the dividing of the account address space is predetermined by hierarchy. Hence, there is no split overhead, meaning that one shard breaks into two shards, each of them keeping only one half of the previous address space in addition to the associated state. Second, the latency is reduced through the state redundancy mechanism, as the merge is prepared by retaining the state in the sibling nodes.
2) Introducing a technique of balancing the nodes in each shard, to achieve overall architecture equilibrium. This technique ensures a balanced workload and reward for each node in the network.
3) Designing a built-in mechanism for automatic transaction routing in the corresponding shards, considerably reduces latency as a result. The routing algorithm is described in chapter IV.4 - Zenone sharding approach.
4) In order to achieve considerable improvements with respect to bootstrapping and storage, Zenone makes use of a shard pruning mechanism. This ensures sustainability of our architecture even with a throughput of tens of thousands of transactions per second (TPS).

4 Secure Proof of Stake (SPoS)

We introduce a Secure Proof of Stake consensus mechanism, that expands on Algorand's[3] idea of a random selection mechanism, differentiating itself through the following aspects:

1) Zenone introduces an improvement which reduces the latency allowing each node in the shard to determine the members of the consensus group (block proposer and validators) at the beginning of a round. This is possible because the last block's aggregated signature is used as the randomization factor r. The block proposer is the validator in the consensus group who's hash of the public key and randomization factor is the smallest. In contrast to Algorand's[3] approach, where the random committee selection can take up to 12 seconds, in Zenone the time necessary for random selection of the consensus group is considerably reduced (estimated under 100 ms) excluding network latency. Indeed, there is no communication requirement for this random selection process, which enables Zenone to have a newly and randomly selected group that succeeds in committing a new block to the ledger in each round. The tradeoff for this enhancement relies on the premise that an adversary cannot adapt faster than the round's time frame. A further improvement on the security of the randomness source, would be the use of verifiable delay functions (VDFs) in order to prevent any tampering possibilities of the randomness source until it is too late. Currently, the research in VDFs is still ongoing - there are no working and tested implementation of VDFs.
2) In addition to the stake factor generally used in PoS architectures as a sole decision input, Zenone refines its consensus mechanism by adding an additional weight factor called rating. The node's probability to be selected in the consensus group takes into consideration both stake and rating. The rating of a block proposer is recalculated at the end of each epoch, except in cases where slashing should occur, when the actual rating decrease is done instantly, adding another layer of security by promoting meritocracy.
3) Zenone uses Bellare and Neven (BN) multisignature scheme[4], which eliminates one communication round in the signing algorithm, because no proof of possession is needed, but maintains the same security level.
4) Zenone considers formal verification for the critical protocol implementations (e.g. SPoS consensus mechanism) in order to validate the correctness of our algorithms.

II Architecture Overview

1 Entities

There are two main entities in Zenone: users and nodes. Users, each holding a (finite) number of public/private (Pk/sk) key pairs (e.g. in one or multiple wallet apps), use Zenone's network to deploy signed transactions for value transfers or smart contracts' execution. They can be identified by one of their account addresses (derived from the public key). The nodes are represented by the devices that form the Zenone network and can be passive or actively engaged in processing tasks. Eligible validators are active participants in Zenone's network. Specifically, they are responsible for running consen- sus, adding blocks, maintaining the state and being rewarded for their contribution. Each eligible validator can be uniquely identified by a public key constructed through a derivation of the address that staked the necessary amount and the node id. Relations between entities in the Zenone protocol are shown in Fig. 1.

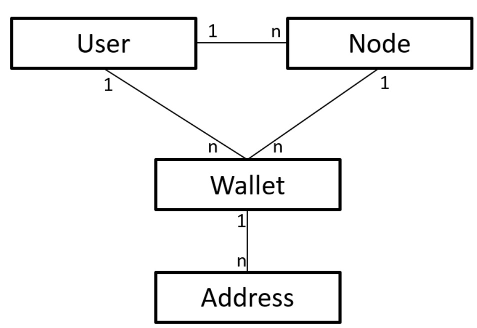

Fig. 1: Relations between Zenone entities

Furthermore, the network is divided into smaller units called shards. An eligible validator is assigned to a shard based on an algorithm that keeps the nodes evenly distributed across shards, depending on the tree level. Each shard contains a randomly selected consensus group. Any block proposer is responsible to aggregate transactions into a new block. The validators are responsible to either reject, or approve the proposed block, thereby validating it and committing it to the blockchain.

2 Intrinsic token

Zenone grants access to the usage of its network through intrinsic utility tokens called Zenones, in short ERDs. All costs for processing transactions, running smart contracts and rewards for various contributions to the network will be paid in ERDs. References to fees, payments or balances are assumed to be in ERDs.

3 Threat model

Zenone assumes a byzantine adversarial model, where at least $\frac{2}{3}$+1 of the eligible nodes are honest. The protocol permits the existence of adversaries that have stake or good rating, delay or send conflicting messages, compromise other nodes, have bugs or collude among themselves, but as long as $\frac{2}{3}$+1 of the eligible validators in a shard are honest/not compromised, the protocol can achieve consensus.

The protocol assumes highly adaptive adversaries, which however cannot adapt faster than a round's timeframe. The computational power of an adversary is bounded, therefore the cryptographic assumptions granted by the security level of the chosen primitives hold firmly within the complexity class of problems solvable by a Turing machine in polynomial time.

The network of honest nodes is assumed to form a well connected graph and the propagation of their messages is done in a bounded time Δ.

Attack vectors' prevention

1) **Sybil attacks:** mitigated through the stake locking when joining the network. This way the generation of new identities has a cost equal to the minimum stake;
2) **Nothing at stake:** removed through the need of multiple signatures, not just from proposer, and the stake slashing. The reward per block compared to the stake locked will discourage such behavior;
3) **Long range attacks:** mitigated by our pruning mechanism, the use of a randomly selected consensus group every round (and not just a single proposer) and stake locking. On top of all these, our pBFT consensus algorithm ensures finality;
4) **DDoS attacks:** the consensus group is randomly sampled every round (few seconds); the small time frame making DDoS almost impossible.

Other attack vectors we have taken into consideration are: Single shard takeover attack, transaction censorship, double spend, bribery attacks, etc.

4 Chronology

In Zenone's network, the timeline is split into epochs and rounds. The epochs have a fixed duration, set to one day (can be modified as the architecture evolves), at the end of which the shards reorganization and pruning is triggered. The epochs are further divided into rounds, lasting for a fixed timeframe. A new consensus group is randomly selected per shard in each round, that can commit a maximum of one block in the shard's ledger.

New validators can join the network by locking their stake, as presented in chapter V.2 - Secure Proof of Stake. They are added to the unassigned node pool in the current epoch e, are assigned to the waiting list of a shard at the beginning of epoch e + 1, but can only become eligible validators to participate in consensus and get rewarded in the next epoch (e+2).

The timeline aspects are further detailed in chapter VII.

III Related Work

Zenone was designed upon and inspired by the ideas from Ethereum[5], Omniledger[6], Zilliqa[7], Algorand[3] and ChainSpace[8]. Our architecture goes beyond state of the art and can be seen as an augmentation of the existing models, improving the performance while focusing to achieve a better nash equilibrium state between security, scalability and decentralization.

1 Ethereum

Much of Ethereum's[5] success can be attributed to the introduction of its decentralized applications layer through EVM[9], Solidity[10] and Web3j[11]. While Dapps have been one of the core features of ethereum, scalability has proved a pressing limitation. Considerable research has been put into solving this problem, however results have been negligible up to this point. Still, few promising improvements are being proposed: $Casper$[12] prepares an update that will replace the current Proof of Work (PoW) consensus with a Proof of Stake (PoS), while $Plasma$ based side-chains and sharding are expected to become available in the near future, alleviating Ethereum's scalability problem at least partially[13].

Compared to Ethereum, Zenone eliminates both energy and computational waste from PoW algorithms by implementing a SPoS consensus while using transaction processing parallelism through sharding.

2 Omniledger

Omniledger[6] Aroposes A Aovel Acale-out Aistributed ledger Ahat Areserves Aong Aerm Aecurity Ander permission-less Ap-eration. At Ansures Aecurity And correctness Ay Asing A Aias-resistant Aublic-randomness protocol Aor Ahoosing Aarge, Atatis-tically Aepresentative shards Ahat Arocess Aransactions. Ao Aom-mit Aransactions atomically Across Ahards, Amniledger Antro-duces Atomix, an Afficient Aross-shard Aommit Arotocol. Ahe Aoncept As a Awo-phase Alient-driven Alock/unlock" Arotocol Ahat the state affected by partially completed transactions. Omniledger also optimizes performance via parallel intra-shard transaction processing, ledger pruning via collectively-signed state blocks, and low-latency ”trust-but-verify” validation for low-value transactions. The consensus used in Omniledger is a BFT variation, named ByzCoinX, that increases performance and robustness against DoS attacks.

Compared to Omniledger, Zenone has an adaptive approach on state sharding, a faster random selection of the consensus group and an improved security by replacing the validators’ set after every round (a few seconds) not after every epoch (1 day).

3 Zilliqa

Zilliqa[7] is the first transaction-sharding architecture that allows the mining network to process transactions in parallel and reach a high throughput by dividing the mining network into shards. Specifically, its design allows a higher transaction rate as more nodes are joining the network. The key is to ensure that shards process different transactions, with no overlaps and therefore no double-spending. Zilliqa uses pBFT[14] for consensus and PoW to establish identities and prevent Sybil attacks.

Compared to Zilliqa, Zenone pushes the limits of sharding by using not only transaction sharding but also state sharding. Zenone completely eliminates the PoW mechanism and uses SPoS for consensus. Both architectures are building their own smart contract engine, but Zenone aims not only for EVM com-pliance, so that SC written for Ethereum will run seamlessly on our VM, but also aims to achieve interoperability between blockchains.

4 Algorand

Algorand[3] proposes a public ledger that keeps the convenience and efficiency of centralized systems, without the inefficiencies and weaknesses of current decentralized implementations. The leader and the set of verifiers are randomly chosen, based on their signature applied to the last block’s quantity value. The selections are immune to manipulations and unpredictable until the last moment. The consensus relies on a novel message-passing Byzantine Agreement that enables the community and the protocol to evolve without hard forks.

Compared to Algorand, Zenone doesn’t have a single blockchain, instead it increases transaction’s throughput using sharding. Zenone also improves on Algorand’s idea of random selection by reducing the selection time of the consensus group from over 12 seconds to less than a second, but assumes that the adversaries cannot adapt within a round.

5 Chainspace

Chainspace[8] is a distributed ledger platform for high integrity and transparent processing of transactions. It uses language agnostic and privacy-friendly smart contracts for extensibility. The sharded architecture allows a linearly scalable transaction processing throughput using S-BAC, a novel distributed atomic commit protocol that guarantees consistency and offers high auditability. Privacy features are implemented through modern zero knowledge techniques, while the consensus is ensured by BFT.

Compared to Chainspace, where the TPS decreases with each node added in a shard, Zenone’s approach is not influenced by the number of nodes in a shard, because the consensus group has a fixed size. A strong point for Chainspace is the approach for language agnostic smart contracts, while Zenone focuses on building an abstraction layer for EVM compliance. Both projects use different approaches for state sharding to enhance performance. However, Zenone goes a step further by anticipating the blockchain size problem in high throughput architectures and uses an efficient pruning mechanism. Moreover, Zenone exhibits a higher resistance to sudden changes in node population and malicious shard takeover by introducing shard redundancy, a new feature for sharded blockchains.

IV Scalability via Adaptive State Sharding

1 Why sharding

Sharding was Arst Ased An Aatabases And As A method for Ais-tributing Aata Across multiple machines. Ahis Acaling technique Aan Ae Ased An Alockchains Ao Aartition Atates and Aransaction Arocessing, Ao Ahat Aach Aode would process Anly A Araction Af All Aransactions An Aarallel with other Aodes. As Aong As Ahere As A Aufficient Aumber Af nodes verifying Aach Aransaction Ao Ahat Ahe Aystem maintains Aigh Aeliability And Aecurity, Ahen Aplitting A blockchain Anto Ahards will Allow At Ao Arocess many transactions An Aarallel, And Ahus Areatly Amproving transaction Ahroughput And Afficiency. Aharding Aromises to Ancrease Ahe Ahroughput As Ahe validator Aetwork expands, A Aroperty Ahat As Aeferred Ao As Aorizontal scaling.

2 Sharding types

A Aomprehensive And Ahorough Antroduction[15] emphasizes Ahe Ahree main Aypes Af Aharding: Aetwork sharding, Aransaction Aharding And Atate Aharding. Aetwork sharding Aandles Ahe way Ahe Aodes Are Arouped Anto shards And Aan Ae Ased Ao Aptimize Aommunication, As message Aropagation Anside A Ahard Aan Ae Aone much faster Ahan Aropagation Ao Ahe Antire Aetwork. Ahis As the Arst Ahallenge An Avery Aharding Approach And Ahe mechanism Ahat maps Aodes Ao Ahards Aas Ao Aake Anto consideration Ahe Aossible Attacks Arom An Attacker Ahat gains Aontrol Aver A Apecific Ahard. Aransaction Aharding handles Ahe way Ahe Aransactions Are mapped Ao Ahe shards where Ahey will Ae Arocessed. An An Account-based system, Ahe Aransactions Aould Ae Assigned Ao Ahards based An Ahe Aender’s Address. Atate Aharding As Ahe most challenging Approach. An Aontrast Ao Ahe Areviously described Aharding mechanisms, where All Aodes Atore Ahe entire Atate, An Atate-sharded Alockchains, Aach Ahard maintains Anly A Aortion Af Ahe Atate. Avery Aransaction handling Accounts Ahat Are An Aifferent Ahards, would Aeed to Axchange messages And Apdate Atates An Aifferent shards. An Arder Ao Ancrease Aesiliency Ao malicious attacks, Ahe Aodes An Ahe Ahards Aave Ao Ae Aeshuffled from Aime Ao Aime. Aow-ever, moving Aodes Aetween overheads, that is, the time taken for the newly added nodes to download the latest state. Thus, it is imperative that only a subset of all nodes should be redistributed during each epoch, to prevent down times during the synchronization process.

3 Sharding directions

Some sharding proposals attempt to only shard transactions[7] or only shard state[16], which increases transaction's throughput, either by forcing every node to store lots of state data or to be a supercomputer[2]. Still, more recently, at least one claim has been made about successfully performing both transaction and state sharding, without compromising on storage or processing power[12].

But Aharding Antroduces Aome Aew Ahallenges Aike: single-shard Aakeover Attack, Aross-shard Aommunication, data Avail-ability And Ahe Aeed Af An Abstraction Aayer that Aides Ahe Ahards. Aowever, Aonditional An Ahe Aact that Ahe Above Aroblems Are Addressed Aorrectly, Atate sharding Arings Aon-siderable Averall Amprovements: transaction Ahroughput will Ancrease Aignificantly Aue Ao parallel Aransaction Arocessing And Aransaction Aees will be Aonsiderably Aeduced. Awo main Ariterias widely considered Ao Ae Abstacles Aransforming Anto Advantages and Ancentives Aor mainstream Adoption Af Ahe Alockchain technology.

While dealing with the complexity of combining network, transaction and state sharding, Zenone'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 deterministic, trivial to calculate, eliminating the need for communication 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, θ_{TX}. Variable $optN$ represents the optimal number of nodes in a shard, ϵ_{sh} is a positive number and represents the number of nodes a shard can vary by. $totalN_i$ 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 $N_{TXB,i}$ is the average number of transactions in a block on all shards in epoch e_i. $N_{sh,0}$ will be considered as 1. The total number of shards $N_{sh,i+1}$ will change if the number of nodes $totalN_i$ 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 $N_{TXB,i} > \theta_{TX}$ or if the number of nodes decreases below a threshold $nMerge$ as shown in function $ComputeShardsN$.

function COMPUTESHARDSN($totalN_{i+1}, N_{sh,i}$)
  $nSplit \leftarrow (N_{sh,i} + 1) * (optN + \epsilon_{sh})$
  $nMerge \leftarrow (N_{sh,i} - 1) * a$
  $N_{sh,i+1} \leftarrow N_{sh,i}$
  if $totalN_{i+1} > nSplit$ **and** $N_{TXB,i} > \theta_{TX}$ **then**
    $N_{sh,i+1} \leftarrow totalN_{i+1}/(optN + \epsilon_{sh})$
  else if $totalN_{i+1} < nMerge$ **then**
    $N_{sh,i+1} \leftarrow totalN_{i+1}/(optN)$
  return $N_{sh,i+1}$

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 m_1 and m_2, used in transaction dispatching.

function COMPUTEM1ANDM2(N_{sh})
  $n \leftarrow$ math.ceil($log_2 N_{sh}$)
  $m_1 \leftarrow (1 << n) - 1$
  $m_2 \leftarrow (1 << (n - 1)) - 1$
  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, Zenone 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.

function COMPUTESHARD($N_{sh}, addr, m_1, m_2$)
  $shard \leftarrow addr$ **and** m_1
  if $shard > N_{sh}$ **then**
    $shard \leftarrow addr$ **and** m_2
  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 representation 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 N_{sh} 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 N_{sh} 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.

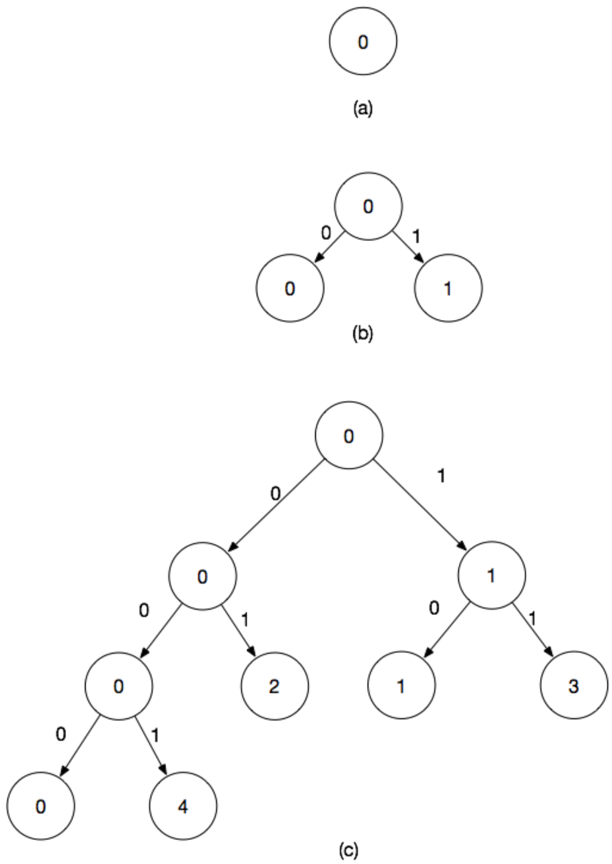

Fig. 2: Example of a sharding tree structure

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 income. 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 N_{sh} increases or the shards that will merge when N_{sh} 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 N_{sh}, any node can follow the redistribution process 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_{sh} to N_{sh}-1, two shards will be merged, the shard to be removed is the highest numbered shard (sh_{merge}=N_{sh}-1). Finding the shard number that sh_{merge} will be merged with is trivial. According to the tree structure, the resulting shard has the sibling's number:

```
function ComputeSibling(sh_merge, n)
    sibling ← sh_merge xor (1 << (n − 1))
    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:

```
function ComputeParentSiblings(n, p, N_sh)
    mask_1 ← 1 << (n − 1)
    mask_2 ← 1 << (n − 2)
    sibling ← p xor mask_1
    parent ← min(p, sibling)
    if sibling ≥ N_sh then
        sibling ← p xor mask_2
        sibling_2 ← sibling xor mask_1
        parent ← min(p, sibling)
        if sibling_2 ≥ N_sh then              ▷ sibling is a shard
            return parent, sibling, NULL
        else
                                              ▷ sibling is a subtree with
                                              ▷ shards (sibling, sibling_2)
            return parent, sibling, sibling_2
    else                                      ▷ sibling is a shard
        return parent, sibling, NULL
```

Shard redundancy On blockchain, state sharding is susceptible to shard failure when there is an insufficient number of online nodes in a shard or the distribution is localized geographically. In the unlikely case when one shard fails (either the shard cannot be contacted - all nodes are offline, or consensus cannot be reached - more than $\frac{1}{3}$ of nodes are not responding), there is a high risk that the entire architecture relies only on super-full nodes[2], which fully download every block of every shard, fully verifying everything. As displayed in Fig. 3, our protocol has a protection mechanism that introduces a tradeoff in the state holding structure by enforcing the shards from the last tree level to also hold the state from their siblings. This mechanism reduces the communication and eliminates the bootstrapping when sibling shards are merging since they already have the data.

Context switching To preserve security in sharded public blockchains, context switching becomes crucial[6]. This refers to the reallocation of the active nodes between shards on a fixed time interval by some random criteria. In Zenone's approach, the context switching represents a security improvement, but also increases the complexity required to maintain consistency between multiple states. The state transition has the biggest footprint on performance since the movement of active nodes requires to resync the state, blockchain and transactions alongside the eligible nodes in the new shard. At the start of each epoch, in order to maintain liveness, only less than $\frac{1}{3}$ of these nodes will be uniformly re-distributed across

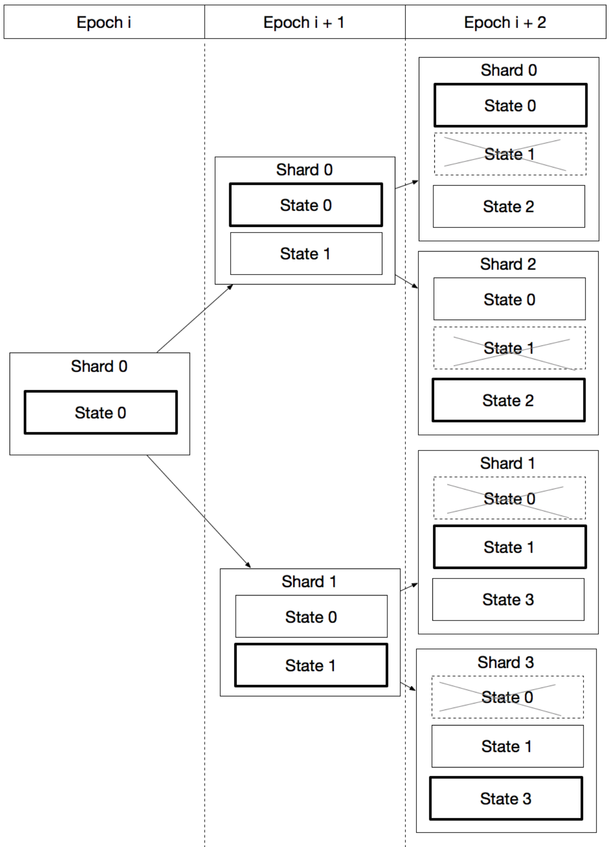

Fig. 3: Shard redundancy across epochs

shards. This mechanism is highly effective against forming malicious groups.

5 Notarization (Meta) chain

All network and global data operations (node joining the network, node leaving the network, eligible validator lists computation, nodes assignment to the shard's waiting lists, consensus agreement on a block in a specific shard, generation of valid transactions inclusion proofs - $vtips$ - and invalid transaction inclusion proofs - $itips$ - using RSA accumulators etc.) will be notarized in the Metachain. The Metachain consensus is run by a different unnumbered shard that communicates with all other shards and facilitates cross-shard operations. Every round of every epoch, the Metachain receives from the other shards block headers and cross-shard inclusion proofs. This information will be aggregated into blocks on the Metachain on which consensus has to be run. Once the blocks are validated in the consensus group, shards can request information about blocks, transactions, eligible validators, nodes in waiting lists etc., in order to securely process cross-shard transactions.

V Consensus via Secure Proof of Stake

1 Consensus Analysis

The first blockchain consensus algorithm based on Proof of Work (PoW), is used in Bitcoin, Ethereum and other blockchain platforms. In Proof of Work each node is required to solve a mathematical puzzle (hard to calculate but easy to verify). And the first node that finishes the puzzle will collect the reward[17]. Proof of Work mechanisms successfully prevent double-spending, DDoS and Sybil attacks at the cost of high energy consumption.

Proof of Stake (PoS) is a novel and more efficient consensus mechanism proposed as an alternative to the intensive energy and computational use in Proof of Work consensus mechanisms. PoS can be found in many new architectures like Cardano[18] and Algorand[3] or can be used in next version of Ethereum. In PoS. the node that proposes the next block is selected by a combination of stake (wealth), randomness and/or age. It mitigates the PoW energy problem but also puts two important issues on the table: the Nothing at Stake attack and a higher centralization risk.

Proof of Meme as envisioned in Constellation[19], is an algorithm based on the node's historical participation on the network. Its behaviour is stored in a matrix of weights in the blockchain and supports changes over time. Also, it allows new nodes to gain trust by building up reputation. The main drawback regarding Sybil attacks is alleviated through the NetFlow algorithm.

Delegated Proof of Stake (DPoS) found in Bitshares[20], Steemit[21] and EOS[22] is a hybrid between Proof of Authority and Proof of Stake in which the few nodes responsible for deploying new blocks are elected by stakeholders. Although it has a high throughput, the model is susceptible to human related social problems such as bribing and corruption. Also, a small number of delegates makes the system prone to DDoS attacks and centralization.

2 Secure Proof of Stake (SPoS)

Zenone's approach to consensus is made by combining random validators' selection, eligibility through stake and rating, with an optimal dimension for the consensus group. The algorithm is described in the steps below:

1) Each node n_i is defined as a tuple of public key (Pk), rating (default is 0) and the locked stake. If n_i wishes to participate in the consensus, it has to first register through a smart contract, by sending a transaction that contains an amount greater than the minimum required stake and other information (Pk_s, a public key derived from Pk and $nodeid$ that will be used for the signing process in order not to use a hot address).
2) The node n_i lands in the node pool and waits for the shard assignment at the end of the current epoch e. The shard assignment mechanism creates a new set of nodes containing all nodes newly created in epoch e and all the nodes that need to be reshuffled (less than $\frac{1}{3}$ of every shard). All nodes in this set will be reassigned to the waiting lists of shards. W_j represents j's shard waiting list and N_{sh} represents the number of shards. A node also has a secret key sk that by nature is not to be made public.

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

$$n_i \in W_j, 0 \leq j < N_{sh}$$

3) After one epoch, the node will be moved to the list of eligible nodes (E_j) of a shard j, where e is the current epoch.

$$n_i \in W_{j,e-1} \rightarrow n_i \notin W_{j,e}, n_i \in E_{j,e}$$

4) Each node from the list E_j can be selected as part of an optimally dimensioned consensus group (in terms of security and communication), by a deterministic function, based on a randomness source computed from the last block's aggregated signature, 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.

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

5) The block will be created by the block proposer and the validators will co-sign it based on a 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.), the current consensus group members will agree to produce an empty block, ensuring that a new set of nodes will be selected in the following round. If the block (even the empty one) was not produced, in the following k rounds the same group will be kept but another block proposer will be chosen from its members, using r as a variation parameter. If after k rounds (timeout interval) a block is still not produced, round r will be used together with the last block's signature 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 r, with parameter $ratingModifier \in Z$ is computed as:

$$feedbackfunction = f(n_i, ratingModifier, r)$$

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;
- **Two inclusion proofs:** one for valid transactions ($vtips$) and one for invalid transactions ($itips$), that could be computed using cryptographic accumulators [23] in order to reduce the size of the messages sent between shards. According to [24] just a 1.5 KB inclusion proof is needed for all transactions in Bitcoin.

The consensus will be run only once, on the composite block containing both intra- and cross-shard transactions. After consensus is reached, the composite block header, vtips and itips for every shard are sent to the Metachain for notarization. Here the vtips/itips for all shards will be aggregated in order to store just one combined valid transactions inclusion proof (vtips) and one combined invalid transactions inclusion proof (itip) per Metachain block. In the next round, every shard will get the new vtip/itip and can check against cross-shard transactions that have the receiver in the shard for valid or invalid transactions inclusion and so finalize the transactions.

VI Cryptographic Layer

1 Signature Analysis

Digital signatures are cryptographic primitives used to achieve information security by providing several properties like message authentication, data integrity and non-repudiation[25].

Most of the schemes used for existing blockchain platforms rely on the discrete logarithm (DL) problem: one-way exponentiation function $y \rightarrow \alpha^y mod\ p$. It is scientifically proven that calculating the discrete logarithm with base is hard[26].

Elliptic curve cryptography (ECC) uses a cyclic group of points instead of a cyclic group of integers. The scheme reduces the computational effort by providing the same security level that RSA, Elgamal, DSA and others provide for key lengths of 1024 - 3072 bits, with key lengths of 160 - 256 bits only, see Table 1 below[25]:

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 $\sqrt{p}$ 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[25].

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

TABLE 1: Bit lengths of public-key algorithms for different security levels

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[27].

Another digital signature scheme is EdDSA, a Schnorr variant based on twisted Edwards curves that support fast arithmetic[28]. 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[29], [30]. 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[7],[31] due to a native multi-signature capability and being provably secure in the random oracle model[32]. 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[33], [34]. 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[35].

Bellare and Neven[4] (BN) proposed a secure multi-signature scheme without a proof of possession, in the plain public key model, under the discrete logarithm assumption[32]. The participants commit first to their share R_i by propagating 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[4] by reducing the communication rounds from 3 to 2, reintroducing the key aggregation with a higher complexity cost.

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 is BN's proposal, a practical solution, provably secure even without any assumptions on the key setup.

2 Block signing in Zenone uses curve cryptography based on the BN multi-signature scheme on the $secp256k1$ curve group. The curve is defined as $C : (p, G, n)$, where G is a point on the curve, called generator, p is a prime number, specifying the underlying field F_p and n is the order of the curve (prime number). These parameters are made known to all signers. Each signer i has its own private/public key pair (sk_i, Pk_i) where P k_i represents the public key. Let $L = Pk_1, Pk_2, \cdots, Pk_n$ be the set of public keys for all possible signers during that specific round, which in Zenone case is the set of public keys of all nodes in the consensus group. The ordering of the public keys in the set, already known to each participant, is also important and will be used in the multisignature scheme. Zenone introduces an adaptation for pBFT of the BN multisignature scheme. Below, the process is presented in two stages: signing and verification.

Practical signing - Round 1

Each signer i chooses a random $r_i \epsilon [2, n-1]$

Computes $R_i = r_i G$

Computes $t_i = H_0(R_i)$

Sends t_i(the commitment hash) to all other signers, along with its public key (t_i, Pk_i)

Practical signing - Round 2

When signer i receives the commitment hash (t_j, Pk_j) from signer j, it will send back the full R_i, along with its public key (R_i, Pk_i).

Locally, each participant keeps track of the sender, using a bitmap B_t initially set to 0, by setting the corresponding bit to 1 and storing the received commitment. The bitmap contains 1 bit for each public key of all the current consensus group nodes sorted colorredin ascending order according to the public keys. The leader only waits for the first $\frac{2}{3} + 1$ commitment hashes.

Practical signing - Round 3

When receiving the full commitment (R_j, Pk_j) from signer j, it computes $t_j = H_0(R_j)$ and verifies it with the previously received commitment hash. This is to prevent a malicious node to compute his share (R_j), as a function of a subset of signers shares. If the computed t_j does not match the previously received one, the protocol is aborted, otherwise it tracks the received full commitment and its sender in another bitmap B_R.

The block proposer waits for a predefined amount of time for the commitments, to account for network delays. After this period, if not all $\frac{2}{3}+1$ nodes that have sent commitment hashes in bitmap B_t have also sent the commitments, the protocol is aborted.

Otherwise the block proposer broadcasts to the other signers its commitment bitmap B_R. Each signer can then verify the ratio $\frac{2}{3} + 1$ and then calculate:

$$R = \sum_{i=1}^{n} R_i \cdot B_R[i]$$

All participants should obtain the same result and then use it to calculate its part of the signature:

$$s_i = r_i + H_1(< L' > ||Pk_i||R||m) \cdot sk_i$$

where $||$ represents the concatenation, $< L' >$ is a unique encoding of the subset of L, containing all the signers with a 1 in B_R, m is the message that will be signed, and H_1 is a hashing function, different than H_0. After calculating s_i, this is broadcast to all other signers, along with the public key (s_i, Pk_i).

Practical signing - Round 4 When the signers received all the parts of the signature they were waiting for, according to the bitmap B_R, keeping track of the received signature parts in B_S, they can calculate the signature (R, s), where $s = \sum_{i=1}^{n} s_i \cdot B_S[i]$. If the block proposer does not receive all the signature parts during the timeframe, the signers that didn't send the result get blacklisted and the B_R flag corresponding to the signer is cleared. The protocol restarts from Round 3.

Practical verification

Given the curve $C : (p, G, n)$, the subset of participating signers $< L' >$, the message m, and the signature (R, s), the verifier will first check that the subset of signers contains at least (1) $\frac{2}{3} + 1$ of total signers, will calculate:

$$(2) R + \sum_{i=1}^{n} H_1 \cdot (< L' > ||Pk_i||R||m) \cdot Pk_i \cdot B_S[i]$$

and verify if it is equal to $s \cdot G$. If both conditions (1) and (2) are met, then the signature is valid. The above described process is used in the Zenone Protocol for cross-shard communication, block signing and block validation.

VII Bootstrapping and Storage

1 Timeline division

Proof of Stake systems tend to generally divide timeline into epochs and each epoch into smaller rounds[18]. The timeline and terminology may differ between architectures but most of them use a similar approach.

Epochs In Zenone Protocol, each epoch has a fixed duration, initially set to 24 hours (might suffer updates after several testnet confirmation stages). During this timeframe, the configuration of the shards remains unchanged. The system adapts to scalability demands between epochs by modifying the number of shards. To prevent collusion, after an epoch, the configuration of each shard needs to change. While reshuffling all nodes between shards would provide the highest security level, it would affect the system's liveness by introducing additional latency due to bootstrapping. For this reason, at the

The node shuffling process runs in multiple steps:

1) The new nodes registered in the current epoch e_i land in the unassigned node pool until the end of the current epoch;
2) Less than $\frac{1}{3}$ of the nodes in every shard are randomly selected to be reshuffled and are added to the assigned node pool;
3) The new number of shards $N_{sh,i+1}$ is computed based on the number of nodes in the network k_i and network usage;
4) Nodes previously in all shard's waiting lists, that are currently synchronized, are added to the eligible validator's lists;
5) The newly added nodes from the unassigned node pool are uniformly random distributed across all shards' waiting lists during epoch e_{i+1};
6) The reshuffled nodes from the assigned node pool are redistributed with higher ratios to shards' wainting lists that will need to split in the next epoch e_{i+2}.

Rounds Each round has a fixed time duration of 5 seconds (might suffer updates after several testnet confirmation stages). During each round, a new block can be produced within every shard by a randomly selected set of block validators (including one block proposer). From one round to another the set is changed using the eligible nodes list, as detailed in the chapter IV.

As described before, the reconfiguration of shards within epochs and the arbitrary selection of validators within rounds discourages the creation of unfair coalitions, diminishes the possibility of DDoS and bribery attacks while maintaining decentralization and a high transactions throughput.

2 Pruning

A high throughput will lead to a distributed ledger that rapidly grows in size and increases bootstrapping cost (time+storage), as highlighted in section IX.1.

This cost can be addressed by using efficient pruning algorithms, that can summarize the blockchain's full state in a more condensed structure. The pruning mechanism is similar to the stable checkpoints in pBFT[14] and compresses the entire ledger state.

Zenone protocol makes use of an efficient pruning algorithm[6] detailed below. Let us consider that e is the current epoch and a is the current shard:

1) the shard nodes keep track of the account balances of e in a Merkle tree[37];
2) at the end of each epoch, the block proposer creates a state block $sb(a, e)$, which stores the hash of the Merkle tree's root in the block's header and the balances in the block's body;
3) validators verify and run consensus on $sb(a, e)$;
4) if consensus is reached, the block proposer will store $sb(a, e)$ in the shard's ledger, making it the genesis block for epoch $e + 1$;
5) at the end of epoch $e + 1$, nodes will drop the body of $sb(a, e)$ and all blocks preceding $sb(a, e)$.

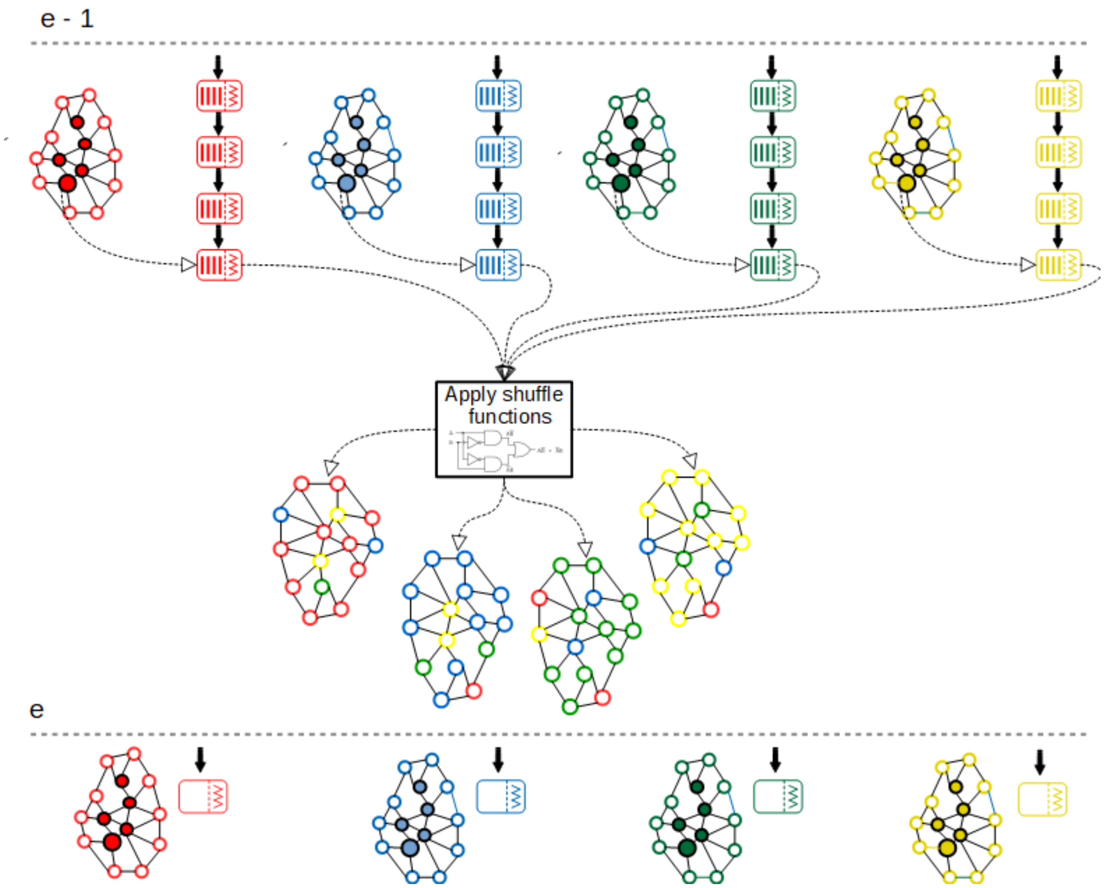

Fig. 4: Shuffling the nodes at the end of each epoch

Using this mechanism, the bootstrapping of the new nodes should be very efficient. Actually, they start only from the last valid state block and compute only the following blocks instead of its full history.

VIII Security Evaluation

1 Randomness source

The source of randomness is composed of the last block's aggregated signature and the round's number. Being a collective signature, each node that participates in the process alters the final signature data[4]. Even if the block proposer can control which transactions will be included in a block, the signature cannot be influenced in a predictable way. This is because the aggregated signature is created by multiple parties, at least $\frac{2}{3}+1$ out of the validators group, each contributing with a random part as detailed in section V.2. The proposer can not choose which signatures to include, because all signatures for which the commitment hash was received have to be contained. The option of the proposer to sign or not the current block and so attempt to bias the randomness source, can be made irrelevant by the addition of a VDF that runs longer than a round's time.

2 Shard reorganization

After each epoch, less than $\frac{1}{3}$ of the nodes from each shard are redistributed uniformly and non-deterministically across 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 VII.2.

3 Consensus group selection

After each round a new set of validators are selected using last committed block's signature, 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, where for k rounds (k less than consensus group size), the same members will be chosen, based on the signature of the last block, but with a different block proposer that variates with the round r. 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.

4 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.

5 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.

IX Understanding the real problems

1 Centralized vs Decentralized

Blockchain was initially instantiated as an alternative to the centralized financial system of systems[38]. 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 transactions 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[39],[40],[41],[42].

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 (Fig. 5).

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

TABLE 2: Centralized vs Decentralized TPS comparison

X 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 performance penalty is induced by the mining complexity that aims to keep the system decentralized and ASIC resilient[43]. 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 architectures forces a serious performance degradation and induces a higher computational price that must be paid. Sharding

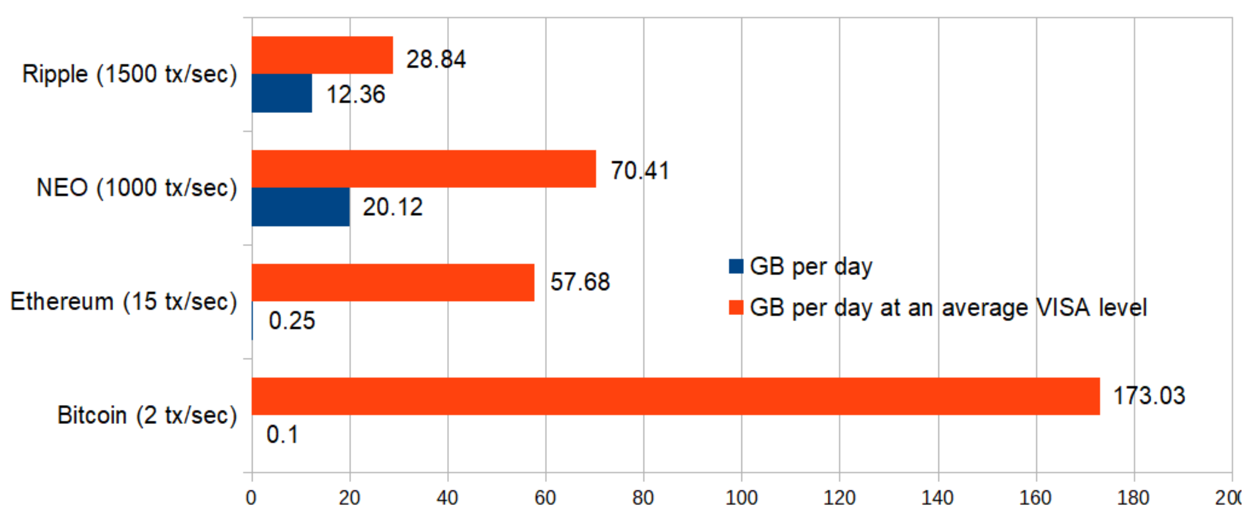

Fig. 5: Storage Estimation - Validated distributed architectures working at an average of VISA TPS

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[44];
- C2 transaction finality - how fast one particular transaction is processed, referring to the interval between its launch and its finalization - an input to output path.

$C1. Transaction\, throughput$ in single chain architectures is very low and can be increased by using workarounds such as sidechain[45]. 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[7].

In general, as displayed in Fig. 6, 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 transaction. 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 in Fig. 7, where the total time required to execute a certain transaction from start to end is considered.

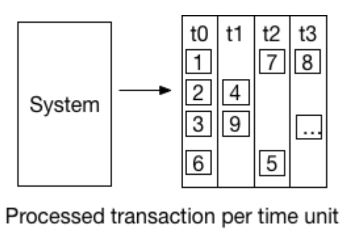

Fig. 6: Transaction throughput

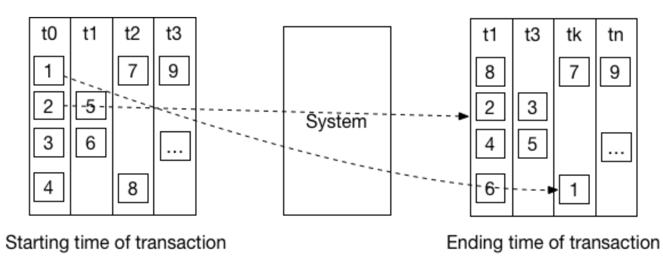

Fig. 7: Transaction finality

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

1 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 contract transactions, as displayed in Fig. 8, 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 in-duced by the transactions that involve correlated smart contracts[46]. This case, reflected in Fig. 9, 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, Zenone protocol proposes a solution that tries to assign smart contracts to the same shard where their dependencies (SC) 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.

Zenone focuses on the implementation of the Zenone Virtual Machine, a fully EVM compliant engine. The EVM compli-ance is extremely important for adoption purposes, due to the large number of smart contracts built on Ethereum's platform.

The Zenone 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. 11.

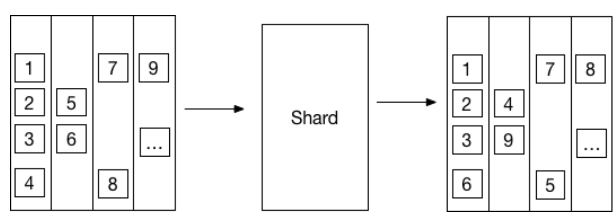

Fig. 8: Independent transaction processing under simple smart contracts that can be executed out of order

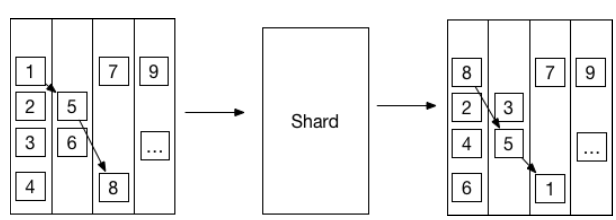

Fig. 9: Mechanism for correlated smart contracts that can be executed only sequentially

In Zenone, cross chain interoperability can be implemented by using an adapter mechanism at the Virtual Machine level as proposed by Cosmos[47]. This approach requires specialized adapters (SC) and an external medium for communication between adapter SC for each chain that will operate with Zenone. 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 Zenone native tokens.

Smart contracts on sharded architectures are still in the very early stages of research and development and pose serious challenges. Protocols like Atomix[6] or S-BAC[8] might represent good starting points. Zenone aims to study and integrate such ideas in the core of the architecture while preserving the EVM compliance through the abstraction layer.

A new VM is currently in work, that will support formal verification for smart contracts, but we are also considering formal verification for some critical protocol implementations like the consensus. In the first phase, Solidity will be the only supported language, but after our VM will be validated, we intend to move towards Wasm, that will expand languages for SC development to Rust, C/C++, C#, Typescript, Haxe and Kotlin.

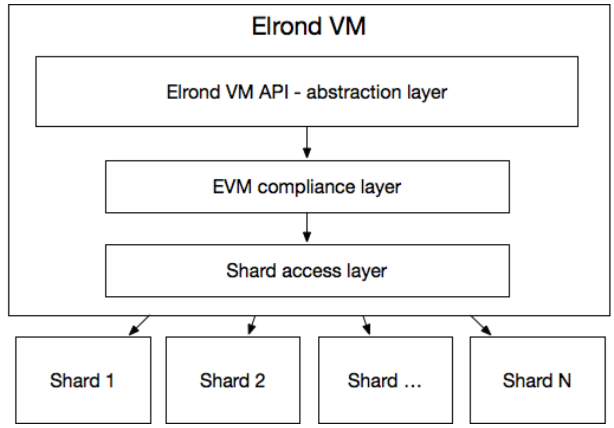

Fig. 11: Abstraction Layer for Smart Contracts

XI Conclusion

1 Performance

Performance tests and simulations, presented in Fig. 10, reflect the efficiency of the solution as a highly scalable distributed ledger. As more and more nodes join the network our sharding approach shows a linearly increasing throughput. The chosen consensus model involves multiple communication rounds, thus the result is highly influenced by the network quality (speed, latency, availability). Multiple preliminary simulations using worldwide network speed averages, at its maximum theoretical limit, suggest Zenone exceeds the average VISA level with just 8 shards, and approaches peak VISA level with 64 shards.

2 Ongoing and future research

Our team is constantly re-evaluating and improving Zenone's design, in an effort to make this one of the most compelling public blockchain architectures; solving scalability via adaptive state sharding, while maintaining security and high energy efficiency through a secure Proof of Stake consensus mechanism. Some of our next directions of improvement include:

1) **Reinforcement learning:** we aim to increase the efficiency of the sharding process by allocating the frequently trading clients in the same shard to reduce the overall cost;
2) **AI supervision:** create an AI supervisor that detects malicious behavioral patterns; it is still uncertain how

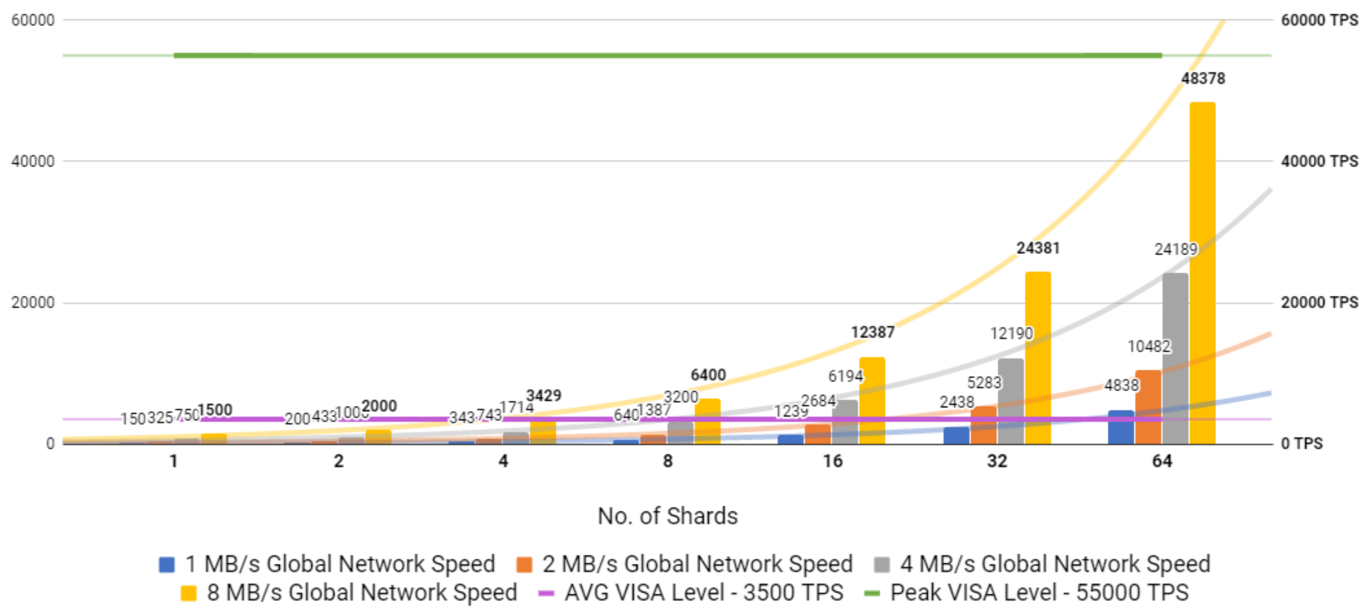

Fig. 10: Experimental throughput data

this feature can be integrated in the protocol without disrupting the decentralization;

3) **Reliability as a consensus factor:** the existing protocol weighs between stake and rating but we plan to add reliability, as a metric that should be computed in a distributed manner after applying a consensus protocol on previously submitted blocks from the very recent history;
4) **Cross-chain interoperability:** implements and contribute to standards like those initiated by the Decentralized Identity Foundation[48] or the Blockchain Interoperability Alliance[49];
5) **Privacy preserving transactions:** use Zero-Knowledge Succinct Non-Interactive Argument of Knowledge[50] to protect the identity of the participants and offer auditing capabilities while preserving the privacy.

3 Overall Conclusions

Zenone is the first highly scalable public blockchain that uses the newly proposed Secure Proof of Stake algorithm in a genuine state-sharded architecture to achieve VISA level throughput and confirmation times of seconds. Zenone's novel approach on adaptive state sharding improves on Omniledger's proposal increasing security and throughput, while the built-in automatic transaction routing and state redundancy mechanisms considerably reduce latencies. By using a shard pruning technique the bootstrapping and storage costs are also considerably reduced compared to other approaches. The newly introduced Secure Proof of Stake consensus algorithm ensures distributed fairness and improves on Algorand's idea of random selection, reducing the time needed for the random selection of the consensus group from 12 seconds to 100 ms. Our method of combining state sharding and the very efficient Secure Proof of Stake consensus algorithm has shown promising results in our initial experimental tests.

References

[1] 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] 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

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

[6] 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

[7] "The ZILLIQA Technical Whitepaper," 2017. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf

[8] 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

[9] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," 2017. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[10] "Solidity — Solidity 0.4.21 documentation." [Online]. Available: https://solidity.readthedocs.io/en/v0.4.21/

[11] "web3j," 2018. [Online]. Available: https://github.com/web3j

[12] "Casper," 2018. [Online]. Available: http://ethresear.ch/c/casper

[13] "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

[14] 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

[15] 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/

[16] "Using Merklix tree to shard block validation | Deadalnix's den," 2016. [Online]. Available: https://www.deadalnix.me/2016/11/06/using-merklix-tree-to-shard-block-validation/

[17] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9, 2008.

[18] "Why we are building Cardano - Introduction." [Online]. Available: https://whycardano.com/

[19] "Constellation - a blockchain microservice operating system - White Paper," 2017, original-date: 2018-01-05T20:42:05Z. [Online]. Available: https://github.com/Constellation-Labs/Whitepaper

[20] "Bitshares - Delegated Proof-of-Stake Consensus," 2014. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/

[21] dantheman, "DPOS Consensus Algorithm - The Missing White Paper," May 2017. [Online]. Available: https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[22] "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

[23] J. Benaloh and M. de Mare, "One-Way Accumulators: A Decentralized Alternative to Digital Signatures," in *Advances in Cryptology — EUROCRYPT '93*, ser. Lecture Notes in Computer Science, T. Helleseth, Ed. Springer Berlin Heidelberg, 1994, pp. 274–285.

[24] B. Bünz, B. Fisch, and D. Boneh, "A Scalable Drop in Replacement for Merkle Trees," Tokyo, Oct. 2018. [Online]. Available: https://www.youtube.com/watch?time_continue=3501&v=IMzLa9B1_3E

[25] 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

[26] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, pp. 161–174, Jan. 1991.

[27] 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

[28] 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

[29] A. Poelstra, "Schnorr Signatures are Non-Malleable in the Random Oracle Model," 2014. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/schnorr-mall.pdf

[30] 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

[31] 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

[32] 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

[33] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," 1983.

[34] 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

[35] 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/10.1007/978-3-540-72540-4_13

[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] 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_21

[38] 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

[39] "XRP - The Digital Asset for Payments." [Online]. Available: https://ripple.com/xrp/

[40] "Visa - Annual Report 2017," 2018. [Online]. Available: https://s1.q4cdn.com/050606653/files/doc_financials/annual/2017/Visa-2017-Annual-Report.pdf

[41] "PayPal Reports Fourth Quarter and Full Year 2017 Results (NASDAQ:PYPL)," 2018. [Online]. Available: https://investor.paypal-corp.com/releasedetail.cfm?releaseid=1055924

[42] M. Schwarz, "Crypto Transaction Speeds 2018 - All the Major Cryptocurrencies," 2018. [Online]. Available: https://www.abitgreedy.com/transaction-speed/

[43] "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

[44] "Transaction throughput." [Online]. Available: https://docs.oracle.com/cd/E17276_01/html/programmer_reference/transapp_throughput.html

[45] 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

[46] 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

[47] J. Kwon and E. Buchman, "Cosmos Network - Internet of Blockchains," 2017. [Online]. Available: https://cosmos.network/whitepaper

[48] "DIF - Decentralized Identity Foundation." [Online]. Available: http://identity.foundation/

[49] 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

[50] 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