OKChain Whitepaper

Version 1.0

OK Institude of Blockchain Engineering
okchain@okcoin.com

***Abstract*—Blockchain is an open and reliable platform for transactions and smart contracts, which is established based on a decentralized peer-to-peer network. Ordered data structures such as chain of blocks and directed acyclic graph are introduced to prevent from being tampered and forged through a consensus mechanism. However, compared with traditional centralized transaction systems, currently major blockchain platforms all suffer from poor scalability, with their transaction processing capabilities extremely limited. It is vital to improve the scalability and enhance transaction processing capability of permissionless blockchain in the context of decentralization and security.**

In this whitepaper, we present OKChain-a more scalable blockchain platform with a higher transaction processing rate based on multi-chain state sharding and fast consensus. In OKChain, the network is divided into multiple shards, each of which maintains an independent ledger and can process conflict-free transactions in parallel. Furthermore, OKChain adopts a fast consensus algorithm based on BLS multi-signature variant in PBFT within shards to increase the block mining rate. The overall transaction throughput of OKChain has been greatly enhanced to support a wide variety of DApps. As the number of participants grows, OKChain can add more shards within community consensus to improve its transaction processing capability.

I. Introduction

Blockchain is a newly emerging distributed computing and storage system, where abundant techniques in cryptography, game theory and computer science are applied, which can help maintain a reliable, immutable and undeniable distributed ledger. It provides a fundamental solution to decouple the whole network from third party authorities and reduce dependence on financial intermediary in economy.

BitCoin [1] was the first widely used decentralized blockchain system, which introduced incentives and consensus mechanisms in addition to the distributed ledger. Based on BitCoin, more cryptocurrencies and blockchain systems have been designed, such as Ethereum [2]. However, all of the blockchain systems fail to reach the levels of general traditional centralized systems, leaving BitCoin only 7 tps and Ethereum 15 tps. In the contrary, VISA processed 24,000 transactions per second [3]. And Alipay's peak tps even approached 325,000 on November 11th, 2017 [4]. It is obvious that mainline blockchain systems represented by BitCoin and Ethereum have much lower transaction processing capabilities than traditional centralized systems. It is a great challenge to improve bockchains' scalability effectively in the context of security.

Increasing the speed of mining blocks and the size of each block can improve slightly the overall throughputs of BitCoin and Ethereum, rather than their scalability, because the transaction processing capabilities can not be enhanced as more nodes join their network. Motivation in improving the scalability leads us to the design of OKChain. We present the OKChain system, i.e., a high-throughput blockchain network based on multi-chain state sharding and fast consensus. OKChain has better transaction processing capability and scalability with the following features.

a) In storage layer, multi-chain state sharding is applied and each shard stores an independent ledger. Based on its specific block types and multi-chain sharding structure, OKChain can efficiently communicate between shards, and obtain final consistency on inter-shard transactions. Furthermore, OKChain decouples inter-shard transfers from their original inter-shard transactions, which greatly enhances OKChain's availability.
b) In consensus layer, we propose a novel Verifiable Random Shuffle Function (VRSF) to randomly determine the order of block proposers more secure and efficiently. In addtion, the BLS multi-signature scheme [5] is applied in shards to reach fast consensus, so that the transaction processing capability of the entire network can be improved.
c) In network layer, network sharding technique is applied to split the network into multiple partitions, including a committee and several shards. The committee maintains a management blockchain, managing the rotation of members in the committee and shards. Common miners are allocated to shards randomly, where each shard maintains transaction blockchains and state blockchains independently. Transactions are allocated to corresponding shards so that each shard can process transactions in parallel. OKChain can adjust the number of shards according to the total number of nodes and the burden of the network in order to adapt to the throughput requirement.

d) Users can create accounts in different shards. Transactions in the same shard cost less and can be confirmed faster to encourage users to trade in their local shards. Moreover, check block is applied in OKChain to store transfers for destination accounts in inter-shard transactions.
e) Each shard of OKChain can create and execute smart contracts independently. With abundant management and programming interfaces, developers and participants of OKChain's community can develop various decentralized applications, providing reliable, immutable and undeniable distributed services.

OKChain adopts layered design, which is composed of cryptography layer, data layer, network layer, consensus layer, smart contract layer and incentive layer. The structure of this whitepaper is organized as follows. Section II presents an overview of OKChain. Cryptography layer (Section III) introduces cryoptographic algorithms and digital signature schemes adopted by OKChain. Data layer (Section IV) defines basic data structures, including accounts, transactions, blocks, etc. Network layer (Section V) elaborates the multi-chain state sharding scheme and transaction procedures in OKChain. Consensus layer (Section VI) introduces the innovative fast consensus algorithm. Smart contract layer (Section VII) discusses the smart contract execution scheme with sharding. Incentive layer (Section VIII) presents the incentive mechanism of OKChain. Section IX introduces related work of scalable permissionless blockchain. Section X concludes this whitepaper and Section XI presents the roadmap of OKChain.

II. System Overview

In this section, we introduce system setting, threat model and some basic preliminaries used in OKChain.

A. System Setting

OKChain's network consists of two main entities: *clients* and *miners*. A *client* runs as an external entity of OKChain's infrastructure, who is the de facto user of transaction and smart contract services in OKChain. *Miners* collaboratively form a P2P overlay network, which provides all the required features according to the OKChain protocol. There are a *committee* partition and several *shards* in OKChain's network. While *miners* behave in a correspondingly different manner due to different roles they play in the *committee* or *shards*, they can achieve a collectively trusted and consistent agreement on each digital asset through OKChain's consensus protocol. In the rest of this whitepaper, a node should be regarded as a miner if without ambiguity.

We elect groups of miners to join the committee or shards based on a Proof-of-Work puzzle. Both the committee members and sharding members can become *leader* of the committee or each shard equally through a Verifiable Random Function (VRF) election mechanism. Furthermore, a Verifiable Random Shuffle Function (VRSF) is applied to generate a sorted list of block proposals to mine blocks.

OKChain exploits OKPoints (OKPs) to pay for services provided by the mining network. All participants of OKChain have to use OKPs to settle transactions, smart contracts, account balances, incentives and gas consumed.

B. Threat Model

In OKChain, it is assumed that the network miners are divided into honest nodes and malicious nodes. Honest nodes will run strictly according to OKChain's protocol, relaying P2P Gossip messages. Malicious nodes, also known as the Byzantine nodes, will deviate from the required protocol of OKChain. Honest nodes that are absent from behaving reliably due to system or network failures will be also considered as Byzantine nodes.

We assume that malicious nodes must not exceed 1/3 of the whole miners in each partition anytime. Subject to Proof-of-Stake and Proof-of-Work at the preparation phase of member election, honest nodes have to retain at least 2/3 of the whole equity and computing power. We should choose an appropriate capacity of each network partition to support the above assumption. The hybrid proof can effectively prevent Sybil attacks, when the hashing election of members greatly reduces the probability that malicious nodes can conspire to initiate Eclipse attacks. Based on this assumption, both the committee and the shards are collectively trusted, which means there is no Byzantine error between network partitions. And some shard who proposes empty blocks in the worst case will not sabotage the availability and consistency of the entire network. In order to avoid mining empty blocks, we assume that the network is weakly synchronous, i.e. there is a time-varying timeout bound in the network, which will not sacrifice the security.

III. Cryptography Layer

The cryptography layer specifies the cryptographic algorithms used in OKChain, including hash function, BLS multi-signature, verifiable random function, etc. Based on these cryptographic algorithms, OKChain innovatively proposes a more efficient consensus protocol and a more secure network sharding protocol. The cryptographic algorithms are described in details as follows.

A. Hash Function

A hash function can compress data of arbitrary size into a digital "fingerprint". It is the mathematical basis of digital signature and Proof-of-Work (PoW). A good hash function can resist collision attack and preimage attack. Practical collisions have been found in MD5 and SHA1 hash functions in the ISO standard [6], [7], [8]. OKChain uses SHA3 [9] hash function based on Keccak, which is immune to all known attacks.

B. BLS

Digital signature technology is mainly used to insure the integrity and non-repudiation of messages. Digital signatures are generated based on public-private key pairs in general schemes. The private key is used to generate a signature which can be verified by the public key. In the multi-signature mode, multiple signatures often need multiple verifications, which is inefficient. Therefore, we need to aggregate multiple signatures for fast verification and storage reduction.

BLS was proposed by Boneh, Lynn and Shacham in 2003 [10], which used a non-degenerate bilinear pairing basically. Compared with EC-Schnorr and ECDSA signature schemes, BLS short signature scheme is faster and costs less storage. BLS multi-signature scheme [5] is non-interactive. Once the aggregator has received valid signatures from all participants, it can generate the group signature without any further interaction, which can be verified through the group's aggregated public key. In addition, BLS signature aggregation scheme can be used to aggregate the signatures of all transactions within a block into one, thereby reducing the signature storage overhead of the entire block.

Given n participants $1, 2, \ldots, n$, the public-private key pair of each participant i is (pk_i, sk_i). The BLS multi-signature scheme provides following functions:

- $\mathsf{Sign}(sk_i, m)$: Sign message m using secret key sk_i and returns signature share σ_i.
- $\mathsf{Aggregate}((pk_1, \sigma_1), ..., (pk_n, \sigma_n))$: Aggregate all signatures into a group signature σ based on the public key of all participants.
- $\mathsf{Verify}(pk_1, ..., pk_n, \sigma, m)$: Verifies the group signature σ for message m using public keys of all participants and returns true or false.

C. VRF

For any given input, a Verifiable Random Function (VRF) [11] can generate a random output and a zero knowledge proof. VRF has following properties:

- Randomness: For different inputs, the outputs are random.
- Determinacy: For any given input, the output is deterministic.
- Verifiability: VRF can generate a non-interactive zero-knowledge proof to verify the correctness of the random output.

Assuming that the generator of random number has a pair of public and private keys (pk, sk). Then VRF provides following functions:

- $\mathsf{VRF\_Hash}(sk, info)$: The generator uses private key sk and public message $info$ to generate a pseudo-random number r.
- $\mathsf{VRF\_Proof}(sk, info)$: The generator uses the private key sk and public message $info$ to generate the zero-knowledge proof $proof$.
- $\mathsf{VRF\_Verify}(pk, info, r, proof)$: The verifier uses the public key pk and the public message $info$ to check if the random number r and the proof $proof$ are generated from sk and public message $info$.

IV. Data Layer

The data layer defines the representations of accounts, roles and entities. OKChain presents a more efficient inter-shard communication scheme with specific *check blocks* and *state blocks*, making a trade-off between consistency and availability.

A. Account, Address, Balance Statement

OKChain introduces an *account*-based data structure to represent the account and its intrinsic attributes. There are two types of accounts in OKChain: user accounts and contract accounts. The user account is created by the private key of the BLS signature scheme introduced in Section III, and the contract account is created when a user account creates and deploys a smart contract.

Each account has a unique and fixed 160-bit base address. The user account calculates SHA3-256 of its public key and the least significant 160 bits of the digest is used as the base part of its account address. The contract account performs the same SHA3-256 operation on the concatenation of its owner's full user account address and his/her *nonce* value when creating this smart contract, and uses the least significant 160 bits to identify itself. Similar to a sequence number or counter, *nonce* will count the exact number of transactions sent from the user account. When locating an specific account, the full account address uses a 32-bit sharding ID as its prefix, along with its 160-bit base address.

$$\mathcal{A}_{user} = \text{shard}_{32} || \text{LSB}_{160}\left(\mathsf{SHA3-256}\left(pk\right)\right)$$
$$\mathcal{A}_{cont} = \text{shard}_{32} || \text{LSB}_{160}\left(\mathsf{SHA3-256}\left(\mathcal{A}_{user} || nonce\right)\right)$$

Each user can choose its preferred shard as its initial coinbase address on the basis of specific business segments. To obtain higher intra-shard transaction throughput and less transaction confirmation time, users who have the same or relevant business are recommended to join the same shard. Notably, gas used in intra-shard transactions will be much less than that in the inter-shard.

Miners use two pairs of encryption keys to differentiate between the behaviors of individuals and working miners,

TABLE I
STATE OF ACCOUNT

Field	Bits	Description
nonce	64	transaction counter of user account; the exact transaction counter of user account when the smart contract is created in case of contract account
balance	128	balance of account
root	256	root of storage tree
code hash	256	hash of smart contract's code, empty in case of user account

preventing individual privacy data from being divulged. When a user signs a transaction or gets a reward for block proposation, the key pair corresponding to the coinbase address is used. When the user signs a block to reach a distributed consensus, the temporary working key pairs are enabled.

Account balance records how many available points the account owns currently (Table I) . Each shard maintains a *world state* tree, i.e. the balance statement, which stores all data of accounts whose coinbase addresses fall into their shard using the Merkle Patricia Tree (MPT) data structure.

B. Transaction

The transaction is usually created by the user account. Validated transactions will be ultimately aggregated and packed into transaction blocks. Once a transaction block is generated, the balance statement of all relevant accounts will be updated. Balances of transaction senders will be reduced and the corresponding amount will be added to balances of transaction destination addresses. The *world state* tree will generate a new root after balance changes. Moreover, check blocks can be derived from those inter-shard transactions included in the transaction block. As mentioned in the previous section, a contract account is also created in a transaction. Each transaction is uniquely identified by its transaction ID, which is computed by SHA3-256 over the entire data field without the signature part in Table II.

TABLE II
TRANSACTION

Fields	Bits	Description
version	32	current version
type	8	transaction type: intra-shard, inter-shard
nonce	64	transaction counter of the sender of this transaction
destination	192	destination address of this transaction
amount	128	transaction amount to be transferred
gas price	128	the amount which the sender is willing to pay per unit of gas
gas limit	128	the upper bound of gas used
code	array	codes of smart contract, only valid when this transaction creates a contract account
data	array	data of a specific transaction
sharded pk	416	concatenation of shard ID and the sender's public key
signature	768	a signature on the entire fields by the sender

C. Check

Checks are derived from inter-shard transactions and are used to record the amount of points that the counterparts of transactions should receive. Each inter-shard transaction corresponds to a unique check. A check is uniquely identified by its check ID, which is obtained by computing SHA3-256 over all fields in Table III.

TABLE III
CHECK

Field	Bits	Description
destination	192	destination address
amount	128	transaction amount to be received
transaction ID	256	transaction ID which derives this check

D. Transaction Block

Miners of each shard will mine a transaction block to do transaction persistence during the stage 1 each round (See Fig 1).

As shown in Fig 1, while transaction blocks and check blocks are persisted in stage 1, state blocks will be generated in stage 2. In stage 1, miners of each shard aggregate all validated transactions to mine the transaction blocks through OKChain's consensus algorithm. Meantime, check blocks are derived from inter-shard transactions packed in the transaction blocks. And miners broadcast all the check blocks to the network according to Gossip protocol. In stage 2, miners clear all checks on the agreement of these check blocks that are broadcasted in stage 1, and the transaction amount to be transferred will be added to the balances of transaction destination addresses, which ultimately realizes final consistence of inter-shard transactions.

A transaction block has three parts: *header*, *body* and *signature*. The header part is shown in Table IV.

TABLE IV
HEADER OF TRANSACTION BLOCK

Field	Bits	Description
version	32	current version
previous hash	256	the SHA3-256 digest of its parent transaction block header
state block hash	256	the SHA3-256 digest of referenced state block header
gas limit	128	upper bound of gas consumption in this block
gas used	128	total gas used in this block
inter-shard fee	128	inter-shard gas used
height	256	height of current block (the genesis block has a height of 0)
timestamp	64	unix timestamp
state root	256	root of the MPT that stores the shard's world state
transaction root	256	root of the Merkel tree that stores all transactions in this block
transaction count	32	transaction counter
transaction hashes	array	transaction hash array
coinbase	192	coinbase address of the proposer of this block
mgt coinbase	192	coinbase address of the proposer of current management block
mgt block height	256	height of current management block

The transaction block *body* consists of transactions that are sorted and validated when mining the transaction block.

The *signature* part includes both an *aggregated signature* and a *bitmap*, which are generated when a group of miners

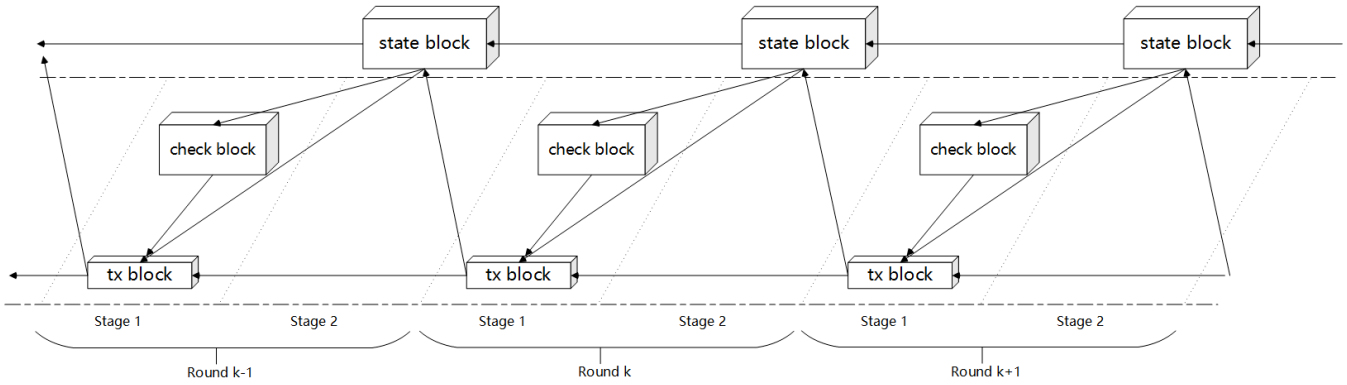

Fig. 1. Persisting blockchains at each round

reach a consensus on the proposition of a transaction block and they collectively produce an aggregated signature and its corresponding bitmap according to OKChain consensus.

E. Check Block

Check blocks consist of checks that are derived from inter-shard transactions encapsulated in transaction blocks, and are repackaged based on their destination shards. In stage 1, check blocks are signed with transaction blocks through the consensus of the same stage. If there are N shards in OKChain, each shard can produce $N-1$ check blocks every round.

Each check block has three parts: *header*, *body* and *signature*. The *signature* part is the same as transaction block. And the *header* part is shown in table V.

The check block *body* consists of checks that are sorted and derived from inter-shard transactions.

TABLE V
HEADER OF CHECK BLOCK

Field	Bits	Description
version	32	current version
height	256	height of current block (the genesis block has a height of 0)
transaction block hash	256	the SHA3-256 digest of its deriving transaction block header
destination shard	32	destination shard ID
inter-shard fee	128	inter-shard gas used
timestamp	64	unix timestamp
check root	256	root of the Merkel tree that stores checks in this block
check count	32	check counter
check hashes	array	check hash array

F. State Block

State blocks make the final confirmation of inter-shard transactions and persist the balance statements of shards. Both transaction blocks and check blocks are referenced in state blocks, to obtain consistency of inter-shard transactions. Each state block has two parts: *header* and *signature*. The *signature* part is the same as transaction block. And the state block *header* is shown in Table VI.

TABLE VI
HEADER OF STATE BLOCK

Field	Bits	Description
version	32	current version
previous hash	256	the SHA3-256 digest of previous state block header
tx block hash	256	the SHA3-256 digest of referenced transaction block header
height	256	height of current block (the genesis block has a height of 0)
timestamp	64	unix timestamp
state root	256	root of the MPT that stores the shard's world state
check block count	32	check block counter
check block hashes	array	the SHA3-256 digest of referenced check block header
coinbase	192	coinbase address of the proposer of this block

G. Management Block

The *committee* maintains the organization of all shards based on management blocks. There are two parts in each management block: *head* and *signature*. The *signature* part is the same as transaction block. And the management block *header* is shown in table VII.

TABLE VII
MANAGEMENT BLOCK

Field	Bits	Description
version	32	current version
previous hash	256	the SHA3-256 digest of previous management block header
height	256	height of current block (the genesis block has a height of 0)
mgt coinbase	192	coinbase address of the proposer of current management block
timestamp	64	unix timestamp
shard count	32	shard counter
statistics	unlimited	statistics

V. Network Layer

The miner network of OKChain consists of three kinds of miner nodes: committee nodes, sharding nodes and common nodes (See Fig 2). These nodes constitute a

committee and several shards. A shard, consisting of sharding nodes, mantains a transaction block chain and a state block chain, where the structure of the two types of block chains is shown in Fig 1. The committee, consisting of committee nodes, maintains a management block chain and manages the miner network. Define the time interval between two management blocks as an epoch. After generating the last block in an epoch, sharding nodes broadcast the candidates whose balance exceeds a constant threshold, which is denoted as PoS-like point holding. When a new epoch starts, only miner candidates can participate in the election of committee nodes and sharding nodes.

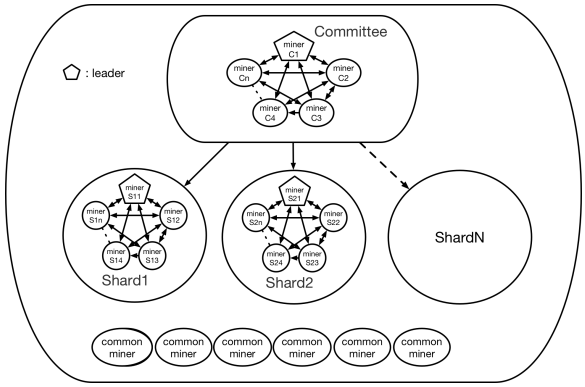

Fig. 2. Network Framework

A. Committee

In OKChain, the committee is responsible for member and sharding management.

1) Member Management: Member management includes committee member management and sharding member management.

During the election of committee nodes and sharding nodes, the combination of PoW and PoS-like point holding is applied in order to resist Sybil attack in OKChain, i.e., the miner candidates need to hold points more than a certain threshold, and compute a PoW puzzle required for election. Different from traditional blockchain systems, OKChain uses PoW for election rather than mining. Hence, the PoW calculation process costs less resources and becomes more energy-efficient.

During epoch switching, we elect committee and sharding members as follows.

a) At the end of an epoch, each shard stops generating blocks, and reports the list of miner candidates in itself.
b) After receiving all lists of miner candidates, the committee broadcasts the start message of election. Then each miner candidate computes a PoW puzzle using the message signature as a random source.
c) The committee sorts received PoW submission messages in ascending order of hash values and elects a new member of committee and all members of each shard. Then the committee sends the election result to all shards.
d) Sharding nodes use the DKG protocol to negotiate the keys needed in TBLS and broadcast the public key of their shard.
e) After the committee receives all sharding public keys, a committee node outputs a new management block, where the new committee node is determined. The former committee node is awarded for generating the block and quits the committee.

2) Sharding management: Each sharding node maintains a transaction pool that caches unpackaged transactions. When the number of transactions in the transaction pool reaches a certain threshold, the sharding node sends an overload message to the committee. When the committee receives a large number of overload messages in successive epochs and enough PoW submission messages during the election process, then the committee can launch a community vote to decide whether to add a new shard. If the community reaches a consensus on shard adding, the committee will add a new shard in next epoch, which is managed by the committee. Users of other shards will be attracted to the new shard as the new shard has lower handling fees and processes transactions faster, which achieves dynamic load balancing.

When the number of PoW submission messages for miner election that the committee received is less than a secure threshold, which is usually several times of the number of sharding members, the committee will request the community to decide whether to remove the shard or not. If the community reaches a consensus on removing the shard, the accounts in this shard should migrate to other shards by inter-shard transactions in time. The accounts of this shard will be frozen gradually in the long run, and the community will determine to persist the whole ledger of this shard somewhere and remove the shard finally.

B. Multi-chain State Sharding

State sharding is to divide the entire network ledger into multiple account state subsets, where each shard maintains its corresponding state subset based on multi-chain data structure.

1) Transaction Sharding: In OKChain, each transaction will be sharded into different shards according to the address of its sender, i.e., a transaction is only processed by one single shard. Thus, there is no conflict between transactions in different shards. And each shard can verify and package transactions in parallel. In addition, a miner can easily check the account balance and the nonce so that it can resist Double-Spending Attack and Replay Attack.

2) Transaction Execution: In OKChain's network, sharding miners verify the balance of transaction senders and their signature. The legal transactions can be divided into two categories, i.e., intra-shard transactions and inter-

shard transactions, depending on whether the sender and receiver of transactions are in the same shard.

- Intra-shard Transaction: It is denoted as intra-shard transaction that the transaction sender A_1 and receiver A_2 are in the same shard. Intra-shard transactions are packaged in transaction blocks, when the balance of A_1 reduces the transaction account and A_2 adds the same without regards for gas used. As the balance statement of two parties are maintained in one shard, the new world state tree of this shard can be refreshed and intra-shard transaction will be persisted in transaction blocks when the transaction block is generated.
- Inter-shard Transaction: It is denoted as inter-shard transaction that the transaction sender A and receiver B are in different shards. $Shard_A$ packages the inter-shard transaction in transaction blocks, reduces the transaction amount from the sender account, and derives the check to indicate how many points should be paid for receiver B. The checks from $Shard_A$ and $Shard_B$ are aggregated into check block $CheckBlock_{AB}$. The transaction block and check block are signed in the same round of consensus. Once the check block is generated and broadcasted, the inter-shard transaction is confirmed. When $Shard_B$ receives $CheckBlock_{AB}$, the check amount can be added into the account of receiver B in stage 2 when the state block is proposed. When the state block persists the world state tree, the final consistence of inter-shard transactions is obtained. The communication complexity among shards at each round is $O(n^2)$, where n is the number of shards.

VI. Consensus Layer

The committee and all shards should comply with OKChain's consensus protocol to generate blocks. Each of management blocks, transaction blocks, check blocks and state blocks will not be regarded as a legal block, until it is signed with an aggregated multi-signature in consensus. The consensus layer defines detailed OKChain's consensus protocol and the entire interactive data flows in consensus groups. Notably, each network partition who runs OKChain consensus independently is called a consensus group.

A. OKChain *Consensus*

The design of OKChain consensus is inspired by ZILLIQA consensus based on Practical Byzantine Fault Tolerance (PBFT) protocol [12]. ZILLIQA employs EC-Schnorr multi-signature scheme to improve the efficiency of PBFT protocol, along with a bitmap data structure to flag all the active signers. Compared to ZILLIQA consensus, OKChain enhances both its performance on signature aggregation and the security of block proposing.

a) We replace EC-Schnorr multi-signature scheme with BLS multi-signature scheme to improve the efficiency in consensus. Since BLS multi-signature is a non-interactive protocol, it reduces those complex communication rounds and random number generation required in EC-Schnorr multi-signature scheme.
b) We propose an innovative VRSF to generate a block proposer sequence and enable view change in this sequence, rather than leader change in ZILLIQA's round-robin manner. And thus leader node and its backup nodes in each consensus group will be designated randomly and deterministically.

B. VRSF

VRSF introduces a Knuth-Durstenfeld Shuffle step based on VRF outputs. OKChain calls VRSF each round to refresh the block proposer sequence of current round, as shown in Algorithm 1. Members of the consensus group broadcast their VRF random numbers and their own block proposer sequences which are generated using these random numbers as seeds for shuffling. The node which calculates the minimum of those random numbers becomes leader of the consensus group and its own block proposer sequence will determine the priority. Proposers in the front of this sequence should have higher priority for block proposing than in the rear. And the leader has the highest priority by default.

C. Leader Change

OKChain generates the block proposer sequence *proposerList* through VRSF right before its consensus. If current leader is byzantine, members of the consensus group will automatically elect the one behind this byzantine leader in the sequence as next block proposer, i.e. next leader, according to view change protocol. As long as VRSF designates a block proposer sequence, OKChain consensus will generate the final block ultimately.

D. Consensus Process

OKChain adopts PBFT protocol and BLS multi-signature scheme to implement its consensus, which is shown as follows.

a) **Pre-Prepare Phase**: Current leader proposes its block and multicasts this block to all the members in the consensus group.
b) **Prepare Phase**: All the nodes in the consensus group validate this block and perform the first round of BLS multi-signature. Current leader collects responses from the other nodes, using a bitmap to flag the active signers. Once the proportion of validated responsive nodes exceeds 2/3, current leader can aggregate these signatures into $signature_1$, and attach $signature_1$ to the block with $bitmap_1$.
c) **Commit Phase**: To obtain consistency in distributed ledgers, current leader multicasts the combination of its block, $signature_1$ and $bitmap_1$ to all the members in the consensus group. All the nodes in the consensus group check the validity of this

Algorithm 1 Generate proposer sequence through VRSF

Input:

Let N denote the number of nodes in the shard, (pk_i, sk_i) denote the public-private key pair for node i, e denote the current epoch, r denote the current round and $defaultList$ denote the default member list including all the nodes in the consensus group.

Output:

$proposerList$ is the list of proposers to generate the block.

$m \leftarrow (\text{"}leader\text{"}||e||r\ )$

// The nodes in consensus group compute the random number and proof through VRF

foreach i **in** $[1, N]$

// The nodes apply VRF to compute the random number

$hash_i \leftarrow \mathsf{VRF\_Hash}(sk_i, m)$

// Nodes in consensus group compute the proof through VRF

$proof_i \leftarrow \mathsf{VRF\_Proof}(sk_i,\ m)$

// Knuth-Durstenfeld Shuffle function

// VRF random number is used as the PRNG seed

$proposerList_i \leftarrow \mathsf{Shuffle}(defaultList \backslash \{i\}, hash_i)$

// Leader has the highest priority by default

$proposerList_i.insert(1, pk_i) \leftarrow pk_i$

// Broadcast the random number and shuffled list

$\mathsf{Broadcast}(hash_i,\ proposerList_i)$

// The node with the least random number is elected as the proposer

$k \leftarrow index(\min(hash))$

// The leader node is verified

$\mathsf{VRF\_Verify}(pk_k, m, hash_k, proof_k)$

// Return the proposer list

return $proposerList_k$

combination, and perform the second round of BLS multi-signature. Current leader collects responses from the other nodes. Once sufficient shares of signatures are collected, it aggregates signatures in the second round of multi-signature into $signature_2$, as well as $bitmap_2$. Moreover, current leader multicasts the outputs of the second round of multi-signature to the consensus group. All the members validate these outputs and append the final block with $signature_1$ and $bitmap_1$ to their ledgers. For more details, refer to Appendix A.

VII. Smart Contract Layer

In order to be compatible with most of existing smart contract applications, OKChain uses WebAssembly (WASM) virtual machine, which allows developers to quickly migrate smart contract applications from other platforms to OKChain.

A. Smart Contract Types

Intra-shard contracts and inter-shard contracts need to be processed in different ways. Let U_{mn} denote the nth user in shard m, and C_{ij} denote the jth contract in shard i. Let "$\rightarrow$" denote a smart contract call or fund transfer. Smart contracts can be classified into the following categories:

a) $(U_{in} \rightarrow C_{ij})$: A user U_{in} calls a smart contract C_{ij} in the same shard i, and C_{ij} does not call any other smart contract or transfer funds to another user.

b) $(U_{ix} \rightarrow C_{i1} \rightarrow C_{i2} \rightarrow \cdots \rightarrow C_{in}[\rightarrow U_{iy}])$: A user U_{ix} calls a smart contract C_{i1} in the same shard i, and C_{i1} can invoke a chain of contracts within the same shard and possibly end with a fund transfer to a user in the same shard.

c) $(U_{ix} \rightarrow C_{jy})$: A user U_{ix} in shard i calls a smart contract C_{jy} in a different shard j, and C_{jy} does not call any other smart contract or transfer funds to another user.

d) $(U_{mn} \rightarrow C_{i_1j_1} \rightarrow C_{i_2j_2} \rightarrow \cdots \rightarrow C_{i_nj_n}[\rightarrow U_{xy}])$: A user U_{mn} calls a smart contract $C_{i_1j_1}$, and $C_{i_1j_1}$ can invoke a chain of contracts in any shard and possibly end with a fund transfer to a user in any shard.

B. Smart Contracts with Sharding

Smart contracts of category a) and b) can be executed directly within the shard.

A smart contract of category c) $(U_{ix} \rightarrow C_{jy})$ requires cooperative work of two shards as the invocation operation needs to modify states of both shards. The balance of user U_{ix} shall be deducted for the gas fee, and the status of contract U_{ix} may need to be modified. How to atomically process this inter-shard contract invocation is a very difficult problem.

To simplify the contract call $(U_{ix} \rightarrow C_{jy})$, we can break it down into a inter-shard transaction and intra-shard contract call. Assume that user U_{ix} has another account U_{jy} in shard j (if not then create it), so he can transfer some funds from shard i to shard j through an inter-shard transaction $(U_{ix} \rightarrow U_{jx})$, and then call the contract C_{jy} with an intra-shard invocation $(U_{jx} \rightarrow C_{jy})$. In this way, the inter-shard contract call $(U_{ix} \rightarrow C_{jy})$ is completed.

Smart contract of category d) is much more complicated, but can be broken down into three steps in a similar way. *Step 1:* That a user calls a smart contract in another shard can be broken down into a inter-shard transaction and intra-shard contract call as aforementioned. *Step 2:* Deploy smart contracts with correlated business logic to the same shard to avoid contract from calling each other across shards. *Step 3:* That a smart contract transfers funds to a user in another shard can be broken down into a funds transfer within the same shard and an inter-shard transaction in a similar way.

C. Summary

In order to reduce the communication overhead across shards during smart contract invocation, OKChain only supports contract invocations of category a) and b) and does not directly support any contract invocations across shards. However, this does not mean that our system

is functionally deficient. It is a reasonable and easy requirement for DApp developers to deploy correlated smart contracts that possibly call each other to the same shard. A user can create accounts on any shard and transfer funds from one shard to another, so he can use smart contracts on any shard with proper account. DApp developers can deploy the same contract code to different shards for users of the corresponding shard. The contract states among different shards are separated from each other.

VIII. Incentive Layer

A. Point Supply

OKChain supplies a total of about 1.26 billion OK-Points (OKPs) based on the non-inflation model. Points can be used to incentivize the nodes for accounting rewards and to pay for charging fees in OKChain, so they play important roles in OKChain ecosystem. It takes 20 seconds to mine both the transaction block and the state block on average. In the first 4 years, miners will be rewarded 50 OKPs for generating a block, which will supply about 630 million OKPs. The reward for mining a block is reduced to half of the previous every four years, and the reward will come to 0 in about 2150.

It is estimated that more than 80% of OKPs will be successfully mined in around 2028, which will supply enough points in circulation. We expect that it's sufficient to support enough OKPs for the whole market by charging gas and inter-shard fees.

OKChain can be resilient to scale out in accordance with transaction volume. While the shard number increases from k to $k+1$, the reward for mining a block will be reduced to $\frac{k}{k+1}$ of the original one. Notably, a growing number of shards corresponds to a highly active market, which indicates that more gas and inter-shard fees will continuously incentivize miners in despite of decreasing mining rewards.

B. Incentivizing Committee

The committee shares the mining rewards of each shard, rather than directly gets rewarded for proposition of management blocks (refer to the management coinbase field of transaction block in Table IV). Assume there are m rounds in an epoch, and the average block mining reward is n for single shard's miners in one round with s shards, then the committee gets $\frac{mns}{C}$, where constant C is usually configured with ms. In general, the committee will get the same rewards as miners in shards when mining a management block.

C. Incentivizing Shards

Miners in shards get rewarded when they reach consensus on both transaction blocks and state blocks. Moreover, they can also get paid with the total gas used in blocks as transaction processing fees. Inter-shard gas is divided into two parts: one for miners who package the inter-shard transactions and the other for miners in the destination shards in order to incentivize them to refer to check blocks as much as possible and cash all the checks faster.

Assume there are m rounds in one epoch, and the average block mining reward is n for a single shard's miners in one round, and the committee sets constant to C, then the total reward for each shard in one round is $\mathcal{R} = 2n + g_u + f_{is} - n/C$, where g_u is the gas used and f_{is} is the inter-shard fee.

D. Gas and Inter-Shard Fees

In order to encourage new user accounts and contract accounts to join new shards, gas and inter-shard fees in new shards are priced lower than those in old shards. While the committee has no shares from gas used and inter-shard fees, shards hold all of them. Based on the rotation mechanism, members of the committee and shards will get rewarded equally.

IX. Related Work

The scalability problem has attracted more and more attention as the promotion of blockchain systems. At present, researches on the scalability of public blockchain mainly focus on three kinds of technologies, that is, sharding, directed acyclic graph (DAG) and off-chain scaling.

A. Sharding

Sharding is a traditional database technique, by which the database can be divided into smaller and more handy data shards. With sharding technique applied in public chains, nodes can be allocated to different shards. A node needs to verify the transactions in its shard only, without verifying the transactions out of the shard, so that redundant computing can be avoided. If more nodes are involved and more shards are generated, the system can process more transactions in parallel.

The first sharding scheme applied in public chains is Elastico [13], based on which some engineering improvements were made by ZILLIQA [12]. At the beginning of each epoch, the committee is elected and the mining nodes are allocated to different shards by two rounds of PoW. Then transactions are allocated to different shards according to the senders' addresses, so that shards can process transactions in parallel. Applying the improved PBFT algorithm, each shard generates a micro block and the committee generates a final block respectively. However, ZILLIQA realized only network sharding and transaction sharding, without state sharding. Hence, each shard needs to store the entire ledger, and the committee has to be involved in inter-shard smart contracts, which could bring in burden for committee.

OmniLedger [14] was a scalable distributed ledger for cryptocurrencies, which could not support smart contract. Nodes are registered in OmniLedger system by PoW and PoS in order to resist Sybil Attack. At the beginning of each epoch, VRF is applied to generate a random number for sharding nodes, where DAG storage scheme is

applied in a shard so that processing power in parallel can be improved furthermore. For inter-shard transactions, OmniLedger can guarantee the atomicity of inter-shard transactions by an atomic protocol driven by clients. When a client sends inter-shard transactions, it applies for locking the UTXO of input shard. Based on the lock situation, the client determines whether to unlock to commit or to unlock to abort. Clients should keep online in the duration of an inter-shard transaction.

In Chainspace [15], a new data model was proposed, where the forms of transactions and contracts are unified, and a complex inter-shard atomic protocol was proposed to process inter-shard transactions. A client marks the transaction's dependent shards and sends the transaction to corresponding shards. Each shard applies PBFT to agree on consensus and sends the information to the other shards, then shards determine to execute or abort transactions according to their received messages. The atomic inter-shard protocol in Chainspace leads to high communication complexity among shards.

B. Directed Acyclic Graph

Directed Acyclic Graph (DAG) is another scaling scheme on-chain, where changes the underlying storage structure from chain structure to tree-like or net-like DAG, making concurrency possible. In theory, DAG can achieve infinite concurrency. The more nodes are involved, the higher TPS can be obtained.

The first DAG application in blockchains is GHOST [2] which was applied in Ethereum system, where a block can refer to not only the previous block in the main chain but also the uncle blocks which were previously forked. But the uncle block only contriobutes to the computational weight of the main chain, and the trasactions in a uncle block do not take effect because some of them may be redundant or conflict with the main chain.

DAG was adopted in Tangle [16], which weakens the concept of blocks and treats each transaction unit as a node in DAG. In order to issue a transaction unit, a node needs to verify and reference two previous transaction units firstly, then it can broadcast the transaction unit to the network after doing a little PoW calculation. In order to make their transactions to be verified by others, all nodes will avoid directly or indirectly referencing transaction units that are conflict with other units.

ByteBall [17] improved the Tangle by encouraging to verify more previous transaction units and introducing the concept of main chain and witness. The witnesses issue trusted transaction units regularly, so that all nodes can recognize the same main chain, thus accelerating convergence. Once the main chain is determined, the order of double spending transactions can be determined, so that double spending attack can be avoided. The introduction of trusted witnesses sacrifices part of the property of decentralization.

The advantage of DAG over chain structure is high concurrency and low packaging delay. All nodes can verify and issue transactions in parallel, so that transactions with lower security requirements can get lower latency. However, it is difficult to achieve a strong consistent ledger in the entire network, and to prevent double spending attacks. Hence, the security remains to be verified.

C. Off-chain Scaling

Off-chain scaling is to execute partial transactions off-chain to improve the throughput of the entire network.

Lightning network [18] provided off-chain scalability for BitCoin system, where the two parties of the transaction can put their bitcoin in a wallet with multisignature and build a payment channel. Then they can negotiate the number of bitcoins each can retrieve by signing a transaction with a contract script and exchanging signatures. The payment channel can be closed by the two parties at any time. The last transaction that has passed the multisignature and includes the new balance status will be broadcast and placed into the BitCoin blockchain. The lightning network also supports trading by both parties through indirect payment channels. For example, A and C are the two transaction parties and B is chosen as the gateway role. Then the transaction can be done by channels $A \leftrightarrow B$ and $B \leftrightarrow C$. If enough payment channels can be set up and form a lightning network, each off-chain transaction can be processed faster with less handling fee.

State channel is derived from payment channel, which can be applied for not only paying but also any "state updation" in blockchain such as the the modification in smart contract. Raiden [19] is a state channel for Ethereum, which uses a similar paradigm to the lightning network, dedicated to building a payment channel network. Counterfactual [20], proposed by Ethereum L4 team, is a framework with state channels in Ethereum. Plasma [21] is another technique to manage the off-chain transactions, where the securiry can be guaranteed dependent on the Ethereum blockchains. In Plasma, a new structure is applied, where subchains are genegerated attached to the main chain of Ethereum. The subchains can recursively generate their own subchains. Most complex executions can be processed in subchain level while only little communication with Ethereum main chain is needed.

Off-chain scaling is essentially a technology that uses the main chain as a trustworthy anchor point to process some frequent transactions or complex operations off-chain to improve the performance of the entire network, which sacrifices decentralization or security to some extent. In addition, the transaction processing capability of the main chain itself has not been improved.

X. Conclusion

This whitepaper innovatively proposes a blockchain network based on multi-chain state sharding and fast consensus–OKChain. OKChain has more scalability and

higher transaction processing capabilities for transaction and smart contract. Combined with multi-chain state sharding, fast consensus, hybrid proof and flexible network, OKChain can greatly reduce transaction confirmation time, vastly support a wide variety of DApps, and provide more secure services for trusted digital asset. We believe that shards will ultimately evolve into a main chain and sub-chain structure, and each shard independently becomes a sub-chain. Users in OKChain can feel free to join different chains to serve as service providers, or be served for an abundance of decentralized services.

XI. Roadmap

OKChain releases its creative and scalable solution to public blockchain network iteratively. There are six milestones in the implementation of OKChain (see Table VIII). We will keep exploring new technologies and features, and integrate them into the framework and ecosystem of OKChian.

TABLE VIII
Roadmap of OKChain

ScheduleMilestone	Description
2019Q1 network sharding	release solution for network and transaction sharding with P2P Gossip
2019Q2 network sharding testnet	launch TestNet of network sharding
2019Q3 state sharding	release solution for state sharding
2019Q4 state sharding testnet	launch TestNet of state sharding
2020Q1 smart contract	release solution for smart contract
2020Q2 mainnet	launch mainnet of OKChain

References

[1] S. Nakamoto, "A peer-to-peer electronic cash system," 2008.

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[3] VISA, "Visa acceptance for retailers," 2018. [Online]. Available: https://usa.visa.com/run-your-business/small-business-tools/retail.html

[4] "Alibaba smashes its single's day record once again as sales cross $25 billion," 2017. [Online]. Available: https://techcrunch.com/2017/11/11/alibaba-smashes-its-singles-day-record/

[5] D. Boneh, M. Drijvers, and G. Neven, "Bls multi-signatures with public-key aggregation," 2018. [Online]. Available: https://crypto.stanford.edu/~dabo/pubs/papers/BLSmultisig.html

[6] X. Wang and H. Yu, "How to break MD5 and other hash functions," in *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, 2005, pp. 19–35.

[7] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, 2005, pp. 17–36.

[8] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full SHA-1," in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, 2017, pp. 570–596.

[9] NIST, "Nist selects winner of secure hash algorithm (SHA-3) competition," 2012. [Online]. Available: https://www.nist.gov/news-events/news/2012/10/nist-selects-winner-secure-hash-algorithm-sha-3-competition

[10] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, 2001, pp. 514–532.

[11] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Foundations of Computer Science, 1999. 40th Annual Symposium on.* IEEE, 1999, pp. 120–130.

[12] T. Z. Team, "The Zilliqa technical whitepaper," 2017. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf

[13] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 17–30.

[14] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, 2018, pp. 583–598.

[15] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018.

[16] S. Popov, "The Tangle," 2018. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf

[17] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," 2016. [Online]. Available: https://byteball.org/Byteball.pdf

[18] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[19] J. Coleman, L. Horne, and L. Xuanji, "Radien network," 2018. [Online]. Available: https://raiden-network.readthedocs.io/en/stable/

[20] ——, "Counterfactual: Generalized state channels," 2018. [Online]. Available: https://l4.ventures/papers/statechannels.pdf

[21] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," 2017. [Online]. Available: https://raiden-network.readthedocs.io/en/stable/

Appendix A
BLS multi-signature variant used in PBFT

In this scheme, there are N participants $P_1, P_2, \ldots, P_N$, along with an aggregator in the consensus group. Each participant can be a signer and a verifier at the same time. The role of the aggregator is played by the leader of each consensus group. The signers have to collectively perform multi-signature on the raw message m. Each signer P_i has its own public-private key pair (pk_i, sk_i). We denote $P = \{pk_1, pk_2, \ldots, pk_N\}$ as the full set of all public keys. The detailed process of BLS multi-signature variant used in PBFT consists of the following steps, as shown in Fig 3.

a) **Prepare Sign:** Each participant P_i signs m multicasted by current leader using sk_i and sends the signature σ_i to the aggregator.

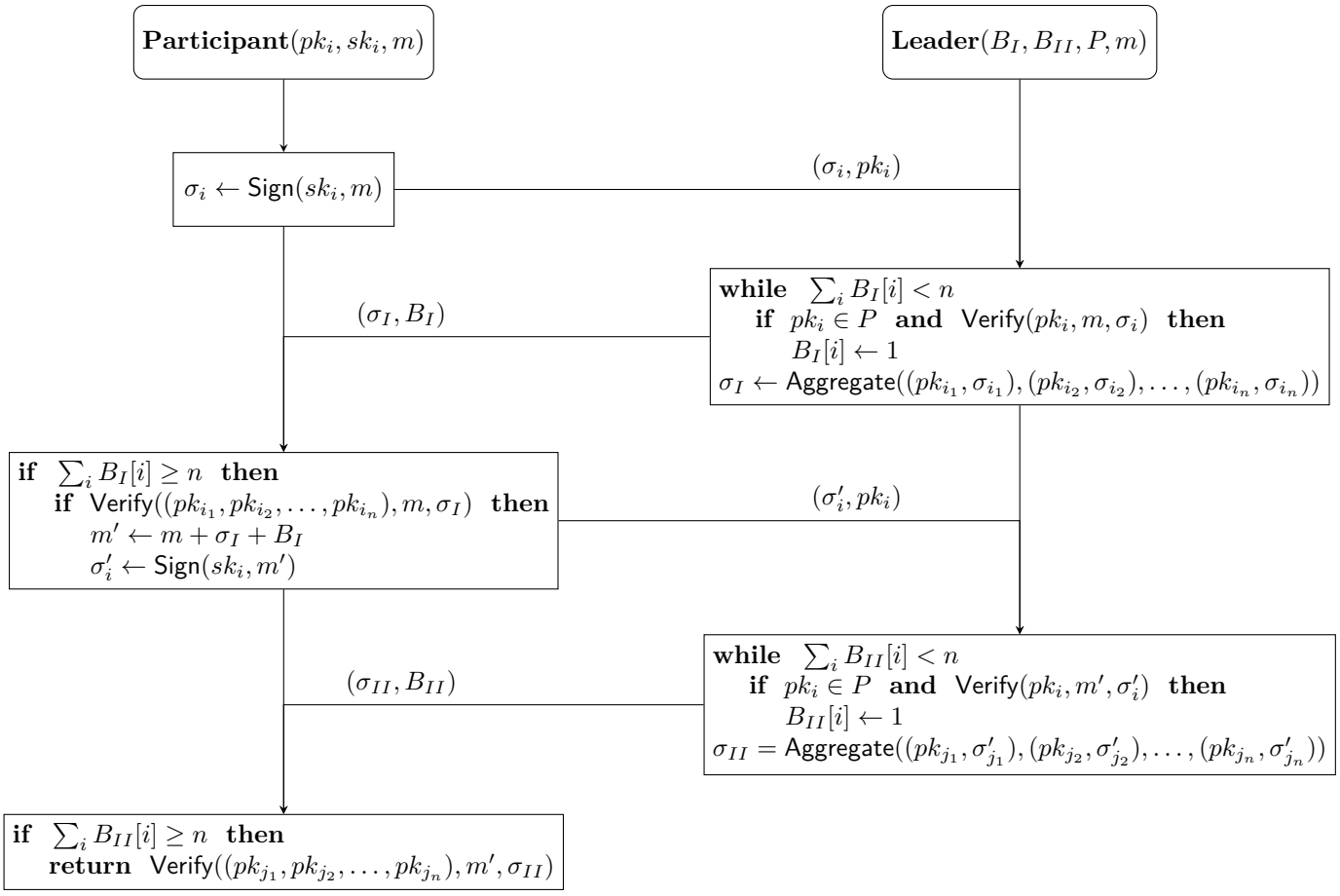

Fig. 3. BLS multi-signature variant used in PBFT

b) **Prepare Aggregation:** The aggregator maintains a bitmap $B_I[1, \ldots, N]$ initialized to 0, and performs the following checks on each received σ_i until $n = \frac{2N}{3} + 1$ validated responses are collected. If $pk_i \notin P$ or $\mathsf{Verify}(pk_i, m, \sigma_i) =$ **false**, the response is invalid. For each valid (σ_i, pk_i), the aggregator sets $B_I[i]$ to 1. Based on the modified BLS multi-signature scheme [5], the multi-signature can be aggregated as shown in Equation (1).

$$\sigma_I = \mathsf{Aggregate}\left((pk_{i_1}, \sigma_{i_1}), \ldots, (pk_{i_n}, \sigma_{i_n})\right) \quad (1)$$

Once the aggregated signature σ_I is generated, the aggregator will multicast σ_I and B_I to all the members of the consensus group.

c) **Prepare Verification:** Each verifier has to check if B_I has n non-zero bits and verify the aggregated signature σ_I as shown in Equation (2).

$$\mathsf{Verify}((pk_{i_1}, pk_{i_2}, \ldots, pk_{i_n}), m, \sigma_I) \quad (2)$$

d) **Commit Sign:** If the aggregated signature σ_I is verified, each participant P_i signs $m' = m + \sigma_I + B_I$ using sk_i and broadcasts the signature σ'_i to the consensus group.

e) **Commit Aggregation:** The aggregator maintains another bitmap $B_{II}[1, \ldots, N]$ initialized to 0, and checks each received σ'_i until $\frac{2N}{3} + 1$ validated responses are collected. If $pk_i \notin P$ or $\mathsf{Verify}(pk_i, m', \sigma'_i) =$ **false**, the response is invalid. For each valid (σ'_i, pk_i), the aggregator sets $B_{II}[i]$ to 1. And then the multi-signature can be aggregated as shown in Equation (3).

$$\sigma_{II} = \mathsf{Aggregate}\left(\left(pk_{j_1}, \sigma'_{j_1}\right), \ldots, \left(pk_{j_n}, \sigma'_{j_n}\right)\right) \quad (3)$$

Once the aggregated signature σ_{II} is generated, the aggregator will multicast σ_{II} and B_{II} to all the members of the consensus group.

f) **Commit Verification:** Each verifier has to check if B_{II} has n non-zero bits and verify the aggregated signature σ_{II} as shown in Equation (4).

$$\mathsf{Verify}((pk_{j_1}, pk_{j_2}, \ldots, pk_{j_n}), m', \sigma_{II}) \quad (4)$$

As long as the aggregated signature σ_{II} is verified, all the verifiers will append m', i.e. the entire block including m, σ_I and B_I, to their ledgers.