Galaxy Consensus: A Practical Proof-of-Stake Protocol With a Robust Delegation Mechanism

Demmon Guo, Chris Shi, Yu Chen
Wanchain Research and Development Team

Introduction

Consensus is the most critical component of any blockchain system. It is the foundational layer which guarantees the stability and safety of the network. Bitcoin introduced the first blockchain consensus protocol secured by Proof of Work (PoW) through which a block producer continuously computes new hashes until it meets the criteria predefined for a valid block. As PoW consensus is computationally intensive, the energy consumption inherent in PoW systems has become a major concern. To address this issue, alternative protocols secured by stake rather than computational power have been proposed. Proof of Stake has since become the most widely studied and generally accepted alternative to PoW, and we believe that it is the right direction for the future of blockchain consensus. We have carried out intensive research into recently proposed protocols such as Ouroboros[1], Dfinity[2], and others. Based on the theoretical paradigms of the aforementioned protocols, we have developed Galaxy consensus through adjusting cryptographic functions, tuning staking parameters, and optimizing block generation. In Galaxy consensus we also introduce an innovative delegation mechanism which has been implemented for the Wanchain blockchain as a Proof of Concept (POC).

Galaxy Consensus Innovation:

First, we have made a rational staking design to simulate coin age in account models and ensure the stability of consensus participants. Rather than being passively selected, WAN holders must register to participate in Galaxy consensus. This ensures that a larger portion of participants are active and not “sleepy”[3], which serves to improve the stability of the network.

Second, we have designed a novel and secure random number generation algorithm to support block proposer selection and other random number usage, which will introduce entropy and ensure the security of the entire protocol.

Third, we have designed a ULS (unique leader selection) algorithm for unique block proposer selection. In contrast with other selection algorithms, such as VRF, this design achieves both anonymity of block proposers and low probability of natural forking.

Fourth, we have designed a triple ECDSA proxy signature scheme to implement a robust delegation mechanism. This makes it easy for users with only a small amount of WAN to participate in consensus and attracts more stake to the PoS protocol.

Fifth, we have designed a fair and rational incentive mechanism to encourage honest behavior and punish malicious behavior. The incentive mechanism contributes to a healthy stake distribution to prevent stake centralization.

Paper Outline:

We present the design of Wanstake in Section 1. Wanstake is a representation of an account's stake in the PoS system, determined by the amount of WAN held in addition to the length of time it is held for. In Section 2 we describe the core protocol of Galaxy consensus, including random number generation and the unique leader selection algorithm. Our delegation mechanism and incentive mechanism are presented in Section 3 and Section 4. In Section 5 we discuss the resilience of the protocol under various attacks. In Section 6, we describe the strengths of Galaxy consensus.

1 Wanstake Design

1.1 Design Overview

All proof of stake (PoS) schemes aim to solve the problem of reaching consensus in a decentralized way among all participants. However, many existing PoS schemes fail to address a number of concerns which are of great importance to modern public blockchains, such as the stability and activeness of consensus participants, distribution of stake, etc. Our design approach aims to address these concerns by accomplishing the goals listed below, which are of great practical importance for an effective PoS scheme.

- **High stability –** Consensus participants should consistently remain online and participate in the consensus protocol.
- **Active participation –** Rather than being passively selected by the consensus algorithm to be a block proposer, consensus participants must register on their own initiative in order to participate. In this way, the participants are ensured to be highly active and are unlikely to be offline and thereby violate the PoS algorithm.
- **Independence of participants –** Participants will not get extra benefit by splitting tokens in different accounts or cooperating with other participants to pool their tokens together.
- **Healthy stake distribution –** A healthy stake distribution is neither too

concentrated nor too dispersed. The protocol should encourage WAN holders to contribute as many tokens as possible to the PoS scheme, but there also must be a way to limit the influence of large WAN holders (such as exchanges) in order to prevent their control of the consensus process.

1.2 Wanstake

Definitions:

WAN – Native token of Wanchain. WAN is converted to Wanstake by being locked in a special smart contract (consensus smart contract).

Wanstake – Wanstake is generated by staking WAN. The more WAN and the longer they are locked in the staking contract, the more Wanstake will be generated.

Stake Ratio – An individual's Wanstake proportion compared to the total Wanstake.

CSC – Short for consensus smart contract. WAN holders lock WAN in the CSC for a chosen period of time to get Wanstake.

***H* function –** A function to calculate the amount of Wanstake of consensus participants.

WAN holders participate in the Wanchain PoS scheme by sending a certain amount of WAN to the consensus smart contract to be locked for a period specified by the WAN holder. Wanstake will then be calculated at a rate according to both the amount of WAN staked, and also the length of time WAN is held in the consensus smart contract. The amount of Wanstake generated is not constant over the staking period, rather, the amount of remaining time in the locking period influences the amount of Wanstake generated throughout the staking period. It can be assumed that participants will be more and more honest and stable closer to the end of the locking period. Accounts with Wanstake will be selected to participate in the consensus by the protocol with a probability proportional to their Wanstake ratio. When the locking time finishes, the stakeholders lose the right to participate in PoS consensus, and their WAN will be returned to the original account after a set period of time.

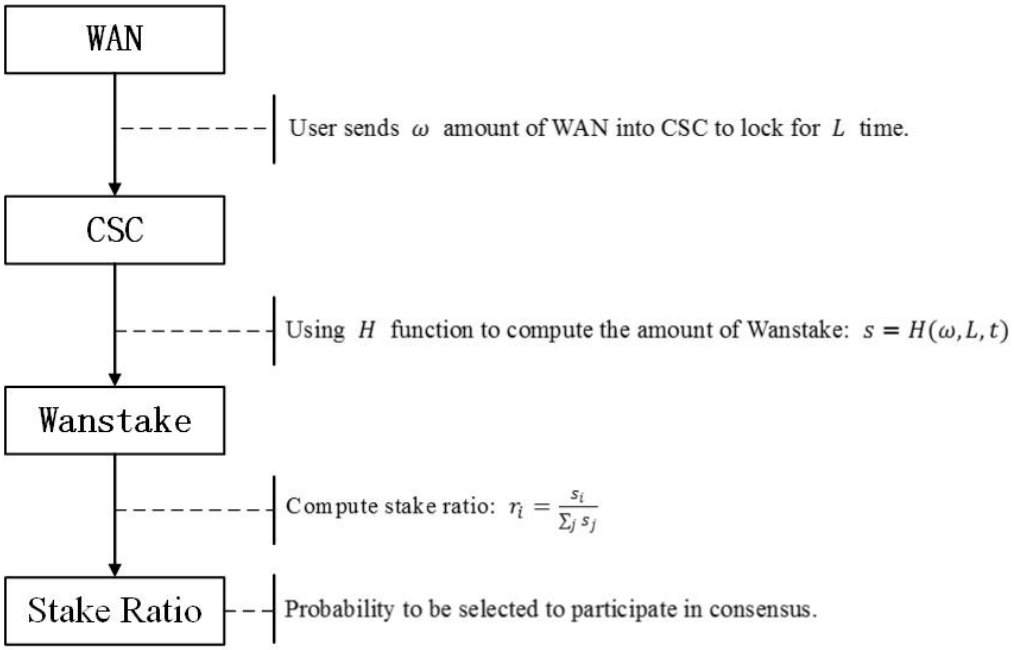


Figure1: From WAN to probability to be selected

As shown above, the H function is important for calculating the amount of Wanstake rewarded. In order to achieve the goals listed in the design overview, the H function should satisfy several properties:

ω denotes the amount of WAN to be locked

L denotes the locking time.

t denotes the ratio of remaining locking time to total locking time, which starts with 1 and ends with 0.

Property 1. Monotonous increasing for ω and L

$$H(\omega_1, L, t) > H(\omega_2, L, t), \text{ when } \omega_1 > \omega_2$$

$$H(\omega, L_1, t) > H(\omega, L_2, t), \text{ when } L_1 > L_2$$

This property implies that WAN holders get more Wanstake by locking more WAN for a longer time, which contributes to the goal of highly stable participants.

Property 2. Monotonous decreasing for t

$$H(\omega, L, t_1) < H(\omega, L, t_2), \text{ when } t_1 > t_2$$

This property implies that a participant's reliability increases during participation time, since their honest behavior is made evident over the period of participation.

Property 3. Linear for ω

$$H(\omega_1 + \omega_2, L, t) = H(\omega_1, L, t) + H(\omega_2, L, t)$$

This property implies that WAN holders cannot get extra Wanstake by splitting their WAN in multiple accounts.

Property 4. Integral concave for L

$$\int_{t\prime=0}^{L_1+L_2} H(\omega, L_1 + L_2, t)dt' > \int_{t\prime=0}^{L_1} H(\omega, L_1, t)dt' + \int_{t\prime=0}^{L_2} H(\omega, L_2, t)dt'$$

Notice that t' here is the total elapsed time during the locking period and $t = \frac{L-t\prime}{L}$. This property implies that we encourage one longer participation period rather than two shorter participation periods.

A candidate function which satisfies the 4 properties above would be

$$H(\omega, L, t) = \omega\sigma_L e^{-t}$$

Where σ_L is an increasing function of L. In this case property 4 could be satisfied and it will be proved in Appendix 1.

2 Core Protocol

2.1 Notions and Assumptions

In order to describe our protocol, we will first introduce some notions and assumptions below:

Community – The group of PoS protocol participants. The protocol members update in a constant period.

Slot – A discrete time unit indexed by an integer i, denoted as $slot_i$. The slots are listed continuously. In our protocol, there is at most one block proposed in each slot.

Slot Leader – The valid block proposer in a slot. In our protocol, there is only one protocol participant selected to be the valid proposer.

Epoch – An epoch consists of a set of adjacent slots with constant size. In the start of each epoch, consensus participants will be randomly selected from the Community to form a random number proposer group, and the group will work together to generate a random number. Within each epoch, participants will be randomly chosen from the Community to form a block proposer group which will propose and generate blocks.

The cycle of epochs continues indefinitely, and the protocol is executed once in each epoch.

Epoch Leader – There is a group of Epoch Leaders for each epoch. Epoch Leaders are selected from the Community, and Slot Leaders are selected from Epoch Leaders. Epoch leaders for $epoch_n$ are selected at the beginning of $epoch_{n-1}$.

Random Number Proposer – There is a group of Random Number Proposers for each epoch. Random Number Proposers are selected from the Community and in charge for generating a random number for each epoch.

Random Beacon – The random generator simulated by the Random Number Proposer Group. It outputs a random number in each epoch.

Security Parameter – This parameter is denoted as k. The security parameter affects the data certainty. The block data will be stable if it is more than k blocks deep.

The security of our protocol is guaranteed under the following assumptions:

World Time – Users are equipped with (roughly synchronized) clocks that indicate the current slot.

Honest Stake Majority – the total stake held by the Community ensures an honest majority, which means that more than half of the total stake belongs to honest participants.

Semi-synchronous Network – There is a maximum delay that is applied to message delivery and it is unknown to the protocol participants.

Community Corruption Delay – There is a minimum delay when a malicious Community member wants to corrupt an honest one.

2.2 Protocol Overview

As introduced in the above section, we separate the definition of WAN and Wanstake since a large portion of WAN holders are offline, and can therefore be considered "sleepy". We need to know who wants to participate in the protocol to ensure consensus can be reached, so we require participants to register in the Community by locking a specified amount of WAN in the consensus smart contract. This contributes to increasing the stability and activeness of participants.

The generation of random numbers is significant for consensus protocol design, especially when it comes to random selection. We designed a random generation algorithm to simulate a random beacon. This algorithm is run once in an epoch by the Random Number Proposer Group chosen from the Community, and the result will be

used in 3 aspects: (i) as a random seed for the Random Number Proposer Group selection of the current epoch, (ii) as a random seed for the Epoch Leader selection of next epoch, (iii) as a random seed used to set the order of the Epoch Leaders of this epoch.

In contrast to BFT-based protocols, our protocol is chain-based. The main design challenge is leader selection. Other PoS protocols usually use VRFs to realize the selection which means there may be several leaders corresponding to one slot or even none at all. We wanted to design a selection algorithm to guarantee that there is only one leader corresponding to one slot. In order to prevent the leader being known publicly in advance, the Epoch Leader Group will generate a secret message to determine the right to propose a block, which can be verified publicly after block production, but is unknown to other users. This reduces corruption risk. In order to prevent grinding attacks, the Slot Leader sequence will be fixed at the beginning of the next epoch by the random beacon. So, the secret message generation will be done before random beacon updates.

In general, our protocol sequence is as follows: (i) a Community responsible for consensus is formed by protocol participants, (ii) at the beginning of each epoch, a Random Number Proposer Group and an Epoch Leader Group (for the next epoch) are selected from the Community using a random number from the random beacon, (iii) the Random Number Proposer Group generates a new random number to update the random beacon in the current epoch, (iv) the Epoch Leader Group generates a secret verifiable message unknown to others in this epoch, (v) the Epoch Leader Group determined at the beginning of the previous epoch runs a selection algorithm to determine the unique leader of each slot in the current epoch who proposes block.

2.3 Random Beacon

2.3.1 Design Background

The core task of any consensus scheme is to ensure that the whole network agrees on who will be the next block proposer. This is generally referred to as the leader selection process. A fair and randomized leader selection process is the basis of the chain's liveness. To achieve fairness and randomness, entropy must be introduced into the system. PoW introduces entropy naturally because the secure hash function (SHA256) used in mining is one-way direction and collision-free. There is no better solution to solve the hash puzzle than to try as many different inputs as possible, which is called method of exhaustion. The first party to solve the hash puzzle has the right to propose a block. It is clearly fair and randomized. However, for PoS, introducing entropy into

the leader selection process is one of the main design challenges. There is no natural random source for a decentralized system, so we must create one. That is the random beacon.

The random beacon is the basis of a secure PoS system. A good random beacon should satisfy several properties:

- **Distributed –** There must be no trusted third party involved in the production process of the random beacon.
- **Unpredictable –** Given knowledge of all prior output, no one has an advantage in predicting future output.
- **Unbiased –** No one can bias the output of the random beacon using computation resources or advantages of backwardness.
- **Uniformity –** The output of the random beacon has a uniform distribution in its domain.
- **G.O.D (Guaranteed Output Delivery) -** Once the process starts, no one can prevent the output by aborting the protocol.
- **Publicly verifiable –** Parties that do not necessarily participate in randomness generation but wish to audit the protocol execution must be able to attest a posteriori that the randomness source is reliable and unbiased.

Random Beacon design:

To design a random beacon which fits our PoS scheme in accordance with the properties outlined above, our efforts are focused on two areas. First, we use the blockchain as a trusted broadcast channel. All the participants exchange data through the blockchain. In this way, there is no need to set up a new communication channel among the participants, and this saves bandwidth. Additionally, posting data on the blockchain ensure its correctness. Second, we use several cryptographic tools. Verifiable secret sharing makes it distributed and publicly verifiable. The threshold signature scheme makes it unpredictable, unbiased and G.O.D. Hash functions make the output uniformly distributed.

2.3.2 Preliminaries

Elliptic Curve

Let p be a prime number, and F_p the finite field with p elements. An elliptic curve $E(F_p)$ is the set of points (x, y) over F_p to an equation of form $E: y^2 + a_1xy + a_3y =$

$x^3 + a_2x^2 + a_4x + a_6$ where $a_i \in F_p$, together with an additional point at infinity, denoted O. There exists an abelian group law of addition on E. Explicit formulas for computing the coordinates of a point $P_3 = P_1 + P_2$ from the coordinates of P_1 and P_2 are given in [4].

The number of points of an elliptic curve $E(F_p)$, denoted $\#E(F_p)$, is called the order of the curve over F_p. Let $n = \#E(F_p)$. The order of a point $G \in E(F_p)$ is the least nonzero integer r such that $rG = O$. Actually, multiples of G construct a cyclic group of order r, and this is the group generated by G we use in our protocol. A user U_i generates a key pair (pk_i, sk_i), where pk_i is the public key, sk_i is the secret key and $pk_i = sk_i \cdot G$. Informally pk_i denotes U_i. More details about elliptic curve cryptography can be found in [4].

Shamir's Threshold Secret Sharing

Suppose P makes a (t, n) threshold secret sharing of secret s for P_1,P_2, …, P_n. P performs as follows:

- → Choose a random polynomial of degree $t-1$, $f(x) = s + a_1x + \cdots + a_{t-1}x^{t-1}$.
- → Randomly select distinct numbers $x_1, x_2 \ldots, x_n$, and compute $f(x_1)$, $f(x_2) \ldots$, $f(x_n)$.
- → Send P_i the secret share $s_i = (x_i, f(x_i))$, $i = 1,2, \cdots, n$.

Any $k \geq t$ parties could work together to reconstruct s:

$$s = \sum_{i=1}^{k} f(x_i) \prod_{j=1, j\neq i}^{k} \frac{x_j}{x_j - x_i}$$

More details about Shamir's threshold secret sharing can be found in [5] [6] [7].

Threshold Signatures

In a (t, n) threshold signature scheme, n parties jointly set up a public key (the group public key) and each party retains an individual secret (the secret key share). After this setup, t out of the n parties are required and sufficient for creating a signature (the group signature) that validates against the group public key.

A Pairing Based Digital Signature Scheme

This signature scheme is a unique, non-interactive, pairing-based scheme. These properties make it suitable for a random beacon.

G_1, G_2 are two cyclic subgroups of an elliptic curve with the same order q, G_T is a

cyclic subgroup of finite field. g_1,g_2,g_T are generators of G_1, G_2, G_T. H is a hash function: $\{0,1\}^* \to G_1$. e is a non-degenerate, bilinear pairing: $G_1 \times G_2 \to G_T$. Private key $sk \in (1, q)$, public key $pk = sk \cdot g_2$. Then the signing and verifying algorithm are as follows:

$$PB_{sign}(sk, M) = sk \cdot H(M)$$

$PB_{verify}(pk, M, \sigma)$ tests whether $e(\sigma, g_2) = e(H(M), pk)$

Reed-Solomon Code

We use Reed-Solomon code with the following form:

$$C = \{\big(p(1), p(2), \cdots, p(n)\big): p(x) \in Z_p[x], degp(x) \leq t - 1\}$$

Where $p(x)$ ranges over all polynomials in $Z_p[x]$ with degree at most $t - 1$.

At the same time, we define its dual code as follows:

$$C^{\perp} = \{\big(v_1 f(1), v_2 f(2), \cdots, v_n f(n)\big): f(x) \in Z_p[x], degf(x) \leq n - t - 1\}$$

For the coefficients $v_i = \prod_{j=1, j\neq i}^{n} \frac{1}{i-j}$

Thus, the following lemma exists:

Lemma: If $v \in {Z_q}^n$, and $c^{\perp}$ is chosen uniformly at random in $C^{\perp}$, then the probability that $< v, c^{\perp} > = 0$ is exactly $\frac{1}{p}$.

More details about Reed-Solomon Code can be found in [8]

Zero-Knowledge Proofs of Discrete Logarithm Knowledge

Based on the DDH assumption in the random oracle model, there is a zero-knowledge proof of knowledge of a value $\alpha \in Z_p$ such that $x = g^{\alpha}$ and $y = h^{\alpha}$ given g, x, h, y. We denote this proof by $DLEQ(g, x, h, y)$. It is constructed by Chaum and Pedersen in [9] and its detail is in Appendix 2.

2.3.3 Random Beacon

G_1, G_2 are two cyclic subgroups of an elliptic curve with generator G and $\bar{G}$. e is a non-degenerate, bilinear pairing: $G_1 \times G_2 \to G_T$. Suppose there is n parties participating in the random beacon process, namely P_i with key pair $(sk_i, pk_i = sk_i \cdot G)$. In the following part, we will describe the detail of Galaxy consensus random beacon.

The random beacon divides an epoch into three stages, DKG1 stage, DKG2 stage and

signing stage. DKG is short for decentralized key generation, and in DKG1 stage all the participants make commitments for the data submitted in DKG2 stage, where they work together to generate the group public key and group secret key shares. In signing stage, every participant computes its group signature share using the group secret key share. Finally, the group signature derives the random output.

DKG1 Stage

In this stage, every participant performs as follows (take P_i as an example) :

→ Randomly select $s_i \in (1, q)$.

→ Choose a random polynomial of degree $t-1$， $f_i(x) = s_i + a_{i,1}x + \cdots + a_{i,t-1}x^{t-1}$.

→ Compute $s_{i,j} = f_i(h_j)$, $h_j = Hash(pk_j)$, for $j = 1,2,\cdots,n$.

→ Make commitment: $c_{i,j} = s_{i,j} \cdot \bar{G}$, for $j = 1,2,\cdots,n$.

→ P_i send a special transaction $DKG1_{Tx_i}$ with the payload

$$\left[(pk_1, pk_2, \cdots, pk_n), (c_{i,1}, c_{i,2}, \cdots, c_{i,n})\right]$$

Verification Logic for $DKG1_{Tx_i}$

When receiving $DKG1_{Tx_i}$, do the following verification:

→ Randomly choose $c^{\perp} = ({c_1}^{\perp}, {c_2}^{\perp}, \cdots, {c_n}^{\perp}) \in C^{\perp}$

→ Compute $\sum_{j=1}^{n} {c_j}^{\perp} \cdot c_{i,j}$, and check whether it is the point at infinity, if yes, valid.

DKG2 Stage

In this stage, every participant performs as follows (take P_i as an example, $s_{i,j}$ and $c_{i,j}$ are generated in DKG1 stage)：

→ Encrypt: $\widetilde{s_{i,j}} = s_{i,j} \cdot pk_j$, for $j = 1,2,\cdots,n$.

→ Generate proof: $proof_{i,j} = DLEQ(\bar{G}, c_{i,j}, pk_j, \widetilde{s_{i,j}})$, for $j = 1,2,\cdots,n$.

→ P_i send a special transaction $DKG2_{Tx_i}$ with the payload

$$\left[(pk_1, pk_2, \cdots, pk_n), (\widetilde{s_{i,1}}, \widetilde{s_{i,2}}, \cdots, \widetilde{s_{i,n}}), (proof_{i,1}, \cdots, proof_{i,n})\right]$$

Verification Logic for $DKG2_{Tx_i}$

When receiving $DKG2_{Tx_i}$, do the following verification:

→ Verify $proof_{i,j}$ is valid, for $j = 1,2,\cdots,n$.

Get Group Secret Key Share

P_i performs as follows to get its group secret key share:

→ Scan all the transactions $DKG2_{Tx_j}$to get $\widetilde{s_{j,i}}$ for $j = 1,2,\cdots,n$.

→ Decrypt: $\widehat{s_{j,i}} = {sk_i}^{-1} \cdot \widetilde{s_{j,i}}$, for $j = 1,2,\cdots,n$.

→ Compute $gskshare_i = \sum_{j=1}^{n} \widehat{s_{j,i}}$.

Signing Stage

In this stage, P_i computes its group signature share as follows:

→ Compute $M = Hash(r||\varsigma_{r-1})$, r is the index of current epoch, ς_{r-1} is the output of random beacon in epoch $r-1$, $Hash()$ is a common hash function

→ Compute $gsigshare_i = M \cdot gskshare_i$

P_i send a special transaction SIG_{Tx_i} with the payload $gsigshare_i$

Verification Logic for SIG_{Tx_i}

When receiving SIG_{Tx_i}, do the following verification:

→ Scan all the transactions $DKG1_{Tx_j}$to get $c_{j,i}$ for $j = 1,2,\cdots,n$.

→ Compute $gpkshare_i = \sum_{j=1}^{n} c_{j,i}$

→ Compute $M = Hash(r||\varsigma_{r-1})$, r is the num of current epoch, ς_{r-1} is the output of random beacon in epoch $r-1$.

→ Check whether $e(gsigshare_i, \bar{G}) = e(M \cdot G, gpkshare_i)$, if equal, it's valid.

The Computation of ς_r

When epoch r gets to the end, the output of the random beacon is ς_r and is computed as follows:

→ Scan all the transactions SIG_{Tx_i} and get $gsigshare_i$, for $i = 1,2,\cdots,n$.

→ Compute $gsig = \sum_{i=1}^{n} \prod_{j \neq i} \frac{h_j}{h_j - h_i} gsigshare_i$, where $h_s = Hash(pk_s)$, for $s = 1,2,\cdots,n$.

→ Scan all the transactions $DKG1_{Tx_i}$to get $c_{j,i}$ for $i = 1,2,\cdots,n, j = 1,2,\cdots,n$.

→ Compute $gpk = \sum_{i=1}^{n} (\prod_{j \neq i} \frac{h_j}{h_j - h_i}) (\sum_{s=1}^{n} c_{s,i})$, where $h_s = Hash(pk_s)$, for $s = 1,2,\cdots,n$.

→ Check whether $e(gsig, \bar{G}) = e(M \cdot G, gpk)$, if not, pause.

→ Else compute $\varsigma_r = Hash(gsig)$.

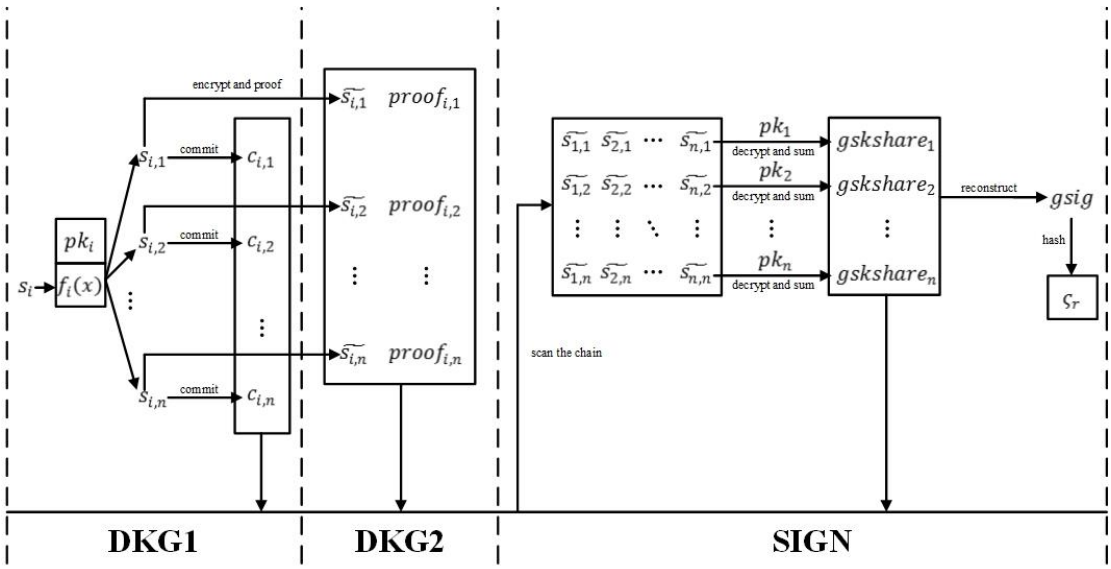


Figure2: Random Beacon working flow

2.5 Epoch Leaders Selection

At the beginning of each epoch, we select the Epoch Leaders of the next epoch according to the participants' stake distribution k blocks before. Actually, as stated above we select the group by the probability determined by the participant's stake. We implement follow-the-stake-rate to simulate the selection process, which is just like

follow-the-satoshi.

We calculate the stake ratio corresponding to each protocol participant in the Community, the total stake ratio in sum will be 1. Then we construct a binary search tree whose leaf nodes are keyed by (a hash of) their public key, and the value of each tree node is the sum of stake ratio in its subtree. If r is the random number output by the Random Beacon, we calculate $cx_i \equiv hash^i$ where N is the number of Epoch Leader group members and $hash^i()$ denotes the ith repeated hash execution. We invoke the ith selection by starting to traverse from the root, branching to the first child if its value is greater than cx_i, otherwise branching to the second child and update cx_i by minus the first child's value until it reaches the leaf node. After N executions, the Epoch Leader group is selected. We emphasize that the group is a multiset which means that a protocol participant may be selected more than once. Meanwhile, the Random Number Proposer Group is selected in this way too, except for $cx_i \equiv hash^i$.

We select the Epoch Leaders according to the participant's stake ratio, and we will select the unique Slot Leader by equal probability. It is important to prove that the probability of a participant to be selected as a Slot Leader is the same in this two-phase selection as with the direct selection. We give the proof in Appendix 3.

2.6 Unique Slot Leader Selection

After Epoch Leader selection, the selected participants need to generate a secret message in this epoch to prepare for the unique Slot Leader selection of the next epoch. Assume $epochleaders = \{P_1, P_2, \dots, P_N\}$, and their key pairs are denoted by $\{(pk_1, sk_1), (pk_2, sk_2), \dots, (pk_N, sk_N)\}$, The 2 stages in the secret message generation are as follows:

Stage 1

In this stage, every participant performs as follows (take P_i as an example):

→ Randomly select $\alpha_i \in (1, q)$.

→ Compute $M_i = \alpha_i \cdot pk_i$.

→ P_i send a special transaction $Stage1_{Tx_i}$ with the payload M_i.

Remarks: This commitment guarantees that the random number generated once could not be changed again

Stage 2

In this stage, every participant performs as follows (take P_i as an example):

→ Compute $\alpha_i \cdot pk_1$, for $i = 1,2,\cdots,N$.

→ Construct $A_i = (\alpha_i \cdot pk_1, \alpha_i \cdot pk_2, \dots, \alpha_i \cdot pk_N)$.

→ Compute $\pi_i = DLEQ(pk_1, \alpha_i \cdot pk_1, pk_2, \alpha_i \cdot pk_2, \dots, pk_N, \alpha_i \cdot pk_N)$.

→ P_i send a special transaction $Stage2_{Tx_i}$ with the payload A_i and π_i

Remarks: The proof π_i here makes sure that α_i stays the same in the scalar multiplication of different public keys. Details are in Appendix 2.

Verification Logic for $Stage2_{Tx_i}$

→ the proof π is valid

→ the *i*th data of A_i is the same as M_i, e.g. $A_i[i] = M_i$.

Computation of the Common Secret Message

P_j performs as follows to get the common secret message:

→ Scan all the $Stage2_{Tx_i}$ to get $\alpha_i \cdot pk_j$, for $i = 1,2,\cdots,N$.

→ Construct $\overline{S_j} = (\alpha_1 \cdot pk_j, \alpha_2 \cdot pk_j, \dots, \alpha_N \cdot pk_j)$.

→ Compute $S = sk_j^{-1} \cdot \overline{S_j} = (sk_j^{-1} \cdot \alpha_1 \cdot pk_j, sk_j^{-1} \cdot \alpha_2 \cdot pk_j, \dots, sk_j^{-1} \cdot \alpha_N \cdot pk_j) = (\alpha_1 \cdot G, \alpha_2 \cdot G, \dots, \alpha_N \cdot G)$

Unique Leader Selection Algorithm

At the beginning of the next epoch, the Slot Leaders will be selected by S and the random number from the random beacon is generated as follows:

→ Sort all Epoch Leaders by $hash(r||pk_i)$, where r is the random output from the random beacon and pk_i is the public key of P_i. We set the Epoch Leader sequence as $sq = (P'_1, P'_2, \dots, P'_N)$, it is a multisequence.

→ Calculate a random number matrix $M = (sr_{ij})_{n \times N}$ where n is the length of an

epoch and

$$sr_{ij} = hash^{j}(RB||epochID||i) mod N$$

$$cr_i = hash\left(\sum_{j=1}^{N} \alpha_{sr_{ij}} \cdot G\right), i = 1,2,\dots,n$$

→ Compute $cs_i = cr_i \ mod \ N$

→ Then the Slot Leader of sl_i is P'_{cs_i+1}.

Proposing a Block

When proposing a block, the Slot Leader P'_{cs_t+1}need to attach extra data to make the leader selection process publicly verifiable:

→ Compute $G_t = \sum_{j=1}^{N} \alpha_{sr_{tj}} \cdot G$

→ Compute $\pi' = DLEQ\left(G, pk_{cs_t+1}, G_t, \sum_{j=1}^{N} \alpha_{sr_{tj}} \cdot pk_{cs_t+1}\right)$

→ Attach G_t,π' to the proposed block.

Verification Logic for Block Leader

When receiving $block_t$, public entities have to be able to verify the correctness of the Slot Leader:

→ Compute $sr_{tj} = hash^{j}(RB||epochID||i) mod N, j = 1,2,\dots,N$

→ Scan the chain to compute $skGt = \sum_{j=1}^{N} \alpha_{sr_{tj}} \cdot pk_{cs_t+1}$.

→ Verify π' is valid with input of $\left(G, pk_{cs_t+1}, G_t, skGt\right)$.

→ Compute $cr_t{}' = hash(G_t)$.

→ Compute $cs_t{}' = cr_t{}' mod N$

→ Check whether $P'_{cs_t{}'+1}$ match the Slot Leader which proposed $block_t$.

Thus, the unique Slot Leader is selected from inside the Epoch Leaders and can be verified publicly.

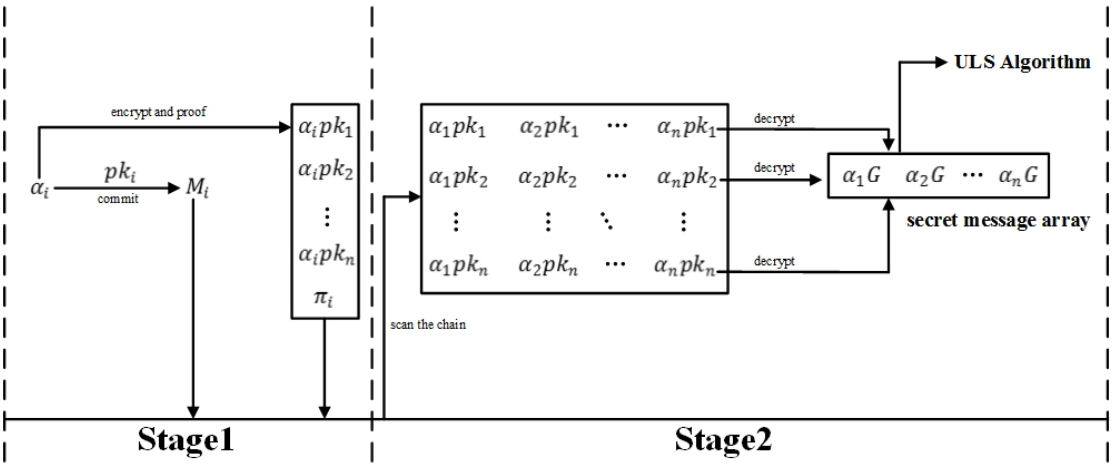


Figure3: Epoch Leader working flow

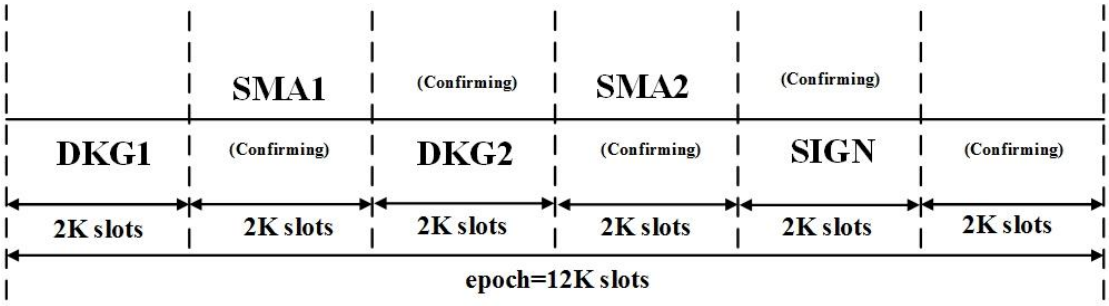


Figure4: Stage order in an epoch

3 Delegation Mechanism

3.1 Design Background

This section describes the design of the delegation mechanism in Galaxy consensus. WAN is distributed widely in different accounts with various amounts. Theoretically anyone holding WAN has the right to participate in the consensus process (i.e. proof of stake). Since the rewards of participating in consensus are proportional to the amount of WAN staked, for those holding a low number of WAN, there is little motivation to take part in consensus because the cost of being a consensus participant (being online all the time, listening to the network, saving chain data, etc.) outweighs the rewards. Thus, it is desirable to have a scheme which ensures that any WAN holder can join in consensus and benefit from it regardless of how many WAN they hold. Our solution is a delegation scheme based on proxy signature. Under this scheme, more WAN holders

will join in the consensus and thus the network will be more strong and secure.

Proxy signature is a practical method for delegation scheme design. A proxy signature protocol allows an entity, called the designator or original signer, to delegate another entity, called a proxy signer, to sign messages on its behalf, in case of say, temporal absence, lack of time or computational power, etc. The delegated proxy signer can compute a proxy signature that can be verified by anyone with access to the original signer's certified public key. Strictly, a proxy signature is a tuple $PS = (G, K, S, V, (D, P), PS, PV, ID)$:

(G, K, S, V) is a digital signature scheme.

- (D, P) is a pair of randomized algorithms forming the proxy-delegation protocol. D takes input the secret key sk_i of the designator i, the identity j of the proxy signer, and a message space descriptor ω for which user i wants to delegate its signing rights to user j, and outputs a cert. P takes input the cert and secret key sk_j of the proxy signer and outputs proxy signing key skp, which j uses to produce proxy signatures on behalf of user i.
- PS is the proxy signing algorithm. It takes as input a proxy signing key skp and a message M and outputs a proxy signature $p\sigma$.
- PV is the proxy verification algorithm. It takes as input a public key pk, a message M and a proxy signature $p\sigma$, and outputs 0 or 1. In the latter case, we say that the proxy signature is valid for M relative for pk.
- ID is the proxy identification algorithm. It takes a valid proxy signature $p\sigma$ and outputs an identity i.

Traditional delegation schemes operate like a mining pool, which requires the delegator to send its tokens to the proxy agent. After collecting all these tokens, the proxy agent participates in the consensus process and gets a reward which will be spilt among the delegators according to amounts of their delegated tokens. This scheme is centralized and insecure, since the proxy agent holds all the tokens and may possibly steal the tokens Proxy signature is naturally suitable for delegation schemes. Anyone may use proxy signature to delegate a trusted party to participate in the consensus on their behalf. The tokens are still in the designator's pocket, and only the signing right is given out. This is much more secure.

3.2 Triple ECDSA Proxy Signature Scheme

In this part, we will introduce our proxy signature scheme which is based on ECDSA. We call it triple ECDSA proxy signature scheme since it uses ECDSA signatures for

standard signing, proxy designation, and proxy signing. Assume the original signer is Alice with key pair (pk_i, sk_i), proxy signer is Bob with key pair (pk_j, sk_j). (G, K, S, V) is the standard ECDSA digital signature scheme. So, we focus on (D, P), PS, PV.

Algorithm D

Alice performs the following operations to delegate the signing right to Bob:

→ Form a message space ω

→ Randomly choose $k \in (1, q)$

→ Compute $R = kG = (x_r, y_r)$

→ Compute $h = H(pk_i || pk_j || \omega)$

→ Set $r = x_r$

→ Compute $s = k^{-1}(h + r \times sk_i)$

→ Output $cert = (pk_i, pk_j, \omega, (R, s))$

Algorithm P

Bob performs the following operations to get proxy signing key:

→ Parse $cert = (pk_i, pk_j, \omega, (R, s))$

→ Compute $h = H(pk_i || pk_j || \omega)$

→ Compute $V(pk_i, h, (R, s))$, if outputs 0, $cert$ is invalid and pause

→ Else Compute $skp = s + r \times sk_j$

Algorithm PS

Bob performs the following operations to do proxy signing:

→ Message M

→ Randomly choose $k_p \in (1, q)$

→ Compute $R_p = k_p G = (x_p, y_p)$

→ Compute $h_p = H(M)$

→ Set $r_p = x_p$

→ Compute $s_p = {k_p}^{-1}\left(h_p + r_p \times skp\right)$

→ Output $proxy_{sig} = \left(M, \left(R_p, s_p\right), cert\right)$

Algorithm *PV*

The third party performs the following operations to do proxy verification：

→ Parse $proxy_{sig} = \left(M, \left(R_p, s_p\right), cert\right)$

→ Parse $cert = \left(pk_i, pk_j, \omega, (R, s)\right)$

→ Verify $M \in \omega$, if not, pause

→ Compute $h = H\left(pk_i || pk_j || \omega\right)$

→ Compute $V\left(pk_i, h, (R, s)\right)$, if outputs 0, $cert$ is invalid and pause

→ Else Compute $pkp = s \cdot G + r \cdot pk_j$, $R = (x_r, y_r)$, $r = x_r$

→ Compute $V\left(pkp, M, \left(R_p, s_p\right)\right)$, if outputs 1, $proxy_{sig}$ is valid

We now prove the correctness of the triple ECDSA proxy signature scheme:

Theorem: For any message space ω, message $M \in \omega$ and users i, j. User i delegates user j the signing rights for ω and user j proxy signs M, then we have

$$PV\left(PS(skp, M)\right) = 1$$

Proof: From the definition of (D, P), PS, PV we have the following equivalent conditions:

$$PV\left(PS(skp, M)\right) = 1$$

$$\Leftrightarrow V\left(pkp, M, \left(R_p, s_p\right)\right) = 1$$

$$\Leftrightarrow V\left(pkp, M, S(skp, M)\right) = 1$$

$$\Leftrightarrow pkp = skp \cdot G$$

$$\Leftrightarrow s \cdot G + r \cdot pk_j = \left(s + r \times sk_j\right) \cdot G$$

$$\Leftrightarrow pk_j = sk_j \cdot G$$

Observe that $pk_j = sk_j \cdot G$ □

3.3 Delegation Scheme

Wanchain's delegation scheme makes use of a trusted ledger based triple ECDSA proxy signature scheme which writes and reads data and also makes use of smart contract functionality. This delegation scheme is more efficient than the original triple ECDSA proxy signature scheme which requires an additional trusted communication channel.

Assume Alice with key pair (pk_i, sk_i), Bob with key pair $\left(pk_j, sk_j\right)$. Alice wants to devote a WAN to the consensus process for a locking time of t. She wants to delegate her signing rights to Bob to participate in the consensus process on her behalf. We design a smart contract, which is named Proxy_SC, to help complete this process. Proxy_SC makes some calculations and stores some critical data. The whole procedure is as follows:

Delegation scheme

Alice performs the following operations:

→ Set message space $\omega = \perp$

→ Randomly choose $k \in (1, q)$

→ Compute $R = kG = (x_r, y_r)$

→ Compute $h = H\left(pk_i || pk_j || \omega\right)$

→ Set $r = x_r$

→ Compute $s = k^{-1}(h + r \times sk_i)$

→ $cert = \left(pk_i, pk_j, \omega, (R, s)\right)$

→ Send a transaction tx to Proxy_SC with payload $(a, t, cert)$

After receiving tx, Proxy_SC will do the following verifications and computations:

→ Parse the payload $(a, t, cert)$

→ Parse the $cert = \left(pk_i, pk_j, \omega, (R, s)\right)$

→ Verify whether pk_i matches the address sending tx

→ Compute $h = H\left(pk_i || pk_j || \omega\right)$

→ Compute $V\left(pk_i, h, (R, s)\right)$, if outputs 0, $cert$ is invalid and break

→ Else Compute $pkp = s \cdot G + r \cdot pk_j$, $R = (x_r, y_r)$, $r = x_r$

→ Compute the deadline time $t_d = t_{now} + t$

→ Save the tuple $\left(pk_i, pk_j, \omega, (R, s), pkp, a, t, t_d\right)$ in the storage of Proxy_SC

→ Take a WAN out of Alice's account (i.e. account pk_i) and lock it for time t

Bob performs the following check and computation to get the proxy secret key:

→ Scan the storage of Proxy_SC to find the tuple that involves pk_j

→ Parse the tuple $\left(pk_i, pk_j, \omega, (R, s), pkp, a, t, t_d\right)$

→ If $t_{now} > t_d$, the deadline time has passed, break.

→ Else compute $skp = s + r \times sk_j$

After this time, a new member of our PoS community (pkp, a, t) is born and it will participate in the consensus process. When pkp is selected as a random number proposer or Epoch Leader, then Bob just behaves normally according to the consensus protocol using skp for signing.

After the deadline time t_d has passed, (pkp, a, t) is moved out of the PoS Community. The locked a WAN will be returned to Alice's account together with the reward which will be split between Alice and Bob proportionally.

3.4 Advantages

- **Compatibility**

Our delegation scheme uses ECDSA, which is the standard digital signature scheme used in blockchain protocols for standard signing, proxy designation, and proxy signing.

So, no extra digital signature schemes need to be introduced. This design has no conflict with the existing technical structure. Whether a member directly joins the PoS Community or joins through the delegation scheme, the verification logic for the member's proposed block and data remains the same.

- **Non-interactive**

The delegation process is non-interactive, so the original signer and proxy signer don't have to establish a channel to communicate with each other. This saves bandwidth and is more practical in the context of blockchain consensus.

- **More efficient**

There are existing ECDSA proxy signature schemes, the most famous of which is introduced in the paper "Design of Proxy signature in ECDSA" by Ming-Hsin Chang, I-Te Chen, and Ming-Te Chen [10]. We refer to this scheme using the author's initials, MIM. Our scheme's delegation process is similar to MIM's; however, our proxy verification process is the same as standard ECDSA, which is more efficient than MIM's. The proxy verification processes comparison is listed below.

Calculation Type	MIM's scheme	Our scheme
Finite Field Calculations	4	3
Scalar Multiplications on ECC	3	2
Point Addition on ECC	1	1
Point Comparison on ECC	1	1

- **Clear division of rewards**

When the original signer delegates its signing rights to the proxy signer, a new proxy public key pkp is generated, which is controlled by the proxy signer and the relationship with the original signer is saved in Proxy_SC. When pkp is selected as the random number proposer or block proposed by the PoS algorithm, the proxy will perform on the behalf of original signer. The benefits made during the consensus process will be split between the original signer and proxy signer. Even if an entity is delegated by several other entities, the division of rewards is still clear and spilt between the corresponding original signer and proxy signer.

- **Message space limited**

Our triple ECDSA proxy signature scheme uses message space to limit the signing space of the proxy signer. Even though we haven't used it (set it as empty) in our

delegation scheme, it still provides the potential for additional application. For example, we could delegate the signing right to different entities under different conditions and so on.

4 Incentive Mechanism

For the purpose of incentivizing the active participation in Galaxy consensus protocol for Wanchain, 10% (21 million) of the total supply of WAN (210 million) has been reserved as reward. Initially this supply will serve as the main incentive for consensus participation, but as Wanchain grows and there are more transactions on chain, this supplemental reward will become a smaller portion of the total reward, and transaction fees will come to serve as the main incentive.

4.1 Basic Principles

The incentive mechanism is created according to the following principles:

(1) The greater the work, the greater the reward.

(2) Passive and malicious participants are discouraged by penalties.

(3) Receipt of reward and withdrawal of locked stake is delayed to improve security.

(4) A fair and benign competitive environment is desired.

(5) Should ensure reasonable and relatively stable revenue.

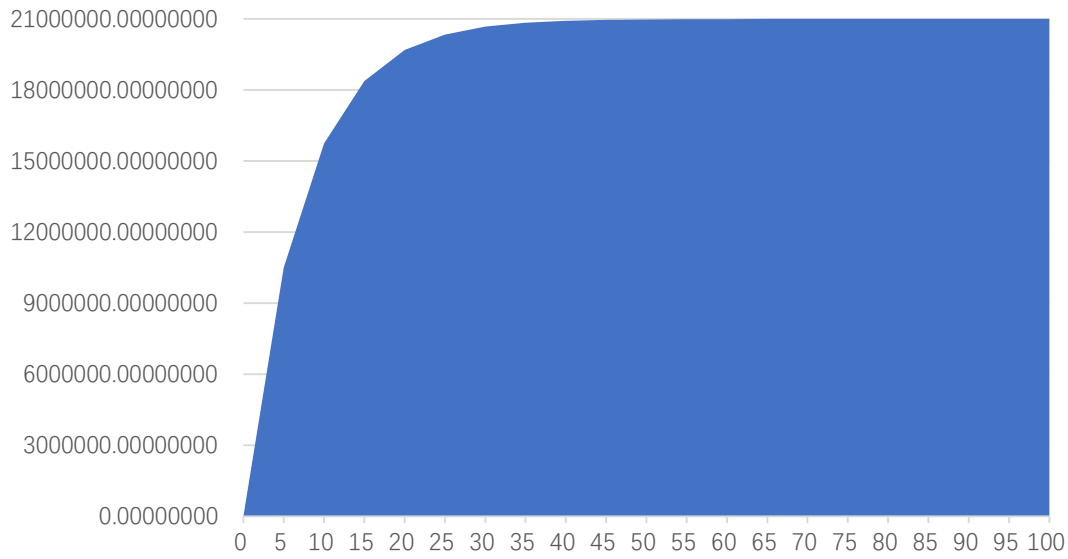
4.2 Incentive Model for Consensus

50% of the supplemental reward will be issued in the first five years, and the issuance of the reward will be cut in half every five years thereafter. Five years is a reasonable period after taking into consideration both Wanchain's growth and also long-term returns for consensus participants. The relation between the total supplemental reward and the year is:

$$R_t = a + a \cdot b + a \cdot b^2 + \cdots = \frac{a}{1-b}$$

Where R_t denotes the supplemental reward, a denotes the reward in the first interval, and b denotes the reduction ratio. In our design, R_t is 21 million, b is 50%, and a is 10.5 million, this should serve as an attractive incentive for consensus participants. The

following figure shows the supplemental reward allocation where the vertical axis denotes the total reward and the horizontal axis denotes the years.



Within each five year period, the supplemental reward will be allocated equally within each epoch. Taking the first five years as an example, if the total supplemental reward in this period is a and there are K epochs, the reward for each epoch R_e would be:

$$R_e = \frac{a}{K}$$

R_e will be given to the Random Number Proposers (RNPs) and Epoch Leaders who behave honestly. In contrast to POW, not all of the transaction fees for a block will be given to the block proposer. The transaction fee must also be distributed amongst all the RNPs because random number generation is a very important parameter for ensuring security. If only the supplemental reward is supplied, the RNPs will not be incentivized to participate. Since the supplemental reward is cut in half every 5 years, RNPs will have less and less motivation to participate. In this way, the reward for each epoch will be settled in the first block of the next epoch. This ensures consistency in the behavior of the participants.

Activeness Index

In Galaxy consensus, two types of participants make contributions in maintaining consensus. RNPs are in charge of generating random numbers to update the Random Beacon. Epoch Leaders are in charge of generating the secret message array and proposing blocks. Both of them need to be rewarded in order to incentivize honest behavior. Therefore, participants are encouraged to consistently participate in random number generation, secret message generation, and block proposal in the slot owned by

the Epoch Leader. An activeness index is thus defined so that participants will be rewarded in accordance with their activeness.

The RNP activeness index is defined as α_{RNP}. If an RNP participates in both of the two stages for random number generation, his α_{RNP} would be 1, and 0 otherwise. Similarly, the activeness index is denoted as α_{EL} for Epoch Leaders. If an Epoch Leader participates in both of the two stages for secret message array generation, his α_{EL} would be 1, and 0 otherwise. These two indexes will influence their reward from random number generation and secret message array generation.

Finally, the activeness index α_{ELG} is defined for the Epoch Leader Group. In our protocol, at the start of an epoch, the Slot Leader can only be known by the Epoch Leaders. After the block is proposed, the Slot Leader can be verified publicly. If some block is not proposed, only the Epoch Leaders know who is responsible for its proposal. This information is not publicly available since lot of work must be done to make it publicly verifiable, and the benefits of doing so are not worth the work necessary. So, the Epoch Leader Group activeness index is defined to solve the problem in a rational way. If an epoch includes K_e slots and only K_v blocks are proposed validly in this epoch, the α_{ELG} of the Epoch Leaders in charge of this epoch would be

$$\alpha_{ELG} = \frac{K_v}{K_e}$$

This index will influence their reward from proposing blocks.

Reward Division

As for the specific allocation in each epoch, the total reward must be divided amongst the RNPs and Epoch Leaders. An Epoch Leader is required to participate in the unique Slot Leader selection stages and propose a block if selected. An RNP is required to participate in the DKG stage and Signing stage. So λ of the total reward for an epoch is given to Epoch Leaders and $(1-\lambda)$ for RNPs. $(1-\mu)$ of the Epoch Leaders' reward will be allocated to incentivize their contributions for generating the secret message array.

Thus, if there are N Epoch Leaders and K_e slots in each epoch, the reward for a block will be:

$$R_s = \mu \cdot (\lambda \cdot \frac{R_e}{K_e} + \lambda \cdot T_s)$$

Where R_e denotes the reward for the epoch and T_s denotes the transaction fee of the block.

An Epoch Leader in charge of proposing blocks in this epoch will get reward R_p

$$R_p = \alpha_{ELG} \cdot \sum R_s$$

Where α_{ELG} denotes the Epoch Leader Group activeness index and $\sum R_s$ denotes the reward corresponding to the blocks proposed by the Epoch Leader.

The reward for the Epoch Leader who participates in secret message array generation in the previous epoch will be

$$R_c = \alpha_{ELG} \cdot (1 - \mu) \cdot (\lambda \cdot \frac{R_e}{K_e} + \lambda \cdot T_s)$$

Although $(1 - \mu)$ may be as low as 10%, it is necessary to set it to encourage the Epoch Leader to be active in secret message array generation, and security is improved as more Epoch Leaders participate in secret message array generation.

The reward for the RNP who participants in random number generation will be

$$R_r = \alpha_{RNP} \cdot \frac{(\,(1 - \lambda) \cdot R_e + \,(1 - \lambda) \cdot \sum T_s)}{N_r}$$

Where α_{RNP} denotes the RNP activeness index, N_r is the number of RNPs and $\sum T_s$ denotes the total transaction fees of the epoch.

The total reward in ideal conditions (one slot for each block, total participation for random number and secret message array generation) would be $(R_e + \sum T_s)$. If any lazy behavior occurs, the reward would be less. The remaining reward will be put back to the pool to be used as reward for PoS.

4.3 Incentive Model for Delegation Mechanism

Since the delegation mechanism is necessary for the stakeholders holding small amounts of WAN, it is necessary to design a reasonable incentive model for it. If someone wants to be a delegate, they need to publish a profit margin m. That means if the delegator is selected to be an RNP or Epoch Leader with reward R after work, the delegate will get $R \cdot m$, and the delegator will get $R \cdot (1 - m)$. This simple scheme will lead delegates to all provide a similar profit margin through natural competition. Besides the max amount of stake a delegate could be delegated is proportional to its principal.

Another goal is the prevention of large staking pools similar to the large mining pools which have arisen in PoW systems such as Bitcoin, as this tends to give rise to centralization. The model should promote the formation of fair staking pools. For this purpose, a ceiling number s_0 is set for the total stake of a single delegate (for example

10% of all currency in circulation). If a delegate's stake exceeds s_0, the reward will be reduced.

If the delegate has total stake s, the reward for the delegator is:

$$R_o = \begin{cases} R \cdot (1-m), s \leq s_0 \\ R \cdot (1-m) \cdot \left(1 - \dfrac{(s-s_0)^2}{s_0^2}\right), s_0 < s \leq 2s_0 \\ 0, s > 2s_0 \end{cases}$$

And the reward for the delegate is:

$$R_d = \begin{cases} R \cdot m, s \leq s_0 \\ R \cdot m \cdot \left(1 - \dfrac{(s-s_0)^2}{s_0^2}\right), s_0 < s \leq 2s_0 \\ 0, s > 2s_0 \end{cases}$$

A ranked list of all potential delegates will be provided which includes their profit margin and current stake, and all users can make the optimal choice. Since any delegation which goes beyond the ceiling s_0 for a certain delegate will lead to a decrease in reward for the both the delegate and delegator, the system naturally incentivizes stake to be distributed in a more decentralized manner. This reduces the risk of conspiracy and incentivizes more active users in the Galaxy consensus ecosystem.

5 Mitigation of Potential Attacks

Double spending attacks – In a double spending attack, the adversary wishes to make two conflicting transactions both appear valid. There are two conditions which must occur for this attack: (i) a block with two conflicting transactions or a transaction in conflict with a previous valid transaction, (ii) two valid chain forks including conflicting transactions. Regarding (i), the protocol requires that any block received must be checked for whether it contains any conflicting transactions. Our chain-based protocol indicates that only the longest chain is valid. So, no two forks can be accepted as both valid, thus preventing double spending.

Grinding attacks – In grinding attacks, the adversary wishes to influence the leader selection process to improve their chances of being Slot Leader. In our protocol, leader selection is based on random beacon and two-phase unique Slot Leader selection. In the random beacon process, we use a threshold signature scheme that means any participant can only determine their own signature share. It is impossible for the attacks to determine the final result as long as more than one participant is honest in the random number generation, and a number of participants less than the threshold number cannot predict the final result either. So, the Epoch Leader selection cannot be influenced by

the random number generation. In the secret message array generation in the two-phase unique Slot Leader selection, the selected participants can only choose to broadcast their information or not. If they don't, they will lose some reward. Additionally, the secret message generation is in front of SIGN stage and the Epoch Leaders are sorted after SIGN, the adversary has no message advantage to influence the unique Slot Leader selection.

Transaction denial attacks – In transaction denial attacks, the adversary wishes to prevent a certain transaction from being confirmed. We emphasize that honest participants would not deny any specific transactions. So, if transaction denial attacks happen it means the Slot Leaders are all malicious. Because of the honest majority assumption and *follow-the-stake-rate* selection, the probability of these attacks reduces exponentially with a base of less than $1/2$. The possibility for an adversary to perform this type of attack comes close to zero.

Bribery attacks – In bribery attacks, the adversary wishes to corrupt the honest participant to work for them in some bad purpose, such as double spending attacks. In our protocol, a rational participant will reject a bribe for two reasons. First, if an honest participant accepts the bribe and turns malicious, he will be punished. Second this malicious behavior will hurt the Wanchain ecosystem and reduces the value of WAN, making the participant's tokens lose value. Thus, no rational actor will commit a bribery attack. As long as the honest majority holds, bribery attacks cannot violate the security of our protocol.

Long-range attacks – In long-range attacks[12], the adversary wishes to reconstruct a chain from a position long before. Then he can make the chain data different from its true state, for example, to double spend. In our protocol, we design a mechanism to set some check points on the valid chain, which means as the chain grows, the past data before the last check point cannot be changed and new blocks before the last check point will not be accepted. So, it is impossible for an adversary to succeed in long-range attacks.

Nothing at stake attacks – In nothing at stake attacks, the participant will generate new blocks in multiple forks from which he could definitely benefit no matter which fork becomes valid. This tends to happen in PoS protocols, because it costs almost nothing to generate an additional block, while in PoW consensus, the miners will not sacrifice computation resources to follow different forks which may get no reward at all. Meanwhile this usually happens when there are more than one valid proposer of a slot or position, just like leader selection by VRFs. In our protocol, there is only one unique Slot Leader of a slot and there are almost no forks. That means there is no motivation to perform nothing at stake attacks.

Past majority attacks – In past majority attacks, the adversary wishes to corrupt some previous participants to take an advantage of stake in some past time. It is reasonable in our assumptions that presently the honest stake majority holds. In order to benefit from past majority attacks, the adversary needs to cooperate with previous majority stake holders to rework new blocks to replace blocks in the past majority epoch. This, similar to long-range attacks described above, will conflict with the last checkpoint. These kinds of newly generated blocks will not be accepted due to the checkpoint, so our protocol is not susceptible to past majority attacks.

Selfish-mining – In selfish-mining, a participant would keep a new valid block in private while constructing the next block in advance. This usually happens in PoW, because of the advantage of computation time. However, in PoS, there is no need to do so, especially in our protocol. Since the Slot Leaders are determined at the beginning of an epoch by the random beacon and the secret message is generated by Epoch Leaders, no matter whether the participant broadcasts the new block or not, it cannot influence the selection of the next slot. So, he cannot benefit from keeping the new block in private. In fact, he may lose rewards by broadcasting the new valid block out of the slot window. It is not rational to perform selfish-mining, and in the case it does occur, cannot influence the security of our protocol.

6 Advantages of Galaxy

Provable security

Galaxy consensus is based on the Ouroboros consensus model which is provably secure. It retains Ouroboros's original consensus backbone while making improvements to the core cryptographic components.

Low probability of natural forking

Galaxy consensus uses a ULS (unique leader selection) algorithm to determine the block proposer. In contrast with VRF, ULS ensures there is a unique proposer for each block. So, it achieves both anonymity of block proposers and low probability of natural forking.

Secure introduction of randomness

Introduction of randomness has a significant impact on the security of consensus. Galaxy consensus introduces randomness using a random beacon, which is based on a threshold signature scheme. Our random beacon is secure in two ways. First, it remains secure as long as no less than one of the random beacon participants is honest. Thus, it reduces the reliance on the honest majority assumption. Second, it ensures G.O.D (Guaranteed Output Delivery). This ensures that even if several participants are offline,

the random beacon will not halt. It functions normally as long as the number of online participants exceeds the predefined threshold.

Rational stake design

A rational stake design is a significant consideration in a PoS protocol. In order to keep the participants live and active, we allow WAN holders to lock their WAN in a special smart contract to join in Galaxy consensus. The amount of WAN, locking time, and remaining time of the locking period are used as parameters to calculate participants' stake score. This design simulates coin age in account model and ensures the stability of the consensus participants.

Robust delegation mechanism

Galaxy consensus is a PoS protocol with robust delegation mechanism. The key technology behind the delegation mechanism is our newly proposed ECDSA proxy signature algorithm, which is compatible, non-interactive, more efficient and message limited. This full delegation mechanism ensures that any WAN holder can participate in the consensus and improves the activeness of Wanstake.

Clear and convincing incentive model

Galaxy consensus has a robust incentive model for consensus participants. Since participants use the blockchain as a broadcast channel to exchange information, all their behavior is reflected on the chain. We introduce the concept of an activeness index to evaluate participants' performance. The more active, the more reward received. This incentive mechanism is clear and convincing.

Reference

1. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, CRYPTO 2017, Part I, volume 10401 of LNCS, pages 357–388. Springer, Heidelberg, August 2017.
2. Ittai Abraham, Dahlia Malkhi, Kartik Nayak, and Ling Ren. Dfinity Consensus, Explored. Cryptology ePrint Archive, Report 2018/1153, 2018. https://eprint.iacr.org/2018/1153.pdf.
3. Iddo Bentov, Rafael Pass, and Elaine Shi. The sleepy model of consensus. IACR Cryptology ePrint Archive, 2016:918, 2016.
4. J.H. Silverman, “The Arithmetic of Elliptic Curves,” Graduate Texts in Mathematics, vol. 106, Springer-Verlag, 1986.
5. Shamir A. How to share a secret. Communications of the ACM, 1979, 24(11): 612~613
6. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In Proceeding 26th Annual Symposium on the Foundations of Computer Science, IEEE, 1985:383~395.
7. Ronald L. Rivest, Adi Shamir, and Yael Tauman, How to Leak a Secret: Theory and Applications of Ring Signatures, Springer Berlin Heidelberg , 2006, 22(11):164-186.
8. Robert J. McEliece and Dilip V. Sarwate. On sharing secrets and reed-solomon codes. Commun. ACM, 24(9):583–584, 1981.
9. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, CRYPTO’92, volume 740 of LNCS, pages 89–105. Springer, Heidelberg, August 1993.
10. Chang, Ming Hsin , I. T. Chen , and M. T. Chen . " [IEEE 2008 Eighth International Conference on Intelligent Systems Design and Applications (ISDA) - Kaohsuing, Taiwan (2008.11.26-2008.11.28)] 2008 Eighth International Conference on Intelligent Systems Design and Applications - Design of Proxy Signature in ECDSA." (2008):17-22.
11. R. Gennaro, S.Jarecki, H. Krawczyk, and T. Rabin. Advances in Cryptology — EUROCRYPT ’99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings,

chapter Secure Distributed Key Generation for Discrete-Log Based Crypto systems, pages 295–310. SpringerBerlinHeidelberg,Berlin,Heidelberg,1999.

12. Vitalik Buterin. Long-range attacks: The serious problem with adaptive proof of work. https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-withadaptive-proof-of-work/, 2014.

13. M. Mambo, K. Usuda, and E. Okamoto, “Proxy signatures: Delegation of the power to sign messages”, IEICE Trans. Fundamentals, vol. E79-A, no.9, pp.13381354,1996

14. Kim S, Park S, Won D. Proxy signatures, Revisited[C], International Conference on Information & Communication Security. 1997.

Appendix 1

In Section 1, we list 4 properties which should be satisfied in the function to calculate the amount of Wanstake of consensus participants. A candidate function would be

$$H(\omega, L, t) = \omega\sigma_L e^{-t}$$

Where σ_L is an increasing function of L. It satisfies the first 3 properties obviously. We need to prove that it also satisfy the property 4.

$$\int_{t\prime=0}^{L_1+L_2} H(\omega, L_1+L_2, t)dt' > \int_{t\prime=0}^{L_1} H(\omega, L_1, t)dt' + \int_{t\prime=0}^{L_2} H(\omega, L_2, t)dt'$$

In property 4, t' here is the total elapsed time during the locking period and $t = \frac{L-t\prime}{L}$.

We calculate the integral $\int_{t\prime=0}^{L_2} H(\omega, L_2, t)dt'$ as an example, and transfer the ratio of remaining locking time to elapsed time below.

$$\int_{t\prime=0}^{L_2} H(\omega, L_2, t)dt' = \int_{t\prime=0}^{L_2} \omega\sigma_{L_2} e^{-\left(\frac{L_2-t\prime}{L_2}\right)}dt' = \omega\sigma_{L_2}\int_{t\prime=0}^{L_2} e^{\frac{t\prime-L_2}{L_2}}dt'$$

$$= \omega\sigma_{L_2}L_2\left(e^{\frac{L_2-L_2}{L_2}} - e^{\frac{0-L_2}{L_2}}\right) = \omega\sigma_{L_2}L_2(1-e^{-1})$$

Then the integral concave should be

$$\omega\sigma_{L_1+L_2}(L_1+L_2)(1-e^{-1}) > \omega\sigma_{L_1}L_1(1-e^{-1}) + \omega\sigma_{L_2}L_2(1-e^{-1})$$

$$\sigma_{L_1+L_2}(L_1+L_2) > \sigma_{L_1}L_1 + \sigma_{L_2}L_2$$

$$(\sigma_{L_1+L_2} - \sigma_{L_1})L_1 + (\sigma_{L_1+L_2} - \sigma_{L_2})L_2 > 0$$

Here σ_L is an increasing function of L. So the property 4 is satisfied.

Actually we want a participant to choose one longer participation period rather than two shorter participation periods. The integral of H function is the accumulative effect of stake which represents the reward of a participant in a sense. Thus the property 4 implies that a participant who chooses one longer participation period rather than two shorter participation periods will get more reward.

Appendix 2

In cryptography, the proof is an important primary to ensure coherence which means that a prover cannot convince a verifier of a fake statement. Our proof scheme is zero-knowledge, similar to that in [9], while we make it in elliptic curve. Its correctness and security is based on the DDH assumption on elliptic curve.

First we consider an array of points with length of $2N$ where $N \geq 2$, and the array is denoted by $pa = (P_1, Q_1, \dots, P_N, Q_N)$ where $P_i, Q_i \in E(F_p)$, $\#E(F_p) = n$. The proof guarantees that there is a value α such that $Q_i = \alpha \cdot P_i, i = 1,2, \dots, N$. The proof is constructed as follows:

- Generate random number $\omega \in [1, n]$ and calculate

$$\overline{P_i} = \omega \cdot P_i, i = 1,2, \dots, N$$

$$e = hash(P_1, Q_1, \dots, P_N, Q_N, \overline{P_1}, \dots, \overline{P_N})$$

- Then calculate

$$z = \omega - \alpha \cdot e \ mod \ n$$

The proof is $\pi = DLEQ(P_1, Q_1, \dots, P_N, Q_N) = (e, z)$.

The verification of the proof is as follows:

- With $(P_1, Q_1, \dots, P_N, Q_N)$ and (e, z), we calculate

$$P'_i = z \cdot P_i + e \cdot Q_i, i = 1,2, \dots, N$$

- Then calculate

$$e' = hash(P_1, Q_1, \dots, P_N, Q_N, P'_1, \dots, P'_N)$$

If $e = e'$, the proof is valid. Easy to prove

$$P'_i = z \cdot P_i + e \cdot Q_i = P'_i = (\omega - \alpha \cdot e) \cdot P_i + e \cdot Q_i = \omega \cdot P_i = \overline{P_i}$$

The generation and verification of the proof is simple, so we denote it as Gen_proof and Ver_proof in the algorithm and protocol description in the paper above.

Appendix 3

We describe the Epoch Leaders selection in Section 3.5. It is necessary to illustrate that the probability of a protocol participant to be a Slot Leader is the same in this two-phase selection and selection directly.

We define the scenario first. There are n protocol participants in the Community and the probability for the participant U_i to be selected as a Slot Leader is p_i. The number of the Epoch Leaders is N, $N < n$. It follows that p_i is the probability for U_i to be selected directly. Then we compute the probability in the protocol two-phase selection and prove the equivalency. As emphasized in Section 3.5, the Epoch Leaders is a multiset and the second phase selection is of equal probability. Then,

$$P(U_i) = C_N^1 \cdot p_i \cdot (1-p_i)^{N-1} \cdot \frac{1}{N} + C_N^2 \cdot {p_i}^2 \cdot (1-p_i)^{N-2} \cdot \frac{2}{N} + \cdots + C_N^N \cdot {p_i}^N \cdot \frac{N}{N}$$

$$\sum_{j=1}^{N} C_N^j \cdot {p_i}^j \cdot (1-p_i)^{N-j} \cdot \frac{j}{N}$$

Here we know

$$\begin{aligned} C_N^j \cdot {p_i}^j \cdot (1-p_i)^{N-j} \cdot \frac{j}{N} &= \frac{N!}{j! \cdot (N-j)!} \cdot {p_i}^j \cdot (1-p_i)^{N-j} \cdot \frac{j}{N} \\ &= \frac{(N-1)!}{(j-1)! \cdot (N-j)!} \cdot {p_i}^j \cdot (1-p_i)^{N-j} \\ &= p_i \cdot C_{N-1}^{j-1} \cdot {p_i}^{j-1} \cdot (1-p_i)^{N-j} \end{aligned}$$

Then

$$\begin{aligned} P(U_i) &= \sum_{j=1}^{N} p_i \cdot C_{N-1}^{j-1} \cdot {p_i}^{j-1} \cdot (1-p_i)^{N-j} = p_i \cdot \sum_{j=1}^{N} p_i \cdot C_{N-1}^{j-1} \cdot {p_i}^{j-1} \cdot (1-p_i)^{N-j} \\ &= p_i \cdot (p_i + 1 - p_i)^{N-1} = p_i \end{aligned}$$

Now we prove the equivalency of probability in two-phase leader selection and direct selection.