Monadring: A lightweight consensus protocol to offer Validation-as-a-Service to AVS nodes

Yu Zhang Xiao Yan Gang Tang Helena Wang

Verisense Network
team@verisense.network

July 26, 2024
Version 0.0.6

Abstract

Existing blockchain networks are often large-scale, requiring transactions to be synchronized across the entire network to reach consensus. On-chain computations can be prohibitively expensive, making many CPU-intensive computations infeasible. Inspired by the structure of IBM's token ring networks, we propose a lightweight consensus protocol called Monadring to address these issues. Monadring allows nodes within a large blockchain network to form smaller subnetworks, enabling faster and more cost-effective computations while maintaining the security guarantees of the main blockchain network.

To further enhance Monadring's security, we introduce a node rotation mechanism based on Verifiable Random Function (VRF) and blind voting using Fully Homomorphic Encryption (FHE) within the smaller subnetwork. Unlike the common voting-based election of validator nodes, Monadring leverages FHE to conceal voting information, eliminating the advantage of the last mover in the voting process.

This paper details the design and implementation of the Monadring protocol and evaluates its performance and feasibility through simulation experiments. Our research contributes to enhancing the practical utility of blockchain technology in large-scale application scenarios.

1 Introduction

Recent blockchain systems have adopted faster and more energy-efficient consensus protocols like Ouroboros [1], BABE [2], and Tendermint [3]. These

networks are typically open and permissionless, with their degree of decentralization defined by the number of participating nodes. Leveraging the aforementioned consensus protocols, new blockchain networks can be rapidly constructed. However, decentralized applications often cannot flexibly customize their degree of decentralization based on the importance of their underlying data.

For instance, decentralized social media applications may require faster response times and lower storage costs compared to decentralized finance applications, necessitating a lower degree of decentralization. This flexibility to adjust the level of decentralization based on the needs of the application is a critical consideration that existing blockchain architectures have not fully addressed.

Building a subnetwork over an existing blockchain system is a potential solution to this issue. Inspired by the token ring architecture, we propose a novel consensus protocol called Monadring. Monadring enables nodes within an existing blockchain network to form smaller, lightweight subnetworks that can perform computations more efficiently and cost-effectively while maintaining the same security guarantees as the main blockchain.

The token ring network operates at the data link layer of the OSI/RM model and was designed to solve the problem of physical link contention, similar to Ethernet. In a token ring network, a token is passed sequentially from one node to the next, granting the holder the right to transmit data. This token-based system for managing access to the shared medium bears some similarity to the consensus mechanisms used in blockchain networks for selecting block producers.

Despite token ring networks largely falling out of favor in modern networking due to their limited scalability and other drawbacks, the underlying principles of their decentralized, token-based structure offer valuable insights for designing a lightweight consensus algorithm for small blockchain systems.

Monadring also involves an FHE-based member rotation mechanism to enhance security. Homomorphic encryption (HE) [4] is a method of encryption that allows computations to be carried out on encrypted data, generating an encrypted result which, when decrypted, matches the outcome of computations performed on the plaintext. This property enables sophisticated computations on encrypted data while maintaining data security. HE schemes protect data privacy by allowing computations to be performed directly on encrypted data. For example, an HE scheme might allow a user to perform operations like addition and multiplication on encrypted numbers, with the same result as if they were performed on the original, unencrypted numbers. This technology is essential for secure cloud computing as it allows

complex data manipulations on completely secure encrypted data.

Fully Homomorphic Encryption (FHE) [5] is a more advanced form of Homomorphic Encryption. FHE allows arbitrary computations to be carried out on encrypted data, unlike normal HE, which may be limited in the types of computations it supports. FHE computations generate a result that, when decrypted, corresponds to the result of the same computations performed on the plaintext. This makes FHE extremely useful for scenarios where sensitive data must be processed or analyzed, but security and privacy considerations prevent the data from being decrypted. With FHE, unlimited calculations can be performed on encrypted data just as on unencrypted data. For instance, in cloud computing, FHE allows users to operate computations on encrypted data stored in the cloud, preserving data confidentiality and privacy.

In small-scale decentralized networks with randomly determined members, the application of FHE for voting can be mathematically equivalent to the Prisoner's Dilemma. By incorporating appropriate incentive structures, it is possible to achieve both efficiency and security in such decentralized networks.

1.1 Overall Framework

The framework is illustrated as Fig. 1. Section 2 demonstrates the mathematical model for transforming perfect information games in voting into imperfect information games through FHE, thereby eliminating the advantage of the last voter. Section 3 outlines the Monadring protocol. The topological structure of Monadring and Token Circulation are detailed in Section 3.1.2, while initialization and Node Rotation are explained in Section 3.2.1. Fault Tolerance mechanisms are discussed in Section 3.3. Section 2.2 covers the processes of Key Sharing and Key Resharing.

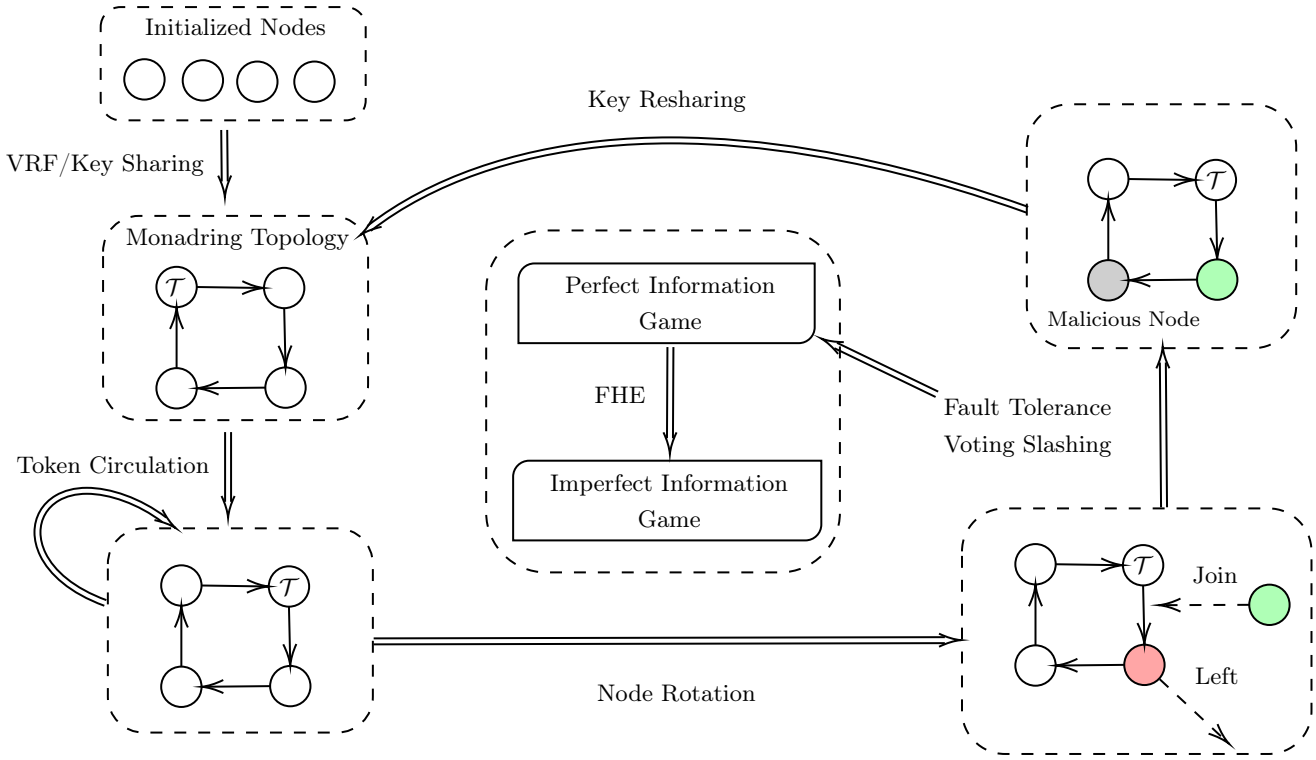

Figure 1: Illustration of Monadring Framework

2 Essential Mathematics

2.1 Fully Homomorphic Encryption (FHE)

We define the fundamental number sets: $\mathbb{Z}$ (integers), $\mathbb{Q}$ (rational numbers), $\mathbb{R}$ (real numbers), and $\mathbb{C}$ (complex numbers). The polynomial cyclotomic ring $\mathcal{R}$ is defined as:

$$\mathcal{R} := \mathbb{Z}[X]/(X^N + 1)$$

Correspondingly, $\mathcal{R}_q$ is defined as $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$, where typically $N = 2^n$ for some integer n.

We denote distributions as $\chi_{\sigma^2,\mu}$, where σ^2 is the variance and μ is the mean. The uniform ternary distribution and discrete Gaussian distribution are represented by $\chi_{\mathcal{T}}$ and $\chi_{\mathcal{N}}$ respectively. In the standard BFV encryption formula:

$$\mathbf{a} \cdot \mathbf{s} + \Delta m + \mathbf{e}$$

We have $\mathbf{a} \sim \chi_{\mathcal{T}}$ and $\mathbf{e} \sim \chi_{\mathcal{N}}$. The probability density function of the discrete Gaussian distribution is given by $\rho_{\mu,\sigma^2} = e^{-\|\mathbf{x}-\mu\|^2/2\sigma^2}$.

The expansion factor $\delta_{\mathcal{R}}$ is defined as $\|\mathbf{a} \cdot \mathbf{b}\|_\infty / (\|\mathbf{a}\|_\infty \cdot \|\mathbf{b}\|_\infty)$ where $\mathbf{a}, \mathbf{b} \in \mathcal{R}$.

As described in [6, 7], the BFV scheme places the ciphertext in the Most Significant Digit (MSD) position, changing the noise growth trend from quadratic to linear. This significantly reduces the impact of noise during computation and eliminates the need for modulus switching, thereby enhancing computational efficiency.

The BFV scheme consists of the following algorithms:

- `BFV.SecretKeyGen(`1^λ`)`: Generate a secret key $\mathtt{sk} \leftarrow \chi_{\mathcal{T}}$.
- `BFV.PublicKeyGen(sk)`: Generate a public key $\mathtt{pk} = (\mathtt{pk}_0, \mathtt{pk}_1) = ([-\mathbf{a} \cdot \mathbf{s} + \mathbf{e}]_q, \mathbf{a})$ where $\mathbf{e} \leftarrow \chi_{\mathcal{N}}$ and $\mathbf{a} \leftarrow \chi_{\mathcal{T}}$.
- `BFV.Enc(pk,` $\mathbf{m}$`)`: For message $\mathbf{m} \in \mathcal{R}_p$, where $p < q$, $\mathbf{u} \leftarrow \chi_{\mathcal{T}}$ and $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi_{\mathcal{N}}$. The ciphertext is:

$$\mathtt{ct} = (\mathtt{ct}_0, \mathtt{ct}_1) = ([\mathtt{pk}_0 \cdot \mathbf{u} + \Delta \cdot \mathbf{m} + \mathbf{e}_0]_q, [\mathtt{pk}_1 \cdot \mathbf{u} + \mathbf{e}_1]_q)$$

- `BFV.Dec(sk, ct)`:

$$\mathbf{m} = [\lfloor p \cdot [\mathtt{ct}_0 + \mathtt{ct}_1 \cdot \mathbf{s}]_q / q \rceil]_t$$

It's noteworthy that the BFV scheme differs from the BGV scheme [8] in ciphertext placement. In BGV, the ciphertext is placed in the Least Significant Digit (LSD):

- `BGV.PublicKeyGen(sk)`:

$$\mathtt{pk} = ([-\mathbf{a} \cdot s + p\mathbf{e}]_q, \mathbf{a})$$

- `BGV.Enc(pk,` $\mathbf{m}$`)`:

$$\mathtt{ct} = ([\mathbf{m} + \mathbf{u} \cdot \mathtt{pk}_0 + p\mathbf{e}_0]_q, [\mathbf{u} \cdot \mathtt{pk}_1 + p\mathbf{e}_1]_q)$$

To evaluate the effect of encryption and decryption on noise, we consider the equation:

$$\mathtt{ct}_0 + \mathtt{ct}_1 \cdot s = \Delta \mathbf{m} + p\mathbf{v} \tag{1}$$

The noise $\mathbf{v}$ can be bounded by $\|\mathbf{v}\|_\infty < (q - pr_t(q))/(2p)$ where $r_p(q) = q - p\Delta$, a value determined by the modulus q, the plaintext scaling factor Δ, and the noise ratio p.

This condition is crucial because the decryption process involves a p/q scaling operation, which may amplify the noise. Therefore, we must ensure that the noise magnitude remains within an acceptable range before decryption.

2.2 Threshold Key Sharing

2.2.1 Shamir's Secret Sharing Scheme

Shamir's Secret Sharing [9], devised by Adi Shamir, is a cryptographic algorithm that enables secure distribution of a secret among multiple participants. Its key feature is the minimal number of shares required to reconstruct the secret. The process for threshold private key sharing using Shamir's scheme is as follows:

1. **Threshold Selection:** Define a threshold t, where knowledge of fewer than t points reveals no information about the secret, but t or more points allow secret reconstruction.

2. **Polynomial Generation:** Create a random polynomial of degree $t-1$, with the constant term being the secret (private key) to be shared:

$$\mathcal{P}(\mathbf{x}) = a_0 + a_1\mathbf{x} + a_2\mathbf{x}^2 + \ldots + a_{t-1}\mathbf{x}^{t-1} \tag{2}$$

3. **Share Creation:** Evaluate the polynomial at n different points to create n shares, where n is the total number of participants. Each participant receives one share, a point on the polynomial:

$$s_i = \mathcal{P}(\mathbf{x}_i) \tag{3}$$

4. **Share Distribution:** Distribute the private key shares among the participants.

5. **Secret Reconstruction:** When the private key is needed, any t participants can combine their shares using polynomial interpolation (e.g., Lagrange interpolation) to reconstruct the polynomial and reveal the constant term (the secret):

$$\mathcal{P}(x) = \sum_{i=0}^{t-1} s_i \prod_{j \neq i} \frac{x - s_j}{s_i - s_j} \tag{4}$$

It's important to note that all polynomials are defined over the ring $(\mathbb{Z}/p\mathbb{Z})[X]/X^t$, and Lagrange interpolation remains valid in this context.

This approach ensures that the private key is never fully revealed to any single party, and no individual can access the secret alone. It provides a balance between accessibility and security, making it particularly useful for managing risks in cryptographic key management systems.

2.2.2 Resharing Scheme

In practical applications, it may be necessary to redistribute keys. This process, known as resharing [10], involves transforming an original (n, t) key sharing scheme into a (n', t') scheme, which can be achieved using Lagrange interpolation.

Assume an initial (n, t) key sharing of s, where $s = \mathcal{P}(0)$. Let $\mathcal{L}_i^{\mathcal{N}}$ be the Lagrange basis for the original (n, t) key sharing user set $\mathcal{N}$, and $\mathcal{M}$ be the user set for the new (n', t') sharing. The resharing process can be expressed as:

$$\begin{aligned} \mathcal{P}(0) &= \sum_{i \in \mathcal{N}} \mathcal{L}_i^{\mathcal{N}} s_i = \sum_{i \in \mathcal{N}} \mathcal{L}_i^{\mathcal{N}} \sum_{j \in \mathcal{M}} \mathcal{L}_j^{\mathcal{M}} s'_{i,j} \\ &= \sum_{j \in \mathcal{M}} \left(\sum_{i \in \mathcal{N}} \mathcal{L}_i^{\mathcal{N}} \mathcal{L}_j^{\mathcal{M}} \right) s'_{i,j} = \sum_{j \in \mathcal{M}} \mathcal{L}_j^{\mathcal{M}} \sum_{i \in \mathcal{N}} \mathcal{L}_i^{\mathcal{N}} s'_{i,j} = \sum_{j \in \mathcal{M}} \mathcal{L}_j^{\mathcal{M}} s'_j \end{aligned} \tag{5}$$

Mathematically, this resharing process is analogous to transforming a one-dimensional vector into a two-dimensional matrix of size $n \times n'$, where any $t \times t'$ submatrix can reconstruct the original key. This transformation effectively shifts users from holding row vectors to holding column vectors.

The resharing scheme provides a flexible method for adjusting share distribution without exposing the original secret or requiring all participants to return their shares. This adaptability is crucial for maintaining long-term security and accommodating changes in the participant group over time.

2.3 Game Theory

This chapter elucidates the design of a game-theory [11] based security framework to ensure system safety and reliability. We begin by introducing fundamental concepts of game theory, followed by a discussion on transforming perfect information games into imperfect information games using fully homomorphic encryption (FHE) techniques. Finally, we present a security framework based on Bayesian games and demonstrate how game theory principles can be applied to design a secure voting mechanism, thereby enhancing overall system security.

2.3.1 Basic Definitions

In game theory, we consider a set of players $\mathcal{P} = \{1, 2, \ldots, n\}$. Each player $i \in \mathcal{P}$ has a strategy space Σ_i, containing all possible strategies available to that player. A specific strategy chosen by player i is denoted as $\sigma_i \in \Sigma_i$.

In multiplayer games, the strategies of other players often influence a player's decision. We denote the strategy profile of all players except player i as $\sigma_{-i} \in \Sigma_{-i}$, where Σ_{-i} represents all possible strategy combinations for players other than i.

Games can be classified based on information availability:

- **Perfect Information Games:** All players have complete knowledge of all past actions and decisions within the game. The entire sequence of play is fully observable to every player.
- **Imperfect Information Games:** The complete sequence of previous actions and decisions is not fully observable for all players. Players may possess private information and typically have different information about past events. Decisions are made based on private and available public information.

2.3.2 Nash Equilibrium

The Nash equilibrium is a fundamental concept in game theory, representing a state where no player can gain an advantage by unilaterally changing their strategy, given that all other players' strategies remain fixed. [12]

The best response function of player i, denoted by $\mathtt{BR}_i(\sigma_{-i})$, is defined as:

$$\mathtt{BR}_i(\sigma_{-i}) = \arg\max_{\sigma_i \in \Sigma_i} \pi_i(\sigma_i, \sigma_{-i}) \tag{6}$$

where π_i is player i's utility or payoff function.

A strategy profile σ^* is a Nash equilibrium if for all players i, $\sigma_i^* = \texttt{BR}_i(\sigma_{-i}^*)$.

In imperfect information games, achieving Nash equilibrium can be challenging. To address this, we consider the concept of ϵ-equilibrium. A strategy profile σ^ϵ is an ϵ-equilibrium if for all players i:

$$\pi_i(\sigma_i^\epsilon, \sigma_{-i}^\epsilon) \geq \pi_i(\sigma_i, \sigma_{-i}^\epsilon) - \epsilon \tag{7}$$

This means that no player can gain more than ϵ by unilaterally deviating from their strategy.

2.3.3 Bayesian Game

A Bayesian game incorporates players' private information that affects their payoffs. Each player has a type $\theta_i \in \Theta_i$, drawn from a known probability distribution p_i, which determines their payoff function.

A Bayesian game is defined by the quintuple:

$$\Gamma = \langle \mathcal{P}, \{\Theta_i\}_{i\in\mathcal{P}}, \{\Sigma_i\}_{i\in\mathcal{P}}, \{\pi_i\}_{i\in\mathcal{P}}, \{p_i\}_{i\in\mathcal{P}} \rangle \tag{8}$$

The Bayesian Nash Equilibrium (BNE) extends the Nash equilibrium concept to Bayesian games. A strategy profile σ^* is a BNE if for all players i and all types θ_i, $\sigma_i^*(\theta_i)$ is a best response to the strategies of all other players, given θ_i.

The best response function in a Bayesian game is defined as:

$$\sigma_i^{\texttt{BNE}} = \texttt{BR}_i^{\texttt{BNE}}(\theta_{-i}, \sigma_{-i}) = \arg\max_{\substack{\sigma_i\in\Sigma_i \\ \theta_i\in\Theta_i}} \int_{\Theta_i} \pi_i(\sigma_i, \sigma_{-i}^{\texttt{BNE}}, \theta_i, \theta_{-i}) dP_i(\theta_{-i}|\theta_i) \tag{9}$$

where $P_i = \prod_{j\neq i} p_j(\theta_j)$.

In this paper, we focus on symmetric Bayesian games, where all players share the same type space and type distribution.

2.3.4 Transform Perfect Game to Imperfect Game with FHE

We propose a method to transform perfect information games into imperfect information games using Fully Homomorphic Encryption (FHE) techniques. The procedure is as follows:

1. **Key Generation:** Each player $i \in \{1, \ldots, n\}$ generates a key pair using the BFV scheme's key generation algorithms:

$$\mathtt{sk}_i \leftarrow \mathtt{BFV.SecretKeyGen}(\lambda) \tag{10}$$

$$\mathtt{pk}_i \leftarrow \mathtt{BFV.PublicKeyGen}(\mathtt{sk}_i) \tag{11}$$

   where λ is the security parameter.

2. **Strategy Encryption:** Each player encrypts their strategy σ_i using the BFV scheme described in Section 2.1:

$$\sigma_i^{\mathtt{Enc}} \leftarrow \mathtt{BFV.Enc}(\mathtt{pk}_i, \sigma_i) \tag{12}$$

3. **Encrypted Strategy Distribution:** Each player distributes their encrypted strategy $\sigma_i^{\mathtt{Enc}}$ to all other players.

4. **Homomorphic Payoff Computation:** Each player i computes their payoff function using homomorphic operations:

$$\pi_i^{\mathtt{Enc}} \leftarrow f_i(\sigma_1^{\mathtt{Enc}}, \ldots, \sigma_n^{\mathtt{Enc}}) \tag{13}$$

   where f_i is player i's payoff function, computed in the encrypted domain using homomorphic operations.

5. **Payoff Decryption and Proof Generation:** Each player i decrypts their payoff $\pi_i^{\mathtt{Enc}}$ and generates a zero-knowledge proof Π_i:

$$\pi_i \leftarrow \mathtt{BFV.Dec}(\mathtt{sk}_i, \pi_i^{\mathtt{Enc}}) \tag{14}$$

$$\Pi_i \leftarrow \mathtt{Zk.Prove}(\mathtt{sk}_i, \pi_i^{\mathtt{Enc}}, \pi_i, \mathtt{BFV.Dec}) \tag{15}$$

   Here, Π_i is a zero-knowledge proof demonstrating that the equation $\pi_i \stackrel{?}{=} \mathtt{BFV.Dec}(\pi_i^{\mathtt{Enc}})$ holds, without revealing any information about the private key $\mathtt{sk}_i$. This proof allows other players to verify the correctness of π_i while maintaining the confidentiality of $\mathtt{sk}_i$.

Each player can only observe the encrypted strategies $\sigma_i^{\mathtt{Enc}}$ of other players, without access to their actual strategies σ_i. Simultaneously, the zero-knowledge proof Π_i enables each player to demonstrate that they have honestly computed and reported their payoff π_i, without disclosing any information about their private key $\mathtt{sk}_i$. The workflow of process is described in Algorithm 1.

Algorithm 1 Transform Perfect Information Game to Imperfect Information Game with FHE

Require: Number of players n, Security parameter λ
Ensure: Encrypted strategies $\{\sigma_i^{\texttt{Enc}}\}_{i=1}^n$, Payoffs $\{\pi_i\}_{i=1}^n$, Proofs $\{\Pi_i\}_{i=1}^n$

for each player $i \in \{1, \ldots, n\}$ **in parallel do**
    $\texttt{sk}_i \leftarrow \texttt{BFV.SecretKeyGen}(\lambda)$
    $\texttt{pk}_i \leftarrow \texttt{BFV.PublicKeyGen}(\texttt{sk}_i)$
    $\sigma_i^{\texttt{Enc}} \leftarrow \texttt{BFV.Enc}(\texttt{pk}_i, \sigma_i)$
    Broadcast $\sigma_i^{\texttt{Enc}}$ to all other players
end for
Synchronization point: wait for all players to broadcast their encrypted strategies
for each player $i \in \{1, \ldots, n\}$ **in parallel do**
    Receive $\{\sigma_j^{\texttt{Enc}}\}_{j \neq i}$ from other players
    $\pi_i^{\texttt{Enc}} \leftarrow f_i(\sigma_1^{\texttt{Enc}}, \ldots, \sigma_n^{\texttt{Enc}})$ ▷ Homomorphic computation
    $\pi_i \leftarrow \texttt{BFV.Dec}(\texttt{sk}_i, \pi_i^{\texttt{Enc}})$
    $\Pi_i \leftarrow \texttt{Zk.Prove}(\texttt{sk}_i, \pi_i^{\texttt{Enc}}, \pi_i, \texttt{BFV.Dec})$
end for
return $\{\sigma_i^{\texttt{Enc}}\}_{i=1}^n$, $\{\pi_i\}_{i=1}^n$, $\{\Pi_i\}_{i=1}^n$

2.4 Voting Game Model

In this chapter, we discuss how to use game theory in Monadring to design voting mechanisms to ensure the normal operation of the subnet. Here, we analyze the mathematical principles of different voting mechanisms. The significance lies in the fact that each AVS needs to design different voting mechanisms for different subnets. For voters, they hope to maximize their voting benefits. For the subnet, it hopes to distribute as few rewards as possible while ensuring that the entire network consensus is reached as soon as possible. We first introduced some basic game theory concepts, then applied these concepts to the subnets in Monadring. For the case of perfect information games, we provided a theoretically optimal voting strategy. For the case of imperfect information games, voters will try to find the right strategy, that is, ξ, to maximize their expected payoffs when reaching the Nash equilibrium. However, AVS designers hope to find the corresponding θ to minimize the overall payoff.

2.4.1 Voting Game

Assume there are n voters, each with a voting weight of ω_i, casting a vote $v \in \{\top, \bot\}$. The consensus threshold percentage is θ, and their payoff function is:

$$b(\omega_i; \theta_\top, \theta_\bot, n) = \begin{cases} \beta(\omega_i), & \text{if } \frac{\#(v=v_i)}{n} > \theta_{v_i} \\ -\alpha(\omega_i), & \text{if } \frac{\#(v\neq v_i)}{n} > \theta_{\bar{v}_i} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

where n and θ refer to the number of participants and threshold percentage in this subgame, with $\theta_\top + \theta_\bot \geq 1$. We use N and θ_g to denote the global threshold and number of participants. The $\beta(\omega_i) \geq 0$ and $\alpha(\omega_i) \geq 0$ are the payoff functions for a voter with a weight of ω_i.

We introduce a deterministic information: when voter i votes, they know that there have already been $n^{(i)}\top$ votes for $\top$ and $n^{(i)}\bot$ votes for $\bot$. The threshold required for them in the subgame is:

$$\theta_\top^{(i)} = \begin{cases} 0, & \text{if } n_\top^{(i)} \geq \theta_\top \cdot n \\ \infty, & \text{if } n_\bot^{(i)} \geq \theta_\bot \cdot n \\ \frac{\theta_\top \cdot n - n_\top^{(i)}}{n - n^{(i)}}, & \text{otherwise} \end{cases} \tag{17}$$

Therefore, at any point in time, each voter is essentially facing a subgame with parameters $\theta^i\top$, $\theta^i_\bot$, and $n^{(i)}$. In this game, the voter is completely unaware of the voting situation of the remaining $n - n^{(i)} - 1$ individuals.

2.4.2 Perfect Information:$n^{(i)} = i - 1$

Let's consider a special case where $n^{(i)} = i - 1$, meaning that voter i knows the voting behavior of all previous voters. Assuming all voters are rational, when the last voter votes, they know the voting behavior of all previous voters, and they only need to choose a vote that will satisfy the voting result.

$$v_n^* = \begin{cases} \top, & \text{if } n_\top^n + 1 > \theta_\top \cdot n \\ \bot, & \text{if } n_\bot^n + 1 > \theta_\bot \cdot n \\ \mathtt{r}^{(n)}(\top, \bot) & \sim \mathcal{B}^{(n)}, \text{otherwise} \end{cases} \tag{18}$$

Here, v^* represents the rational voting result (not the actual vote), where $\mathcal{B}$ is the Bernoulli distribution. For the $n-1$ voters, if we assume they are rational, they would know the results of $n-2$ votes and v_n^*. Then, they can

vote similarly to Eqn. 18 based on these $n-1$ results. Therefore, for any i, it can be inferred based on $\{v_1, v_2, \ldots, v_{i-1}\} \cup \{v^*_{i+1}, v^*_{i+2}, \ldots, v^*_n\}$.

$$v^*_i = \begin{cases} \top, & \text{if } n^{(i)}_\top + 1 + \sum_{j>i} \mathbb{E}[v^*_j = \top] > \theta_\top \cdot n \\ \bot, & \text{if } n^{(i)}_\bot + 1 + \sum_{j>i} \mathbb{E}[v^*_j = \bot] > \theta_\bot \cdot n \\ \mathtt{r}^{(i)}(\top, \bot) & \sim \mathcal{B}^{(i)}(v; \theta_\top \cdot n - n^{(i)}_\top, \theta_\bot \cdot n - n^{(i)}_\bot, n - n^{(i)}) \end{cases} \tag{19}$$

If we assume that all other information for all voters is ideal, meaning there is no additional information that makes them more inclined to $\top$ or $\bot$, then v^* is no longer a random variable, but:

$$\mathtt{r}^{(i)} = \begin{cases} \top, & \text{if } \theta_\top \cdot n - n^{(i)}_\top < \theta_\top \cdot n - n^{(i)}_\bot \\ \bot, & \text{otherwise} \end{cases} \tag{20}$$

However, the real situation is not like the ideal one, especially for the voters who are closer to the front ($i < \mathtt{min}((1-\theta_\top) \cdot n + n^{(i)}_\top, (1-\theta_\bot) \cdot n + n^{(i)}_\bot)$). According to the Central Limit Theorem, the variance will be higher because they have to estimate the $\mathtt{r}$ of other voters.

2.4.3 Imperfect Information: $n^{(i)} = 0$

For the case of $n^{(i)} = 0$, where voter i is the first voter and does not know the voting behavior of other voters, their voting result should be a random variable. This situation degenerates into a Bayesian game. Assume that its distribution is $v^* \sim B(X; \xi)$, where B is the binomial distribution. For voter i, the expected payoff is:

$$\begin{aligned} &\mathbb{E}[b(\omega_i; \theta_\top, \theta_\bot, n) | v_i = \top] \\ =&\beta \cdot \sum_{i > \theta_\top \cdot n - 1} \binom{n-1}{i} \xi^i (1-\xi)^{n-1-i} - \alpha \cdot \sum_{i > \theta_\bot \cdot n} \binom{n}{i} \xi^{(n-i)} (1-\xi)^i \end{aligned} \tag{21}$$

$$\begin{aligned} &\mathbb{E}[b(\omega_i; \theta_\top, \theta_\bot, n) | v_i = \bot] \\ =&\beta \cdot \sum_{i > \theta_\bot \cdot n - 1} \binom{n-1}{i} \xi^{n-1-i} (1-\xi)^i - \alpha \cdot \sum_{i > \theta_\top \cdot n} \binom{n}{i} \xi^i (1-\xi)^{(n-i)} \end{aligned} \tag{22}$$

According to Bayes' theorem, the posteriori expectation is:

$$\begin{aligned} &\mathbb{E}[b(\omega_i; \theta_\top, \theta_\bot, n)] \\ =&\mathbb{E}[b(\omega_i; \theta_\top, \theta_\bot, n) | v_i = \bot] \cdot (1-\xi) + \mathbb{E}[b(\omega_i; \theta_\top, \theta_\bot, n) | v_i = \top] \cdot \xi \end{aligned} \tag{23}$$

We consider a Pareto optimal solution to Eqn. 23, maximising the posterior expectation:

$$\begin{aligned}
&\frac{\partial \mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)]}{\partial \xi} \\
=&\frac{\partial \mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\top]}{\partial \xi}\cdot\xi+\mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\top] \\
&+\frac{\partial \mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\bot]}{\partial \xi}\cdot(1-\xi)-\mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\bot] \\
&=0
\end{aligned} \tag{24}$$

Deriving Eqn. 21 and Eqn. 22 separately, we get:

$$\begin{aligned}
&\frac{\partial \mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\top]}{\partial \xi} \\
=&\beta\cdot(n-1)\binom{n-2}{\theta_\top\cdot n-1}\int_0^{1-p} t^{(n-1)-(\theta_\top\cdot n-1)-1}(1-t)^{(\theta_\top\cdot n-1)}\mathrm{d}t \\
&-\alpha\cdot n\cdot\binom{n-1}{\theta_\bot\cdot n}\int_0^{1-p} t^{n-(\theta_\bot\cdot n)-1}(1-t)^{\theta_\bot\cdot n} \\
=&-\beta\cdot(n-1)\cdot\binom{n-2}{n\cdot\theta_\top-1}\xi^{n\cdot\theta_\top-1}(1-\xi)^{n\cdot(1-\theta_\top)-1} \\
&+\alpha\cdot n\cdot\binom{n-1}{n\cdot\theta_\bot}\xi^{n\cdot\theta_\bot}(1-\xi)^{n\cdot(1-\theta_\bot)-1}
\end{aligned} \tag{25}$$

Similarly, we can find the partial derivative of $\mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\bot]$.

$$\begin{aligned}
&\frac{\partial \mathbb{E}[b(\omega_i;\theta_\top,\theta_\bot,n)|v_i=\bot]}{\partial \xi} \\
=&-\beta\cdot(n-1)\cdot\binom{n-2}{n\cdot\theta_\bot-1}\xi^{n\cdot\theta_\bot-1}(1-\xi)^{n\cdot(1-\theta_\bot)-1} \\
&+\alpha\cdot n\cdot\binom{n-1}{n\cdot\theta_\top}\xi^{n\cdot\theta_\top}(1-\xi)^{n\cdot(1-\theta_\top)-1}
\end{aligned} \tag{26}$$

Solve Eqn. 24 by using Eqn. 25 and Eqn. 26 to find the Nash equilibrium $\xi=\frac{\theta_\bot}{\theta_\top+\theta_\bot}$. With any parameters, $\xi=0$ and $\xi=1$ are the trivial Nash Equilibrium points. When $\alpha=\beta$,$\theta_\top=\theta_\bot$, the Nash equilibrium degenerates to $\xi=0.5$.

Evaluation of θ For the designers of voting mechanisms, their goal is to achieve network-wide consensus at the minimum cost. That is, we should

design θ, β, α in such a way that consensus can be effectively reached at the lowest cost (note that these three are not necessarily constants, they could very well be in the form of functions such as $\beta(\theta)$ and $\alpha(\theta)$). Here we only discuss the design of θ, while β and α need to be evaluated based on functional analysis, which we will publish in the form of a paper. We derivate the expectation of the payoff function with respect to $\theta_\top$ and $\theta_\perp$. According to Eqn. 21, We approximately estimate that the likelihood expectation of the derivative of $\theta_\top \cdot n$ at the point $\theta_\top \cdot n - 1$ is a Beta distribution. We use $\mathbb{E}_\top$ to abbreviate $\mathbb{E}[b(\omega_i; \theta_\top, \theta_\perp, n)|v_i = \top]$

$$\frac{\partial \mathbb{E}_\top}{\partial \theta_\top} = \frac{\partial \mathbb{E}_\top}{\partial (\theta_\top \cdot n)} \cdot \frac{\partial (\theta_\top \cdot n)}{\partial \theta_\top} = \beta \cdot n \cdot \mathrm{f}_b(\xi; n \cdot \theta_\top, n - n \cdot \theta_\top + 1) \cdot n \tag{27}$$

$$\frac{\partial \mathbb{E}_\top}{\partial \theta_\perp} = -\alpha \cdot n \cdot \mathrm{f}_b(1 - \xi; \theta_\perp \cdot n + 1, n - n \cdot \theta_\perp + 1) \cdot (n + 1) \tag{28}$$

$$\frac{\partial \mathbb{E}_\perp}{\partial \theta_\perp} = \beta \cdot n \cdot \mathrm{f}_b(\xi; n \cdot \theta_\perp, n - n \cdot \theta_\perp + 1) \cdot n \tag{29}$$

$$\frac{\partial \mathbb{E}_\perp}{\partial \theta_\top} = -\alpha \cdot n \cdot \mathrm{f}_b(1 - \xi; \theta_\top \cdot n + 1, n - n \cdot \theta_\top + 1) \cdot (n + 1) \tag{30}$$

where

$$f_b(x; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1 - x)^{b-1}$$

and $\Gamma(\cdot)$ is the gamma function. Then the condition for Bayesian Nash Equilibrium is:

$$\begin{aligned} \frac{\partial \mathbb{E}}{\partial \theta_\top} &= \frac{\partial \mathbb{E}_\top}{\partial \theta_\top} \cdot \xi + \frac{\partial \mathbb{E}_\perp}{\partial \theta_\top} \cdot (1 - \xi) = 0 \\ \frac{\partial \mathbb{E}}{\partial \theta_\perp} &= \frac{\partial \mathbb{E}_\top}{\partial \theta_\perp} \cdot \xi + \frac{\partial \mathbb{E}_\perp}{\partial \theta_\perp} \cdot (1 - \xi) = 0 \end{aligned} \tag{31}$$

Bringing in Eqn. 27, Eqn. 28, Eqn. 29, Eqn. 30, we get the analytic solution of $\theta_\top$ and $\theta_\perp$. But generally this analytical solution is not easy to solve, we can solve it by numerical methods.

Regularization Term To implement a reasonable voting mechanism, we need to introduce a regularization term. The purpose of this term is to prevent voters from voting too extremely, that is, their voting behavior is overly concentrated on $\top$ or $\perp$. By introducing a regularization term, we can encourage a more balanced voting behavior among voters. We can define the

regularization term as follows:

$$\mathcal{F}(\xi,\theta,n,\alpha,\beta) = \mathbb{E}[b(\omega_i;\theta_\top,\theta_\perp,n)] + \underbrace{\lambda\cdot\left\|\frac{\partial\mathbb{E}}{\partial\theta_\top}+\frac{\partial\mathbb{E}}{\partial\theta_\perp}\right\|^2}_{\text{Regularization}} + \underbrace{\lambda_1\rho(B(X\le\theta_\top;\xi))+\lambda_2\rho(B(X\le\theta_\perp;\xi))}_{\text{Invalid Voting Mechanism Penalty}} \tag{32}$$

Where $\rho(\cdot)$ is a step function, similar to activation. When there is a high probability that the vote can be passed, the value is 0; otherwise, the value is 1. λ is the regularization coefficient. We can find the optimal θ, α, and β by minimizing $\mathcal{F}(\theta,n,\gamma,\beta)$. Use Eqn.31 and Eqn.24 to find the optimal θ and ξ.

Then the optimal θ and ξ can be obtained by computing the minimum (for θ) or the maximum (for ξ) of $\mathcal{F}$. Here we can use gradient descent method to solve it. We can solve θ and ξ by iteration. In each iteration, we can update θ and ξ by calculating $\frac{\partial\mathcal{F}}{\partial\theta}$ and $\frac{\partial\mathcal{F}}{\partial\xi}$. During the iteration process, we can determine whether there is convergence by calculating the value of $\mathcal{F}$. When the value of $\mathcal{F}$ converges, we can get the optimal θ and ξ.

2.4.4 Semi-Perfect Information: $0 < n^{(i)} < i-1$

It is worth noting that in general, $\theta_\top$ and $\theta_\perp$ are the same. However, in the typical voting process, especially regarding blockchain consensus schemes, the situation often lies between a perfect information game and an imperfect information game. We refer to this as a semi-perfect game. For instance, in a consensus network, some nodes have already seen part of the voting results before voting, while they haven't seen the rest. After excluding the results they have seen, a voting subgame forms, which is an imperfect information game. And there is a high probability that $\theta_\top \neq \theta_\perp$. In this case, we can refer to the method described in Section 2.4.3 to characterize this subgame problem.

3 Monadring Protocol

We want to formalize the procedure of organizing some partial nodes of an existing blockchain network to form a smaller subnet, i.e., implementing this protocol as a plugin of some specific blockchain node programs.

3.1 Network Architecture

3.1.1 Model and definitions

Hostnet and subnet. Consider a network composed of a set of participants $\mathbb{V}$ in which the majority obey a protocol to reach Byzantine Agreement and finality [13] over ledger $\mathcal{L}$. $\mathcal{S}_i$ is a subset of $\mathcal{V}$, $\mathcal{V} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup .. \cup \mathcal{S}_{n-1}$. We call $\mathcal{V}$ is a `hostnet` and $\mathcal{S}_i$ is a `subnet` of $\mathcal{V}$.

- A node participant $v \in \mathcal{V}$ could be a member of any $\mathcal{S}_i$ at meantime.$v \in \mathcal{V}, v \in \mathcal{S}_m \cap \mathcal{S}_n$ is valid.
- We assume the ledger $\mathcal{L}$ of hostnet maintains all the subnets information in form of a mapping `subnetid`→`nodelist`.
- Any participant of the hostnet could join a specific subnet through proposing a modification over ledger $\mathcal{L}$.
- Since the ledger $\mathcal{L}$ is under Byzantine Agreement by all participants, the map can be considered a provable information outside any subnets.

Subnet ledger. For each subnet $\mathcal{S}_i$, all participants $v_j \in \mathcal{S}_i$ maintain an independent ledger $\mathcal{L}_i$ different from the ledger $\mathcal{L}$ of hostnet, while the root state of each subnet ledger will be recorded in the hostnet ledger, $\mathcal{L}_i \nsubseteq \mathcal{L}, \mathcal{F}(L_i) \in \mathcal{L}$.

A subnet could handle query and update requests independently from the hostnet ledger. We can assume that the subnet ledger has properties as below:

- The subnet ledger contains all the modification events and each event has an incremental number as index. The ledger is expected to be in a deterministic state after the n_{th} event being applied, $\mathcal{S}_{n+1} = f(\mathcal{S}_n, e_n)$.
- Particularly, changing the function f is also a kind of event. The first event e_0 is loading the function.
- The time complexity of looking up the n_{th} event is $\mathcal{O}(1)$.
- The time complexity of retrieving the maximum event id is $\mathcal{O}(1)$.

Token. A `token` $\mathcal{T}$ of Monadring is a special signal circulates within a subnet. A node in a subnet can write the subnet ledger only when it holds the token $\mathcal{T}$. The token is similar to a write lock in distributed networks.

3.1.2 Subnet topology

The participants of the subnet strictly follow the sequence of the `nodelist` to form a ring topology. The `token` $\mathcal{T}$ of a subnet with n nodes circulates around the subnet following the sequence of the `nodelist`.

- The `token` $\mathcal{T}$ of a subnet carries groups of events noted as $\mathbf{G}$ from its sender and the sender's forehead recursively.
- Group $\mathcal{G}_i$ is composed by the node v_i. It contains a list of modification events originally from its pending request queue, the node's digital signature, a digest of its local ledger after these modifications applied, a number q indicates how many times this group should delivered, the `nonce` of the signer and `ct` represents the voting data using FHE:

  $$\mathcal{G}_i = (\mathbf{E} = [e_k, e_{k+1}..e_{k+n}], \mathcal{S}_{k+n}, \texttt{signature}, \texttt{nonce}, q, [\texttt{ct}, ..]) \quad (33)$$

  Where the $\mathbf{E}$ could be empty, the q could be negative, the $\texttt{signature} = \texttt{sig}(\texttt{nonce}, \mathbf{E}, S_{k+n}, \texttt{nodekey})$.
- Normally, a token circulates in a subnet with n nodes should always include n groups unless there were malicious behaviors or some nodes went offline. Whenever a node receives the token, it ought to check the signature for each group. Then applying all the events of each group and compare the digest with the local ledger. The q of executed groups should be decreased by 1.
- If all checks pass, the node should handle transactions from its local queue and compose them as a new group with initial $q = n - 1$ to replace previous one in the token. Then try to deliver the token to its successor.

Assume a subnet with n nodes $v_0, v_1..v_{n-1}, n \geq 3$. If no nodes joined the subnet during last round, when a node v_i receives the token $\mathcal{T}$, noted as $\mathcal{T}_i$.

$$\mathcal{T}_i = \mathcal{G}_i, \mathcal{G}_{(i+1) \mod n}, .., \mathcal{G}_{(i+n-1) \mod n} \quad (34)$$

The $\mathcal{G}_i$ is the group generated by v_i last time. After executed all the groups except the $\mathcal{G}_i$, v_i should replace $\mathcal{G}_i$ with $\mathcal{G}'_i$ and put it at the tail of the token, i.e., the $\mathcal{T}_{i+1}$ is:

$$\mathcal{T}_{i+1} = \mathcal{G}_{i+1}, \mathcal{G}_{(i+2) \mod n}, .., \mathcal{G}'_i; i + 1 < n \quad (35)$$

Optimization. A node could broadcast the events it just executed to others as an optimization. The broadcasting doesn't have to be through reliable communication since the token contains all the essential information that if some nodes don't receive the broadcasting could execute it later.

A node could execute the events from broadcasting only if the `eventid` strictly follows the sequence in case it missed the previous broadcasting, e.g., a node has maximum event e_i can pre-commit a group received from broadcasting in which the events start from e_{i+1}.

3.2 Procedures

3.2.1 Launching a subnet and rotating subnet members

To launch a subnet, a publishing procedure similar to deploying a smart contract is required. This involves a deterministic function f in the form of executable binary code. Once the transaction is confirmed on the host network, any nodes can register to join the subnet. When the number of registered nodes reaches a certain `threshold`, these nodes will synchronize the function f specified in the registration information and begin executing it locally. The token $\mathcal{T}$ initializes following the order of member registration, and the `FHEKeyGen` procedure mentioned previously will be executed.

After surpassing the threshold for registered nodes, subnet nodes must undergo periodic rotation to ensure randomness and reduce the potential for collusion. This rotation process can utilize a verifiable random function (VRF) or a simple pseudorandom function with a deterministic seed, such as the block hash at the end of each epoch, to select nodes for rotation in and out of the subnet.

3.2.2 Node offline and token recovery

The token might be disappeared during circulates within the subnet if some nodes go offline or due to malicious behaviors.

Assume a node v_i of a subnet $\mathbf{S} = [v_0, v_1, .., v_{n-1}]$ holds the token $\mathcal{T}$ at moment t. At the same time, all the rest nodes $\forall v_j \in \mathbf{S}, (i \neq j)$ keep the copy τ_j of the token $\mathcal{T}$ when they received it, e.g., v_0 keeps a copy τ_0 of $\mathcal{T}_0$ delivered from v_{n-1}. We call τ_j is the *last seen token* of v_j. The τ_j could be `null` since $v_j (j > i)$ may be a new member and never received the token before.

To recover the missing token, all nodes $v_i \in \mathbf{S}$ need to set a timer with countdown $\texttt{len}(\mathbf{S}) * \epsilon$ after they delivered it, where ϵ is the rough estimation of the execution time on a single node or a constant setting of the subnet.

When the timer of v_i triggered, v_i should resend the copy τ_i to its successor and reset the timer. For a node v_i receives a copy τ_i of the token, it is clearly to recognize that τ_i is a copy since v_i had handled the events within it already.

3.2.3 Function upgrade

A function upgrade is an event that should be included in the token $\mathcal{G}$. Only specific users can initiate this event, where the subnet manager submits a transaction to modify the hostnet ledger $\mathcal{L}$, similar to registering a subnet. The transaction should include not only the function itself but also a digest and version, such as an executable WASM binary with its corresponding digest and version.

Once an upgrade transaction is confirmed by the hostnet, any members of the subnet can include this event into token $\mathcal{T}$ if the hostnet ledger contains a higher version of the function but the associated event is absent in the token.

3.3 Fault Tolerance with FHE

The hostnet can reach the Byzantine Agreement over $\mathcal{L}$ based on an assumption that the ratio of honest participants $r >$ `threshold`. We don't expect that a subnet has a same ratio as the honest participants of hostnet, neither we expect that a node was honest on ledger $\mathcal{L}$ would be honest on subnet legder $\mathcal{L}_i$.

We want to find a solution to detect the malicious behaviours or nodes offline then exclude them from the subnet by a provable invalidity.

For simplicity, we note $\mathcal{T}_{-i}$ as $\mathcal{T}_i$ excluding the i_{th} group. An valid $\mathcal{T}_i$ must satisfy properties as below:

- The q of $\mathcal{G}_i$ satisfies $q \leq 0$. $q = 0$ implies no new nodes joined the subnet in this round, while $q < 0$ implies some new nodes have joined the subnet. The node v_i can check the `nodelist` from $\mathcal{L}$. For rest q of $\forall \mathcal{G} \in \mathcal{T}_{-i}$ are similar.

- Group $\mathcal{G}_i \in \mathcal{T}$ contains a signature using its `nodekey`.

- Assume the maximum event id of $\mathcal{G}_i$ is m. Combine all event lists of $\mathcal{T}_{-i}$ as a single list $\mathbf{E}_{-i}$. Assume the minimal event id of $\mathbf{E}_{-i}$ is n. $n = m + 1$ and m is the maximum event id of node v_i.

Given a token $\mathcal{T}$, any nodes can simply validate it. If there is a malicious behavior noted as $\mathcal{G}_i = (.., \mathbf{E} = [e_i, e_{i+1}, ..], ..)$, the nearest honest successor could place its event group with the smallest common agreed `eventid`, i.e., there are two conflicted groups G_i' and $\mathcal{G}_i$ in a token $\mathcal{T}$. The invalid one contains the node's signature could be used for slashing.

The difference between this procedure and other BFT consensus algorithm is that the former one involves FHE to hide the voting information on each group. Thus, the subnet could efficiently adapt to small-scale networks and avoid collusion actions.

3.4 Framework Workflow

The Framework Workflow of Monadring is described in Algorithm 2.

Algorithm 2 Monadring Workflow

Intialization: $sk, pk \leftarrow$ `FHEKeyGen`()
Sends a transaction to the hostnet $\mathcal{V}$ to opt-in a subnet $\mathcal{S}_i$.
Ultilize `VRF` using Voting Model described in Section 2.4 to select the subnet members.
for each epoch **do**
    Sharing/Resharing the keys using Shamir's Secret Sharing Scheme.
    Compose a new group with initial $q \leftarrow n - 1$.
    while $q > 0$ **do**
        if receives $\mathcal{T}$ **then**
            Check the signature of each group $\mathcal{G}_i$.
            Apply the events of each group $\mathcal{G}_i$.
            Compare the digest with $\mathcal{L}_i$.
            Decrease the q of executed groups.
            Handle transactions from the local queue.
            Deliver $\mathcal{T}$ to its successor.
        end if
        $q \leftarrow q - 1$
    end while
    Rotate the subnet members described in Section 3.2.1.
    if Malicious behaviours detected **then**
        ApplyFault tolerance with FHE described in Section 3.3.
        Voting for slashing using the voting model described in Section 2.4.
    end if
end for

4 Future Work

We are currently in the process of implementing several key solutions to advance our research:

- Integration of Zama [14]'s FHE library: We are working on incorporating Fully Homomorphic Encryption techniques using Zama's library to enhance privacy and security in our system.
- Adoption of Substrate framework: We are actively implementing the Substrate blockchain development framework to improve the scalability and interoperability of our solution.
- Decoupling of runtime and consensus: Our team is focusing on separating the runtime and consensus mechanisms, a change we anticipate will significantly enhance both flexibility and performance.
- Additional improvements: We are addressing numerous other areas for enhancement, including optimizing data structures, improving network protocols, and refining our cryptographic approaches.

These initiatives represent our current focus and the direction of our ongoing work. We will continue to refine and expand upon this paper over the coming months, providing more detailed analyses, implementation specifics, and preliminary results as our work progresses.

References

[1] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Paper 2016/889, 2016. `https://eprint.iacr.org/2016/889`.

[2] Jeff Burdges, Alfonso Cevallos, Peter Czaban, Rob Habermeier, Syed Hosseini, Fabio Lama, Handan Kilinc Alper, Ximin Luo, Fatemeh Shirazi, Alistair Stewart, et al. Overview of polkadot and its design considerations. *arXiv preprint arXiv:2005.13456*, 2020.

[3] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph, 2016.

[4] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4):1–35, 2018.

[5] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[6] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

[8] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[9] Liao-Jun Pang and Yu-Min Wang. A new (t, n) multi-secret sharing scheme based on shamir's secret sharing. *Applied Mathematics and Computation*, 167(2):840–848, 2005.

[10] Jens Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.

[11] Drew Fudenberg and Jean Tirole. *Game theory*. MIT press, 1991.

[12] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.

[13] Eleftherios Kokoris-Kogia Alistair Stewart. Grandpa: a byzantine finality gadget. *arXiv:2007.01560*, 2020.

[14] Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022. `https://github.com/zama-ai/tfhe-rs`.