星系共识

拥有完整委托机制的实用 PoS 共识协议

Demmon Guo, Chris Shi, Yu Chen

万维链理论研究团队

目 录

引 言

共识协议是区块链系统的核心组成要素，是整个系统稳定性和安全性的保证。比特币引入了第一个区块链共识协议，即工作量证明（PoW）。在这一共识协议下，为获得出块权，矿工需要不断进行哈希计算，直到结果小于某个特定值，而持续的哈希计算耗费大量的能源，这也是 PoW 被人诟病的主要原因。为解决能源浪费的问题，很多新型的共识机制被提出，它们将安全基础由计算能力转为依赖其他资源，其中权益证明（PoS）被广泛研究和认可，作为 PoW 最有利的替代者，我们坚信，PoS 将是区块链共识未来的发展方向。立足深入研究已有优秀共识协议（如 Ouroboros[1]、Dfinity[2]）的基础上，充分借鉴其设计优势，结合多种密码学手段，我们设计出星系共识协议。同时，星系共识引入了全新的委托机制，并在万维链完成了概念验证（POC）。

星系共识的创新点:

1.重新定义了权益并在账户模型下（account model）模拟了币龄（coin age）的概念，保证了共识参与者的稳定性。与其被动地被选择，所有 WAN 的持有者需要主动注册去参与到星系共识中，这样保证了参与者在共识过程中是活跃状态，而不是沉睡状态（sleepy）[3]。由此增加的稳定性使得整个协议在现实应用中具有更高的实用性。

2.设计了新型安全的随机数生成算法，可用于出块者的选择以及其他用途，为系统引入熵（entropy）并保证了整个协议的安全性。

3.设计了 ULS（unique leader selection）算法并用于出块者选择。与其他选择算法相比（如 VRF），ULS 算法在保持出块者匿名的同时，保证了出块者的唯一性，降低了自然分叉的概率。

4.设计了三重 ECDSA 委托签名方案并实现了完整的委托机制，持有较少 WAN 的用户也能够参与到共识过程中，从而吸纳更多的权益加入到共识协议中。

5.设计了公平合理的奖励机制，激励共识过程中的诚实行为并对恶意行为进行惩罚，保证权益分布健康，防止权益集中（stake centralization）。

论文组织结构：

我们在第一章中介绍 Wanstake 的设计；在第二章介绍星系共识的核心协议，包括随机数生成过程和 ULS 算法；第三章和第四章分别介绍委托机制和经济激励机制；第五章介绍协议对不同攻击行为的抵抗性；在第六章介绍星系共识的优势。

第一章 Wanstake 设计

1.1 设计概览

所有的 PoS 共识方案都致力于使参与者通过去中心化的方式达成共识，然而往往忽略了影响共识的一些重要因素，如共识参与者的稳定性和活跃度、权益的分布等等。我们的设计通过实现以下目标，保证了共识机制在现实应用中的可行性。

- 高稳定性：共识参与者应保持在线并参与到共识过程中。
- 高活性：与其被动地被选择为出块者，所有参与者需要通过主动注册的方式参与到星系共识中，这样能够保证共识参与者具有很高的活性，从而不会因为掉线等原因而影响共识过程的正常进行。
- 独立性：共识参与者无法通过将持有的 WAN 分布在不同的账户中或与其他参与者合作将 WAN 放在同一账户中而获得额外收益。
- 权益的健康分布：健康的权益分布不能够太集中，同时也不能够太分散。星系共识鼓励 WAN 持有者将尽可能多的币投入到共识协议中，同时也采取方式限制 WAN 的大型持有者（如交易所）对共识过程的影响，防止其控制共识过程。

1.2 Wanstake

定义：

WAN：万维链的原生币，通过锁定在一个特定的智能合约（即共识智能合约）中转化为 Wanstake。

Wanstake：通过锁定 WAN 得到，WAN 的数量越多，锁定时间越长，生成的 Wanstake 越多。

权益比率：个人持有的 Wanstake 占全部 Wanstake 的比例。

CSC：共识智能合约（consensus smart contract）的缩写，WAN 持有者通过将 WAN 锁定在 CSC 中获取 Wanstake。

***H*函数：**计算共识参与者 Wanstake 量的函数。

WAN 的持有者通过将一定数量的 WAN 锁定在共识智能合约中去参与万维链星系共识协议。共识智能合约根据锁定的 WAN 的数量和锁定时间计算得到 Wanstake 的数量。值得注意的是，考虑到越临近锁定时间结束，参与者的诚实性和稳定性就越高，Wanstake 的数量在锁定时间内并不是固定值，而是根据剩余锁定时间发生变化。Wanstake 的持有者以正比于其权益比率（Wanstake ratio）的概率被选择参与到共识过程中。当参与者的锁定时间结束后，将自动失去参与共识的权利，共识智能合约会在一段时间之后将锁定的 WAN 返还到原始账户中。

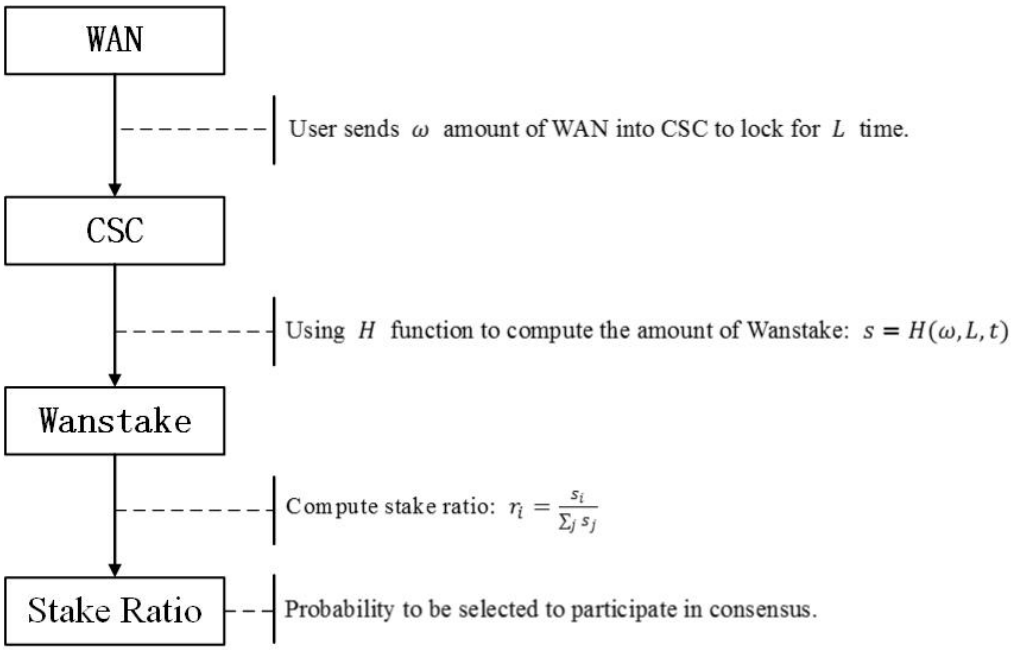

图 1: WAN 到共识参与概率计算过程

如上图所示，H函数在计算 Wanstake 的过程中具有重要作用。为达到上文提到的设计目标，H函数应当满足以下几个性质:

ω为锁定的 WAN 的数量。

L为锁定时间长度。

t为余下锁定时间占全部锁定时间的比例，锁定开始时为 1，结束时为 0。

性质 1. 对ω和L单调递增

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

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

这一性质表明通过将更多的 WAN 锁定更长的时间，能够提高 Wanstake 的量，这样有助于提高共识参与者的稳定性。

性质 2. 对t单调递减

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

这一性质表明，参与者的可靠性随着其在共识过程中的诚实行为而不断提升。

性质 3. 对ω满足线性

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

这一性质表明 WAN 的持有者无法通过将其持有的 WAN 拆分到多个账户中而获取额外的收益。

性质 4. 对L的积分为凹函数

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

其中，t'是已经流逝的锁定时间，$t = \frac{L-t'}{L}$。这一性质表明我们鼓励参与者锁定一个较长的时间段，而不是两个较短的时间段。

满足以上 4 个性质的候选函数为：

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

其中σ_L是关于L的增函数即可满足性质 4，详细证明在附录 1。

第二章 核心协议

2.1 概念和假设

为描述协议，我们首先介绍协议中用到的概念和假设：

Community：PoS 共识参与者组成的集合，成员定期更新。

Slot：协议中最小的时间单位，连续放置，用自然数顺次标记，如第i个 slot 记为$slot_i$，每个 slot 最多有一个块与之对应。

Slot Leader：与 slot 对应的合法出块者，在我们协议中，一个 slot 只有一个参与者会被选作合法出块者。

Epoch：由相邻的固定数量的 slot 相连而成，它是共识协议执行的基本时间单位。在 epoch 开始，从共识参与者中选择成员作为 Random Number Proposer 完成随机数的生成；同时，在 epoch 中每个 slot，都会选择一个合法的出块者生成对应的区块。共识协议以 epoch 为单位，循环往复执行。

Epoch Leader：每个 epoch 都会从 Community 中选取一部分成员成为 Epoch Leader，而 Slot Leader 就是从 Epoch Leader 中选择得到，$epoch_n$的 Epoch Leader 在$epoch_{n-1}$的开始被确定。

Random Number Proposer：每个 epoch 都会从 Community 中选取一部分成员成为 Random Number Proposer，负责随机数的生成。

Random Beacon：由 Random Number Proposer Group 成员模拟的随机数生成器，在每个 epoch 中输出一个随机数。

安全参数：记作k，影响数据的确定性，深度超过k的区块数据便不再变化。

协议的安全性由以下假设保证：

世界时间：所有参与者都能获取到当前绝对时间，这一时间在所有参与者中基本同步，可以推算出当前 slot 编号。

诚实权益大多数：参与共识过程中的权益超过一半掌握在诚实参与者手中。

半同步网络：消息的传输存在一个最大的延迟时间，它对协议的参与者是未知

的。

腐化延迟：恶意的共识成员腐化诚实的共识成员存在一个最小延迟时间，超过这个时间之后，腐化才会生效。

2.2 协议概述

正如上文所述，我们将 WAN 和 Wanstake 的概念进行了分离，因为大部分 WAN 的持有者是离线的，或者说是“沉睡”的。为保证共识过程顺利进行，要确保参与者是在线的。因此有意向参与共识的 WAN 持有者需要注册成为 Community 成员，注册的方式是在一个特定的智能合约（即共识智能合约）中锁定一部分 WAN。这样能够保证共识参与者具有较高的稳定性和活性。

随机数的生成对于共识协议有重要意义，特别是在随机选取过程中。 我们设计了随机数生成算法去模拟 random beacon。这个算法在每个 epoch 由 Random Number Proposer Group 成员运行一次，而这些成员是从 Community 选择得到。random beacon 的输出有三个作用：(i) 确定当前 epoch 中 Random Number Proposer Group 的成员，(ii) 确定下一 epoch 中 Epoch Leader Group 的成员，(iii) 确定当前 epoch 中出块者的顺序。

与基于拜占庭容错算法（BFT-based）的协议不同，我们的协议是基于链的（chain-based），因此设计过程中最大的挑战就是如何确定出块者。很多已有的 PoS 共识协议通过 VRF 算法实现出块者的选择，带来的后果是每一 slot 会对应多个出块者或者没有出块者。我们期望设计一种选择算法，保证每个 slot 只有单一的出块者与其对应。为保证出块者身份不提前暴露，Epoch Leader Group 通过生成秘密消息决定出块的权利，出块者的身份只有在区块生成并广播之后才会暴露，这样降低了被腐化的风险。同时，为了防止 grinding attack，出块者的顺序由 random beacon 在每个 epoch 开始前确定。因此秘密消息会在 random beacon 更新之前产生。

整体而言，协议的执行流程为：(i) 协议的参与者组成 Community，执行共识过程，(ii) 在每个 epoch 开始，通过 random beacon 产生的随机数从 Community

选择成员组成 Random Number Proposer Group 和 Epoch Leader Group，(iii) Random Number Proposer Group 成员负责合作生成一个随机数作为当前 epoch 中 random beacon 的输出，(iv) Epoch Leader Group 成员在当前 epoch 生成秘密可验证的消息，(v) 在前一个 epoch 开始出选择的 Epoch Leader Group 成员运行选择算法决定当前 epoch 中每一个 slot 的出块者身份。

2.3 Random Beacon

2.3.1 设计背景

任何共识机制的核心都是保证全网对于下一出块者的身份达成一致意见，也就是常说的出块者选择过程。一个公平且随机的出块者选择过程是整条链保持活性的基础，而这种公平性和随机性依赖于为系统引入“熵”。PoW 以一个很自然的方式引入了熵，因为在挖矿过程中使用的哈希函数 SHA256 是安全的，即单向且无碰撞。对于矿工而言，只有一种方法能够解开挖矿难题，那就是不断尝试各种输出，即所谓的穷举法。第一个解开哈希难题的矿工便获得了下一区块的出块权，整个过程公平而且随机。如何在出块者选择过程中引入熵是在设计 PoS 共识机制过程中诸多重大挑战之一，我们需要设计一种高效安全的方式去引入熵，这就是 random beacon。

random beacon 是 PoS 共识系统的安全基础，需满足如下性质：

- 去中心化：在 random beacon 的运行过程中，无需可信第三方参与。
- 不可预测：即使获得之前的输出，也无法在预测未来输出中占有优势。
- 无偏性：任何人无法通过计算能力或者后发优势去针对性左右 random beacon 的输出。
- 均匀分布：random beacon 的输出在其值域内是均匀分布。
- 保证输出：一旦协议开始执行，任何参与者无法通过违背协议的方式来影响结果的输出，即协议能够保证输出结果。
- 公开可验证：没有参与到随机数生成过程中的个体能够以后验的方式，监督协议的执行，验证随机数是可信和无偏的。

Random Beacon 设计思路：

为了设计一个既契合我们 PoS 共识机制，又同时满足上述良好性质的 random beacon，我们从两方面入手解决了这一难题。首先，我们将区块链作为一个可信任的广播通道，所有参与者通过区块链完成数据的交互。一方面，参与者之间不必再单独建立通信联系，节省了带宽；另一方面数据上链能够保证交互数据的正确性。其次，我们在 random beacon 的设计中使用了多种密码学工具去保证了其具备良好的性质，比如可验证秘密分享保证了它是去中心化且可被公开验证，门限签名保证了它具备不可预测性和保证输出性，哈希函数保证了最终的输出是分布均匀的。

2.3.2 基础知识

- 椭圆曲线

p是一个素数，F_p为有p个成员的有限域。椭圆曲线$E(F_p)$是F_p上所有满足等式$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$的点$(x, y)$以及无穷远点$O$构成的集合，其中$a_i \in F_p$。$E$是一个阿贝尔群，由点$P_1$、$P_2$坐标计算点$P_3 = P_1 + P_2$坐标的公式请见[4]。

椭圆曲线$E(F_p)$上点的个数记作$\#E(F_p)$，称为椭圆曲线的阶。点$G \in E(F_p)$的阶r定义为满足$rG = O$的最小非负数。事实上所有G的倍点构成一个阶为r循环群，而生成元为G。用户U_i生成一个密钥对(pk_i, sk_i)，其中为pk_i公钥，sk_i为私钥，并且$pk_i = sk_i \cdot G$，一般用pk_i表示U_i。关于椭圆曲线密码的更多细节请见[4]。

- Shamir 门限秘密分享

P有秘密s，要通过(t, n)门限秘密分享方案向P_1、P_2、⋯、P_n分享，操作如下：

→ 随机选择一个$t-1$次多项式$f(x) = s + a_1x + \cdots + a_{t-1}x^{t-1}$

→ 随机选择n个不同的数字x_1、x_2、⋯、x_n，并计算$f(x_1)$、$f(x_2)$、⋯、$f(x_n)$

→ 向P_i发送秘密份额 $s_i = (x_i, f(x_i))$， $i = 1,2,\cdots,n$

任何$k \geq t$个参与者可以合作去重构出s：

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

关于 Shamir 门限秘密分享的更多细节请见[5] [6] [7]。

- 门限签名

在一个(t, n)门限签名方案中，n个参与者联合计算一个公钥（即组公钥），每个参与者保留各自私钥份额。之后，只需大于或等于t个参与者就可以合作计算出一个签名（即组签名），并且能够被组公钥验证通过。

- 基于序对的数字签名方案

基于椭圆曲线序对实现的数字签名方案具有唯一性、非交互性，非常适合用于 random beacon 的设计。

G_1、G_2是椭圆曲线上两个阶为q的循环群，G_T为有限域上的循环群。g_1、g_2、g_T分别是G_1、G_2、G_T的生成元。H是哈希函数：$\{0,1\}^* \to G_1$。e是非退化的双线性映射：$G_1 \times G_2 \to G_T$。私钥$sk \in (1, q)$，公钥$pk = sk \cdot g_2$。签名以及验证算法如下：

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

$PB_{verify}(pk, M, \sigma)$ 验证 $e(\sigma, g_2) = e(H(M), pk)$是否成立

- Reed-Solomon 码

我们使用的 Reed-Solomon 码具有以下形式：

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

其中$p(x)$取遍$Z_p[x]$上所有次数小于等于$t - 1$的多项式。

同时我们定义其对偶码（dual code）为：

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

其中参数$v_i = \prod_{j=1, j\neq i}^{n} \frac{1}{i-j}$。

那么以下定理成立:

定理：如果$v \in {Z_q}^n$，$c^{\perp}$从 $C^{\perp}$ 中均匀选择，那么$< v, c^{\perp} > = 0$成立的概率为 $\frac{1}{p}$。

关于 Reed-Solomon 码的更多细节请见[8]。

- 离散对数知识的零知识证明

基于随机预言模型的 DDH 假设，可对以下事实做出零知识证明：给定g、x、h、y，存在$\alpha \in Z_p$，满足$x = g^{\alpha}$和$y = h^{\alpha}$。将这一零知识证明记为$DLEQ(g, x, h, y)$。它最早是由 Chaum 和 Pedersen 在[9]中构造，更多细节请见附录 2。

2.3.3 Random Beacon

G_1、G_2是椭圆曲线的两个循环群，生成元分别为G、$\bar{G}$。e为非退化的双线性映射：$G_1 \times G_2 \to G_T$。假设在 random beacon 过程中有n个参与者，记为P_i，密钥对为$(sk_i, pk_i = sk_i \cdot G)$，$i = 1,2,\cdots,n$。在接下来的部分，我们详细介绍星系共识中 random beacon 的工作流程。

random beacon 将一个 epoch 分为三个阶段，分别为 DKG1、DKG2 和 SIGN 阶段，其中 DKG 是分布式密钥生成（decentralized key generation）的简称。在 DKG1 阶段，所有参与者对在 DKG2 要提交的数据做出承诺；在 DKG2 阶段，所有参与者协同计算得到组公钥（group public key）和组私钥份额（group secret key share）；在 SIGN 阶段，所有参与者使用其组私钥份额计算组签名份额（group signature share）；最终，通过组签名份额合成组签名，并计算得到 random beacon 的输出。

DKG1 阶段

在这一阶段，每个参与者做如下操作（以P_i为例）：

→ 随机选取 $s_i \in (1, q)$

→ 随机选取$t-1$次多项式$f_i(x) = s_i + a_{i,1}x + \cdots + a_{i,t-1}x^{t-1}$

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

→ 计算承诺$c_{i,j} = s_{i,j} \cdot \bar{G}$，其中$j = 1,2,\cdots,n$

→ P_i发送一笔特殊交易$DKG1_{Tx_i}$，并将以下数据包含进去：

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

$DKG1_{Tx_i}$交易的验证逻辑

当收到$DKG1_{Tx_{i_i}}$交易时，做如下验证：

→ 随机选取 $c^{\perp} = (c_1{}^{\perp}, c_2{}^{\perp}, \cdots, c_n{}^{\perp}) \in C^{\perp}$

→ 计算$\sum_{j=1}^{n} c_j{}^{\perp} \cdot c_{i,j}$，并验证结果是否为无穷远点，如果成立，则交易验证通过

DKG2 阶段

在这一阶段，每个参与者做如下操作（以P_i为例，$s_{i,j}$和$c_{i,j}$是在 DKG1 阶段生成）：

→ 加密 $\widetilde{s_{i,j}} = s_{i,j} \cdot pk_j$，其中 $j = 1,2,\cdots,n$

→ 构造证明 $proof_{i,j} = DLEQ\left(\bar{G}, c_{i,j}, pk_j, \widetilde{s_{i,j}}\right)$，其中$j = 1,2,\cdots,n$

→ P_i发送一笔特殊交易$DKG2_{Tx_i}$，并将以下数据包含进去：

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

$DKG2_{Tx_i}$交易的验证逻辑

当收到$DKG2_{Tx_i}$交易时，做如下验证：

→ 验证$proof_{i,j}$是否成立，其中 $j = 1,2,\cdots,n$

计算组私钥份额

P_i通过以下操作计算得到其组私钥份额（group secret key share）：

→ 扫描所有 $DKG2_{Tx_j}$交易，获取 get $\widetilde{s_{j,i}}$，其中$j = 1,2,\cdots,n$

→ 解密 $\widehat{s_{j,i}} = {sk_i}^{-1} \cdot \widetilde{s_{j,i}}$，其中$j = 1,2,\cdots,n$

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

SIGN 阶段

在这一阶段，P_i计算其组签名份额（group signature share），过程如下：

→ 计算$M = Hash(r||\varsigma_{r-1})$，其中$r$是当前 epoch 的编号，$\varsigma_{r-1}$是 random beacon 在 epoch $r-1$的输出，$Hash()$是安全哈希函数

→ 计算$gsigshare_i = M \cdot gskshare_i$

P_i构造一笔包含$gsigshare_i$的特殊交易SIG_{Tx_i}，并将其发送到链上

SIG_{Tx_i}交易的验证逻辑

当收到 SIG_{Tx_i}交易时，做如下验证：

→ 扫描所有$DKG1_{Tx_j}$交易，获取$c_{j,i}$，其中$j = 1,2,\cdots,n$

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

→ 计算$M = Hash(r||\varsigma_{r-1})$，其中$r$是当前 epoch 的编号，$\varsigma_{r-1}$是 random beacon 在 epoch $r-1$的输出

→ 验证$e(gsigshare_i, \bar{G}) = e(M \cdot G, gpkshare_i)$是否成立，如果成立，则交易验证通过

计算 ς_r

在 epoch r 结束时，random beacon 输出ς_r，它的计算过程如下：

→ 扫描所有SIG_{Tx_i}交易，获取$gsigshare_i$，其中$i = 1,2,\cdots,n$

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

→ 扫描所有$DKG1_{Tx_i}$交易，获取$c_{j,i}$，其中$i = 1,2,\cdots,n$，$j = 1,2,\cdots,n$

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

→ 验证 $e(gsig, \bar{G}) = e(M \cdot G, gpk)$是否成立，如果不成立，则计算失败

→ 否则计算并输出 $\varsigma_r = Hash(gsig)$

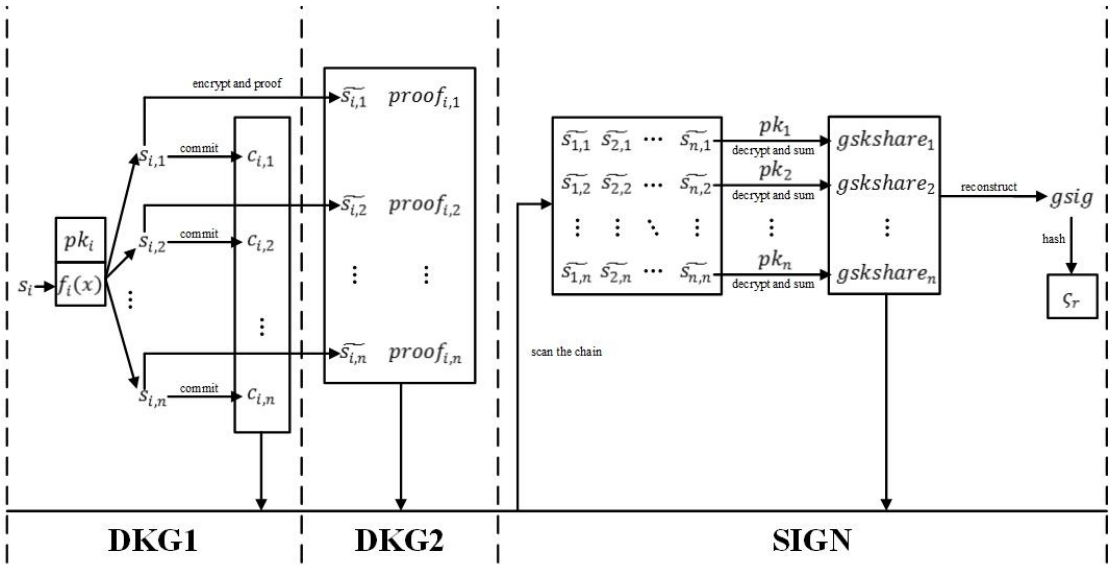

图 2: Random Beacon 工作流程

2.5 Epoch Leaders 选择

在每个 epoch 开始时，我们根据k个块之前的权益分布去选取下一 epoch 的 Epoch Leader，共识参与者被选到的概率正比于其持有权益的比例。我们实现了 *follow-the-stake-rate* 算法去模拟选择过程，与 *follow-the-satoshi* 算法基本一致。

我们为每位共识参与者计算其权益比例，显然所有权益比例之和为 1。之后我们构造一个二叉树，每个叶节点用参与者公钥标记，节点的权重为其子树的权益比例之和。不妨设r为 random beacon 的输出，计算$cx_i \equiv hash^i(0||r)$， $i = 1,2,\cdots,N$，其中N是 Epoch Leader 的个数，$hash^i()$指将哈希函数重复i次。第i个 Epoch Leader 的选择方式为，从二叉树的根节点开始往下游动，如果节点权重大于cx_i，则往左子树游动，否则往右子树游动，同时将cx_i更新为减去节点权重的结果。在N轮游动之后，全部N个 Epoch Leader 的身份就会确定。值得注意的是，选出的 Epoch Leader 集合为多重集，因为一个参与者可能会不止一次被选到。Random Number Proposer 的选择方式也是一样，不同的是$cx_i \equiv hash^i(1||r)$，$i = 1,2,\cdots,N$。

Epoch Leader 是按照正比于权益比例的概率从共识参与者中选择得到，而 Slot Leader 是从 Epoch Leader 中等概率选取得到。这种两阶段的选择方式与直接从共识参与者中选择 Slot Leader 的方式效果是一致的，具体证明请见附录 3。

2.6 唯一出块者选择

Epoch Leader 选择结束后，被选到的参与者需要在本 epoch 中生成秘密信息，为下一 epoch 的唯一出块者选择做准备。不妨假设被选到的参与者为 $epochleaders = \{P_1, P_2, \dots, P_N\}$，它们对应的密钥对为 $\{(pk_1, sk_1), (pk_2, sk_2), \dots, (pk_N, sk_N)\}$。生成秘密信息的两个阶段如下：

SMA1

在这一阶段，所有参与者如下操作（以P_i为例）：

→ 随机选取$\alpha_i \in (1, q)$

→ 计算$M_i = \alpha_i \cdot pk_i$

→ P_i构造一笔包含M_i的特殊交易$SMA1_{Tx_i}$，并将其发送到链上

注：M_i本质上是对生成随机数的承诺，保证了其不再发生更改。

SMA2

在这一阶段，所有参与者如下操作（以P_i为例）：

→ 计算$\alpha_i \cdot pk_j$，其中$j = 1,2,\cdots,N$

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

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

→ P_i构造一笔包含A_i和π_i的特殊交易 $SMA2_{Tx_i}$，并将其发送到链上

注：零知识证明π_i保证了不同公钥标量乘的系数是一致的，详细内容请见附录 2。

$SMA2_{Tx_i}$交易的验证逻辑

→ π_i验证通过

→ 向量A_i的第 i 个坐标与M_i一致，即$A_i[i] = M_i$

计算 Secret Message Array

P_j做如下计算得到 secret message array：

→ 扫描所有$SMA2_{Tx_i}$交易，获取$\alpha_i \cdot pk_j$，其中$i = 1,2,\cdots,N$

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

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

ULS 算法

在每个 epoch 开始之前，所有的出块者由S和 random beacon 产生的随机数通过以下方式确定：

→ 计算$hash(r||pk_i)$，其中r是 random beacon 输出的随机数，pk_i是P_i的公钥，并按照计算结果将所有 Epoch Leader 重新排序为$sq = (P'_1, P'_2, \dots, P'_N)$

→ 计算随机数矩阵$M = (sr_{ij})_{n \times N}$，其中$n$是 epoch 的长度，

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

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

→ 计算$cs_i = cr_i\ mod\ N$

→ Slot sl_i 的出块者为P'_{cs_i+1}

产生区块

当提出一个区块时，出块者P'_{cs_t+1}需要附加额外的信息去让整个选择信息公开可验证：

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

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

→ 将G_t和π'添加到所产生区块中

Slot Leader 身份的验证逻辑

当接受到区块$block_t$，出块者身份的合法性验证过程如下：

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

→ 扫描链上数据，计算$skGt = \sum_{j=1}^{N} \alpha_{sr_{tj}} \cdot pk_{cs_t+1}$

→ 验证$\pi' = DLEQ(G, pk_{cs_t+1}, G_t, skGt)$成立

→ 计算$cr_t{}' = hash(G_t)$

→ 计算$cs_t{}' = cr_t{}' \ mod \ N$

→ 验证$P'_{cs_t{}'+1}$是否与$block_t$的出块者身份匹配

这样一来，从 Epoch Leader 中选择 Slot Leader 的过程便可以公开验证。

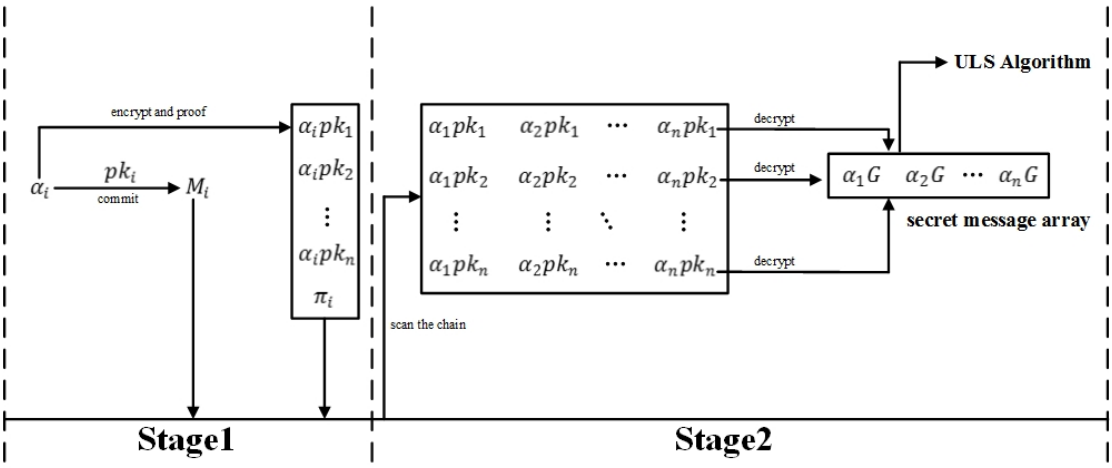

图 3: Epoch Leader 工作流程

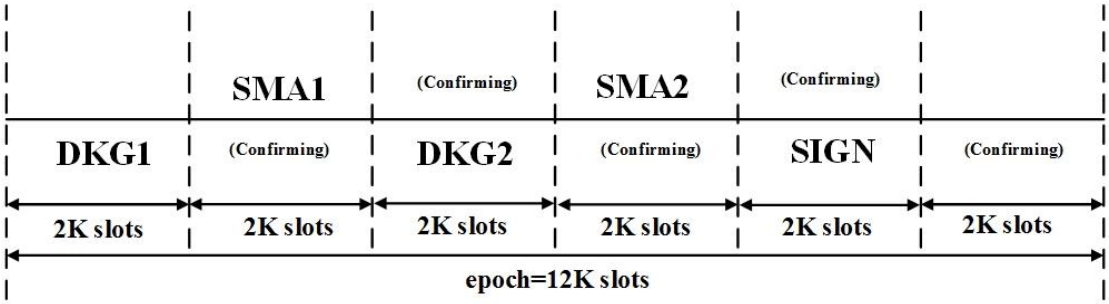

图 4: epoch 内各阶段顺序

第三章 委托机制

3.1 设计背景

本章介绍星系共识协议的委托机制。WAN 以不同的数量广泛分布在各个账户中，理论上任何一个 WAN 的持有者都有权利加入到共识过程中，这是 PoS 协议的精神所在。然而，由于参与共识过程的收益与锁定的 WAN 的量成正比，导致持有少量 WAN 的用户共识收益不足，甚至低于其参与的成本（计算、存储、网络成本等），大大降低了其参与共识的积极性。因此，需要建立一套机制去保证持有任何数量 WAN 的用户都能够参与共识并且从中获益。我们通过基于委托签名（proxy signature）实现的委托方案解决了这一问题，在这一方案下，更多的 WAN 持有者会加入到共识中，从而使整个网络更加安全健壮。

委托签名是实现委托方案的一种有效方式，它能够使一个个体，称为委托者或原始签名者，将自己对消息签名的权利委托给另外一个个体，即被委托者。被委托者能够计算出一个委托签名，任何拥有原始签名者公钥的个体均可以验证签名的合法性。严格的说，委托签名算法为多个算法的集合，$PS = (G,\ K,\ S,\ V,\ (D,P),\ PS,\ PV,\ ID)$：

- (G,K,S,V) 是一个标准数字签名算法。
- (D,P)是委托和接受委托的算法。其中D输入委托者i的私钥sk_i、被委托者j的身份、i制定的消息空间ω，输出一个委托证书；P输入委托证书、被委托者j的私钥sk_j，输出委托签名密钥skp，j可以通过skp生成i名义的委托签名。
- PS是委托签名生成算法，输入为委托签名密钥skp和消息M，输出委托签名$p\sigma$。
- PV是委托签名验证算法，输入公钥pk、消息M、委托签名$p\sigma$，如果签名合法则输出 1，否则输出 0。
- ID是委托签名身份认证算法，输入为一个合法的委托签名$p\sigma$，输出为委托者身份。

传统的委托方案运作方式类似于矿池，要求委托者将权益转到被委托者的

账户，在收集到全部委托者权益后，被委托者参与共识过程，并且将收益按照委托权益比例分发给每位委托者。这种方案是中心化的，而且缺乏安全保障，因为存在被委托者窃取权益的可能性。而委托签名算法可以有效解决这一问题，任何人都可以使用这一算法将其权利委托给可信方，不同的地方在于，权益仍然在委托者手中，仅仅是将签名的权利释放出去而已，这种方式更加安全可靠。

3.2 三重 ECDSA 委托签名算法

这一部分将介绍我们的基于 ECDSA 的委托签名算法。它被称为三重 ECDSA 委托签名算法，因为它在普通签名、权利委托、委托签名中均使用 ECDSA 算法。假设原始签名者 Alice 的密钥对为(pk_i, sk_i)，委托签名者 Bob 的密钥对为$\left(pk_j, sk_j\right)$。(G, K, S, V)与标准 ECDSA 算法一致，(D, P)、PS、PV算法如下：

Algorithm D

Alice 将数字签名的权利委托给 Bob，需做如下操作:

→ 构造消息空间ω

→ 选取随机数$k \in (1, q)$

→ 计算 $R = kG = (x_r, y_r)$

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

→ 设置 $r = x_r$

→ 计算 $s = k^{-1}(h + r \times sk_i)$

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

Algorithm P

Bob 获取委托签名密钥，需做如下操作：

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

→ 计算 $h = H(pk_i||pk_j||\omega)$

→ 计算 $V(pk_i, h, (R, s))$，如果输出为 0，$cert$ 不合法，失败

→ 否则计算 $skp = s + r \times sk_j$

Algorithm *PS*

Bob 进行委托签名，需做如下操作：

→ 确定签名消息 M

→ 选取随机数 $k_p \in (1, q)$

→ 计算 $R_p = k_p G = (x_p, y_p)$

→ 计算 $h_p = H(M)$

→ 设置 $r_p = x_p$

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

→ 输出委托签名 $proxy_{sig} = (M, (R_p, s_p), cert)$

Algorithm *PV*

第三方验证委托签名的合法性，需做如下操作：

→ 解析签名 $proxy_{sig} = (M, (R_p, s_p), cert)$

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

→ 验证 $M \in \omega$，如果不成立则失败

→ 计算$h = H(pk_i||pk_j||\omega)$

→ 计算 $V(pk_i, h, (R, s))$，如果输出为 0，则证书不合法，失败

→ 否则计算 $pkp = s \cdot G + r \cdot pk_j$，$R = (x_r, y_r)$，$r = x_r$

→ 计算 $V\left(pkp, M, (R_p, s_p)\right)$，如果输出为 1，则委托签名 $proxy_{sig}$ 合法

现在对三重 ECDSA 委托签名算法的正确性进行证明：

定理：任意消息空间ω，消息$M \in \omega$，用户i将签名权利委托给用户j，并且用户j对消息M进行了签名，那么

$$PV\big(PS(skp, M)\big) = 1$$

证明: 由(D, P)、PS、PV的定义，我们有以下等价条件：

$$\begin{aligned} & PV\big(PS(skp, M)\big) = 1 \\ \Leftrightarrow\ & V\left(pkp, M, \left(R_p, s_p\right)\right) = 1 \\ \Leftrightarrow\ & V\big(pkp, M, S(skp, M)\big) = 1 \\ \Leftrightarrow\ & pkp = skp \cdot G \\ \Leftrightarrow\ & s \cdot G + r \cdot pk_j = \big(s + r \times sk_j\big) \cdot G \\ \Leftrightarrow\ & pk_j = sk_j \cdot G \end{aligned}$$

显然$pk_j = sk_j \cdot G$成立 □

3.3 委托方案

星系共识的委托机制基于三重 ECDSA 委托签名算法实现，通过区块链进行数据的读取，借助智能合约进行数据的验证，无需在建立可信通信联系，更加高效。

假设 Alice 的密钥对为(pk_i, sk_i)，Bob 的密钥对为$\big(pk_j, sk_j\big)$，Alice 将数量为a的 WAN 锁定t时间参与到共识过程中。Alice 希望将其签名权利委托给 Bob，由 Bob 代其参与共识。我们建立一个专门的智能合约 Proxy_SC 去完成整个过程，这个智能合约需要进行一些计算和存储关键数据。整体流程如下：

委托方案

Alice 进行如下操作：

→ 设置消息空间 $\omega = \perp$

→ 选取随机数 $k \in (1, q)$

→ 计算$R = kG = (x_r, y_r)$

→ 计算$h = H(pk_i||pk_j||\omega)$

→ 设置 $r = x_r$

→ 计算$s = k^{-1}(h + r \times sk_i)$

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

→ 向智能合约 Proxy_SC 发送一笔带有数据$(a, t, cert)$的交易tx

收到tx后，Proxy_SC 做如下计算和验证：

→ 解析交易载荷$(a, t, cert)$

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

→ 验证pk_i 是否与发送 tx的地址吻合

→ 计算$h = H(pk_i||pk_j||\omega)$

→ 计算$V(pk_i, h, (R, s))$，如果结果为 0，则$cert$不合法，失败

→ 否则计算$pkp = s \cdot G + r \cdot pk_j$，$R = (x_r, y_r)$，$r = x_r$

→ 计算截止时间 $t_d = t_{now} + t$

→ 将数据元组 $(pk_i, pk_j, \omega, (R, s), pkp, a, t, t_d)$ 存储在 Proxy_SC

→ 从 Alice 的账户中扣除数量a的 WAN，将其锁定t时间

Bob 做以下验证和计算去获得委托签名密钥：

→ 扫描 Proxy_SC 存储空间找到包含pk_j的数据元组

→ 解析数据元组 $(pk_i, pk_j, \omega, (R, s), pkp, a, t, t_d)$

→ 如果 $t_{now} > t_d$，当前时间已经超过截止时间，中断操作

→ 否则计算$skp = s + r \times sk_j$

到此为止，PoS 协议的一个新的参与者(pkp, a, t)诞生。当pkp被选为 Random Number Proposer 或 Epoch Leader 时，Bob 将正常参与到协议中，并且使用skp进行签名。

委托截止时间t_d到期后，(pkp, a, t)将被从 Community 移除，锁定的数量为a的 WAN 将会随同收益返回到 Alice 的账户，Bob 也会分享其中一部分收益。

3.4 优势

- 通用性

我们的委托方案在标准签名、权利委托和委托签名中均使用ECDSA签名算法，这一算法也在区块链领域广泛使用。因此本方案与已有的区块链技术架构完全兼容。无论是直接注册加入到共识机制中，还是通过委托机制加入，对参与者出块的验证逻辑完全一致。

- 非交互

委托的过程是非交互的，因此原始签名者和被委托者之间不必建立安全的通信信道，这样节省了带宽，使其在区块链共识应用中更加可性。

- 高效

在本文之前就有基于 ECDSA 的委托签名方案，最出名的方案是由 Ming-Hsin Chang、I-Te Chen 和 Ming-Te Chen 在论文“Design of Proxy signature in ECDSA”[10]中提出，我们用三位作者名字缩写MIM代指这一方案。我们方案的委托过程与 MIM 方案相似，但是委托验证过程与标准 ECDSA 算法一致，比MIM方案更加高效。委托验证过程的计算量比较见下表：

计算类型	MIM方案	本文方案
有限域计算	4	3
椭圆曲线点标量乘	3	2
椭圆曲线点加	1	1
椭圆曲线点比较	1	1

- 收益分配清晰

原始签名者将其签名权利委托给被委托者之后，一个新的公钥pkp生成，它由被委托者控制，而且与原始签名者的关系也一同保存在智能合约 Proxy_SC 中。当pkp被选为Random Number Proposer或Epoch Leader时，被委托者将会代替原始签名者参与共识。共识过程中的收益将会在二者之间分配。即使某个体被多个个体委托权利，共识收益的分配仍然非常清晰，不会混淆。

- 消息空间限制

我们设计的三重 ECDSA 委托签名算法通过消息空间限定被委托者签名的范围，虽然在本委托方案中并未使用（设置为空），但是能为未来潜在的应用提供理论基础，比如实现在不同状况下将签名权利委托给不同个体等等。

第四章 经济激励机制

WAN 发行总量（210 million）的 10%（21 million）将被拿出作为追加奖励来鼓励参与者积极参与万维链共识，起初这部分奖励将占参与者共识收益的大部分，但随着万维链的发展，链上交易越来越多，这部分额外的奖励就将变成总奖励中的小部分，交易费将成为收益主体。

4.1 基本原则

经济激励机制依据下述原则设计：

（1）贡献越多，收益越多；

（2）消极和恶意参与者将通过惩罚被抑制；

（3）收益和锁定权益延迟返还以提高安全性；

（4）营造公平的良性竞争环境；

（5）确保合理和相对稳定的收益。

4.2 共识经济激励模型

追加奖励的 50%将在第一个 5 年内投放，随后每五年减半，五年对于万维链的发展和共识参与者的长期收益来说都是一个深思熟虑的合理时间。追加奖励和年份的关系是：

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

这里R_t代表追加奖励，a代表第一个时间区间（5 年）投放奖励量，b代表减少比率。在我们的设计中，R_t是 21 million，b是 50%，a是 10.5 million，对于共识参与者来说，这应该是一个比较有吸引力的激励方案。下图说明追加奖励投放情况，其中纵坐标代表已投放追加奖励（WAN），横坐标代表时间（年）。

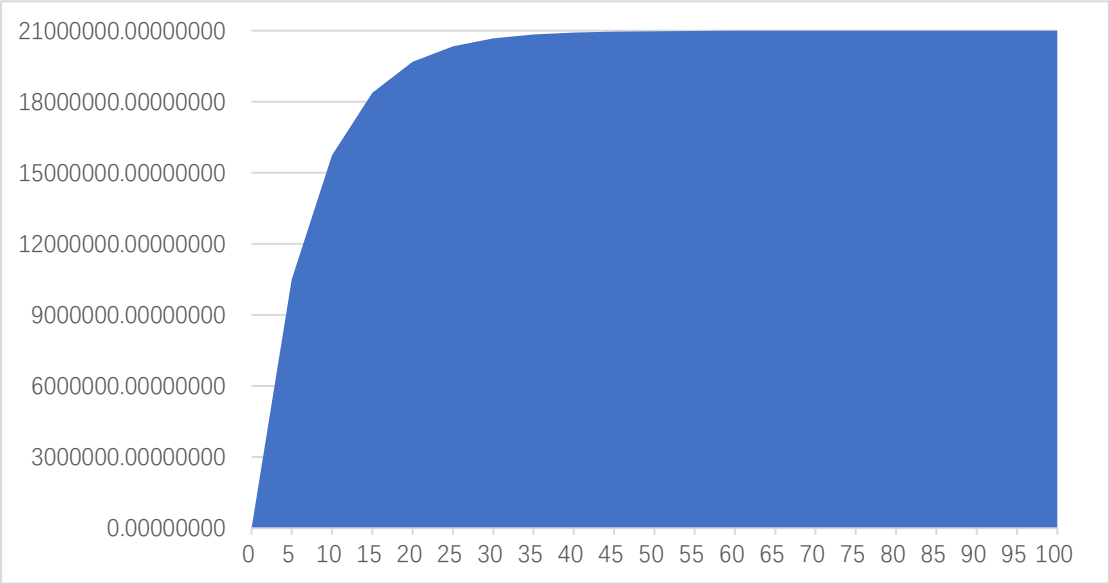

图 5: 追加奖励年化投放量

每个 5 年时间段内部，追加奖励将在每个 epoch 中均匀分配。以第一个 5 年为例，如果这 5 年内追加奖励总量为a，一共有K个 epoch，那么每个 epoch 的奖励R_e为：

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

R_e将被奖励给诚实参与的 Random Number Proposers (RNPs)和 Epoch Leaders。与 PoW 不同，1 个区块内的交易费并不会都奖励给出块者，交易费中的一部分将被分配给 RNPs，因为随机数的生成是确保安全性的重要一环，如果只有追加奖励的部分奖励给 RNPs，由于追加奖励每 5 年减半，RNPs 的收益将持续减少，最终失去参与的动机和兴趣。这样，对于每个 epoch 内的奖励分配将在下一个 epoch 的第一个区块进行结算，保证了参与者行为统计的一致性和连贯性。

活性系数

在星系共识中，两类参与者为达成共识做出了贡献，RNPs 负责生成随机数来更新 Random Beacon，Epoch Leaders 负责生成秘密信息序列和生成区块，两者都需要给予奖励来鼓励诚实行为。因此，参与者被鼓励参与随机数生成、秘密信息序列生成和在他负责的 slot 内生成区块，而为了依据参与者的活跃度进行精准奖励，我们定义了活性系数的概念。

我们把 RNP 的活性系数定义为α_{RNP}。如果一个 RNP 诚实的参与了随机数生成的三个阶段工作，那么α_{RNP}就是 1，否则是 0。类似的，定义 Epoch Leaders 的活性系数为α_{EL}，如果一个 Epoch Leader 诚实参与了秘密信息序列生成的两个阶段工作，那么α_{EL}就是 1，否则为 0。这两个系数会影响 RNP 和 Epoch Leader 从随机数生成和秘密信息序列生成中得到的收益。

最后，我们为 Epoch Leader Group 这一群组定义活性系数α_{ELG}。我们的协议中，在每个 epoch 的起始，只有 Epoch Leaders 知道 Slot Leader 的归属，而在区块被提出之后，Slot Leader 的合法性就可以被公共验证。那么如果某些区块没有被提出，只有 Epoch Leaders 知道谁应该对此负责，而欲将此部分信息转变为公共验证信息需要增加大量的工作，得不偿失。所以我们定义 Epoch Leader Group 的活性系数，以一种相对理智的方式来解决这一问题。如果 1 个 epoch 内含有K_e个 slot，而只有K_v个合法区块被提出，那么这个 epoch 里 Epoch Leader Group 的活性系数α_{ELG}为

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

这个系数将会影响生成区块所得收益。

收益划分

对于 1 个 epoch 内部，总的收益将被划分给 RNPs 和 Epoch Leaders。Epoch Leader 需要参与 ULS 算法同时在被选中的时候生成区块，RNP 需要参与 DKG1、DKG2 和 SIGN 三个阶段工作，所以我们设定总收益中的λ奖励给 Epoch Leaders，$(1-\lambda)$奖励给 RNPs，而 Epoch Leaders 所获奖励中的$(1-\mu)$将用来奖励他们参与秘密信息序列生成。

因此，如果 1 个 epoch 里有N个 Epoch Leaders 和K_e个 slot，那么 1 个区块的奖励为：

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

这里R_e代表 1 个 epoch 的追加奖励值，T_s代表这一区块内的交易费。

1 个 epoch 内 Epoch Leader 生成区块所获奖励R_p为：

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

这里α_{ELG}是 Epoch Leader Group 的活性系数，$\sum R_s$代表那些由 Epoch Leader 生成的区块所对应的收益之和。

Epoch Leader 参与了前一个 epoch 进行的秘密信息序列生成将得到的收益为：

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

虽然$(1-\mu)$很低，甚至只有 10%，但这部分收益也必须提供来鼓励 Epoch Leader 积极参与秘密信息序列生成，越多 Epoch Leaders 参与秘密信息序列生成，安全性越高。

RNP 参与随机数生成获得奖励为：

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

这里α_{RNP}是 RNP 的活性系数，N_r是 RNP 个数，$\sum T_s$代表这个 epoch 内的所有交易费。

理想状态下（每个 slot 有一个合法区块，随机数生成和秘密信息序列生成完全参与），总的奖励值为$(R_e + \sum T_s)$。如果有惰性行为发生，奖励将减少，剩余部分的奖励将返回追加奖励池以用作后续 PoS 奖励。

4.3 委托机制经济激励模型

由于委托机制是对小额 WAN 持有者的一种必要设置，我们就需要为其设计相应的激励模型。如果参与者想要接受委托，他需要公布一个收益率m，即如果委托者被选为 RNP 或 Epoch Leader 并得到收益R，那么被委托者将分得$R \cdot m$，委托者将分得$R \cdot (1-m)$。这样的简单框架下将促使被委托者通过自然竞争维持一个类似的收益率。此外，被委托者能够接受的委托权益总量正比于其本身持有的权益量。

另一个目的，为了避免类似 Bitcoin 等 PoW 共识中出现的大规模矿池，我们尽量避免出现大规模权益池，以防权益集中带来的安全风险，我们在模型中要鼓励公平的权益池形成。为实现这一点，我们为每个被委托者设置委托权益峰

值s_0（例如发行总量的 10%），如果被委托者接受委托权益超过s_0，收益将减少。

如果被委托者权益总量为s，委托者将获得的收益为：

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

被委托者收益为：

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

所有潜在被委托者的收益率和当前持有权益量将形成一个排序列表，供委托用户做出最优选择。由于委托超出委托峰值后委托者和被委托者的收益都将降低，系统自然促进形成更均匀的权益分布，降低了合谋风险，鼓励更多的活跃用户参与星系共识经济生态系统。

第五章 攻击抵抗分析

- Double spending attacks

在 double spending attacks 中，攻击者希望令两笔矛盾交易都生效，想实现这种攻击只有两种情况：(i)含有两笔矛盾或一笔与之前生效交易相矛盾交易的区块被确认合法生效；(ii)出现含有矛盾交易的两条合法分叉链。对于(i)，协议要求对任何收到的区块进行合法性检查，包含任何矛盾交易的区块将被认定无效。同时我们的 chain-based 共识只接受最长链作为合法连，不会出现两条合法分叉链的情况，进而避免 double spending 发生。

- Grinding attacks

在 grinding attacks 中，攻击者希望影响出块者选择过程以提高自己被选中的几率。我们的共识中，出块者选择基于 random beacon 和两阶段的 ULS 算法，在随机数生成过程中我们采用了门限签名方案，任何参与者只能决定自己的签名部分，只要有至少一个参与者是诚实的，攻击者就无法决定最终随机数结果，同时只要参与者中攻击者数量低于门限值就无法预知最终随机数结果，所以 Epoch Leader 的选择无法通过控制随机数生成进行影响。在两阶段的 ULS 算法的 SMA 生成过程中，被选中的参与者只能选择是否广播他们的数据，如果不公布将失去部分收益。而且 SMA 的生成在 SIGN 阶段之前，而 Epoch Leaders 排序在 SIGN 阶段之后，攻击者没有信息优势来影响 ULS 算法的选择结果。

- Transaction denial attacks

在 transaction denial attacks 中，攻击者希望阻止一笔合法交易被确认，我们强调诚实节点不会拒绝任何合法交易，所以如果要实现 transaction denial attacks，就意味着所有的 Slot Leaders 都是恶意的。由于诚实主体假设和 *follow-the-stake-rate* 选择方案，攻击概率是随着时间以小于 1/2 的底数指数下降的，攻击者实现攻击的概率趋近于零。

- Bribery attacks

在 bribery attacks 中，攻击者希望通过腐蚀诚实参与者为其所用来实现某种恶意目的，例如实现 double spending attacks。在我们的共识中，理性的参与者由于两点原因不会接受这样的贿赂腐蚀。一是诚实参与者接受这样的贿赂腐蚀而变成恶意参与者的话，他将失去抵押在智能合约中的权益。二是恶意行为会损害 Wanchain 的经济生态系统，进而减低 WAN 的价值，参与者的 WAN 就会贬值。所以理性参与者不会接受贿赂腐蚀。而且只要保证诚实主体，bribery attacks 也无法影响到共识的安全性。

- Long-range attacks

在 long-range attacks[15]中，攻击者希望从链上一个之前较远的位置开始重构一条合法链，这样就可以使链上数据与当前真实状态不同，进而实现双花攻击等。协议中我们在有效链上设置了一些检测点（check point），随着链不断延长，在最后一个检测点之前的数据将不再改变，即新出现的检测点的区块无效，所以攻击者无法实现 long-range attacks。

- Nothing at stake attacks

在 nothing at stake attacks 中，参与者会在多个分叉上产生新的区块，无论最终哪个分叉被确认有效都将获得收益。由于 PoS 共识中生成区块几乎没有消耗，这种攻击就很容易发生，而在 PoW 共识中，矿工不会浪费算力来在不同分叉上产生区块，这样很可能让他完全得不到收益。同时，这种攻击常发生在某一时间段或某一区块高度有多个合法出块者的情况下，例如通过 VRF 进行出块者选择的时候，然而在我们的共识中，某一时间段内只有唯一合法出块者，几乎不会出现分叉情况，参与者就没有动机发动 nothing at stake attacks。

- Past majority attacks

在 past majority attacks 中，攻击者希望腐化某些之前的参与者以取得在过去

某段时间里的权益优势。我们假设当前的诚实权益大多数是合理的，而为了在 past majority attacks 中获益，攻击者需要与过去某一时间的大部分权益持有者重新产生区块来代替之前已有的区块，这就类似于前面提到的 long-range attacks，会与检测点（checkpoint）产生冲突，这种新产生的区块会因为位置早于检测点而不被接受，所以我们的共识不会受到 past majority attacks 影响。

- Selfish-mining

在 selfish-mining 中，参与者会私下保留一个合法区块然后提前去构造下一个合法区块，由于可以获得计算时间上的优势，这种情况常发生在 PoW 共识中，然而在 PoS 中尤其是我们的共识之中无需如此。由于 Slot Leaders 是在一个 epoch 开始时由 random beacon 和 Epoch Leaders 产生的秘密信息决定的，无论参与者是否广播新的区块都无法影响下一个 slot 中出块者的选择，所以私下保留当前的合法区块是无利可图的。事实上，如果超出了一个 slot 时间窗口没有广播新的合法区块的话，出块者很可能失去此区块的收益。所以发起 selfish-mining 是不理智的，及时发生也不会影响我们共识的安全性。

第六章 星系共识优势

1.可证明安全

星系共识基于 Ouroboros 的可证明安全模型设计，在保留其可证明安全共识框架下对核心密码组件进行了设计提升。

2.自然分叉概率低

星系共识采用 ULS(unique leader selection)算法进行出块者选择，与 VRF 算法不同，ULS 算法在保持出块者匿名的同时，保证了出块者的唯一性。

3.安全随机性引入

随机性的引入对于保证共识的安全性有着积极的影响，星系共识利用基于门限签名方案的 random beacon 引入随机性，而 random beacon 的安全性由两方面保证：一是只要至少 1 名参与者是诚实的即可保证安全，降低了诚实主体假设要求；二是保证 G.O.D (Guaranteed Output Delivery)性质，即使部分参与者掉线，只要在线参与者数量高于预设门限值就可以成功执行。

4.合理的权益设计

PoS 共识中合理的权益设计是一个很有意义的考量，为保持参与者的稳定性和活性，我们要求参与者将自身持有的 WAN 锁定在一个特殊的智能合约之中以参与共识，锁定 WAN 的数量、锁定时间和剩余锁定时间比例都是用于计算参与者权益值的参数，这样的设计在 account model 下模拟了币龄的概念同时保证了星系共识参与者的稳定性。

5.完整的委托机制

星系共识采用我们全新设计的非交互高效 ECDSA 委托签名算法，实现了完整的委托机制，限制签名空间且与现有区块链签名体系相兼容，保证了任何 WAN 持有者皆可参与共识，提升了 Wanstake 的活性。

6.清晰可信的经济激励模型

星系共识拥有对共识参与者完整的经济激励模型，由于参与者在链上进行

信息交互，他们的行为也将被反映在链上，我们引入活性系数的概念来评估参与者的表现，活性越高收益越多，这样的经济激励机制是清晰明确并令人信服的。

参考文献

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.

附录 1

在第一章中，我们列出了共识参与者权益计算函数需要满足的 4 个性质，并给出候选函数：

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

其中 σ_L是L的单调递增函数，很显然满足前 3 条性质，下面证明其满足性质 4：

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

在性质 4 中，t'是已经流逝的锁定时间，$t = \frac{L-t'}{L}$。以$\int_{t\prime=0}^{L_2} H(\omega, L_2, t)dt'$为例计算积分，此时利用$t'$计算函数中的锁定时间剩余比例：

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

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

则积分的凹函数关系转化为：

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

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

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

因为σ_L是L的单调递增函数，所以得证候选函数满足性质 4。

事实上我们希望共识参与者倾向于选择一段长时间而不是多段短时间进行锁定，我们认为H函数的积分作为权益累积效应可以在某种意义上反应参与者的收益预期，所以性质 4 预示着共识参与者选择一段长时间相比于多段短时间锁定将得到更多收益，从而鼓励参与者锁定一段长时间参与共识。

附录 2

密码学中，零知识证明是用于在不提供关键信息的情况下保证数据可验证正确的一种重要手段。类似于[12]，我们采用了椭圆曲线上的零知识证明方案，其正确性和安全性基于椭圆曲线上的 DDH 假设。

首先我们考虑一个$2N(N \geq 2)$长的点列$pa = (P_1, Q_1, \dots, P_N, Q_N)$，其中$P_i, Q_i \in E(F_p), \#E(F_p) = n$。零知识证明保证了存在$\alpha$使得$Q_i = \alpha \cdot P_i, i = 1,2, \dots, N$。证明构造方法如下：

- 生成随机数$\omega \in [1, n]$，计算

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

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

- 计算

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

零知识证明结果为$\pi = DLEQ(P_1, Q_1, \dots, P_N, Q_N) = (e, z)$.

证明验证方法如下：

- 对于点列$(P_1, Q_1, \dots, P_N, Q_N)$和证明$(e, z)$，计算

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

- 计算

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

若$e = e'$，则证明有效。观察易得：

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

零知识证明的生成和验证较简单，在正文算法和协议表述中以 Gen_proof 和 Ver_proof 代表零知识证明的生成和验证过程。

附录 3

在 3.5 节中，我们描述了 Epoch Leaders 的选择过程，有必要说明共识参与者通过直接选择还是两阶段选择被选为 Slot Leader 的概率是一样的。

先定义场景，Community 中有n个共识参与者，其中参与者U_i被选为 Slot Leader 的概率是p_i，Epoch Leaders 集合元素个数为$N(N < n)$，那么对于参与者U_i来说，直接被选为 Slot Leader 的概率就是p_i。下面我们计算两阶段选择下参与者U_i被选为 Slot Leader 的概率，并证明其与p_i相等。在 3.5 节中强调，Epoch Leaders 集合为多重集，且第二阶段为等概率选择，那么有

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

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

由于

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

有

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

综上得证，在 Slot Leader 选择过程中，直接选择与两阶段选择概率相同。