HNB

技术白皮书

基于区块链的下一代去中心化经济体

HNB
HASHED NEXUS BITECO

© HNB Foundation 2018Email: service@HNB.eco

目录

1 概述 4

2 术语说明 4

3 设计原则 5

4 整体技术架构 6

4.1 HNB 生态技术架构 6
4.2 HNB 底层链技术体系 8

5 身份管理 9

5.1 身份标识 9
5.2 身份注册 9
5.3 信任模型 10

6 共识机制 11

6.1 区块链不可能三角 11
6.2 HNB 共识算法 13
6.3 共识激励 24

7 数据存储 25

7.1 分布式存储 25
7.2 数据分片 26
7.3 存储可插拔 26
7.4 加密存储 26

8 多资产账本 27

8.1 多资产账本架构 27
8.2 账本模型 29

9 交易管理 30

9.1 交易验证 30
9.2 交易池 31
9.3 排序策略 31

10 智能合约 31

10.1 HNB 智能合约设计 31
10.2 合约管理 32
10.3 合约运行环境 32
10.4 形式化验证 33

11 节点管理 33

11.1 节点分类 33
11.2 全节点 33
11.3 记账节点 33

11.4 SPV 节点 34
11.5 节点授权管理 34
11.6 节点小节 34

12 网络通信 35

12.1 服务接入 35
12.2 P2P 组网 35
12.3 节点间通信 35

13 密码学算法 36

13.1 签名算法 36
13.2 摘要算法 36
13.3 加密算法切换 36
13.4 零知识证明 37
13.5 盲签名 38
13.6 环签名 39
13.7 同态加密 39

14 实体经济应用 40

14.1 DAPP 去中心化应用 40
14.2 商业隐私保护 42
14.3 跨链交互 43
14.4 链外极速网络 43
14.5 账本归档 44
14.6 并行记账 44
14.7 算法银行 44

15 参考文献 46

1 概述

HNB 是下一代基于区块链的去中心化经济体，旨在建立服务实体经济体的可持续发展的经济模型。从技术层面来讲，HNB 是一个支持具有多侧链架构且支持多数字资产流通的区块链技术体系，用于建立基于公有区块链的去中心化经济生态。基于 HNB 的业务理念和发展规划，我们对市场上现有的区块链项目和技术进行了分析和梳理，提出了实现 HNB 区块链经济模型的技术方案。

本技术白皮书详细阐述了 HNB 公有区块链的设计原则，整体技术架构，以及核心技术的实现方法。技术白皮书主要描述的核心技术包括身份管理，共识机制，数据存储，多资产账本，交易管理，智能合约，节点管理，网络通信和密码学算法等。同时结合实体经济应用，技术白皮书还介绍了 HNB 的去中心化应用 DApp 体系、应用场景例如算法银行等的方面的内容。

2 术语说明

术语	说明
HNB	HNB Share Token 是代表 HNB 经济实体所有权的数字资产。其总供应量固定为 10 亿
HGS	HGS Stable Token 是与美元等法定货币挂钩的支付媒介，由“HNB 储备委员会”发布并仅在 HNB 经济体内部流通
委员会成员	共识过程中可能成为提议者的候选人，通过 DPoS 选举产生
提议者	共识过程中，参与拜占庭容错共识算法的成员，即共识出块的节点
DPoS	英文全称 Delegated Proof of Stake，委托权益证明，通过选举委员会再通过权益证明方式进行共识的共识算法
加密抽签	即可验证随机函数英文全称 Verifiable Random Function，简称 VRF。用以完成提议者的随机选择，VRF 的返回值应尽力难以预测
零知识证明	英文全称 Zero-Knowledge Proof，证明者能够在不向验证者提供任何有用的信息的情况下，使验证者相信某个论断是正确的算法。
盲签名	英文全称 Blind Signature，是一种特殊的数字签名方案，可以保证被签名数据的保密性和不可追踪性，从而保护用户数据的隐私
环签名	可以实现签名者的无条件匿名性，即任何人都无法追踪到签名人的身份
同态加密	英文全称 Homomorphic Encryption 又称隐私同态，同态加密中对密文执行的运算等价于对明文执行了同样的运算

3 设计原则

HNB 被设计用于建立服务实体经济体的可持续发展的经济模型。HNB 为下一代基于区块链的去中心化经济体打造，目标为上亿消费者和商家的生态圈提供服务，其经济规模和社区成员会随着生态圈的扩大而不断壮大。因此，HNB 区块链的设计原则为：

可扩展性 Scalability

- 可支撑HNB社区的持续增长, 包括区块链底层节点数的增长、更多商户的接入、用户数增加等

高性能High Performance

- 支持数百万用户和并发事务，可支持大并发交易峰值的处理，降低交易确认延时，秒级共识效率

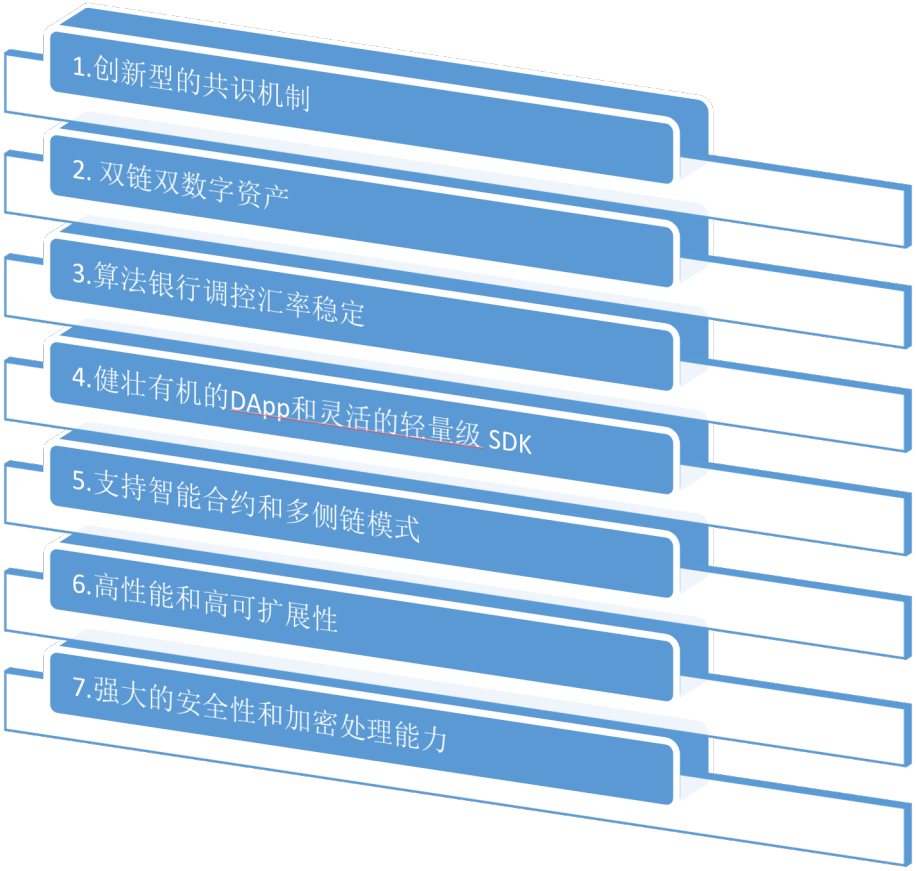
安全性 Security

- 在应用和基础设施级别的强安全性，保护商业数据和个人数据的隐私性

互操作性 Interoperability

- 支持与现有系统的快速集成和持续开发，便于与其他区块链系统和应用系统对接

综合上述应用场景的考虑，我们设计 HNB 区块链具备以下的特性：



- 创新的共识机制，可支持多节点快速共识
- 双链双数字资产：HNB 和 HGS; HGS 与法币锚定的调控机制；
- .算法银行调控汇率稳定
- 健壮有机的 DApp 集和灵活的轻量级 SDK 支持商户应用对接集成
- 支持智能合约和支持多侧链模式
- 高性能和高可扩展：支持.链外极速网络等技术
- 强大的安全性和加密处理能力

4 整体技术架构

4.1 HNB 生态技术架构

HNB 生态技术架构遵循层次化、模块化的设计理念，强调框架的可扩展性、可交互性、性能和安全性。HNB 生态技术架构是包含底层区块链模块以及其相关的外围生态组件的集合体，是基于用户层、HNB 业务服务层、区块链核心层、基础组件层和通用服务层的结构化设计。

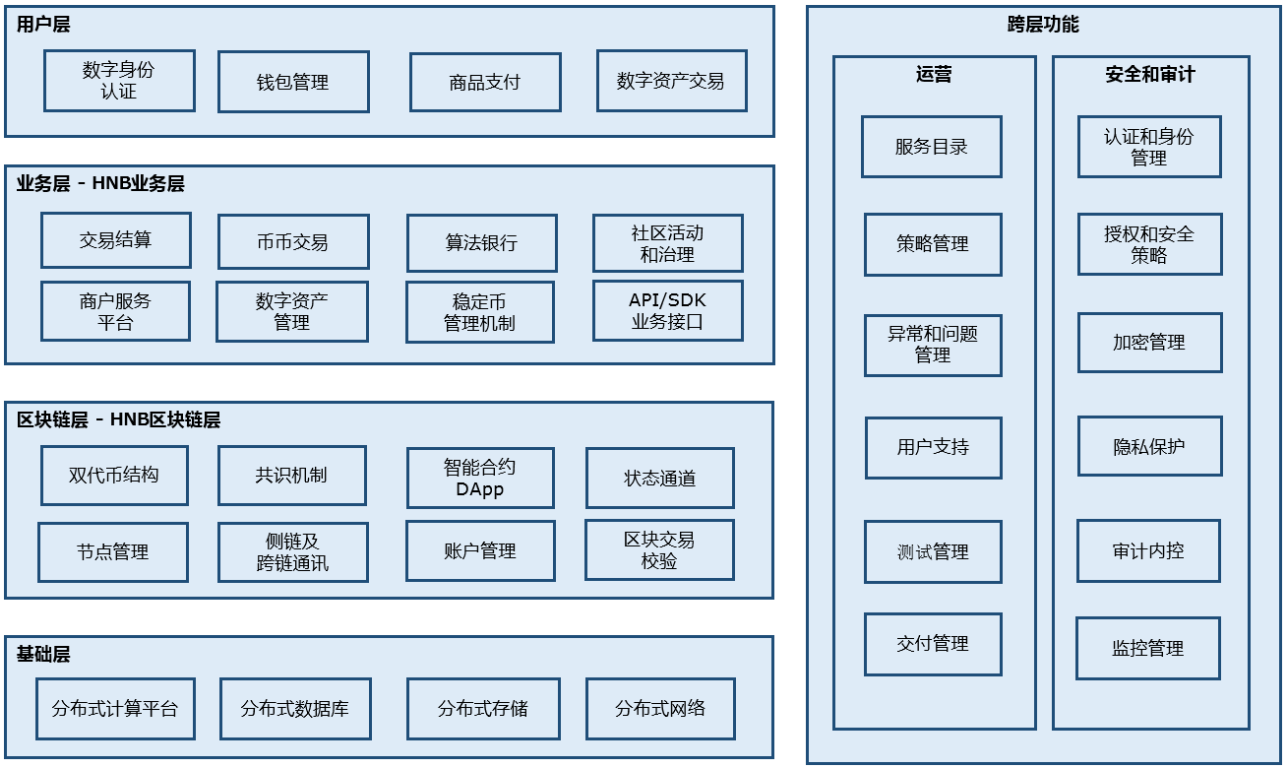


图 1　HNB 区块链生态技术架构

用户层	用户层是终端用户的接入点。用户通过用户层访问 HNB 区块链服务。从技术角度来看，HNB 终端用户服务的特性由后续章节的去中心化应用 DApp 服务支持，包含数字钱包、数字资产管理等功能，将在第 14.1 小节 DApp 去中心化应用中进行介绍。另外该层还包含用户管理的功能，将在第 5 章中详细阐述。
HNB 业务服务层	HNB 业务服务层实现了 HNB 去中心化经济社区的所有核心业务逻辑。从技术角度来看，HNB 的 DApp 去中心化应用为该层提供服务支持，将在第 14.1 小节 DApp 去中心化应用中进行介绍。
区块链核心层	区块链核心层，即 HNB 区块链底层，具备共识机制、账本管理、多资产管理、资产交易管理、智能合约等区块链核心功能，其技术方案将分别在第 6 章至第 10 章进行介绍。另外，HNB 区块链在隐私保护、跨链交互、链外极速交易处理、并行记账等特性进行创新，详细方案可以参见第 14 章的介绍。
基础组件层	主要为区块链核心层提供分布式基础组件，包括：分布式数据库、分布式存储、分布式网络。其中，数据库和存储服务支持可插拔，为社区参与者提供了更多的选择空间和更广泛的业务支撑能力。存储可插拔将在第 7.3 小节详细介绍。
通用服务层	**通用服务层**为上述四层的模块提供服务，包括：安全、审计、运维等功能。安全加密算法相关的方案将在第 13 章介绍。

4.2 HNB 底层链技术体系

HNB 底层区块链技术体系，是针对生态技术架构的区块链核心层、基础组件层和通用服务层的进一步细化设计。

HNB 区块链技术体系同样采用层次化和模块化的设计思想，分为合约层、共识层、激励层、数据层、网络层、安全层、基础层的七层架构。

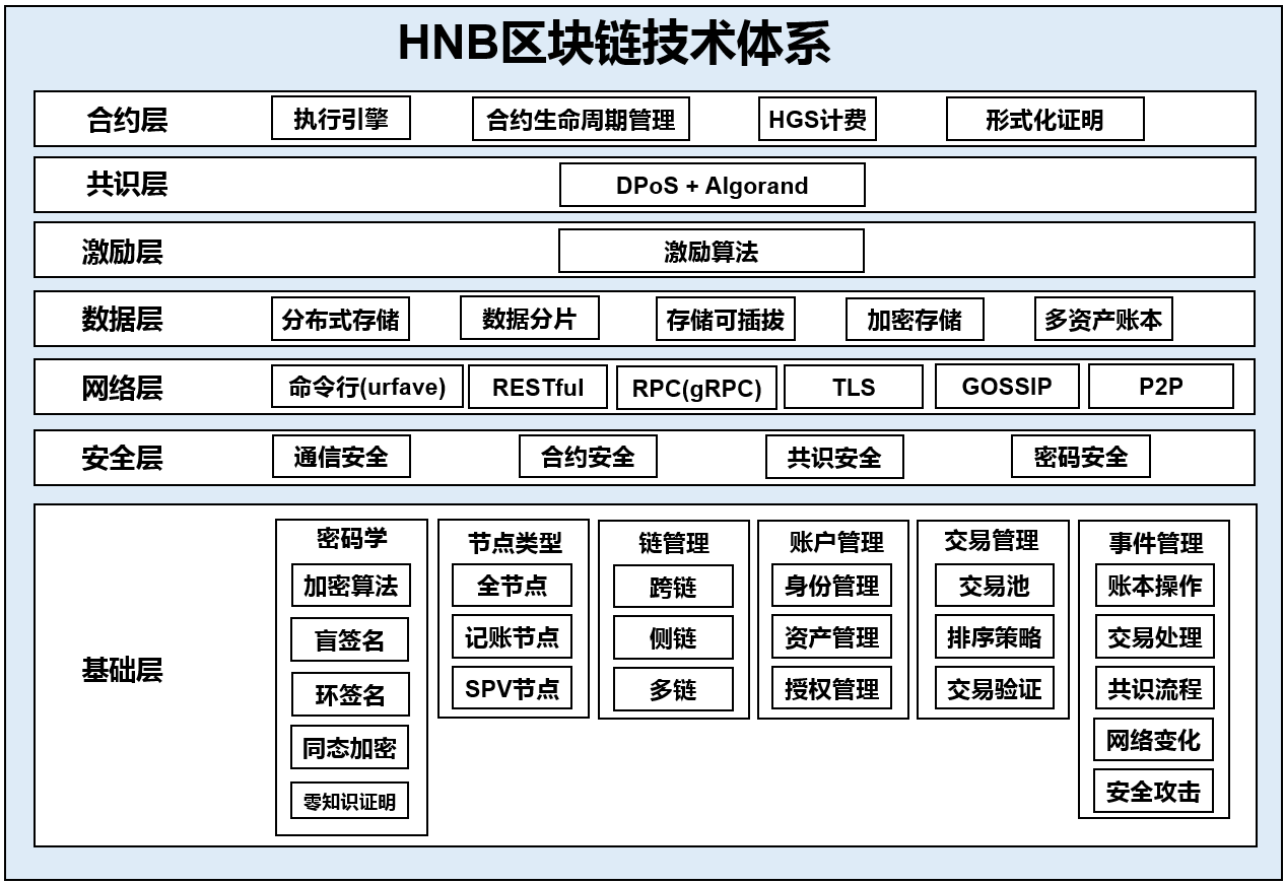


图 2　HNB 区块链底层技术体系

合约层	合约层即智能合约服务层，为智能合约的生命周期管理、运行成本管理、运行环境、形式化证明逻辑提供服务。HNB 的智能合约支持采用 golang、JavaScript 等高级语言编写，满足图灵完备的特性，其成熟的运行环境保障合约运行的安全可靠。该部分内容将在第 10 章详细阐述。
共识层	共识层，采用 DPoS 与 Algorand 结合的混合共识算法，鼓励 HNB 社区内贡献多的节点参与记账，同时基于加密抽签保证共识提案人的公平选举，最终以 bft 共识算法保证共识的高效和一致。该部分内容将在第 6 章介绍。
激励层	激励层，基于 HNB 生态中算法银行的设计，公开、透明、统一的管理激励策略，对社区内贡献度较高的节点，进行激励 HGS 数字资产。算法银行将在第 14.7 小节详细阐述。
数据层	数据层，作为 HNB 基础设施层中分布式数据库和分布式存储模块的细化，在区块链底层技术架构中体现出来。主要为区块链的分布式数据库、分布

	式文件存储提供基础存储服务，同时支持 HNB 双资产账本管理。该部分内容将在第 7、8 章中介绍。
网络层	网络层，作为区块链底层的网络通信基础模块，主要包含：区块链底层对应用层提供的访问接口、区块链节点间 P2P 组网、区块链节点间广播数据的用途。该部分内容将在第 12 章详细介绍。
安全层，	安全层，从通信安全、数据安全、合约安全、密码安全的维度分析 HNB 对高安全区块链底层的方案设计，可分别参考第 7、10、12、13 章的相关内容。
基础层	基础层，为区块链底层提供通用的基础服务。包含：密码学算法库、节点管理、多资产链管理、账户管理、交易管理、事件管理等

5 身份管理

5.1 身份标识

5.1.1 去中心化身份标识

传统身份标识是对人本身进行识别，识别方式有：身份证、户口簿、驾照等。随着技术发展，指纹、DNA 等新兴身份识别方法逐步应用。在互联网世界中，所有事物非实体存在，需要网络标识来映射与管理实体身份。

HNB-ID 是根据用户的 HNB 公钥进行数学算法计算得到。HNB-ID 不需要通过中心化系统计算生成，也无需通过中心化系统进行冲突检测，由用户自身生成与管理。

5.1.2 隐私保护

HNB-ID 生成算法决定根据 ID 无法推导公钥信息，根据公钥信息也无法与用户其他身份属性信息进行关联。真正意义的做到了保护用户隐私。

5.1.3 自证身份

HNB-ID 与用户公钥具有一一映射关系，用户可通过身份验证算法证明该 HNB-ID 属于某线下实体，以便进行其他商业活动。

5.2 身份注册

HNB 支持可配置的节点准入机制，加入 HNB 生态需要进行节点授权。该授权需要由 HNB 证书授权中心 CA (Certificate Authority)中心颁发证书，表示允许某 HNB-ID 加入到 HNB 生态。与 HNB-ID 关联的属性信息将不会被发送至 HNB 公链底层。节点启动、网络组建时，将带有 CA 颁发的证书信息发送至其他节点，表明被授权身份。

5.3 信任模型

5.3.1 密钥生成安全

密钥生成过程在离线钱包中进行，用户私钥不能以任何形式在网络上传输，保证攻击者无法盗取用户密钥，杜绝盗取 HNB 的可能。

5.3.2 密钥托管安全

根据用户需求，HNB 支持密钥托管服务。用户可将密钥交给分布式私钥服务器进行管理，密钥存储过程使用对称加密，加密密钥由用户以其他形态进行管理。即使私钥服务器被攻破，攻击者无法还原用户私钥。

5.3.3 身份注册安全

用户进行身份注册过程中，用户将实体属性信息与 HNB-ID 进行绑定，并发送至证书授权中心 CA 中心，CA 中心验证实体身份后颁发证书。HNB 系统对用户实体属性信息进行非对称算法加密，中间人无法解析交易报文并获取关联信息，从而在最大程度上保护用户隐私。

如下图所示，Alice 身份注册时，需要将实体信息与用户公钥信息进行打包，使用证书授权中心 CA (Certificate Authority)的公钥对信息数据进行加密，保证传输安全。同时使用 HTTPS 协议进行安全通信，这样使用了双重加密机制来保护数据安全和用户隐私。

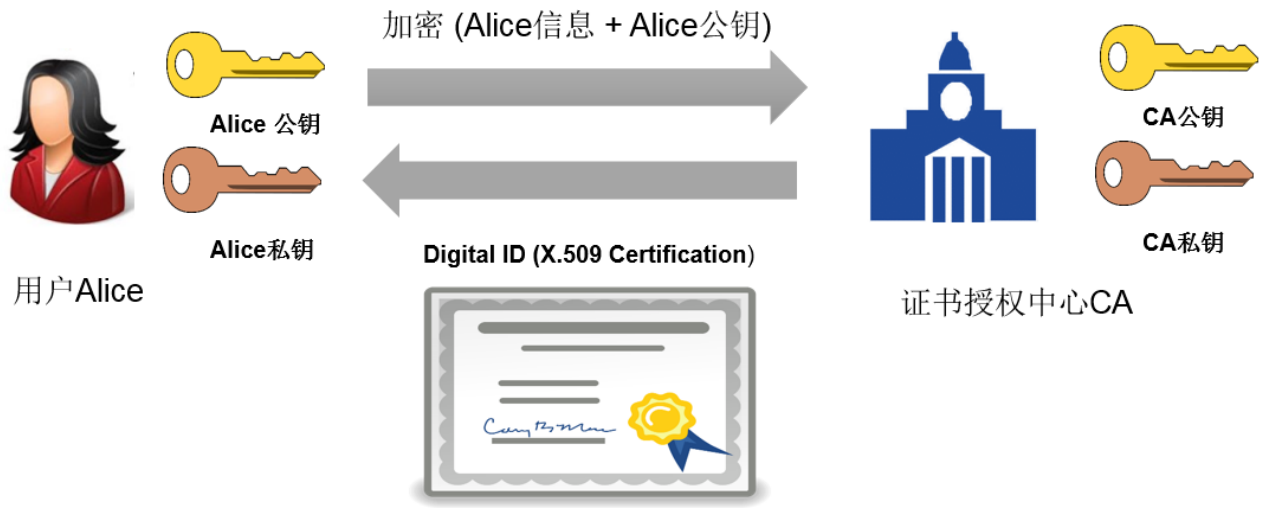


图 3　身份注册流程

6 共识机制

对于任何一个去中心化自治系统，其共识机制往往是社区信任建立的根基。共识是用来来描述了整个区块链系统中多个节点之间，对某个状态达成一致结果的过程。要保障系统满足不同程度的一致性，往往需要共识算法来达成。有效的共识算法解决的是对某个提案达成一致意见的过程，这里的提案可以指任何能达成一致的信息，无论是基础的区块产生和校验还是社区运行的规则。

6.1 区块链不可能三角

CAP 定律（一致性 Consistency，可用性 Availability，分区容错性 Partition Tolerance Theorem）是指在一个分布式计算机系统中，一致性，可用性和分区容错性这三种特性，无法保证同时得到满足，最多同时满足其中的两个特性；系统设计者需要根据不同的需求进行取舍。

图 4　CAP 定律

在区块链共识算法设计上，我们把焦点集中在了 SDE(安全 Security, 去中心化 Decentralization, 和效率 Efficiency) 三大核心特性上。 安全，去中心化和效率构成一个不可能三角形，三者不能同时满足。如果我们设计一个安全的去中心化系统，它必然是以牺牲效率，以付出能源与计算力为代价(比如比特币的工作量证明机制)。我们要求设计一个既保证效率又去中心化的密码学货币，它将会是不足够安全的。

目前的主流的共识机制为 PoW / PoS / DPoS / PBFT，经典的 PoW,满足安全和去中心化的要求,但是效率非常低下，达成共识的时间很长; PoS 满足去中心化和效率，但是安全性不足，其确认仅仅为概率上的表达而没有所谓的“最终确认”，具有遭受攻击的可能，此外在机制设定上很难避免无利益攻击（nothing-at-the-stake attack）。DPoS 满足效率，但是去中心化不足，容易被少数利益集团的节点把持，导致单个节点存在作恶的可能。单纯的 PBFT 虽然满足去中心化和效率，但是当节点数增多时网络开销就会变得很大，很难在短时间内得出共识，并不适合大型网络。

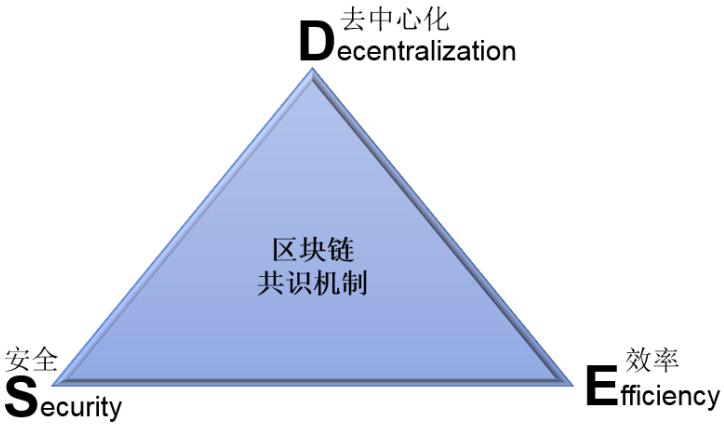


图 5　区块链不可能三角

	PoW	PoS	DPoS	BFT/PBFT	RAFT	Dpos＋Algorand
拜占庭容错	50%	50%	50%	33%	No	50%
崩溃故障容错	50%	50%	50%	33%	50%	50%
块验证时间	BTC: 60 分钟 ETH: 1 分钟	<100 秒	<100 秒	<10 秒	<5 秒	<60 秒
可扩展性	强	强	强	弱	弱	强
每秒交易量 TPS	<100	<1000	>500	>1,000	>5,000	>1,000
资源消耗	高	中	低	低	低	低

表 1　主流共识算法对比分析

我们分析了 HNB 当前应用场景与经济模型，效率应当是首先被考虑的，需要支持更大的并发事务和高性能。同时作为一个围绕支付作为核心的系统，在保证低延迟的前提下，需要避免出现双花问题，因此安全可靠也是至关重要的。基于上述分析，HNB 提出了 DPoS+Algorand 的算法，通过 DPoS 进行共识组的选举，然后 Algorand 进行区块的生成。

基于这样组合的共识算法，主要的优势为：

- 利用身份属性的 Dpos 选出的节点充分考虑了利益分配和公平性，避免了单一群体作恶的情况。
- 共识达成速度较快，且速度不会随着系统节点增多而降低，能够支持海量节点.此外利用 Dpos 来竞选候选节点会比利用 Algorand 来纯粹随机挑选普通用户更为稳定和相对高性能的配置
- 记账者完全随机且隐秘，无法对其产生特定攻击
- 在保证安全性的前提下不做无意义的运算，节省算力资源
- 单一记账者出现分叉的概率接近 0，容错率高。

6.2 HNB 共识算法

6.2.1 共识算法概要

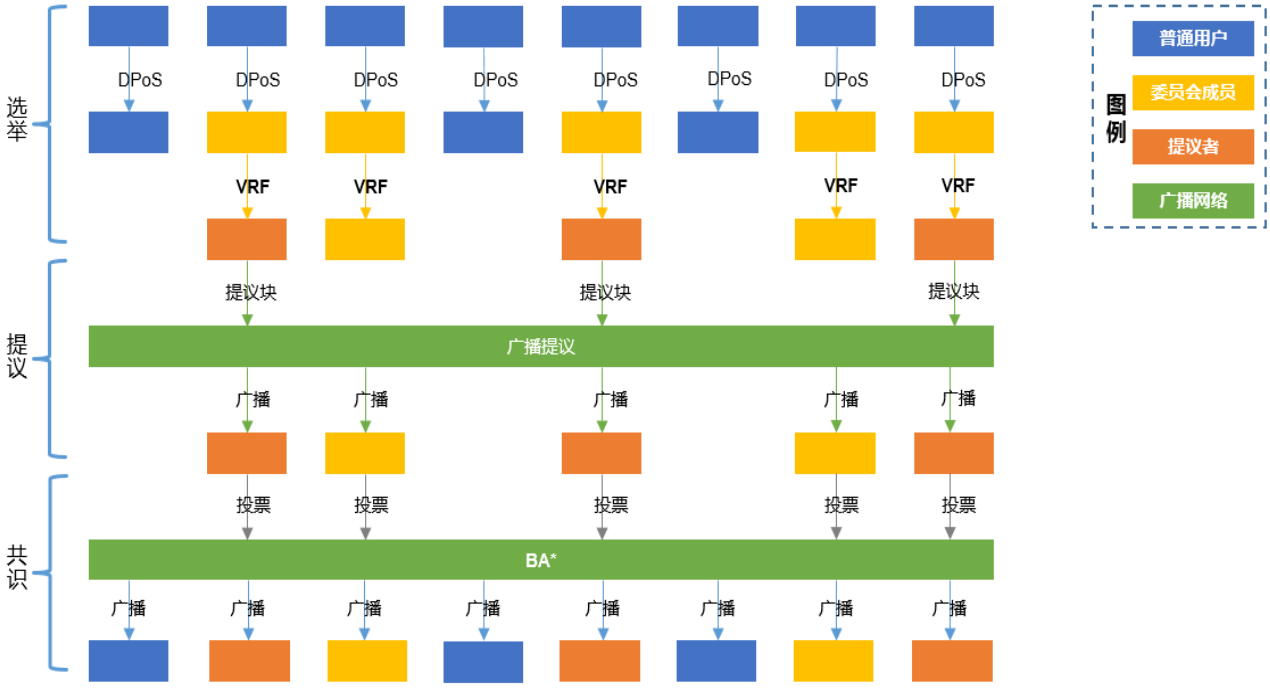


图 6 DPoS + Algorand 共识算法示意图

HNB 底层链采用 DPoS + Algorand 共识算法，提议者为从 HNB 社区选出的 99 名委员会成员，被分成 3 组，代表 3 个不同的利益集团，分为：消费者、生产者（或商户）和贡献者。DPoS 将从三大利益团体中选举提议者，并且保证没有单一利益群体持续获得提议权。

块提议流程将使用加密抽签方法（Verifiable Random Function，简称 VRF），随机选择提议者，加密抽签每次根据前块数据信息选择提议者，保证提议者选择的绝对随机性。攻击者无法预测、修改下一轮次的提议者，增加系统容错能力。

BA*流程将使用拜占庭容错算法在提议者间进行共识，HNB BA*算法利用多轮交互保证最终提议块在提议者间绝对一致性。BA* 两轮投票必须票数大于 2/3 才能进入下一流程，保证永远不会出现共识分区、区块分叉等问题。

6.2.2 提议流程

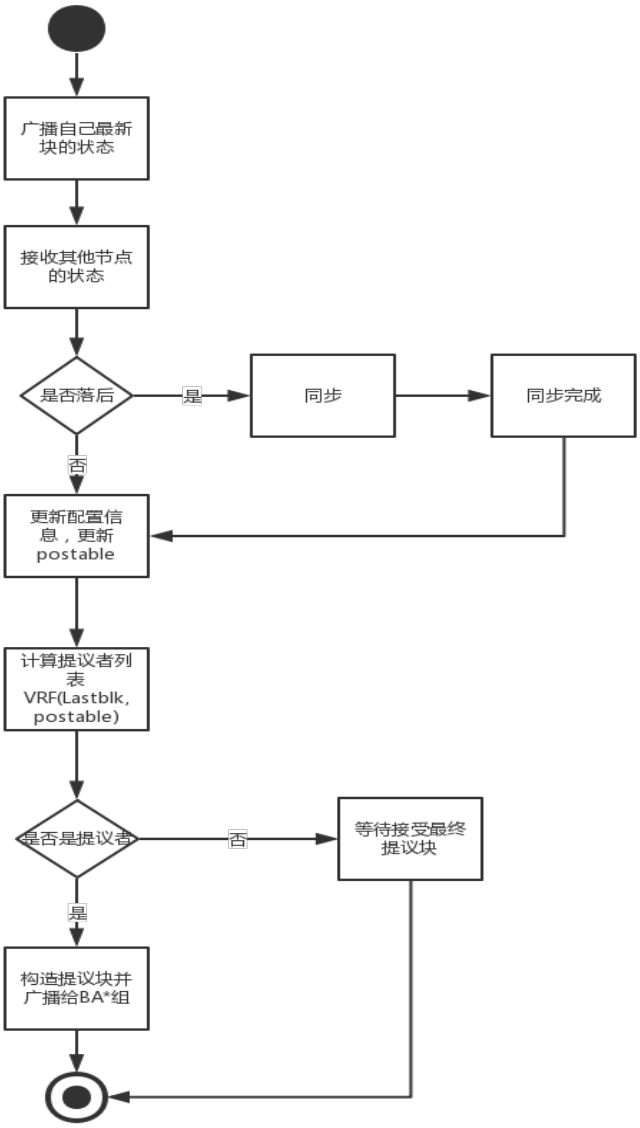


图 7　提议流程

- <u>具体步骤</u>

  1. 广播自己最新块的状态信息，并接收其他节点的状态信息。
  2. 检查自己是否落后，如果落后则进入同步流程，否则进入步骤 3。
  3. 更新配置信息，更新共识节点资产表。
  4. 各个节点独立计算提议者列表。（根据上一个块的信息和资产表来计算）
  5. 如果自己是提议者，去交易池获取交易构建提议消息。（提议者不止一个）
  6. 把提案消息广播给 BA*共识组。

- <u>实现说明</u>

抽签需要一个角色（role）参数来区分用户可以选择的不同角色；例如，用户可能在某个回合中被选择提出一个块，或者在 BA⋆的某个步骤他们被选择为委员会的成员。 Algorand 指定了一个阈值 τ，用于确定为该角色选择的用户的预期数量。

procedure Sortition (sk, seed, τ, role, w, W):
<hash, π> ← VRF $_{sk}$ (seed||role) p ← τ/ W j ← 0 **while** hash / 2 ^{hashlen} **NOT IN** [$\sum_{k=0}^{j} B(k; w, p)$, $\sum_{k=0}^{j+1} B(k; w, p)$) **do** j++ **return** <hash, π, j>

用户通过计算⟨hash,π⟩ ← VRF$_{sk}$(seed||role)执行抽签，其中 sk 是用户的密钥。伪随机 hash 决定了选择多少个子用户，如下所述。在 w（用户的权重）子用户中选择恰好 k 的概率遵循二项分布。由于 B(k_1; n_1， p)+B(k_2; n_2, p)= B(k_1 + k_2; n_1 + n_2, p)，所以划分用户的权重（货币）不会影响在他/她的控制下选择的子用户的数量。

为了确定有多少用户的 w 子用户被选择，抽签算法将区间[0,1）划分为连续区间的形式 I^j = $\left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p)\right) for\ j \in \{0,1,000, w\}$ 。如果 hash / $2^{hashlen}$（其中 hashlen 是散列的比特长度）属于间隔 I^j 中，则用户恰好具有 j 个选定的子用户。选择的子用户数可以使用证明 π（来自 VRF 输出）进行公开验证。

procedure VerifySort (pk, hash, π, seed, τ, role, w, W):
if VerifyVRF$_{pk}$(hash, π, seed||role) **then return** 0; p ← τ/ W j ← 0 **while** hash / 2 ^{hashlen} **NOT IN** [$\sum_{k=0}^{j} B(k; w, p)$, $\sum_{k=0}^{j+1} B(k; w, p)$) **do** j++ **return** j

抽签提供了两个重要的属性。首先，给定一个随机种子，VRF 输出一个伪随机哈希值，它基本上是均匀分布在 0 和 $2^{hashlen}$ − 1 之间的。因此，用户是根据它们的权重随机选择的。其次，一个不知道 sk_i 的对手不能猜测用户 i 被选择了多少次，或者究竟用户 i 是否被选中了，更准确地说，攻击者根据权重随机猜测是猜不到的。

在 VerifySort 算法中显示了用于验证抽签证明的伪代码，遵循相同的结构以检查是否选择了该用户。 该函数返回所选子用户的数量，如果用户没有被选中，则返回 0。

6.2.3 同步流程

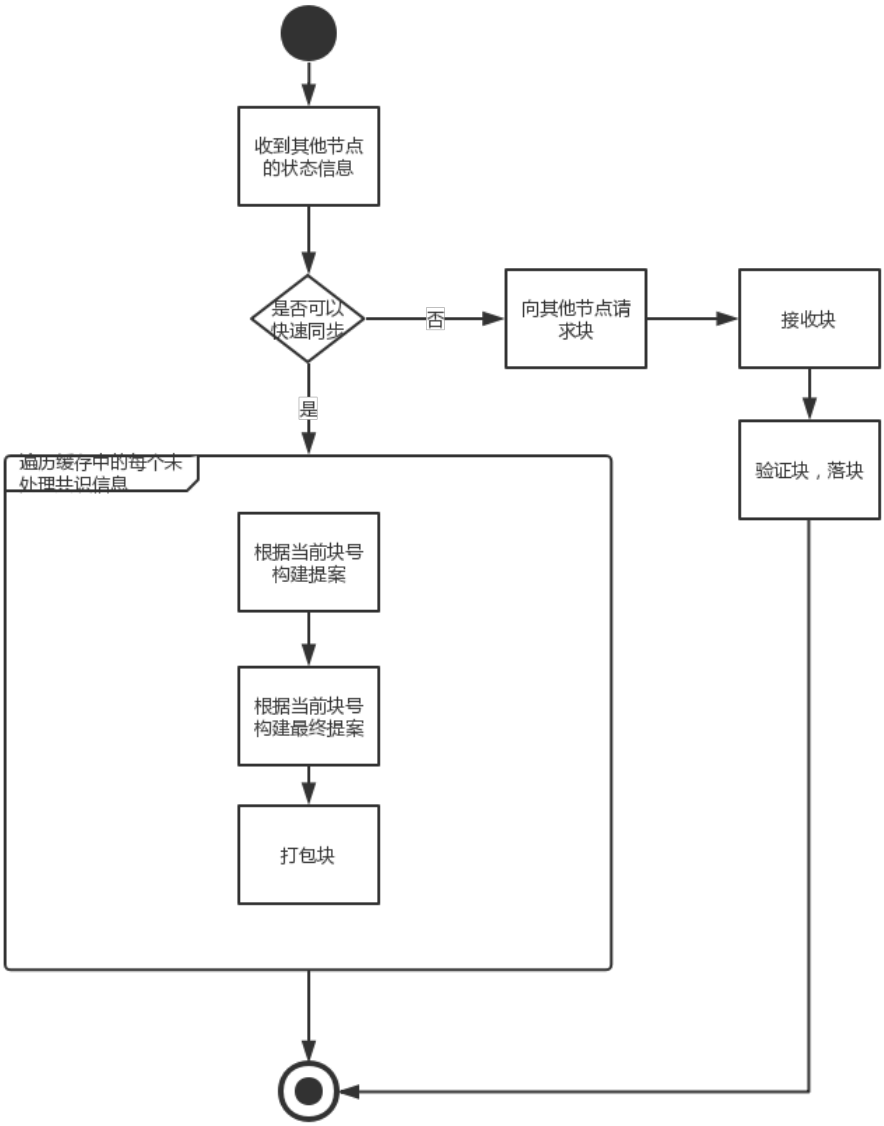


图 8　同步流程

- <u>具体步骤</u>

1. 判断自己是否可以快速同步，可以进入步骤 2，否则进入步骤 3。
2. 遍历内存中的所有未处理的共识信息。
3. 2.1 根据当前块号构建提案信息。
4. 2.2 根据当前块号构建最终提案信息。
5. 2.3 打包块，并更新状态。
6. 向其他节点发送块请求，并等待收到响应块，验证块并落块。

- <u>实现说明</u>

同步假设允许 Algorand 在对手知道选择种子之前选择用户的公钥，确保用户对 BA⋆委员会的随机抽样。我们接下来利用这个属性非正式地表明委员会成员最后一步代表系统中的所有用户。因此，如果足够多的用户投票决定一个区块，那么有足够多的用户在第一步完成 BinaryBA⋆并且不会在将来的步骤中投票。

我们设置 τ_{final}= 10,000，并假设用户的加权诚实比例是 h = 0.8。因此，最后一步委员会中诚实委员的预期数量是 0.8*1 万= 8000，而预计恶意委员的数量是 0.2*10000 = 2000。我们接下来表明，产生了一个 round-r'方块 A 的 7400 个最后一个投票的概率，而另一个方块 B 也被认证为同一回合的概率是微不足道的。

设 $seed_{r-1-(r \bmod R)}$ 为用于选择圆 r'∈[r，r + R]委员会的种子。如果当种子发布的时候，即网络没有强烈同步的时候，发生轮 r-1(r mod R)，那么攻击者可以通过丢弃来自诚实用户的块提议来操纵种子(seedr-1-(r mod R))的选择。 我们首先表明，尽管攻击者有操纵种子的能力，计算种子是不可行的：（1）在 r'轮的最后一步中给了 i 个恶意的委员会成员;（2）j 恶意委员会成员在其他一些 r'，其中 i+3j>4,100。

i 和 j 的预期值分别是 0.2*1000 = 2000 和 0.2*2000 = 400。因此，对于固定的一轮和一步，我们应用公式 2 中的公式，以找到随机种子给出的概率 i+3j>4,100：

$$\sum_{i+3j>4,100} \frac{2000^i}{i!\,e^{2000}} \frac{400^j}{j!\,e^{400}} \cong 2^{-100}$$

为了检查种子值是否满足 i + 3j> 4,100，攻击者必须至少执行一次加密哈希或 VRF 操作。每轮最多有 MaxSteps = 150 步的委员会。对于每一轮，150 个（即 100 个）委员会中只有 2/3 可以创建一个证书（只有那些在 BinaryBA⋆循环中的第一步或第二步投票的人）。 所以通过联合约束，一个种子值对于一个特定的回合（以及该回合的任何一个步骤）给出 i + 3j> 4,100 的概率是≈2^{-93}。所以对手找不到这样的种子是不可能的。

请注意，我们上面的论点并不排除多于一对的圆桌决赛委员会有 i 个恶意的成员和一个有针对性的 BinaryBA⋆委员会有 j 个恶意成员，其中 i + 3j = 4,100。然而，直觉上来说，如果攻击者把所有的计算能力集中在寻找一轮（具有高度恶意的最终委员会来完成 A 块）并且在该轮内有一个有针对性的步骤（用高度恶意的委员会来证实 B 块），使得 i + 3j = 4,100，那么对于所有其他轮次和步骤，恶意委员会成员的数量是随机的，这使得攻击者成功进入其中一个循环/步骤对的概率可以忽略不计(<10^{-20})。

更准确地说，如果攻击者以 i+3j>3850 为目标的两个循环/步骤对，那么这两个循环对将具有 i+3j<3950。这是事实，因为 i+3j>3850 的概率大约是 0.41*10^{-18}，而 i+3j>3950 的概率大约为 0.6*10^{-23}，所以两个事件发生的概率小于 2.5*10^{-41}。因为对于一个特定的选择种子，R=10^3 个回合，每个回合有 10^2 个步骤，有 $\binom{10^2 \cdot 10^3}{2}$ = (1/2)10^{10} 个可能的回合/步骤对。攻击者设法瞄准多于一个回合/步骤的概率，其中 i+3j>3950，因此在联合限制下小于 2.5*10^{-41}*0.5*10^{10} = 1.25*10^{-31} <2^{-93}，所以它不应该发生。

总之，攻击者有两种策略：（1）以 i +3j≤4100 为目标，在目标轮一步进行，或者（2）以 i

+3j≤3950 为目标的多个步/轮对。接下来分析这两个策略。

策略 1：一轮/一对，其中 i+3j≤4100。如果 r'轮的最后一步委员会有 i 个恶意成员，那么同一轮的任何其他委员会最多只有 4100 个恶意成员。显然，最坏的情况是 i+3j=4100。因此，为了证明失败的概率是 10^{-7}，我们需要证明对于 i + 3j 中所有的整数元组（i，j）4100，两个委员会有足够的用户的概率小于 10^{-7}。等价地，对于区间[0,4100/3]中的每个整数值 j，我们需要证明对于 i = 4100-3j，两个委员会有足够的诚实成员（即最终委员会的概率为 0.74·10000-i = 7400-i 和另外一个委员会的 0.685·2000-j = 1370-j 诚实成员）小于 10^{-7}（自 T_{final} = 0.74，τ_{final}= 1000，T_{step} = 0.685，τ_{step}= 2000）。对于每个积分值 j∈[0,41003]我们做了一个单独的计算来证明期望的概率总是小于 10^{-7}。

接下来用于上限每个元组（i，j）=（4100-3j，j）的概率的方法。固定（i，j）。i 个恶意票进入最后的 A 区，而这张 j 票是去证明另一个区块 B。因此，最终委员会需要 7400-i 个签名，另一个委员会需要 1370-j 个签名。假设 A 表示将投票批准 A 块的诚实用户集合作为最终（如果在第一步完成 BinaryBA⋆并选入最终委员会），以及如果他们在一个委员会里，并且没有在第一步完成 BinaryBA⋆，那么 B 将会投票证实另一个区块 B（≠A）的诚实用户。显然，A∩B=∅。 所以每个诚实的用户最终都会成为两套中最多一个的成员。在 7400 和 1370 两个门槛都达到的情况下，集合 A 必须包括 7400-i 个委员会成员进行决赛，而 B 组必须包括 1370-j 个委员会成员进行另一个步骤。

我们说，一个诚实的用户"加入"集合 A，当该用户计数足够多（超过 $T_{step} \cdot \tau_{step}$）的投票 block_hash 在第一步完成 BinaryBA⋆。否则（在 BinaryBA⋆的第一步之后），我们说诚实用户"加入"集合 B。如果没有很强的同步性，网络就处于攻击者的控制之下，所以攻击者可以选择用户是否加入集合 A 或 B（通过投票给用户）。这允许攻击者得到一些反馈：如果攻击者让用户加入集合 A，则他可以观察用户是否被选入最终委员会（即查看该用户是否投票）。以这种方式，攻击者可以让用户一个接一个地加入设置，直到产生足够多的选票来批准块 A 为止，然后加入剩下的诚实用户给集合 B。

因此，用户以某种顺序加入集合 A。假设设置的用户第一次在目标回合中可以产生 7400-i 个最后一步投票，那么 n 就是"A"的大小。设 H 为诚实用户持有的 Algorand 货币单位总数; 为了简化我们的证明，我们假设每个用户只有一个货币单位，所以 H 也是用户数量（如果用户有多个单位，我们认为用户由多个"子用户"代表，每个用户有一个单位 的货币）。对于 n 的固定值，至多 H -n 用户加入集合 B。攻击者要成功，这些 H-n 用户必须在 BinaryBA⋆的目标步骤中为 B 块产生 1370-j 票。设 Pk 表示下列概率：（1）n = k;（2）集合中剩余的 H-k 个用户对 B 块产生 1370-j 票。Pk 由下式给出：

$$P_k = P_r[n = k]\, P_r\,[the\ H - k\ users\ in\ B\ produce\ at\ least\ 1370 - j\ votes] =$$
$$P_r[n = k]\ \sum_{c=1370-j}^{\infty} P_r\,[the\ H - k\ users\ in\ B\ produce\ exactly\ c\ votes]$$

使用方程式 2 中的公式，我们可以计算 Pr [H 中的 H-k 用户产生正确的 C 票]。 在集合 B（包括 H-k 用户）中为委员会选择的诚实用户的预期数量是 h*2000*(H-k)/H，因此对于 h = 0.8，预期的用户数量是 1600*(H-k)/H。 因此，等式 3 中的表达式等于：

$$P_r[n=k]\left(\sum_{c=1370-j}^{\infty}\frac{\left(1600\frac{H-k}{H}\right)^c}{c!\,e^{1600\frac{H-k}{H}}}\right)$$

攻击者成功概率 $\sum_{k=0}^{H} P_k$ ，等于：

$$\sum_{k=0}^{H} P_k = \sum_{k=0}^{H} P_r\ [n=k]\left(\sum_{c=1370-j}^{\infty}\frac{\left(1600\frac{H-k}{H}\right)^c}{c!\,e^{1600\frac{H-k}{H}}}\right)$$

内和收敛，可以评估。 然而，外部总和重复了总数可能是任意大的诚实货币单位。 因此，我们评估一个上限：

$$\leq \sum_{f=1}^{100} P_r\left[\frac{f-1}{100}H \leq n \leq \frac{f}{100}H\right]\left(\sum_{c=1370-j}^{\infty}\frac{\left(1600\frac{H-\frac{f-1}{100}H}{H}\right)^c}{c!\,e^{1600\frac{H-\frac{f-1}{100}H}{H}}}\right)$$

以上是 $\sum_{c=1370-j}^{\infty}\frac{\left(1600\frac{H-\frac{f-1}{100}H}{H}\right)^c}{c!e^{1600\frac{H-\frac{f-1}{100}H}{H}}}$从 0 到 H 的积分的上限黎曼和(Riemann Sum)，其中区间[0，H]被分割成 100 个等份。 由于函数$\sum_{c=1370-j}^{\infty}\frac{\left(1600\frac{H-k}{H}\right)^c}{c!e^{1600\frac{H-k}{H}}}$ 正在减少 k，所以这是一个上限。 评估如下：

$$P_r\left[\frac{f-1}{100}H \leq \frac{f}{100}H\right] =$$

$$P_r\left[the\ \frac{f}{100}\ H\ Honest\ users\ in\ A\ Produce\ 7400-i\ votes\right] -$$

$$P_r\left[the\ \frac{f-1}{100}\ H\ Honest\ users\ in\ A\ Produce\ 7400-i\ votes\right]=$$

$$\frac{\left(8000\frac{f}{100}\right)^{7400-i}}{(7400-i)!\,e^{8000\frac{f}{100}}}-\frac{\left(8000\frac{f-1}{100}\right)^{7400-i}}{(7400-i)!\,e^{8000\frac{f-1}{100}}}$$

因此，失败的概率由下式计算：

$$\leq\ \sum_{f=1}^{100}\left(\frac{\left(8000\frac{f}{100}\right)^{7400-i}}{(7400-i)!e^{8000\frac{f}{100}}}-\frac{\left(8000\frac{f-1}{100}\right)^{7400-i}}{(7400-i)!e^{8000\frac{f-1}{100}}}\right)\left(\sum_{c=1370-j}^{\infty}\frac{\left(1600\left(1-\frac{f-1}{100}\right)\right)^{c}}{c!e^{1600\left(1-\frac{f-1}{100}\right)}}\right)$$

策略 2：多个循环/步骤对，其中 i+3j≤3950。我们首先表明，对手不能瞄准 3 循环/步骤对，其中最终委员会至少有 i 个恶意成员和步委员会有 j 个恶意成员，其中 i + 3j> 3850。这是因为瞄准一循环/步骤对，其中 i+3j> 3850 最多为 $0.41*10^{-18}$，因此以三个这样的对为目标的概率是（$0.41*10^{-18}$）^{3} <10^{-51}。对于每个选择种子，R = 10^3 轮，每轮最多有 10^2 个步骤，攻击者可以瞄准（即 BinaryBA⋆可以结束和证实一个块的步骤）。所以我们最多有 10^3*10^2 个步骤委员会为每个选择种子设置，因此攻击者可以瞄准的循环/步骤对的数量是 $\binom{10^2 10^3}{3} < 10^{15}$ 攻击者成功的概率是 $10^{15}*10^{-51} = 10^{-36}$。因此，攻击者最多可以有两对，其中 i + 3j> 3850，在这种情况下，我们都有 i + 3j <3950。这使得我们可以分析以下两种情况：

(1) 两个目标循环/步骤对，其中 3850 <i + 3j≤3950。可以表明（对于情况 i +3j≤4100，类似的计算）对于一个目标循环/步骤对，成功对手成功的概率小于 10^{-9}，因此小于 $2*10^{-9}$ 的总。
(2) 所有循环/步骤对都有 i + 3j≤3850。同样可以看出对手的成功概率小于 10^{-20}。由于每轮最多有 10^2 步，并且由相同的选择种子设置 R = 10^3 轮，所以攻击者成功的概率以 $10^2*10^3*10^{-20} = 10^{-15}$ 为界。

总之，策略 2 比策略 1 提供了更低的攻击者成功率，其中攻击者的成功率以 R = 10^3 的每个序列的 10^{-7} 为界。

6.2.4 BA*流程

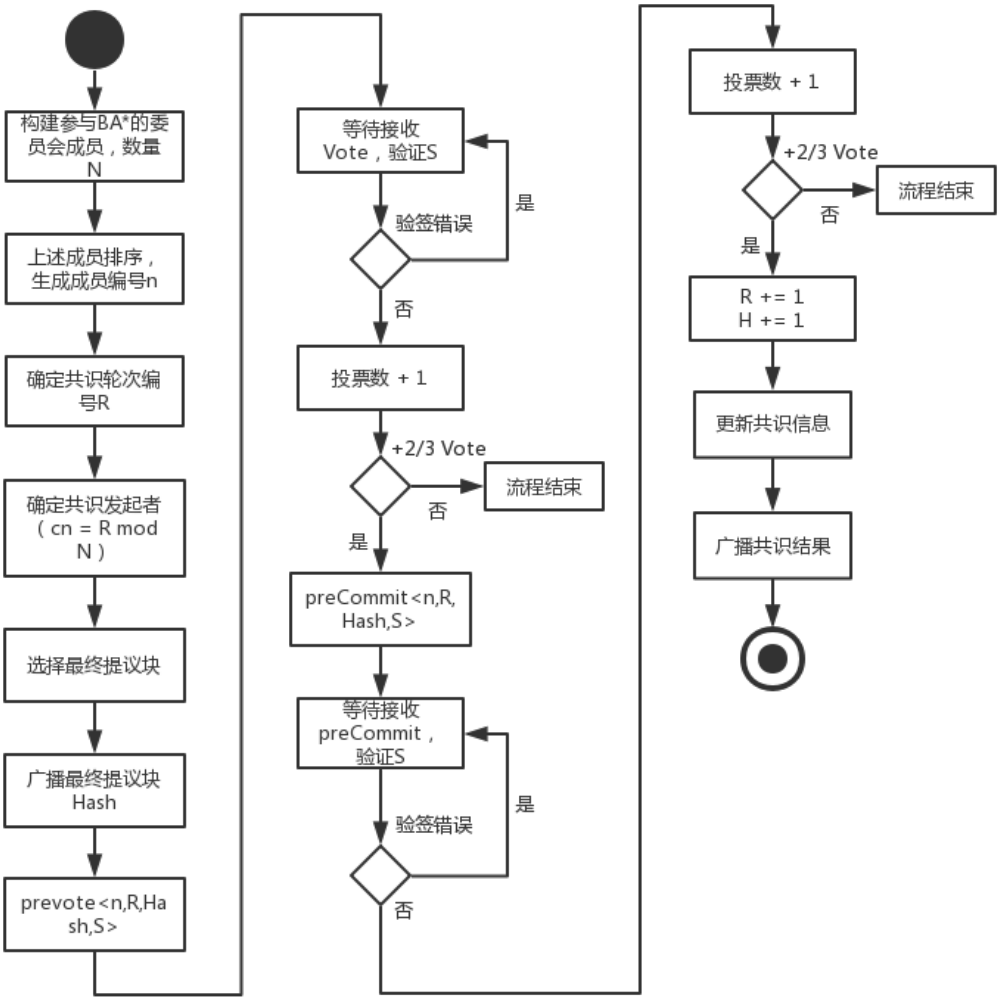


图 9 BA*流程

- **BA*处理流程简述**

1. 提议者将提议块发送给参与 BA*的提议者，其中参与 BA*的提议者总数 N = 3f + 1。
2. 对参与 BA*的提议者信息进行 Hash 计算进行降序排序，得到序号 n。
3. 获取本轮次共识编号 R。
4. 确定本轮次共识发起者编号 curNum = R mod N。
5. 取提议块 Hash 最小值最为最终提议块。
6. 共识发起者将最终提议块的 Hash 广播至其他参与 BA*的提议者。
7. 参与 BA*的提议者对该提议块进行预投票，预投票需要包含自身节点编号 n、预投票轮次 R、投票的块 Hash、validater 投票签名 S。
8. 参与 BA*的提议者在超时时间内等待接受投票数，对投票报文进行身份识别操作。
9. 收集预投票个数，超过 2/3 进入 preCommit 流程。
10. 参与 BA*的提议者对 preVote 块进行预提交，预投票需要包含自身节点编号 n、预投票轮次 R、投票的块 Hash、validater 投票签名 S。

11. 收集预提交个数，超过 2/3，完成 preCommit 流程。
12. 更新块高度 H、轮次 R，广播共识结果。

- 实现说明

为了提高效率，BA⋆投票表决块的散列，而不是整个块的内容。在 BA⋆算法的最后，使用 BlockOfHash()函数来表明，如果 BA⋆还没有收到商定散列的前映像，它必须从其他用户那里获得。BA⋆算法还决定是否建立最终的或暂定的共识，其主函数流程如下：

```
procedure BA⋆(ctx, round, block):
hblock ← Reduction(ctx, round, H(block))
hblock ← BinaryBA*(ctx, round, hblock)
// Check if we reached "final" or "tentative" consensus
r ← CountVotes(ctx, round, FINAL, T_FINAL, τ_FINAL, λ_STEP)
if hblock* = r then
    return <FINAL, BlockofHash(hblock*)>
else
    return <TENTATIVE, BlockofHash(hblock*)>
```

CountVotes()程序从 incomingMsgs 缓冲区中读取属于当前轮和步骤的消息。它通过调用每个消息的 ProcessMsg()程序来处理投票，这确保了投票的有效性。 请注意，不需要私有状态来处理这些消息。

```
procedure CountVotes(ctx, round, step, T, τ, λ):
start ← Time()
counts ← {} //hash table, new keys mapped to 0
voters ← {}
msgs ← incomingMsgs[round, step].iterator()
while TRUE do
    m ← msgs.next()
    if m = NULL then
        if Time() > start + λ then return TIMEOUT;
    else
        <votes, value, sorthash> ← ProcessMsg(ctx, , m)

        if pk ∈ voters or votes = 0 then continue;

        voters.add(pk)
        counts[value] += votes
        // if we got enough votes, then output value
        if counts[value] > T * τ then
            return value
```

其中，ProcessMsg()方法细化的实现如下：

```
procedure ProcessMsg(ctx, τ, m):
<pk, signed_m> ← m
if VerifySignature(pk, signed_m) ≠ OK then
    return <0, NULL, NULL>
<round, step, sorthash, π, hprev, value> ← signed_m
// discard msg that do not extend this chain
if hprev ≠ H(ctx, last_block) then return <0, NULL, NULL>
votes ← VerifySort(pk, sorthash, π, ctx.seed, τ, <"committee", round, step>, ctx.weight[pk],
ctx.W)
return <vote, value, sorthash>
```

ProcessMsg()不仅返回消息中包含的值，还返回与该值关联的投票数。如果消息不是来自选定的委员会成员，则 ProcessMsg()返回零票。如果委员会成员多次被选中，ProcessMsg()返回的选票数量也反映了这一点。ProcessMsg()也返回抽签散列。

Reduction()程序将在任意值上达成一致的问题转换为在两个值中的一个上达成一致：或者是特定建议的块散列，或者是空块的散列。Turpin 和 Coan 的两步技术启发了我们的减少。这种减少对于确保活力非常重要。

```
procedure Reduction(ctx, round, hblock):
// step 1: gossip the block hash
CommitteeVote(ctx, round, REDUCTION_ONE, τSTEP, hblock)
// other users might still be waiting for block proposals, so set timeout for λBLOCK + λSTEP
hblock1 ← CountVotes(ctx, round, REDUCTION_ONE, TSTEP, τSTEP, λBLOCK + λSTEP)
// step 2: re-gossip the popular block hash
empty_hash ← H(Empty(round, H(ctx, last_block)))
if hblock1 = TIMEOUT then
    CommitteeVote(ctx, round, REDUCTION_TWO, τSTEP, empty_hash)
else
    CommitteeVote(ctx, round, REDUCTION_TWO, τSTEP, hblock1)
hblock2 ← CountVotes(ctx, round, REDUCTION_TWO, TSTEP, τSTEP, λSTEP)
if hblock2 = TIMEOUT then return empty_hash;
else return hblock2;
```

BinaryBA⋆()，它在两个值之一上达成共识：要么传递给 BinaryBA⋆()的散列，要么是空块的散列。BinaryBA⋆()依赖于 Reduce()来确保至多一个非空块的散列被所有诚实的用户传递给 BinaryBA⋆()。

```
procedure BinaryBA⋆(ctx, round, block_hash):
step ← 1
r ← block_hash
empty_hash ← H(Empty(round, H(ctx, round, block_hash)))
while step < MAXSTEPS do
```

```
    CommitteeVote(ctx, round, step, τSTEP, r)
    r ← CountVotes(ctx, round, step, TSTEP, τSTEP, λSTEP)
    if r = TIMEOUT then
        r ← block_hash
    else if r ≠ empty_hash then
        for step < s'≤ step+3 do
            CommitteeVote(ctx, round, s', τSTEP, r)
        if step = 1 then
            CommitteeVote(ctx, round, FINAL, τFINAL, r)
        return r
    step++
    CommitteeVote(ctx, round, step, τSTEP, r)
    r ← CountVotes(ctx, round, step, TSTEP, τSTEP, λSTEP)
    if r = TIMEOUT then
        r ← empty_hash
    else if r = empty_hash then
        for step < s'≤ step+3 do
            CommitteeVote(ctx, round, s', τSTEP, r)
        return r
    step++
    CommitteeVote(ctx, round, step, τSTEP, r)
    r ← CountVotes(ctx, round, step, TSTEP, τSTEP, λSTEP)
    if r = TIMEOUT then
        if CommonCoin(ctx, round, step, τSTEP) = 0 then
            r ← block_hash
        else
            r ← empty_hash
    step++
// No consensus after MAXSTEPS; assume network problem
HangForever()
```

6.3 共识激励

虚拟货币的生态体系的可持续发展，离不开完善的生态体系模型，也离不开合理优化的共识激励模型。中本聪创造的新型经济激励模型保证比特币 10 年的长治久安，拥有强大的支持者群体，并因此建立新型的行业生态。可见共识激励模型在数字货币生态发展中占有举足轻重的作用。

HNB 激励算法是根据经济学模型建立，极大程度提高持币用户参与生态建设的积极性、维护生态可持续发展。HNB 为每轮共识成功出块的记账节点提供 HGS 奖励，对于 HNB 作恶节点将扣除一定比例 HGS 以示惩戒。

HNB 经济体的激励机制遵循以下准则：

1. HNB 根据币龄与持币数量计算权重 w，被选为共识组概率与 w 值正相关。
2. 共识成功后，明确本次共识的提议者及参与 BA*的提议者。将 HGS 奖励数量根据比例发放至本次共识相关成员。
3. HGS 奖励数量与该块内交易复杂度总和正相关，存在相关系数 k。
4. HNB 支持分级惩罚，区分节点作恶及节点故障。若提议者发送虚假、错误提案，参与 BA*的提议者发送异常共识协议判定为节点作恶。若节点共识期间网络交互无响应判定为节点故障。
5. HGS 奖励的相关系数 k 根据出块总数负相关。（是否奖励衰减）

7 数据存储

7.1 分布式存储

7.1.1 分布式数据库

Google 公司发表了 Spanner / F1 论文，并基于论文实现了 Spanner 数据库，提供的分布式读写事务（严格的两阶段锁+两阶段提交），以及 F1 实现了全局索引。开启了分布式 HTAP (Hybrid Transactional and Analytical Processing) 数据库的时代，HTAP 是结合了传统的 RDBMS 和 NoSQL 的最佳特性的数据库。

分布式 HTAP 数据库支持无限的水平扩展，具备强一致性和高可用性。其目标是为 OLTP (Online Transactional Processing) 和 OLAP (Online Analytical Processing) 场景提供一站式的解决方案。

对于区块链账本存储，分布式数据库提供了很好的账本存储、账本访问、账本扩容的支持，适用于商业环境的区块链底层系统。

HNB 区块链的账本存储，除了支持 RDBMS 和 NoSQL 数据库以外，还将支持分布式 HTAP 数据库，以便对实体商业业务提供更好的数据管理支撑。HNB 的 HTAP 数据库技术将基于最新的 NewSQL 开源数据库的 TiDB（HTAP Database for TiDB），同时支持联机事务处理（OLTP）和联机分析处理（OLAP）两种业务类型的分布式数据库产品；支持强一致性的分布式事务，在线弹性扩容，异地多活及自动故障恢复；同时兼容 MySQL 数据库。

7.1.2 分布式文件存储

IPFS（Inter-Planetary File System，星际文件系统）是一种协议，其定义了：一个内容寻址的文件系统；协调内容分发；结合了 Kademlia + BitTorrent + Git。

IPFS 文件系统，每个文件及其中的所有块都被赋予一个加密散列的唯一指纹。IPFS 通过

网络删除重复具有相同哈希值的文件，通过计算是可以判断哪些文件是冗余重复的。并跟踪每个文件的版本历史记录。每个网络节点只存储它感兴趣的内容，以及一些索引信息，有助于弄清楚谁在存储什么。查找文件时，通过文件的哈希值就可以在网络查找到储存改文件的节点，找到想要的文件。

HNB 底层系统是为实体商业服务的，社区应当为合作商户提供高可用、高安全的文件存储服务。IPFS 文件系统具有多副本、自动冗余管理的特性，非常适合自治型组织用于提供文件服务。HNB 底层文件存储将与 IPFS 文件系统集成，为社区商户提供文件管理服务。

7.2 数据分片

区块链的数据访问效率和可扩展性是数据存储方案的重要考虑因素，HNB 的数据存储方案采用分片的方式以满足效率和可扩展性的要求。

分布式 HTAP 数据库提供数据分片的能力支持。对于数据写入操作，HTAP 数据库自动将底层数据按照 Key（分片数据）的范围进行分片，每个分片是一个区间。分片中的 Key-Value 触发规则设置（如：数据量超过一定阈值），就会自动分片，以支持数据横向扩展。对于数据读取操作，HTAP 数据库通过负载均衡器，根据存储集群的状态，对集群的负载进行调度。调度是以分片为单位，以负载均衡器配置的策略为调度逻辑，全部过程自动完成，保证数据读取可以高效完成。

7.3 存储可插拔

区块链的账本存储应当根据应用场景的不同，设计采用不同的存储工具和存储方案，因此，账本存储不应与存储介质强绑定。HNB 的数据存储可支持 RDBMS、NoSQL、HTAP 不同类型的数据库，以及 IPFS 文件系统。

对于账本存储，推荐采用数据库进行存储方案，可通过配置的方式，根据具体需求在 RDBMS、NoSQL 和 HTAP 之间选择。HNB 还将提供存储介质可插拔的功能，社区参与者可根据实际运行情况，进行数据存储介质的切换。

7.4 加密存储

区块链可以防篡改，去中心化，在非信任网络环境运行，但是用户的账本对参与节点是透明的，任何参与机构都可以访问到相同数据，如果将用户和商业隐私数据放链上会增加隐私泄露的风险。当前在比特币等公有链系统中，所有的交易要素信息都是公开的（如交易金额）。在实体商业业务中，交易要素信息属于敏感数据，非业务相关方不能查看。

HNB 区块链交易解决方案：

1. 账本数据加密存储：数字资产交易信息加密存储于区块链上；
2. 账户标识脱敏，账本上仅存储 HNB-ID 标识用户身份，HNB-ID 对应的身份信息独立于链外管理；
3. 商业数据加密独立保存，商户的商业数据包含商业机密信息，该信息独立于链外管理。

8 多资产账本

8.1 多资产账本架构

8.1.1 多链架构

多链，即抛弃了“一链治所有”的传统方案，采用“一链一合约”的方案保障每个合约都可以独立运行。这一创新极大程度上简化了架构，降低了数据处理压力，确保一条链上流量激增不会影响到另一条链的效率，在链上进行的任何业务都不会受到其他业务干扰，有效实现了资源隔离。

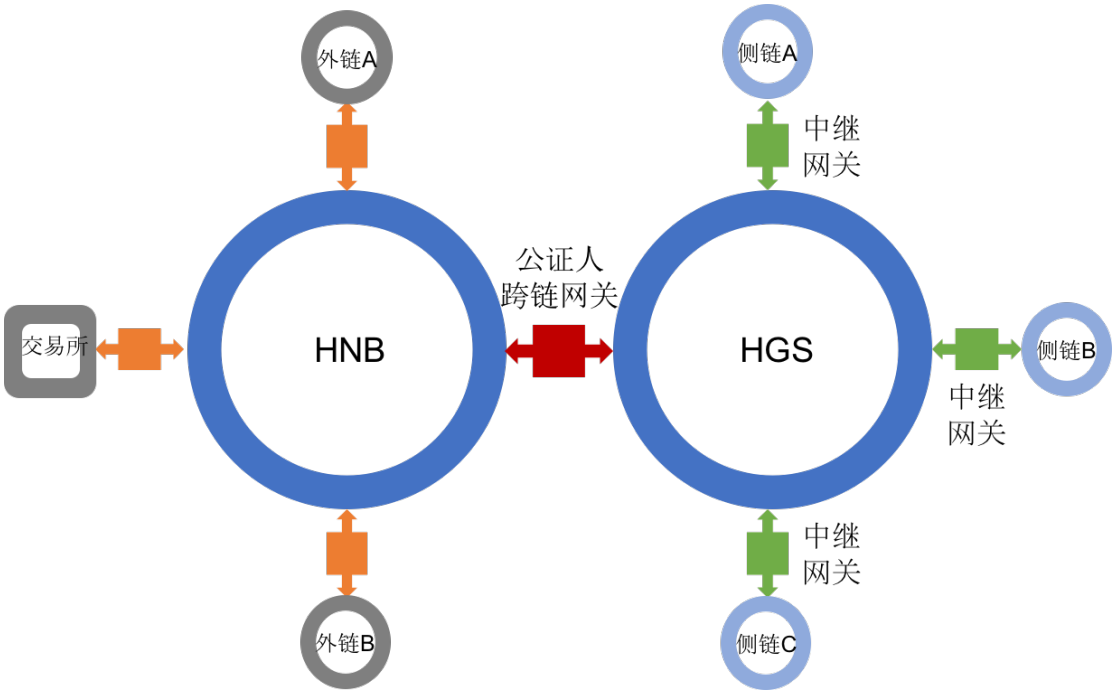


图 10　HNB 多链架构

HNB 底层链运行采用“一链一币”的架构思想，HGS 与 HNB 基于双链架构并行运行，互不影响，互不干扰。多链架构保证双币在性能、隔离性、拓展性等方面有所提高，增强 HNB 底层链的可扩展性和生态包容性。

在性能方面，多链架构保证双原生数字资产运行期间共识模型状态机相互独立，将交易检查、投票等待、数据库访问等耗时操作由串行执行转为并行执行，通过扩展技术资源的方式，可以极大程度地降低系统等待时间、减少 IO 阻塞和执行时间、提高系统吞吐量。

在隔离性方面，多链架构将 HNB 与 HGS 进行分通道共识以及交易处理。HNB 与 HGS 链无法获取交易相关信息，也无法读取或修改对方的账本数据。从物理上进行双原生资产的隔离，保证 HNB 与 HGS 的隔离性，保障了用户及交易的隐私。

在拓展性方面，当 HNB 底层链需要对 HNB 或 HGS 升级时，多链架构将允许对一条链单

独升级，而不会影响其他链业务执行。当 HNB 底层链有增加链需求时，多链架构将十分便捷的增加新链，采用跨链交互技术与原生资产链通信，实现数字资产间的流转，保证 HNB 的高度拓展性。

8.1.2 异构账本架构

HNB 底层链支持异构账本架构，根据 HNB 与 HGS 的货币特点设计独特的双资产账本结构。异构账本架构极大地提高账本模块可用性，提供更加多样化的账本查询服务。

其中 HNB 资产作为与外界交互的数字资产，需要在账本中记录与外界数字资产交易的信息，如：源数字资产标识、目标数字资产标识、交易所标识等。

HGS 资产作为生态内的稳定数字资产，在账本中需记录生态内的相关信息，如：交易侧链标识、衍生数字资产标识、商户标识、商品订单摘要等。

8.1.3 默克尔树

HNB 区块头部包含默克尔树（Merkle Tree）树根值，用于对块内交易进行快速校验。

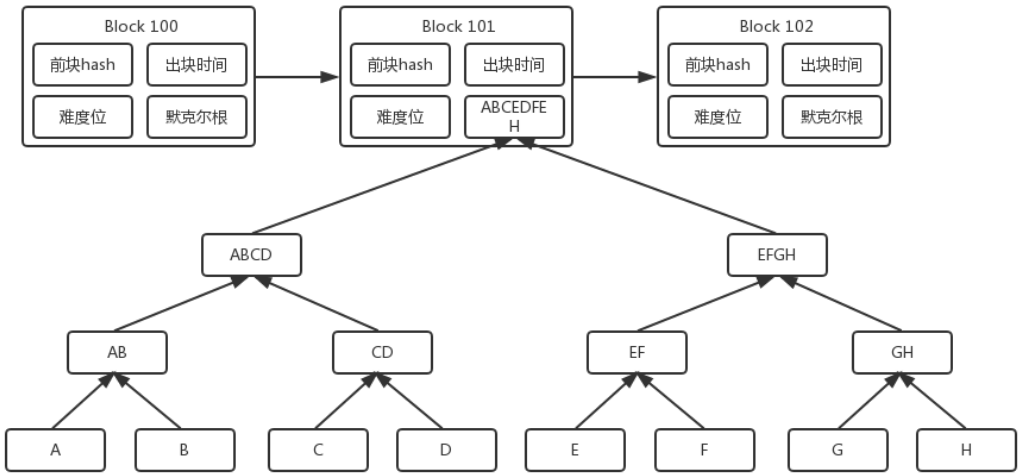


图 11　默克尔树

区块链块内每个交易都计算一个 Hash 值，然后两两连接再次计算 Hash，一直到最顶层的默克尔树根。默克尔树的叶子节点的 value 是数据集合的单元数据或者单元数据 Hash。非叶子节点的 value 是根据它下面所有的叶子节点值，然后按照 Hash 算法计算而得出的。

基于默克尔树交易验证算法的最大好处就是，每个交易都可以单独直接删除，只保留这个交易的 Hash 值即可。对整个区块来说，并没有改变他的密码学安全性和完整性，但是数据量可以大大减小。轻节点只保存块头就可以保证交易不可篡改和伪造。

举例：

只需证明“F”的存在且未被篡改，只需要提供“E”、“GH”、“ABCD”、默克尔根就可以证明“F”交易未被伪造和篡改。

8.2 账本模型

8.2.1 UTXO 模型

UTXO 的全称是 Unspent Transaction Output。 UTXO 模型通过链式的方式组织所有交易的输入和输出，每一个交易的输入必须是某个交易的输出。UTXO 模型是中本聪在 bitcoin 模型中初创并引用的，但并非比特币独有且和区块链没有必然联系。

UTXO 模型优点：

1. 可扩展性——由于可以同时处理多个 UTXO，因此可以实现并行事务并鼓励可伸缩性创新。
2. 隐私——比特币也不是一个完全匿名的系统，但只要用户为每笔交易使用新地址，UTXO 就可以提供更高级别的隐私。
3. 防止双花——UTXO 的每个交易输入必须是未被花费的交易输出，具有天然防范双花的能力。

8.2.2 余额模型

余额模式即传统记账模型，要求必须有账户系统支持，余额与账户相关联。每笔交易只需要验证发送账户是否有足够的余额来支付交易。

余额模型的一个缺点是暴露于双重支出攻击。可以实施递增的随机数来抵消这种类型的攻击。 在以太坊中，每个帐户都有一个公共可见的随机数，每次进行交易时，随机数都会增加一个。 这可以防止相同的事务被多次提交。

余额模型的优点：

1. 简单性——UTXO 的无状态模型会迫使交易包含状态信息，这不必要地使合约的设计复杂化。
2. 效率——除了简单之外，账户/余额模型更加高效，因为每笔交易只需要验证发送账户是否有足够的余额来支付交易。

8.2.3 HNB 混合账本模型

HNB 底层链采用混合账本模型策略，HGS 与 HNB 原生货币采用 UTXO 模型提高底层并发能力。而通过智能合约产生的数字资产采用余额模式，该模式开发简单，降低智能合约的开发

难度，降低由此导致的安全问题。

9 交易管理

9.1 交易验证

HNB 作为一个支撑实体商业的区块链底层系统，对交易的有效性核验是重中之重。交易的验证主要分成交易完整性验证和交易的有效性验证两个部分。

交易完整性验证，是指主要校验交易的签名，保证交易报文完整没有被篡改。

交易的有效性验证，包含交易字段格式校验和交易内容的合法性。其中，交易字段格式校验，用于验证所有字段的格式长度必须正确；交易内容的合法性校验（包括但不限于金额、双花），针对 HNB 不同的节点类型，存在不同的校验处理。

对于全节点，可以独立地进行建立并校验区块链，从第一区块（创世区块）直建立到网络中最新的区块。完整区块链节点可以独立自主地校验任何交易信息，而不需要借助任何其他节点或其他信息来源。该节点会维护一个 UTXO 池对交易的有效性进行校验。

对于 SPV 节点，不必储存完整的区块链账本，需要简化的支付验证（SPV）的方式。这种类型的客端被称为 SPV 客户端或轻量级客户端。SPV 节点只需下载区块头，而不用下载包含在每个区块中的交易信息。由此产生的不含交易信息的区块链，大小只有完整区块链的 1/1000。由于 SPV 节点没有能力构造全部的账本信息，他的方法有所不同。主要步骤如下：

1) 确定交易所在的块号。一般的是通过交易的时间进行撒范围的筛查，然后具体的定位；
2) 确定请求交易的默克尔树审计路径并验证。

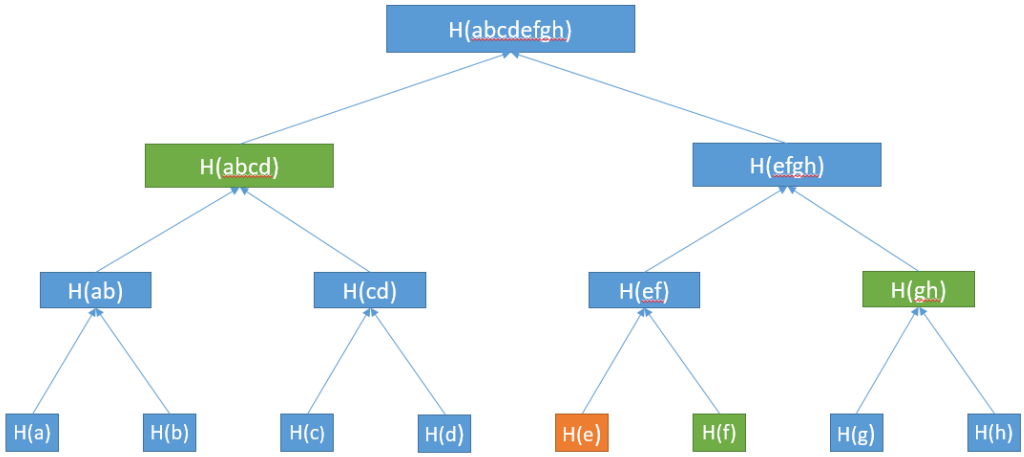


图 12 默克尔树验证路径

如上图所示，要验证 H(e)交易的存在性，只需要 H(f)交易、H(gh)默克尔树节点、H(abcd)默克尔树节点和 H(abcdefgh)默克尔树根节点这 4 个就可以构造出来一条审计路径，然后计算根 Hash 与 SPV 节点的根 Hash 做对比就可以验证该交易的有效性。

9.2 交易池

HNB 系统在完成交易检验后，需要广播交易并讲待记账交易放入交易池管理。如果交易量较大，内存可能会成为交易池管理的瓶颈，HNB 会扩展维护一个额外的交易池，大于内存交易池容量的交易，HNB 会选择对交易进行持久化，通过 DB 或者文件的手段进行存储。

当一个交易被确保打包进区块之后，HNB 将从交易池删除该交易，扩展交易池的交易会被移入内存交易池中。

9.3 排序策略

交易进入到交易池之后，HNB 会通过预置策略决定哪些交易从池中取出打包进块。

传统的基于交易手续费的排序方式存在弊端，即提交一笔交易的交易费过低，导致这笔交易始终无法被确认出块，在资产被退还之前，由于双花的验证，对无法确认的交易造成锁定的效果，出现“冻结”的情况。

由于 HNB 服务的实体商业场景，不向 HGS 的使用者收取费用，因此采用先进先出 FIFO 模式进行交易排序。相比之下，HNB 的 FIFO 排序模式，充分体现了先来先处理的公平性，越早提交的交易越容易被打包进块。

10 智能合约

10.1 HNB 智能合约设计

HNB 区块链系统的智能合约采用标准高级语言作为合约开发语言，支持：golang 和 JavaScript 语言，从而使智能合约的开发者不需要学习新的语言，降低了商户接入门槛。使用 golang 作为智能合约的执行环境，可以为 HNB 应用层框架实现智能控制逻辑。golang 虚拟机具备图灵完备性，可以实现任意逻辑，同时具有高度的确定性，非常适合用于对确定性要求很高的商业业务场景。

合约运行层上层高级语言解析转换相结合，灵活支持虚拟机的基础应用。通过定制化的 API

操作实现虚拟机的外置接口，可以灵活地与账本数据以及外部数据进行交互。

10.2 合约管理

10.2.1 合约发布

HNB 合约开发平台用于 HNB 用户开发自定义智能合约并完成基本测试。HNB 合约开发平台提供完整的工具链，开发、测试、问题定位、API 说明等等，高效定位合约 Bug 问题、检测合约编写异常，有效减少由于智能合约的漏洞导致的安全性问题。利用合约开发平台可非常便捷的进行合约发布，实现一键部署。

10.2.2 合约访问

HNB 底层平台为公链类型，智能合约的查询操作无需增加访问权限。智能合约的写入操作根据密码学算法进行身份识别后，才可以操作。HNB 提供两种智能合约访问方式：外部访问、合约间访问。

外部访问：HNB 支持通过钱包发送交易信息，也支持通过钱包、浏览器查询区块信息、交易信息。支持以 gRPC 或 RESTful 接口访问合约相关信息。

合约间访问：HNB 充分思考各业务领域的交互、关联关系，因此支持合约间数据交互，打通行业壁垒、实现数据互联互通。

10.3 合约运行环境

HNB 采用自研的 HNB-VM 虚拟机平台，HNB-VM 对指令码解析执行过程中进行深度优化，在保证安全性的同时提高指令执行效率。HNB-VM 特色如下：

1. **高速访问缓存**。HNB-VM 创建多层高速缓存，并根据指令码出现概率不同有针对性的设计缓存结构，提高 Cache 命中概率，减少计算、读取对象地址数据的次数。

2. **即时编译**。传统的虚拟机平台在指令码被解释成入栈、出栈、读写变量区、读取方法区等一系列操作，这些操作严重影响虚拟机解释执行效率。HNB 将合约热点程序进行即时编译，减少寄存器的操作步骤。

3. **存储结构优化**。优化对象存储结构，建立对象标识与存储地址的映射关系，实现高效检索对象存储地址。同时，优化对象结构中属性信息，提高内存资源利用率。

10.4 形式化验证

2016 年黑客通过 The DAO，利用智能合约中的漏洞，成功盗取 360 万以太币。The DAO 事件发生后，以太坊创始人 Vitalik Buterin 提议修改以太坊代码，对以太坊区块链实施硬分叉，将黑客盗取资金的交易记录回滚，在得到社区大部分矿工支持的同时也遭到了少数人的强烈反对，最终导致了以太坊社区的分裂。

加强智能合约审计是提高区块链安全的重要保证，其中形式化验证是解决智能合约审计的一个有效方法。HNB 的形式化验证就是基于已建立的形式化规格，对所规格系统的相关特性进行分析和验证，以评判系统是否满足期望的特性。形式化验证并不能完全确保系统的性能正确无误，但是可以最大限度地理解和分析系统，并尽可能地发现其中的不一致性、模糊性、不完备性等错误。

11 节点管理

11.1 节点分类

HNB 节点根据功能性分为全节点、SPV 节点、记账节点，HNB 使用者可根据自身条件（硬件资源、参与身份）不同，选择不同类型节点参与到 HNB 生态体系中。其中：

- 全节点拥有全量账本数据并对外提供查询交易信息、块信息服务；
- SPV 节点可判定某笔交易是否已经存在于区块链网络；
- 记账节点实现节点间共识，生成并广播账本数据。

11.2 全节点

全节点是拥有完整 HNB 账本的节点，全节点需要同步记账节点的账本数据，能够独立验证区块有效性并广播交易，同时为 SPV 节点提供区块查询服务。

HNB 全节点记录全部的账本信息，非常便捷地验证交易余额、双花交易、交易签名等，无需依赖其他节点数据即可判定交易有效性。但与此同时带来的是节点存储开销较大，随着账本数据的逐渐增多，数据库读写操作将受到严重影响。HNB 支持横向扩展的数据存储，来解决全节点的存储空间问题。

11.3 记账节点

记账节点即参与共识的区块链节点，负责运行 Algorand 共识协议，保障 HNB 底层系统正常、稳定运行。记账节点将共识成功的块信息写入账本并广播至全节点。由于记账节点进行 Algorand 共识，需对交易有效性进行验证。

记账节点基于 DPoS 算法从全节点中选出，记账节点是全节点的一个子集。从功能上讲，

记账节点与全节点的功能相同。

基于 HNB 的共识算法设计，同一时刻，全网记账节点数为 99 个。在下一记账周期，全网会通过 DPoS 重新选出 99 个节点作为记账节点。记账周期按照出块数作为一个批次，一批次为 3,000,000 块（约为 6 个月时间周期），每到达一批触发重新选举。

11.4 SPV 节点

SPV 是“Simplified Payment Verification”，简单支付验证的缩写。SPV 节点只存储账本的块头信息，根据存储的块头信息及交易构造的默克尔树判定某交易是否存在与区块链网络。

SPV 节点可以验证交易存在性，却无法验证交易有效性。SPV 节点向全节点请求交易信息，并验证全节点返回的交易信息是否被篡改。但 SPV 节点没有其他交易信息，无法判定该交易输出是否被双花，因此有效性无法验证。

11.5 节点授权管理

由于 HNB 区块链底层系统服务于 HNB 生态系统，非社区成员不可享受 HNB 生态服务，因此 HNB 区块链系统设计严格的节点准入机制，对加入 HNB 区块链网络的节点进行严格的身份审核确认。

从 HNB 社区选出的 99 名委员会成员将参与 DPoS + Algorand 共识，委员会成员需向社区申请成为 HNB 记账节点，社区向委员会成员颁发相应的授权证书。当节点加入到区块链网络时，主动向邻居节点出示授权证书，表明其记账节点的身份。

委员会成员由全节点选举产生，有意成为全节点的社区成员向社区发起申请，社区检查该代表提供的硬件资源、网络情况是否满足成为全节点的基本要求。社区对满足要求的节点颁发全节点授权证书。

SPV 节点由社区成员发起申请，社区只需验证该社区成员的合法身份即可申请 SPV 节点。

11.6 节点小节

综合上述三类节点的介绍，下面对节点进行汇总如下：

节点类型	共识	记账	账本同步	交易验证	交易查询	智能合约
全节点	✔	✔	✔	✔	✔	✔
记账节点	✔	✔	✔	✔	✔	✔
SPV 节点	—	—	—	✔	—	—

表 2　节点类型功能对比表

12 网络通信

12.1 服务接入

HNB 对外提供服务的方式主要有有两类：命令行和 API 接口。

命令行接口是基于 urfave 封装的，可以提供丰富的命令参数，并提供了足够多的默认参数，在控制台的界面简单的输入简单的命令就可以完成全部的操作。HNB 还支持交互式的命令行，在输入命令的时候有更加友好的体验。

为了提供更友好的接入体验，HNB 的 API 接口采用的是 gRPC 和 RESTful 两种方式，风格友好。

gRPC 是一个高性能、开源、通用的 RPC 框架，基于 HTTP/2 标准设计，提供诸如双向流、流控、头部压缩、单 TCP 连接上的多路复用、请求等待等特性，使得网络传输的效率有较大的提升。

RESTful 风格的 API，同样支持 HTTP/2，提供强大的路由能力和反射能力，在处理 URL GET 参数的时候优化了键值对的传输方式，减小了网络报文的传输，对现有区块链技术生态的接入友好。

12.2 P2P 组网

HNB 的 P2P 网络采用的是 Kademlia 算法，该算法适合分布式海量节点网络的组网，其节点关系管理机制可以很有效的抵御 DDoS 攻击。该算法基于两个节点之间的距离进行计算，该距离是两个网络节点 ID 号的异或，计算的结果最终作为整形数值返回（如 01000000）。然后，根据该距离对网络进行分组，分组的依据是前面所有位数相同，从倒数第 n 位开始不同（如 00000000，00000001 为一组，00000000 和 00000010,为不同的组 ）。基于分组，节点可以很快地找到一个它不曾连接到的节点，方便资源的查找和节点的快速接入和退出。

节点之间维持的链接采用的是 gRPC 通信协议。

12.3 节点间通信

节点之间的通信通过 gRPC。节点之间的通信协议采用的是 Protobuf 代码库。Protobuf 代码库支持高性能的转码，支持多种开发语言。HNB 的各种节点之间可以支持四种方式进行数据交互：

- 单向通信（完全同步）
- 流式服务端
- 流式客户端
- 双向流（完全异步）

同时 HNB 支持节点之间的加密传输，采用 TLS 协议，保证节点间通信的安全。通用 TLS 通信存在不足之处，如：启动 TLS 之前，就已经被攻，TLS 的保护就失效。

HNB 区块链也可提供基于硬件解决方案，端到端的保证区块链节点间通信安全。

13 密码学算法

13.1 签名算法

签名算法是指数字签名的算法。数字签名，就是只有信息的发送者才能产生的别人无法伪造的一段数字串，这段数字串同时也是对信息的发送者发送信息真实性的一个有效证明。数字签名是通过一个单向函数，对要传送的信息进行处理得到的用以认证信息来源，并核实信息在传送过程中是否发生变化的一个字母数字串。

HNB 采用的签名算法为 Secp256k1 椭圆曲线，Secp256k1 是在比特币中使用的 ECDSA(椭圆曲线数字签名算法)曲线的参数，并且在高效密码学标准中进行了定义。Secp256k1 为基于 Fp 有限域上的椭圆曲线，由于其特殊构造的特殊性，其优化后的实现比其他曲线性能上可以特高 30%，有明显以下两个优点：

- 占用很少的带宽和存储资源，密钥的长度很短。
- 让所有的用户都可以使用同样的操作完成域运算。

13.2 摘要算法

消息摘要算法的主要特征是加密过程不需要密钥，并且经过加密的数据无法被解密，只有输入相同的明文数据经过相同的消息摘要算法才能得到相同的密文。消息摘要算法不存在密钥的管理与分发问题，适合于分布式网络上使用。

HNB 支持 Hash256、Hash512 等多种 Hash 算法，用户可根据应用场景自行选择 Hash 算法。

13.3 加密算法切换

随着科技不断发展，越来越多的密码算法被攻破，MD5、SHA1、DES 等先后退出了历史舞台。随着量子计算机等高运算能力的高科技产品出现，曾经不可破解的密码算法不堪一击。为保证能够有效抵御量子攻击，HNB 底层支持 ECDSA256、ECDSA384、ECDSA521 等加密算法切换。密钥长度越长，抗攻击能力越强，安全系数越高。

13.4 零知识证明

零知识证明（Zero-Knowledge Proof）是由 S.Goldwasser、S.Micali 及 C.Rackoff 在 20 世纪 80 年代初提出。早期零知识证明需要证明者与验证者通过交互消息的方式才能完成证明，这种证明过程被称之为“交互式零知识证明”。在 20 世纪 80 年代末，Blum 等人提出使用短随机串交互过程实现零知识证明，只由证明者发出一次消息，无需证明者与验证者交互，验证者就可以验证消息正确性，该证明过程被称之为“非交互式零知识证明”。

HNB 区块链将会提供零知识明能力，对用户和商户的隐私数据进行保护，减少用户商户隐私数据泄露的风险。HNB 对 zk-SNARK 零知识证明库进行优化、升级，并封装了高可用、功能丰富的调用接口。zk-SNARK 是 zero-knowledge Succinct Non-interactive ARguments of Knowledge 的简称，是非交互零知识证明的一种，具有证据信息短、验证速度快的特点。

零知识证明主要处理流程——Mint

procedure Mint(pp, v, $addr_{pk}$):
1） Parse $addr_{pk}$ as (a_{pk}, pk_{enc})
2） Randomly sample a PRF^{sn} seed p
3） Randomly sample two COMM trapdoors r, s
4） Compute k := $COMM_r(a_{pk} \| p)$
5） Compute cm := $COMM_s(v \| k)$
6） Set c := ($addr_{pk}$, v, p, r, s, cm)
7） Set tx_{Mint} := (cm, v, *), where * := (k, s)
8） Return c and tx_{Mint}

零知识证明主要处理流程——VerifyTx

procedure VerifyTx(pp, tx, L):
if tx = tx_{Mint} **then**
Parse tx_{Mint} as (cm, v, *) and * and (k, s)
Set cm’ := $COMM_s(v \| k)$
if cm = cm’ **then**
return b := 1
else
return b := 0
else if tx = tx_{Pour} **then**
Parse tx_{Pour} as (rt, sn_1^{old}, sn_2^{old}, cm_1^{new}, cm_2^{new}, v_{pub}, info, *), and * as (pk_{sig}, h_1, h_2, π_{POUR}, C_1, C_2, τ)
if sn_1^{old} or sn_2^{old} **IN** L (or sn_1^{old} = sn_2^{old}) **then return** b := 0
if mr **NOT IN** L **then return** b := 0
h_{sig} := CRH(pk_{sig})
x := (rt, sn_1^{old}, sn_2^{old}, cm_1^{new}, cm_2^{new}, v_{pub}, h_{sig}, h_1, h_2)
m := (x, π_{POUR}, info, C_1, C_2)
b := $V_{sig}(pk_{sig}, m, \tau)$
b’ := Verify(vk_{POUR}, x, π_{POUR})

```
return    b ^ b'
```

13.5 盲签名

传统数字签名方案，都是由签名者对数据明文实施数字签名。当签名者和数据所有者分属不同的个体时，无法做到“被签名数据”对签名者保密。

HNB 实现盲签名算法，具有以下隐私保护能力：

- 盲性：待签消息的具体内容签名者无法获知。
- 不可追踪性：消息的最终签名 S(m)和签名者自己保留的信息 m’无法联系、对应起来。
- 不可伪造性：任意不知道签名者私钥的人，都无法有效地计算出一个能够通过签名验证方程的消息签名对，即除了签名者本人外，任何人都不能以他的名义生成有效的签名消息。

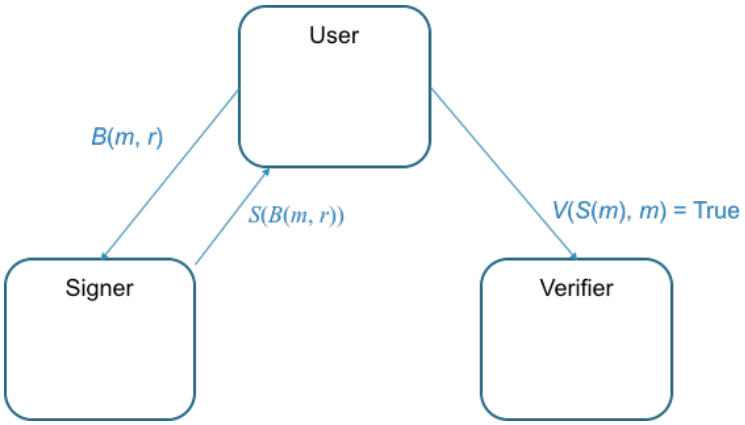


图 13　盲签名处理流程

盲签名数学公式证明：

- 签名消息对：　((S(m), m))
- 对盲签名内容的解密处理：U(S(B(m,r)), r) = S(m)
- 验证处理：V(S(m), m) = True

盲签名的处理流程如下：

1. 签名者在对文件签名的时候，对所签名的文件内容不可见，只能知道签名请求方的身份。
2. 签名者对文件黑盒签名。该签名值能够透过黑盒印在文件上，形成对文件本身的签名。
3. 当签名者检索到文件的内容以及自己对文件的签名，签名者也无法识别出该签名是由自己签出的。

13.6 环签名

环签名可以实现签名者的无条件匿名性，即任何人都无法追踪到签名人的身份。在环签名生成过程中，真正的签名者任意选取一组成员包含它自身作为可能的签名者，用自己的私有密钥和其它成员的公开密钥对文件进行签名。签名者选取的这组成员称作环(Ring)，生成的签名称作环签名。签名接收者能证明签名者是来自环中的某一个成员但却无法确定签名者的真实身份。

<u>环签名原理</u>

1. N 个用户参与，每个用户都有一对秘钥（公钥和私钥），其中 N 个公钥都是公开的。
2. 签名者使用 N 个公钥和自己的私钥对明文进行签名。
3. 验证者使用 N 个公钥对签名信息进行验证。如果签名者的私钥属于 N 个用户中一个私钥，则验证通过，否则验证失败。

验证者在验证的过程中，准确猜出签名者身份的概率是 1/N。实现了隐藏用户身份的目的。

<u>环签名处理流程</u>

1、随机产生一个 v 作为起始参数，$v\in (0,1)^b$，例如：（011010…1100101）。

2、对 r 个用户生成 r 个 $x_i \in (0,1)^b$。其中 r-1 个用户的 x_i 是随机产生的。x_s 不是随机产生。

3、利用 $y=g_i(x)$求对应的 y_i。其中 y_s 暂时未空，因为 x_s 依然是空缺。

存在 r 个用户。采用类 RSA 加密算法。每个用户都有一堆秘钥（P_i,S_i）。类 RSA 加密算法存在两个门限函数：

$y=g_i(x)$——g_i 函数必须使用 P_i
$x=g_i^{-1}(y)$——g_i^{-1} 函数必须使用 S_i

4、利用 $z=C_{k,v}=v$，快速计算出 y_s。

v 是第 1 步生成的起始参数，k 是 H(M)，E 是对称加密算法：Ck,v(y1, y2,…, yr) = Ek(yr ⊕ Ek(yr-1 ⊕ Ek(yr-2 ⊕ Ek(…⊕ Ek(y1 ⊕ v) …))))

5、已知 y_s 然后根据 $x=g_i^{-1}(y)$，求出 x_s。这一步是核心，由于只有 s 用户拥有私钥 s_s，所以只有 s 用户才能计算出 x_s。环签名的安全性就是是基于非对称加密算法的安全性。

6、已经构造了一个环。只有这 r 个用户才能构造完整的环。因为攻击者能够计算出 y_s，但是无法得到 x_s。

13.7 同态加密

在许多数据交易场景中，数据提供商仅仅希望提供数据的使用权，而不是将数据直接就转

让给买方。提供对商户数据的隐私保护是极为迫切的需求。同态加密技术提供了一个很好的解决方案。商户首先利用自己的公钥，使用同态加密算法对数据 m 加密，将密文C 提供给算法提供商，对 C 进行加法或者乘法运算（以及由加法和乘法合成的复杂运算），经过这些运算之后，得到密文C'并返回给商户，商户使用其私钥恢复出明文结果 f(m)。

HNB 支持加法同态加密。同态加密又称隐私同态，同态加密中对密文执行的运算等价于对明文执行了同样的运算。乘法同态支持对密文的乘法运算，加法同态则支持对密文的加法运算。通过同态加密所得到的密文，无需对密文进行解密，即可对密文执行计算任务，对密文执行的计算任务与对明文执行的计算任务是相同的。

设明文空间为 P，密文空间为 C，加密算法为 Enc，解密算法为 Dec。用*表示明文空间内的运算，＃表示密文空间内相应的运算。对于两个明文$m_1, m_2 \in P$，两个密文$c_1, c_2 \in C$，$c_1 = Enc(m_1)$，$c_2 = Enc(m_2)$,同态加密方案細足：$c_1 \# c_{21} = Enc(m_1 * m_2)$。

14 实体经济应用

14.1 DApp 去中心化应用

14.1.1 HNB 的 DApp 体系

HNB 区块链底层支持 DApp（Decentralized Application）作为 HNB 去中心化经济社区的核心特征之一。

DApp 的后端代码在 HNB 的去中心化 P2P 网络上运行。在 HNB 区块链底层系统中，DApp 的后端代码由智能合约实现。前端可以是任何终端用户的应用，如：HTML5、手机 app 等，通过提供的 API 与后端智能合约进行交互，实现其特定的商业应用处理。

可以理解为，HNB 的 DApp 是由用户 API 和智能合约共同组成的。

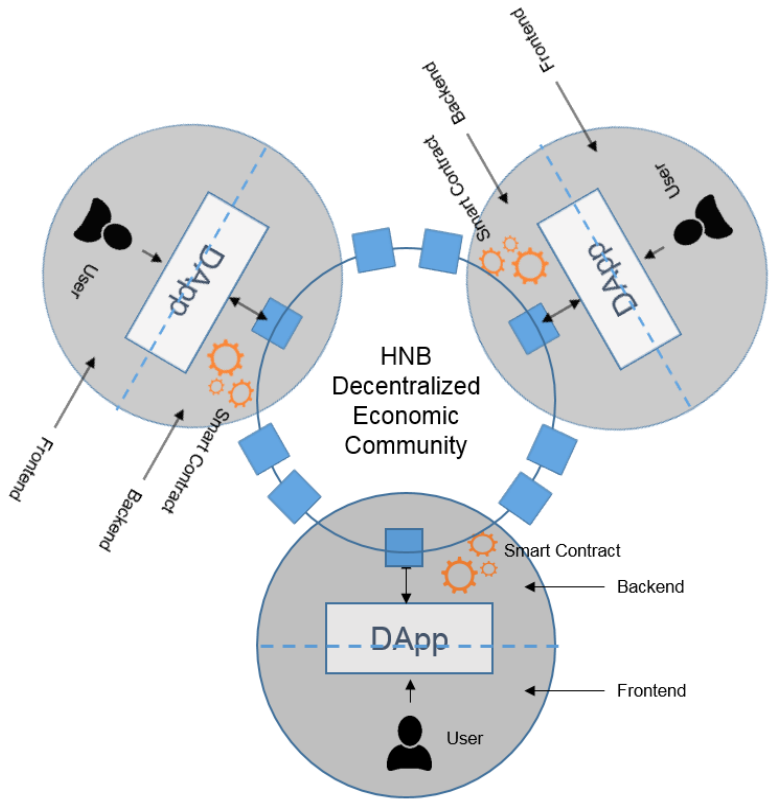


图 14　HNB 系统 DApp 体系示意

DApp 的接入可以采用下述两种方式：

1. **API/SDK 接入**——轻量级的 SDK 接入，让生态共建者能够更加便捷的将原有的商业应用转换为 DApp，使其最关心的资产和支付上链；

2. **HNB 前端接入**——即通过 HNB 约定的协议接入生态中，如：gRPC、RESTful 等。

14.1.2 DApp 功能

HNB DApp 包括以下特性：

- **开源公开**：应用程序的源代码可供所有人使用，同时也可以被其他社区参与者监督。
- **去中心化**：在 HNB 区块链网络上运行，由社区共同参与，共同维护，共同治理。
- **参与激励**：HGS 作为节点/用户的激励奖励，有助于 DApp 运行。
- **共识算法**：基于 HNB 区块链的共识机制，保证 DApp 逻辑执行的单一输入单一输出效果，执行结果在区块链全网一致。

14.1.3 DApp 架构说明

进一步细分，HNB 的 DApp 架构可以分为前端用户 API 和后端智能合约两层架构。

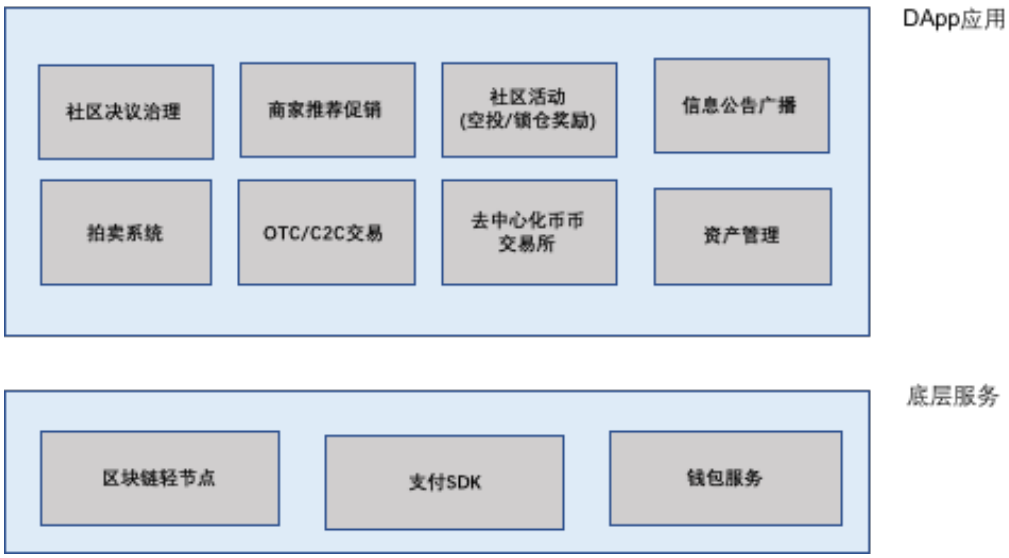


图 15　HNB 系统 DApp 架构

14.1.3.1　DApp 前端

HNB 将在社区中提供一系列有机且强大的 DApp。 前端服务包括但不限于：社区治理、拍卖系统、商家促销、OTC / C2C 交易、去中心化化的币币交易、社区活动、公告板、资产管理等。

用户层和业务层中描述的一些上述功能由嵌入业务逻辑的 DApps 和后端智能合约支持。

14.1.3.2　DApp 后端

DApp 后端服务包括轻量级或 SPV（简化支付验证）节点、钱包服务和支付 SDK。其中，支付 SDK 的目的是为第三方应用程序提供 API 接口，使得支付过程可以易于集成或重定向到 HNB 系统。

14.2 商业隐私保护

HNB 对于商业数据提供基于加密算法和数据隔离的多种隐私保护方案。

首先，采用对称加密算法，对 HNB 合作商户的订单信息、商品信息中，涉及到商业机密的数据进行对称加密保护，保证数据非明文存储。密钥在 HNB 社区统一管理。

其次，用户的核心机密数据，通过非对称加密进行保护。用户使用公钥对该类型的数据加密，之后在 HNB 社区中进行保存。因此，该数据仅对用户本身可见。当其他用户需要访问该数据时，需向数据拥有者提出申请。数据拥有者同意后，与数据访问者协商会话密钥，同时用自己的私钥将原始数据解密，再通过会话密钥保护，提供至数据访问者。保证数据的交互定向传递给数据访问者，期间不会泄露用户机密数据。

第三，由于区块链的共享总账本的特性，HNB 将账本数据与商业数据隔离管理。账本数据

用于保存交易信息，以数字资产转移为主，用户标识使用 HNB-ID，且不会体现订单、商品等信息。采用中心化存储管理商业化数据，如：订单、商品、身份信息等。以此保证商业数据的更强保护效果。

14.3 跨链交互

以太坊创始人 Vitalik Buterin 为银行联盟链 R3 写了一份关于跨链互操作的报告，文中提到三种跨链方式，分别是：公证人机制（Notary schemes）、侧链/中继（Sidechains/relays）、哈希锁定（Hash-locking）。

结合 HNB 的应用场景，HNB 底层区块链系统采用采用公证人机制、侧链/中继、可扩展其他第三方协议三种跨链交互方式。

公证人机制：用于支持 HNB 内部两种原生数字资产间跨链交易。公证人机制采用偏中心化的机制处理，由公证人分别识别其中一种资产的跨链交易请求，在另一种数字资产提交相应的处理。HNB 和 HGS 两种数字资产，由 HNB 社区管理，资产间的跨链交易通过公证人机制，由 HNB 社区充当公证人角色，可以有效的管理两种资产间的兑换交易。同时，采用该种方式，可以支持较高的跨链交易吞吐量。

侧链/中继：用于 HGS 数字资产与侧链间的跨链交互。HGS 链与侧链发生跨链交互请求，以事件的方式通过中继网关将交易转发至对方，实现两条链跨链交易的事务一致性。HNB 底层平台将提供中继网关模块，用于侧链与 HGS 链的对接。

可扩展第三方协议：用于 HNB 数字资产与其他第三方区块链的资产交易的协议。未来的区块链架构将是由多个区块链共同组成的链网结构，跨链交互是必然趋势。因此，HNB 区块链底层将预留第三方跨链协议接口，用于未来与其他区块链底层对接时扩展使用。

14.4 链外极速网络

区块链社对交易扩容方案的争论与尝试由来已久，现有主要方案包括区块扩容，共识算法改良，安全硬件辅助，隔离见证，闪电网络，交易分片，状态分片，多子链等。但无论哪种方案都难以同时兼顾去中心化，可扩展性，安全性三个关键需求。区块链具有应用相关性，重点是结合特性场景需求在各特性间找到平衡点。

HNB 的目标是服务实体经济，在大规模 DApp 应用中，往往小额支付占据了大部分交易请求，而小额交易并无必要在主链及时获得确认，例如实体经济中广泛存的小支付场景。如果将海量小额高频交易在链外极速网络处理，交易过程中不与主链交互，而在交易通道关闭或退出时才请求主链记录交易最终状态，这将极大缓解主链的处理压力，也是离链微支付通道的设计思想。典型应用包括比特币框架下的闪电网络 (Lightning Network)和以太坊的雷电网络(Raiden Network)。

14.5 账本归档

商业环境的数据需要持续性的保障，在一定周期内有保存的要求，这就对账本的存储提出了要求。历史交易数据随着时间的推移，访问的频度越来越小，如果还存储在访问热度高的账本中，代价较大。因此，需要一种历史账本归档方案，保证账本完整性的同时降低账本管理的资源占用。

HNB 采用账本归档技术，定义基于数据库的账本存储为热账本，基于其他存储介质的账本存储为冷账本，按照预定规则对历史账本进行归档处理，移动至冷账本保管，热账本依然保留最新的交易数据。

账本归档功能由全节点提供，全节点完成归档操作后，会在本地保存冷账本。如需要访问历史交易记录，客户端必须访问至全节点。

对于 SPV 节点和共识节点，大部分交易和验证操作都可以通过热账本实现，无需保存冷账本。如涉及到沉默账户的交易，需要从冷账本获取交易信息，则可以通过访问全节点获取历史交易数据。

账本归档的简单操作为：

1. 全节点由规则触发账本归档操作，选择热账本的起始点（该起始点之前的交易数据需要迁移至冷账本中）；
2. 将该起始点之前的交易以文件方式打包保存至冷账本中；
3. 建立冷账本索引，便于冷账本的交易检索；
4. 热账本的块高度保持不变，热账本可在最新高度继续追加记录新交易块。

14.6 并行记账

HNB 采用多链架构，每条链采用独立的网络进行共识。可保障每条链的共识操作互不干扰，不会因为某条链的瞬时并发量较大而影响其他链的交易执行。同时，多条链的共识操作并行执行，可以保证整体网络的共识高效性。

14.7 算法银行

HGS 是一种与法币锚定的数字资产。根据系统内的用户数/货币流通/消费规模 / 真实世界通胀率来确定 HGS 的货币供给；利用算法银行的方式进行增发或销毁 HGS， 保证 HGS 的价值与与法币进行锚定，实现币值稳定。

• HGS 作为稳定货币

HGS 作为稳定货币，它不同于传统的 USDT，TUSD 等稳定货币通过资产背书，中心化及平台承兑的，这种模式要么不透明要么审计成本过高，而用加密资产作抵押的 bitcny 和 DAI 又经常会出现黑天鹅事件而出现连环爆仓。同时也不同于 BASIS，CARBON 完全依赖算法和金融衍生品的方式去进行算法调控，这种方式很难避免信任陷阱会产生所谓的死亡螺旋使得币值崩溃。

• **HGS 的稳定机制更加合理有生命力和以实体经济作为支撑**

HNB 的基础调控方式为基于货币数量理论通过机器学习来实现的算法银行。我们基于生态中的用户数 / 货币供给 / 货币流通速率等因子通过增强学习的方式进行建模，当监测出来的结果异常会对货币供给进行调整（增发或者销毁）。

除此之外，HNB 还设计了由社区商户进行背书和承兑的机制来保证其币值稳定。首先是利用生态中的商品来做托价。由于我们与实体经济密切相连，其中入住商家有一些刚需商品如农产品快销品等，商家可能对自己的大米在自己商城标价 10 美元，而在 HNB 生态中标 100HGAS，如果 HGAS 因为某种原因导致币值下降到 0.95 美元时，此时商家在 HNB 中的大米对于消费者来说就有其套利空间，会更多的去购买 HGAS 来购买大米，随着购买的需求的增加 HGAS 币值就会重新往 1 美元靠拢，可以说生态中的所有商家的商品来为 HGAS 做了承兑。另外一种方式是用拍卖来保证，DAO 会持续的进行拍卖 HNB 用于运营，而用户只能通过 HGAS 来购买 HNB，拍卖的价格理论上会一直低于二级市场价格，那么对于用户来说持有 HGAS 就比法币更加值钱，HGAS 的价值就自然与法币锚定。

15 参考文献

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, 2008. http://bitcoin.org/bitcoin.pdf

[2] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. “Algorand: Scaling Byzantine agreements for cryptocurrencies.” Cryptology ePrint Archive, Report 2017/454, Version 20170924:210956, Sept. 2017. http: //eprint.iacr.org/.

[3] Silvio Micali. ALGORAND: the efficient and democratic ledger. CoRR, abs/1607.01341, 2016.

[4] 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.

[5] Zerocoin Electric Coin Company. ZCash: All coins are created equal, 2017. https://z.cash.

[6] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin, 2013 IEEE Symposium on Security and Privacy.

[7] Eli Ben-Sasson_, Alessandro Chiesay, Christina Garmanz, Matthew Greenz, Ian Miersz, Eran Tromerx, Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin, 2014 IEEE Symposium on Security and Privacy.

[8] R. Turpin and B. A. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. Information Processing Letters, 18(2):73–76, Feb. 1984.

[9] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending Bitcoin’s proof of work via proof of stake. In Proceedings of the 2014 Joint Workshop on Pricing and Incentives in Networks and Systems, Austin, TX, June 2014.

[10] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without proof of work. In Proceedings of the 2016 Financial Cryptography and Data Security Conference, 2016.

[11] D. J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Proceedings of the 9th International Conference on Theory and Practice in Public-Key Cryptography (PKC), pages 207–228, New York, NY, Apr. 2006.

[12] Vitalik Buterin. Proof of stake faq. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ, 2016.

[13] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. Algorithmic Game Theory. Cambridge University Press, New York, NY, USA, 2007.

[14] Rafael Pass. Cryptography and game theory. Securty and Cryptography for Networks, 2016, invited talk., 2016.

[15] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Jo¨el Alwen, Georg Fuchsbauer, and Peter Gazi. Spacemint: A cryptocurrency based on proofs of space. IACR Cryptology ePrint Archive, 2015:528, 2015.

[16] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. J. Cryptology, 23(2):281–343, 2010.

[17] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost,

JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, Dale Woodford. Spanner: Google's Globally-Distributed Database. Proceedings of OSDI, 2012.

[18] Jeff Shute, Chad Whipkey, David Menestrina, Radek Vingralek, Eric Rollins, Stephan Ellner, Bart Samwel, Mircea Oancea, John Cieslewicz, Ben Handy, Kyle Littlefield, Ian Rae*, Traian Stancescu, Himani Apte. F1: A Distributed SQL Database That Scales. 2012.

[19] Bitcoin Wiki. Confirmation. https://en.bitcoin. it/wiki/Confirmation, 2017.

[20] BitcoinWiki. Mining hardware comparison, 2016. https://en.bitcoin.it/wiki/Mining_hardware_comparison.

[21] BitcoinWiki. Bitcoin scalability. https://en. bitcoin.it/wiki/Scalability, 2017.

[22] BitcoinWiki. Proof of stake. https://en.bitcoin. it/wiki/Proof_of_Stake, 2017.

[23] Ethereum Foundation. Ethereum, 2016. https://www. ethereum.org/.

[24] Ethereum Foundation. Create a democracy contract in Ethereum, 2016. https://www.ethereum.org/dao.