HNB

Technical White Paper

A Next Generation of Blockchain Based Decentralized Economic Entity

HNB
HASHED NEXUS BITECO

Table of Contents

1 Summary

HNB is a next generation of blockchain-based decentralized economic entity. Its goal is to build a sustainable business model to serve and support real business scenarios. From technology point of view, HNB is a blockchain system supporting multi-chains architecture and the circulation of multiple digital assets, it can be used to build a decentralized economic ecosystem founded on public blockchain. Aligning with HNB business vision and development plan, HNB team has formulated the technical solution to implement HNB economic model after overhaul of current blockchain projects in the market.

In this technical white paper, there are detailed descriptions on design principles of HNB public chain, overall architecture design and how core technologies are implemented. The core technologies elaborated on in the white paper include identity management, consensus mechanism, data store, multiple asset accounting, transaction management, smart contract, node management, network communication and cryptography. Other content such as introduction of HNB DApp structure, algorithm bank for application scenario for supporting real business applications are also included.

2 Terminology Description

Terminology	Description
HNB	HNB Token is the digital asset representing the ownership of HNB economic entity. Its total supply is fixed to one billion.
HGS	HGS is the payment medium pegged to US dollar, issued by “HNB Reserve Committee” and only circulated in HNB community.
Committee Member	Candidates for delegates through consensus process, elected by DPoS
Delegate	Members participating in Byzantine Agreement process to reach consensus. It is the node selected by consensus.
DPoS	Delegated Proof of Stake. It is the mechanism to vote for committee members based on proofing the stake/voting weight.
VRF	Verifiable Random Function. It is for random sortition of delegates. Its output is unpredictable.
Zero-Knowledge Proof	A method that the prover can prove to the verifier that something is true without disclosing any information.
Blind Signature	A special form of digital signature to guarantee the data signed confidentiality and un-traceability to protect data privacy.
Ring Signature	A type of signature to ensure unconditional anonymity of signers. No one can trace the identity of the signer.
Homomorphic Encryption	A form of encryption that allows computation on encrypted data as if it had been performed on the plain text.

3 Design Principles

HNB system is designed to enable the founding of sustainable economic model to serve real business entities. The goal is to use HNB system to build a next generation of blockchain-based decentralized economic entity serving an economic ecosystem comprised of over 100 million of consumers and merchants. The HNB economy of scale and its members are ever-increasing as the ecosystem grows. Therefore allowing for growth drives how HNB blockchain is designed and defines HNB blockchain design principles as following.

Scalability

- Support ever-increasing size of HNB community, e.g the number of nodes increase in HNB blockchain, more and more merchants to join, user growth

High Performance

- Support millions of users in concurrent transcations. Gracefully handle bursting concurrent transactions at the peak. Reduce the latency of transcation confirmation, achieving the speed of consensus at the level of second.

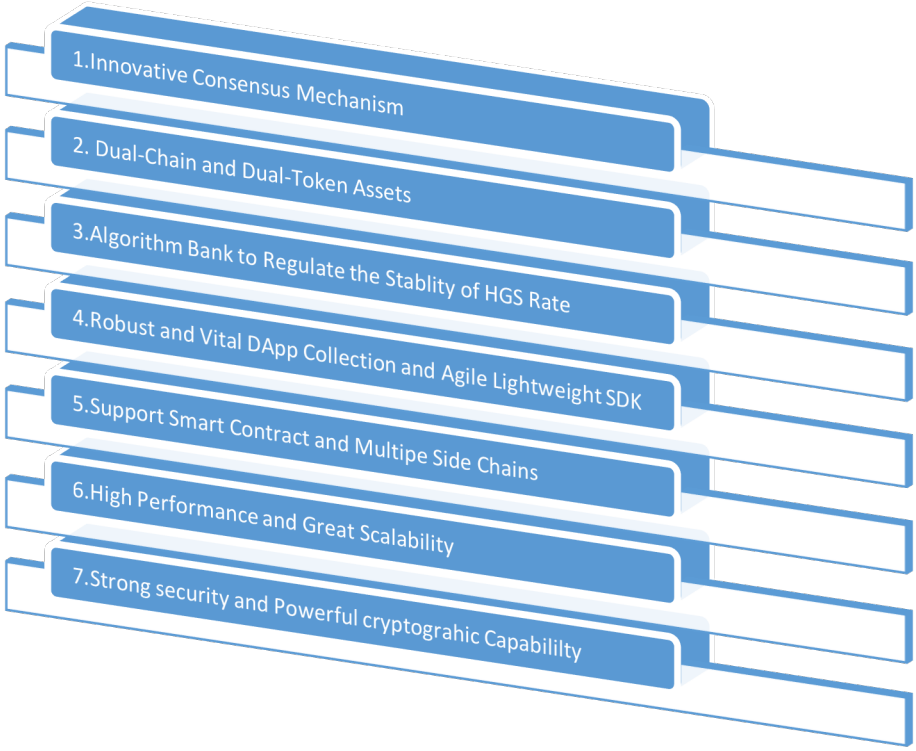
Security

- Robust security at application and infrastructure levels, protecting the privacy of business and invidivitual data.

Interoperability

- Support speedy integration of existing systems and ongoing iterative developement, easy interfacing with other blockchain systems and applications.

Based on above design considerations, HNB blockchain embodies the following characteristics.



- Innovative consensus mechanism to support multiple nodes to reach consensus speedily.
- Dual-chain and dual-token assets : HNB and HGS; HGS pegged to FIAT
- Algorithm bank to regulate the stability of HGS exchange rate to FIAT.
- Robust and vital DApp collection and agile lightweight SDK to support easy integration with merchant applications.
- Support smart contract and multiple side chains
- High performance and great scalability to support off-chain liquidity network technologies.
- Strong security and powerful cryptographic capability

4 HNB Architecture Design

4.1 HNB Ecosystem Architecture

HNB ecosystem design follows layered and modular architecture philosophy with the emphasis on system scalability, interoperability, performance and security. HNB ecosystem architecture is a collection of the foundational blockchain building blocks and their related components. It is structured as multiple layers – User layer, HNB business layer, Blockchain layer, Infrastructure layer and Common service layer.

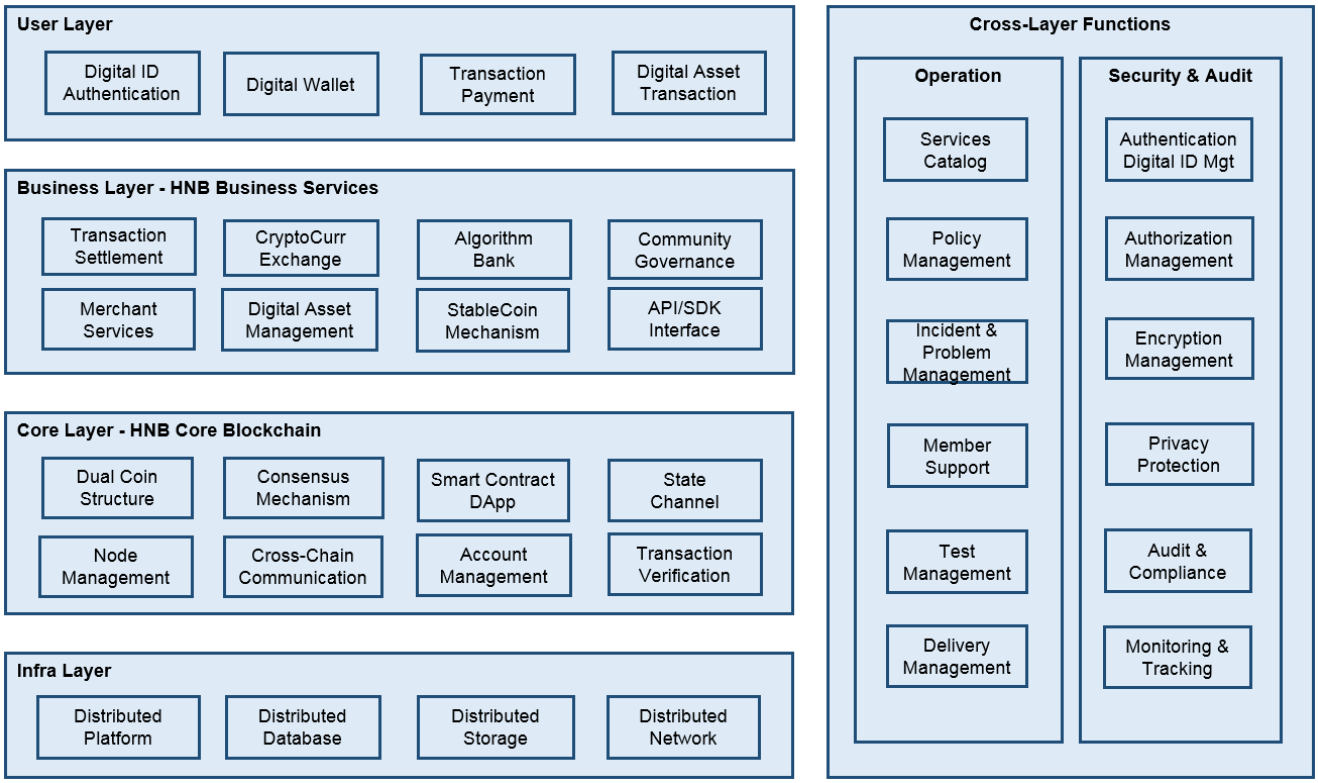


Figure 1 HNB Blockchain Ecosystem Architecture

User Layer	A layer of end user access point. Users access HNB blockchain services through this layer. From technical perspective, HNB end user service enabled by DApp which are described in chapter 14.1 including digital wallet, digital asset management etc. User layer also includes user management function described in Chapter 5.
HNB Business Layer	HNB business layer implements all the core business logics of HNB decentralized economic community. From technical perspective, it is supported by DApp which are described in chapter 14.1
Blockchain Core Layer	Blockchain layer which served as HNB blockchain foundation, fulfills core functions of consensus, ledger management, multi-assets management, asset trading, smart contract. The technical solutions are described respectively from chapter 6 to 10. In addition, unique innovations are applied by HNB blockchain in the areas of privacy protection, cross-chain interaction, off-chain liquidity network, concurrent ledgering. Details are seen in chapter 14.
Infrastructure Layer	Infrastructure layer provides distributed foundational components such as distributed database, distributed storage, and distributed networking. Database and storage service support plugin capability, providing greater choices to the users and comprehensive business enablement. Pluggable storage is elaborated on in chapter 7.3.
Common	Common Service layer provides services to above four layers including

Service Layer	security, audit and operation. Solutions to cryptographic algorithms are described in chapter 13.

4.2 HNB Blockchain Core Technology Architecture

HNB blockchain technology architecture is further granular design of HNB blockchain core layer, infrastructure layer and common service layer.

HNB blockchain technology architecture is also based on layered and modular approach. It is made up of 7 top-down layers of contract, consensus, incentive, data, network, security and foundation.

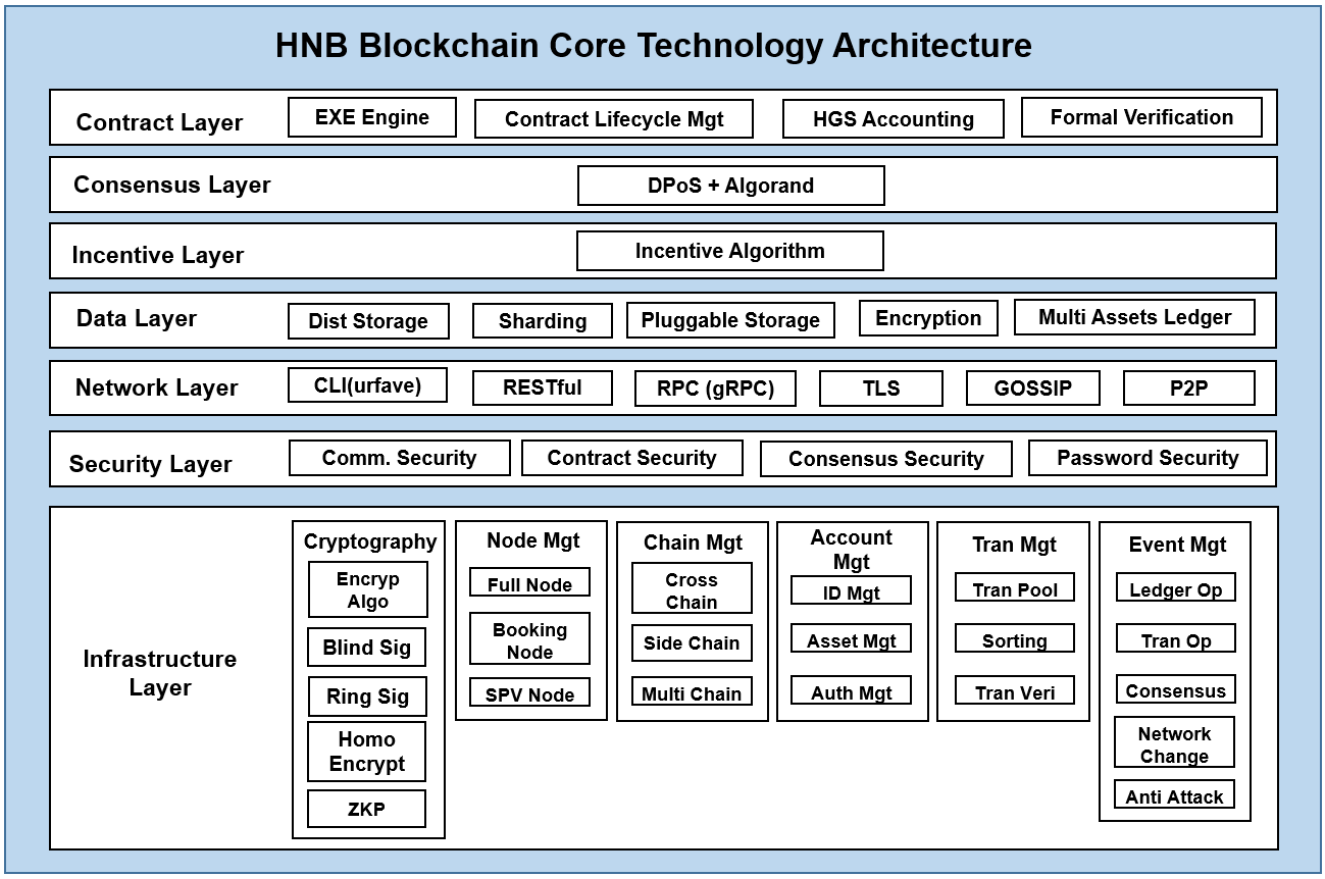


Figure 2 HNB Blockchain Core Technical Architecture

Contract Layer	Contract layer provides the services of life cycle management, operation cost management, run-time environment, formal verification for smart contracts. HNB smart contracts are developed by advanced programming languages such as Golang and JavaScript which not only meet Turing completeness but also ensure smart contracts to be executed smoothly and safely by their mature run-time environments.
Consensus Layer	Consensus layer adopts hybrid consensus mechanism of blending DPoS and Algorand, encouraging the nodes making great contributions in HNB community to

	participate in accounting, ensuring fair voting on consensus of delegates based on encrypted sortition, ultimately achieving efficiency and consistency of consensus process through consensus algorithms.
Incentive Layer	Incentive layer provides open, transparent, unified incentive strategy management based on the design of algorithm bank in HNB ecosystem. HGS digital assets are rewarded to the nodes making great contributions in the community. Details about Algorithm bank are described in chapter 14.7
Data Layer	Data layer is the granularity of decentralized database and decentralized storage in HNB infrastructure layer, described in blockchain technology architecture. It provides storage service for decentralized database and decentralized filesystem, supporting HNB dual-token asset ledge management. The details are described in chapter 7 and 8.
Network Layer	Network layer is the infrastructure module providing network communication. Its functions include providing interfaces for blockchain infrastructure layer to access applications, providing P2P network for blockchain nodes and broadcast among blockchain nodes. Details are described in chapter 12.
Security Layer	Security layer is the HNB solution to address demanding security requirements of blockchain through the analysis of four security pillars of communication, data, contract and key. Details can be referenced in chapter 7,10,12,13.
Infrastructure Layer	Infrastructure layer provides foundational services for blockchain including cryptographic algorithm library, node management, multi-asset chain management, account management, transaction and event management.

5 Identity Management

5.1 Identification

5.1.1 Decentralized Identifier

Conventional identification is tied to individual identity document. Identification includes ID, residence document and driver license. With technology advancement, new identification approaches such as fingerprint and DNA are gradually accepted. In Internet world, things are not always tangible and many are virtual so the network identifiers are needed to map them to real entities.

HNB-ID is obtained by computation based on mathematic algorithm on HNB public key. It is unnecessary to generate HNB-ID nor conduct conflict detection of it through centralized computation. HNB-ID is produced and managed by user directly in decentralized way.

5.1.2 Privacy Protection

HNB-ID generation algorithm determines impossibility of hacking public key information based on HNB-ID, nor can user attributes of HNB-ID be correlated to public key. Therefore user privacy is surely protected.

5.1.3 Identity Authentication

HNB-ID and user public key maintains one-to-one relationship. Users can prove their identities belonging to certain business entity through identity authentication so to maintain the integrity of their business activities.

5.2 Identification Registration

HNB supports acceptance mechanism of configurable nodes into HNB community. Authorization is required to accept nodes into HNB community. The authorization is obtained through certificate issued by HNB CA(Certificate Authority) to allow a HNB-ID to join HNB community. HNB-ID related attributes are not sent to HNB public blockchain to protect privacy. When the node starts up and joins blockchain network, its certificate signed by CA is distributed to other nodes to declare the authorization of its identity.

5.3 Trust Model

5.3.1 Security of key generation

Key is generated in offline wallet so the private key is not exposed to network whatsoever in any form which ensure hackers are unable to steal user private key and rule out the possibility of HNB security breach.

5.3.2 Security of storing the key

HNB provides the service of storing key upon user requirement. User can defer its key management to decentralized private key servers who encrypt user's private key through symmetric-key encryption algorithm before storing it. Symmetric key is kept by user directly apart from private key servers. So even the servers get hacked, the attackers are unable to restore user private key.

5.3.3 Security of identity registration

In the process of user identity registration, user attributes combined with HNB-ID, are sent to CA and CA will issue certificate upon verification of user identity. The user attributes are encrypted through asymmetric-key encryption algorithm by HNB system so the information must not be tampered by

middleman and data privacy is protected to maximum degree.

As shown in the following diagram, when Alice registers her identity, the user information packaging with the public key are encrypted by CA public key to ensure safety of data transmission. In addition, HTTPS is used for data transfer to impose double encryption mechanisms to protect data integrity and user privacy.

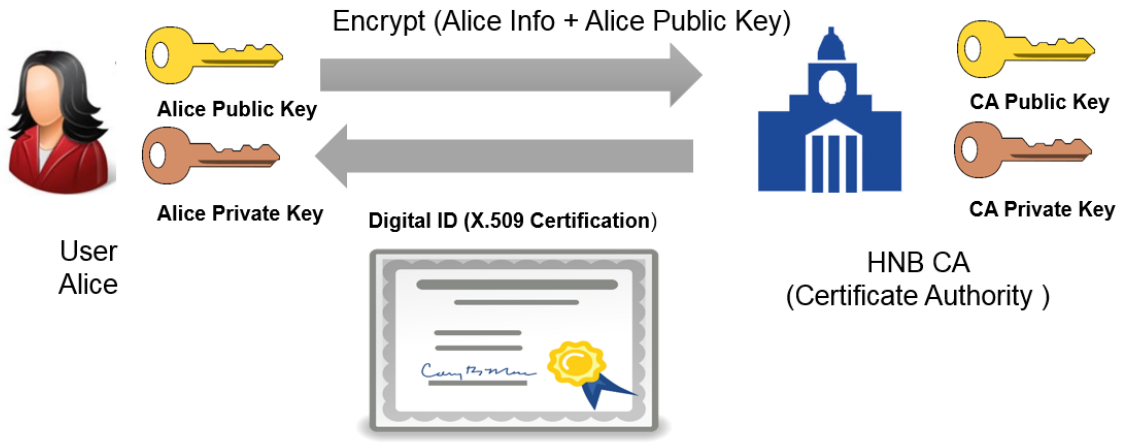


Figure 3 ID Registration Process

6 Consensus Mechanism

For any decentralized autonomous systems, the consensus mechanism is the foundation of community trust. Consensus is a process of reaching agreement between distrusting nodes regarding the final state of the data in the entire Blockchain system. To achieve consensus, different approaches could be used.

A consensus mechanism is a set of steps taken by most or all nodes in a Blockchain to agree on a proposed state or value. The researches on consensus mechanism have been conducted by computer scientists in industry and academy over 3 decades. Despite its long history, only recently until the advent of Blockchain, Bitcoin, Ethereum etc., has it been placed in the limelight and gained considerable popularity.

Consensus algorithm can also be used on reaching agreement on a proposal. The proposal can refer to any information that can be agreed upon, such as basic block generation and verification, or the rules/policies of community operations and governance.

6.1 Blockchain Impossible Trinity

CAP theorem, also known as Brewer's theorem, was introduced by Eric Brewer in 1998 as a conjecture.

In 2002, it was proven as a theorem by Seth Gilbert and Nancy Lynch. It states that any distributed system cannot have consistency, availability, and partition tolerance simultaneously:

- **Consistency** is a property which ensures that all nodes in a distributed system have a single, current, and identical copy of the data.
- **Availability** means that the nodes in the system are up, accessible for use, and are accepting incoming requests and responding with data without any failures as and when required. In other words, data is available at each node and the nodes are responding to requests.
- **Partition tolerance** ensures that if a group of nodes is unable to communicate with other nodes due to network failures, the distributed system continues to operate correctly. This can occur due to network and node failures.

It has been proven that any distributed system cannot have consistency, availability, and partition tolerance simultaneously.

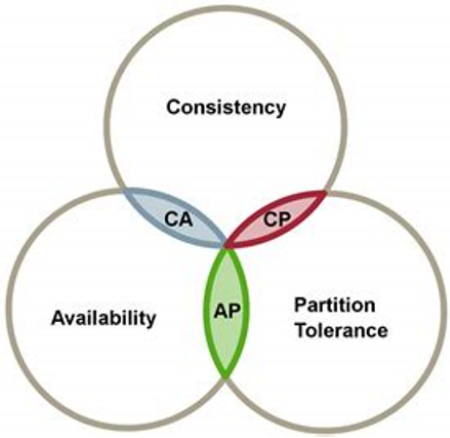


Figure 4 CAP Theorem

Based on HNB project team's research, the current consensus algorithm focuses on three core characteristics of **S**ecurity, **D**ecentralization, and **E**fficiency.

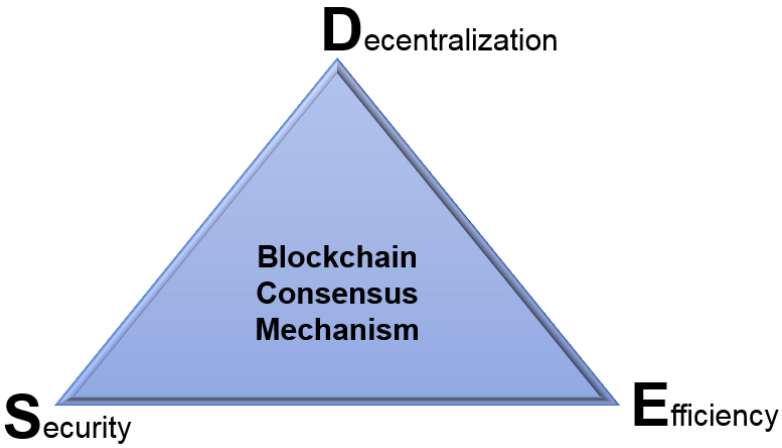


Figure 5 Blockchain Impossible Trinity

According to CAP principle of distributed computing, security, decentralization, and efficiency are impossible to be achieved simultaneously. Trade-off has to be made for the design of consensus mechanism.

HNB team analyzed the current mainstream consensus mechanisms including PoW/PoS/DPoS/PBFT.

Consensus Mechanism	Brief Description	Concerns
Proof of Work (PoW):	This type of consensus mechanism relies on proof that adequate computational resources have been spent before proposing a value for acceptance by the network. This scheme is used in Bitcoin, Litecoin, and other cryptocurrency Blockchain's. Currently, it is the only algorithm that has proven to be astonishingly successful against any collusion attacks on a blockchain network	PoW meets the security and decentralization requirements, but the efficiency is relatively low
Proof of Stake (PoS)	This algorithm works on the idea that a node or user has an adequate stake in the system; that is, the user has invested enough in the system so that any malicious attempt by that user would outweigh the benefits of performing such an attack on the network. This idea was first introduced by Peercoin, and it is going to be used in the Ethereum blockchain version called Serenity. Another important concept in PoS is coin age, which is a criterion derived from the amount of time and number of coins that have not been spent. In this model, the chances of proposing and signing the next block increase with the coin age.	PoS satisfies decentralization and efficiency, but it lacks security; PoS validates transactions based kind of probability calculation, is vulnerable to attack. PoS also fails to avoid nothing-at-the-stake attack
Delegated Proof of Stake (DPoS)	This is an innovation over standard PoS, whereby each node that has a stake in the system can delegate the validation of a transaction to other nodes by voting. It is used in the BitShares blockchain	DPoS satisfies the efficiency, but not decentralization. The consensus result could be easily controlled/influenced by the nodes of specific interest groups, resulting in the high possibility of the attack by malicious single node.
Practical Byzantine Fault Tolerance (PBFT)	This mechanism achieves state machine replication, which provides tolerance against Byzantine nodes. Various other protocols including PBFT, PAXOS, RAFT, and Federated Byzantine Agreement (FBA) are also being used or have been proposed for use in many different implementations of distributed systems and Blockchain.	PBFT satisfies decentralization and security, and when the number of nodes increases, the network overhead becomes heavy and it's difficult to reach consensus efficiently; so not suitable for large network

The key consideration factors comparison is listed in the following table.

	PoW	PoS	DPoS	BFT/PBFT	RAFT	Dpos＋Algorand
BFT(Byzantine Fault Tolerance)	50%	50%	50%	33%	No	50%
CFT(Crash Fault Tolerance)	50%	50%	50%	33%	50%	50%
Block Validation Time	BTC: 60 min ETH: 1 min	<100s	<100s	<10s	<5s	<60s
Scalability	Strong	Strong	Strong	Weak	Weak	Strong
TPS (Transactions per second)	<100	<1000	>500	>1,000	>5,000	>1,000
Resource Consumption	High	Medium	Low	Low	Low	Low

Table 1　Comparison Of Main Consensus Algorithm

Under current HNB application scenario and economic model, efficiency is the priority so that HNB platform will be able to support the large numbers of concurrent transactions. Massive real-time transactions cannot tolerate long confirmation time. At the same time, the system operating around payments, while the low latency is emphasized, shall not cause double-spending problems, the reliability is crucial. Security is another critical for systems that operate around financial payments and data. As the trades-off, decentralization requirement takes the back seat.

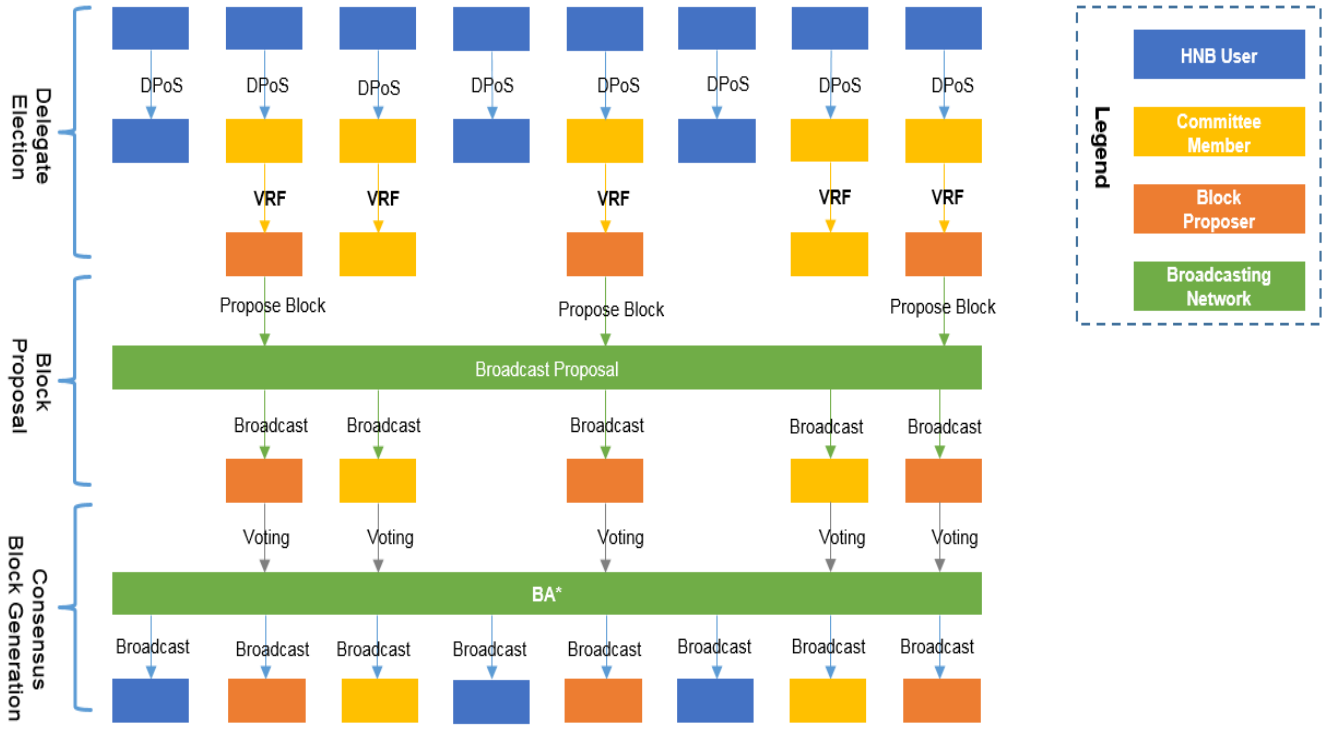
Based on the above analysis and assumptions, HNB architecture embraces an innovative consensus algorithm based on DPoS + Algorand for HNB decentralized economic community blockchain.

Summary of HNB Consensus Mechanism Advantages:

- DPoS based delegate election takes full account of interests distribution and fairness, and avoid attackers;
- Using DPoS to run for delegate nodes will be more stable and relatively high performance than using Algorand to select users with Verifiable Random Functions (VRFs)
- Fast to reach consensus
- Algorand’s Cryptographic Sortition ensured that the accounting node is completely random and secretive and specific attacks will not work
- Scalable to support large number of nodes without performance degrade
- Ensure Security without heavy computation overhead and energy-saving
- Never hard fork
- High fault-tolerant

6.2 HNB Consensus Algorithm

6.2.1 Consensus Algorithm Overview



Figure#6 DPoS + Algorand Consensus Algorithm Process

HNB adopts consensus algorithm based on DPoS + Algorand for HNB decentralized economic community blockchain.

- **Delegate Election Process based on DPoS**

  - Election of 99 accounting delegates from entire HNB community
  - 99 delegates are divided evenly into 3 groups, which represents 3 different interest groups, consumer, producer (or merchant) and contributor.
  - The constitution of delegates ensures no single interest group can have overwhelming advantage over and impact on the normal operations of the entire community
  - HNB community users vote for candidate based on the weight of their currency and holding time (Token Age)

- **Block Validation Process Based on Algorand**

HNB architecture adopts Algorand as its core consensus algorithm for transaction validation process.

- Using Algorand algorithm's **Cryptographic Sortition** to select 21 accounting nodes including 1 master notes among 99 selected delegates during the current accounting cycle
- Cryptographic Sortition is an algorithm for choosing a random subset of users according to per-user weights; that is, given a set of weights and the weight of all users. Sortition is implemented using **Verifiable Random Functions (VRFs)**
- **Verifiable Random Functions (VRFs)** ensures randomly select users in a private and non-interactive way. This prevents an adversary from forecasting, modifying and attacking accounting node selection
- The selected Master Node proposes a new block. Then we use a new **Byzantine Agreement (BA)** protocol called **BA★** to reach consensus among 21 accounting nodes
- **BA★** is designed to guarantee consensus as long as a weighted fraction (a constant greater than 2/3) of the tokens are controlled by honest users
- HNB BA* algorithm uses multiple rounds of interaction to ensure absolute consistency of the final proposed block among accounting nodes, which ensures that there will never be problems such as consensus partitioning and block forking

6.2.2 Selection Procedure

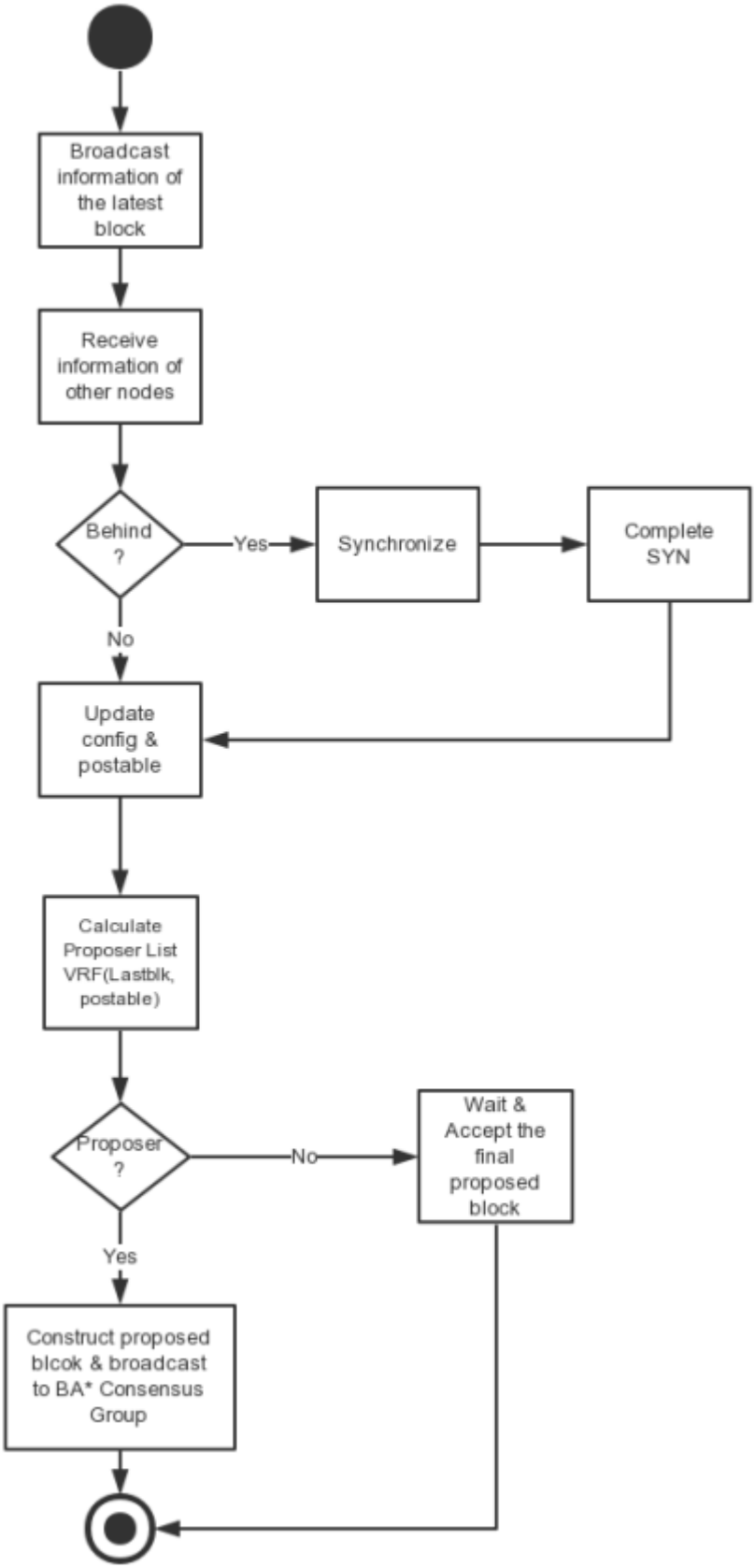


Figure 7 Selection Procedure

- **Steps**

1. Broadcast status information of the latest block of your own and receive status information of other nodes.
2. Check if you are behind, if it is behind, enter the synchronization process, otherwise go to step 3.
3. Update the configuration information and update the consensus node asset table.
4. Each node independently calculates the list of proposers. (calculated based on the information of the previous block and the asset table)
5. If you are a proposer, go to the transaction pool to get a deal build proposal message. (more than one proposer)
6. Broadcast the proposal message to the BA★ Consensus Group.

- **Implementation Details**

Sortition requires a role parameter that distinguishes the different roles that a user may be selected for; for example, the user may be selected to propose a block in some round, or they may be selected to be the member of the committee at a certain step of BA. Algorand specifies a threshold τ that determines the expected number of users selected for that role.

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>

A user performs sortition by computing ⟨hash,π⟩ ← VRFsk(seed||role)， where sk is the User's secret key。The pseudo-random hash determines how many sub-users are selected, as follows. The probability that exactly k out of the w (the user's weight) sub-users are selected follows the binomial distribution. Since $B(k_1;n_1,p)+B(k_2; n_2, p)= B(k_1 + k_2; n_1 + n_2, p)$， splitting a user's weight (currency) among Sybils does not affect the number of selected sub-users under his/her control.

To determine how many of a user's w sub-users are selected, the sortition algorithm divides the interval [0,1） into consecutive intervals of the form

$I^j = [\sum_{k=0}^{j} B(k; w, p) , \sum_{k=0}^{j+1} B(k; w, p)) \; for \; j \in \{0,1,000, w\}$ .If hash / $2^{hashlen}$ (where hashlen is the bit-length of hash) falls in the interval I^j， then the user has exactly j selected sub-users. The number of selected sub-users is publicly verifiable using the proof π (from the VRF output).

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** [$\sum_{k=0}^{j} \text{B}(\text{k}; \text{w}, \text{p})$, $\sum_{k=0}^{j+1} \text{B}(\text{k}; \text{w}, \text{p})$) **do**

    j++
return j

Sortition provides two important properties. First, given a random seed, the VRF outputs a pseudo-random hash value, which is essentially uniformly distributed between 0 and $2^{\text{hashlen}} - 1$. As a result, users are selected at random based on their weights. Second, an adversary that does not know sk_i cannot guess how many times user i is chosen, or if i was chosen at all (more precisely, the adversary cannot guess any better than just by randomly guessing based on the weights).

The pseudocode for verifying a sortition proof, shown in VerifySort Algorithm, follows the same structure to check if that user was selected (the weight of the user's public key is obtained from the ledger). The function returns the number of selected sub-users (or zero if the user was not selected at all).

6.2.3 Synchrony Process

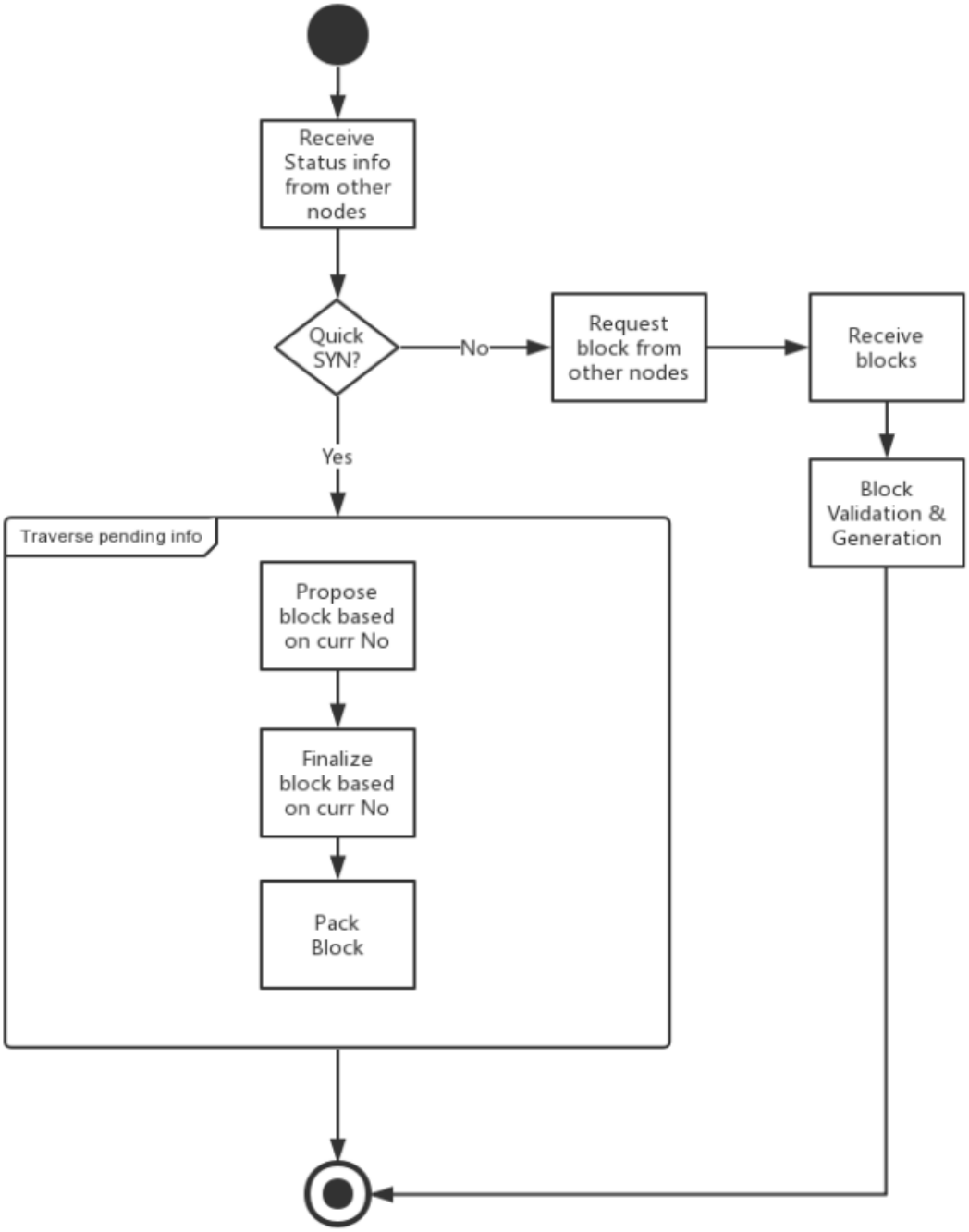


Figure 8 Synchrony Process

- **Steps**

1. To determine if you can synchronize quickly, you can go to step 2, otherwise go to step 3.
2. Traverse all unprocessed consensus information in memory.
3. 2.1 Construct proposal information based on the current block number.
4. 2.2 Construct the final proposal information based on the current block number.
5. 2.3 Pack the block and update the status.

6. Send a block request to the other node and wait for the response block to be received, verify the block and generate the block.

- **Implementation Details**

Synchrony assumption allows Algorand to select the users' public keys (and associated weights) before the adversary knows the selection seed, ensuring random sampling of users for BA★ committees. We next leverage this property to, informally, show that the committee members for the final step represent well all the users in the system. Therefore, if sufficiently many of them vote for a block to be final, then sufficiently many of the users have finished BinaryBA⋆ at the first step and will not vote in future steps.

We set τ_{final}= 10,000， and assume that the weighted honest fraction of users is h = 0.8. Thus, the expected number of the honest members in the final step committee is 0.8*10,000= 8000， while the expected number of malicious members for that step is 0.2*10,000=2000. We next show that the probability that 7400 step-final votes for round-r' block A are produced, but another block B is also certified for the same round, is negligible.

Let $seed_{r-1-(r \bmod R)}$ be the seed used for selecting the committees for round r'∈[r， r + R]. If the round when that seed was published, i.e. round r −1−(r mod R), happens when the network is not strongly synchronous, then the adversary may manipulate the selection of the seed (seedr−1−(r mod R)) by discarding block proposals from honest users. We first show that despite the ability of the attacker to manipulate the seed, it is infeasible to compute a seed that: (1) gives i malicious committee members in the r' round final step, and (2) j malicious committee members in some other step of round r', where i +3j > 4,100.

The expected numbers for i and j are 0.2*1000 = 2000 and 0.2*2000 = 400 respectively. So for a fixed round and step, we apply the formula in Equation 2, to find the probability that a random seed gives i +3j > 4,100:

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

To check if a seed value satisfies i + 3j > 4,100, the adversary has to do at least one cryptographic hash or VRF operation. Each round has committees for at most MaxSteps = 150 steps. For each round, only 2/3 of those 150 (i.e., 100) committees can create a certificate (only those that vote in the first or second step in the BinaryBA⋆ loop). So by union bound, the probability that a seed value gives i +3j > 4,100 for a specific round (and any step of that round) is $\simeq 2^{-93}$. So it is infeasible for the adversary to find such a seed.

Notice that our argument above does not exclude the possibility that more than a single pair of round final committee has i malicious members and a targeted BinaryBA⋆ committee has j

malicious members, where i +3j = 4,100. Intuitively, however, if the adversary concentrates all of his computational power on finding a round (with highly malicious final committee to finalize block A) and a targeted step within that round (with a highly malicious committee to certify block B) such that i +3j = 4,100, then for all other rounds and steps, the number of malicious committee members is random, which makes the probability that the attacker succeeds in one of those round/step pairs negligible ($<10^{-20}$).

More precisely, if the adversary targets two round/step pairs where i +3j > 3850, then both of those pairs will have i +3j < 3950. This is true, because the probability that i +3j > 3850 is approximately $0.41*10^{-18}$, while the probability that i +3j > 3950 is approximately $0.6*10^{-23}$, so the probability that both events happen for two specific round-step pairs is less than $2.5*10^{-41}$. Since for a specific selection seed there are $R = 10^3$ rounds with 10^2 steps each, there are $\binom{10^2.10^3}{2} = (1/2)10^{10}$ possible pairs of round/step. The probability that the attacker manages to target more than one round/step where i+3j > 3950 is therefore, by union bound, less than $2.5*10^{-41}*0.5*10^{10} = 1.25*10^{-31} < 2^{-93}$, so it should never happen.

In sum, the attacker has two strategies: (1) target one round and step within that round such that i +3j≤4100, or (2) target multiple step/round pairs where i +3j≤3950. We next analyze these two strategies.

Strategy 1: one round/step pair where i+3j≤4100. If the final step committee at round r' has i malicious members, then any other committee at the same round has at most $\frac{4100-i}{3}$ malicious members. Clearly, the worst case is when i +3j = 4100. Thus, in order to prove that the probability of failure is 10^{-7}, we need to show that for all integral tuples (i, j) on the line i +3j = 4100, the probability that both committees have enough users is less than 10^{-7}. Equivalently, for each integral value j in the interval $[0,\frac{4100}{3}]$, we need to prove that for i = 4100−3j, the probability that both committees have enough honest members (i.e., the final committee has 0.74*10,000−i = 7400−i and the other committee has 0.685*2000−j = 1370−j honest members) is less than 10^{-7} (since $T_{final} = 0.74$, $\tau_{final}= 1000$and Tstep = 0.685, $\tau_{step}= 2000$). For each integral value $j\in[0, \frac{4100}{3}]$ we did a separate calculation to prove that the desired probability always is less than 10^{-7}.

We next describe the method used to upper-bound the probability for each tuple (i, j) = (4100−3j, j). Fix (i, j). The i malicious votes go towards the final block A, while the j votes go towards certifying another block B. So 7400−i more signatures are required for the final committee and 1370−j more signatures for the other committee. Let $\dot{A}$ denote the set of honest users who would vote to approve block A as final (if completed BinaryBA⋆ at the first step and selected to the final committee), and $\dot{B}$ the set of honest users who would vote to certify another block B (≠ A) if they were in a committee and did not finish BinaryBA⋆ at the first step. Clearly, $\dot{A} \cap \dot{B}=\emptyset$.

So each honest user eventually becomes a member of at most one of the two sets. For the 7400 and 1370 thresholds to both be reached, set $\dot{A}$ must include 7400−i committee members for final, while set $\dot{B}$ must include 1370−j committee members for another step.

We say that an honest user "joins" set $\dot{A}$ when that user counts sufficiently many (more than $T_{step} \cdot \tau_{step}$) votes for block_hash to complete BinaryBA⋆ in its first step. Else (after the first step of BinaryBA⋆), we say that the honest user "joins" set $\dot{B}$. Without strong synchrony, the network is under the attacker's control, so the attacker can choose whether a user joins set $\dot{A}$ or $\dot{B}$ (by delivering votes to that user). This allows the attacker to get some feedback: if the attacker lets a user join set $\dot{A}$, then he can observe whether that user is selected to the final committee (i.e., see if that user votes). In this fashion the attacker can let users join set $\dot{A}$ one by one until sufficiently many votes are produced to approve block A as final, and then join the rest of the honest users to set $\dot{B}$.

Therefore, users join set $\dot{A}$ in some order. Let n be the size of $\dot{A}$ when the users in set $\dot{A}$ can, for the first time, produce 7400−i final step votes in the targeted round. Let H be the total number of units of Algorand currency held by honest users; to simplify our proof, we assume that each user has exactly one unit of currency so H is also the number of users (if a user has more than one unit, we consider that user to be represented by multiple "subusers" each with one unit of currency). For a fixed value of n, at most H −n users join set $\dot{B}$. For the attacker to succeed, these H −n users must produce 1370 − j votes for block B in the targeted step of BinaryBA⋆. Let Pk denote the probability that: (1) n = k, and (2) the remaining H−k users in set $\dot{B}$ produce 1370-j votes for block B. Pk is given by:

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

Using the formula in Equation 2 we can compute Pr [the H-k users in $\dot{B}$ produce exactly c votes]. The expected number of honest users selected for the committee in the set $\dot{B}$ (which includes H-k users) is h*2000 * $\frac{H-k}{H}$, so for h = 0.8 the expected number of users is 1600 · $\frac{H-k}{H}$. Therefore, the expression in Equation 3 equals to:

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

The adversary succeeds with probability $\sum_{k=0}^{H} P_k$, which is equal to:

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

The inner sum converges and can be evaluated. However, the outer sum iterates over total number amount of honest currency units which might be arbitrarily large. We therefore evaluate an upper bound:

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

The above is the upper Riemann sum of the integral of $\sum_{c=1370-j}^{\infty} \frac{\left(1600\frac{H-\frac{f-1}{100}H}{H}\right)^c}{c!e^{1600\frac{H-\frac{f-1}{100}H}{H}}}$ from 0 to H, where the interval [0,H] is partitioned into 100 equal parts. Since the function $\sum_{\mathrm{c}=1370-\mathrm{j}}^{\infty} \frac{\left(1600\frac{H-k}{H}\right)^c}{c!e^{1600\frac{H-k}{H}}}$ is decreasing in ,k this is an upper bound. It is evaluated as follows:

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

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

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

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

Thus, the probability of failure is upper-bounded by the following formula, which can be

computed:

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

Strategy 2: multiple round/step pairs where i+3j ≤ 3950. We first show that the adversary cannot target 3 round/step pairs, where the final committee has at least i malicious members and the step committee has j malicious members where i +3j > 3850. This is because the probability of targeting one round/step pair where i +3j > 3850 is at most $0.41*10^{-18}$, so the probability to target three such pairs is $(0.41*10^{-18})^3 < 10^{-51}$. For each selection seed there are $R = 10^3$ rounds and each round has at most 10^2 steps that the attacker can target (i.e. steps where BinaryBA⋆ can conclude and certify a block). So we have at most 10^3*10^2 step committees set for every selection seed, therefore the number of step/round pairs that the attacker can target is $\binom{10^2 10^3}{3} < 10^{15}$. The probability of attacker success is thus bounded by $10^{15}*10^{-51} = 10^{-36}$. Thus, the attacker can have at most two pairs with i +3j > 3850, in which case for both we will have i + 3j < 3950. This allows us to analyze the following two cases (within the above attacker strategy):

(1) Two targeted round/step pairs where 3850 < i + 3j ≤ 3950. It can be shown (with similar calculations to the case i +3j ≤ 4100) that the probability of success for the adversary success is less than 10^{-9} for one targeted round/step pair, and therefore less than $2* 10^{-9}$ in total.

(2) All round/step pairs have i + 3j ≤ 3850. It can be similarly shown that probability of success for the adversary is less than 10^{-20}. Since each round has at most 10^2 steps and there are $R = 10^3$ rounds set by the same selection seed, the attacker's probability for success is bounded by $10^2*10^3*10^{-20} = 10^{-15}$.

In conclusion, strategy 2 provides a lower attacker success rate than strategy 1, where the attacker's success rate is bounded by $10^{-7}$for every sequence of $R = 10^3$ rounds.

6.2.4 BA★Process

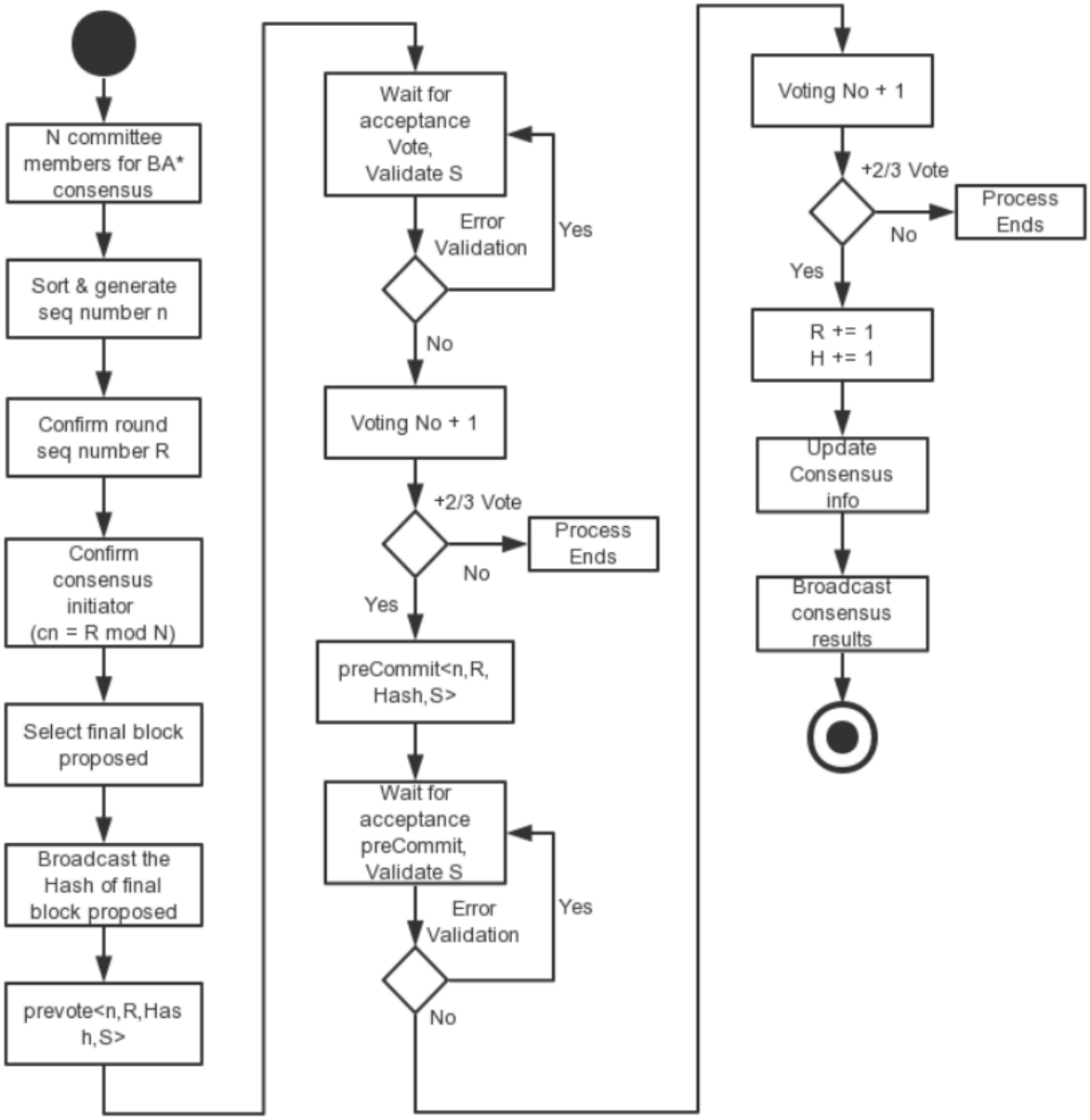


Figure 9 BA★Process

- **BA★Process Description**
  1. The proposer sends the proposed block to the proponents participating in the BA★, where the total number of proponents participating in the BA★ is N = 3f + 1.
  2. Perform a hash calculation on the Hash calculation of the proponent information participating in the BA★ to obtain the sequence number n.
  3. Obtain this round of consensus number R.
  4. Determine the current round of consensus initiator number curNum = R mod N.
  5. Take the proposed block hash minimum and the final proposed block.
  6. The Consensus Initiator broadcasts the Hash of the final proposed block to other proponents participating in the BA★.

7. The proponent who participates in the BA★ pre-votes the proposed block. The pre-voting needs to include the own node number n, the pre-voting round R, the voting block hash, and the validator voting signature S.
8. The proponent who participated in the BA★ waited for the number of votes to be accepted within the timeout period to identify the voting message.
9. Collect the number of pre-voting, more than 2/3 into the preCommit process.
10. The proponent of the participating BA★ pre-submits the preVote block, which needs to include its own node number n, pre-voting round R, voting block hash, and validator voting signature S.
11. Collect the number of pre-submissions, more than 2/3, complete the preCommit process.
12. Update block height H, round R, and broadcast consensus results.

.

- **Implementation Details**

For efficiency, BA⋆ votes for hashes of blocks, instead of entire block contents. At the end of the BA⋆ algorithm, we use the BlockOfHash() function to indicate that, if BA⋆ has not yet received the pre-image of the agreed-upon hash, it must obtain it from other users (and, since the block was agreed upon, many of the honest users must have received it during block proposal). The BA⋆ algorithm also determines whether it established final or tentative consensus.

Running BA⋆ for the next round, with a proposed block. H is a cryptographic hash function.

```
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*)>
```

The CountVotes() procedure reads messages that belong to the current round and step from the incomingMsgs buffer. (For simplicity, our pseudocode assumes that a background procedure takes incoming votes and stores them into that buffer, indexed by the messages' round and step.) It processes the votes by calling the ProcessMsg() procedure for every message, which ensures that the vote is valid. Note that no private state is required to process these messages.

```
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
```

ProcessMsg(): validates incoming vote message m.

```
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() returns not just the value contained in the message, but also the number of votes associated with that value. If the message was not from a chosen committee member, ProcessMsg() returns zero votes. If the committee member was chosen several times, the number of votes returned by ProcessMsg() reflects that as well. ProcessMsg() also returns the sortition hash.

The Reduction() procedure, converts the problem of reaching consensus on an arbitrary value (the hash of a block) to reaching consensus on one of two values: either a specific proposed block hash, or the hash of an empty block. Our reduction is inspired by Turpin and Coan's two-step technique. This reduction is important to ensure liveness.

```
procedure Reduction(ctx, round, hblock):
// step 1: gossip the block hash
CommitteeVote(ctx, round, REDUCTION_ONE, τ_STEP, hblock)
// other users might still be waiting for block proposals, so set timeout for λ_BLOCK + λ_STEP
```

```
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⋆(), which reaches consensus on one of two values: either the hash passed to BinaryBA⋆() or the hash of the empty block. BinaryBA⋆() relies on Reduction() to ensure that at most one non-empty block hash is passed to BinaryBA⋆() by all honest users.

```
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 Consensus Incentive Mechanism

The sustainable development of the virtual currency ecosystem is inseparable from the well-designed ecosystem model, and it is also inseparable from the rationally optimized consensus incentive mechanism. The new economic incentive mechanism created by Satoshi Nakamoto guarantees the long-term stability of Bitcoin for 10 years, has a strong supporter community, and thus establishes a new industry ecosystem. It can be seen that the consensus incentive model plays an important role in the development of digital currency ecology.

The HNB incentive algorithm is based on the economic model, which greatly enhances the enthusiasm of the participating users in ecological construction and maintains the ecological sustainable development. HNB provides HGS rewards for each round of consensus-successful accounting nodes. In HNB community a certain percentage of HGS will be deducted for disciplinary action.

The incentives for HNB economies follow the following guidelines:

1. HNB calculates the weight w according to the age of the coin and the number of coins held, and the probability of being selected as the consensus group is positively correlated with the value of w.
2. After the consensus is successfully reached, identify the proponents of this consensus and the proponents who participate in BA⋆. The number of HGS rewards will be distributed to the relevant members of this consensus.
3. The number of HGS rewards is positively correlated with the sum of transaction complexity within the block, and there is a correlation coefficient k.
4. HNB supports graded penalty, distinguishing between malicious node and node failure. If the proposer sends a false or incorrect proposal, or the proponent who participates in the BA⋆ sends an abnormal consensus agreement, it will be determined as malicious node. If the node doesn't respond to network interaction during the consensus process, it is determined as a node failure.
5. The correlation coefficient k of the HGS reward is negatively correlated according to the total number of blocks. (Whether reward attenuation or not)

7 Data Storage

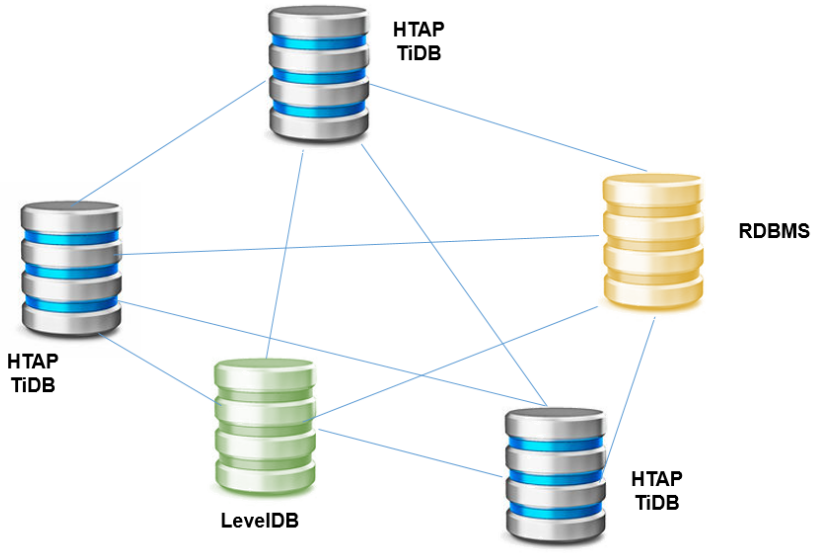
7.1 Distributed Storage

7.1.1 Distributed Database

Google published the Spanner / F1 paper and implemented the Spanner database based on the paper. It provides distributed read and write transactions (strict two-stage lock + two-phase commit), and F1 implements global indexing. With the era of distributed HTAP (Hybrid Transactional and Analytical Processing) databases, HTAP is a new database technology that combines the best features of traditional RDBMS and NoSQL.

The distributed HTAP database supports unlimited horizontal scaling with strong consistency and high availability. The goal is to provide a one-stop solution for OLTP (Online Transactional Processing) and OLAP (Online Analytical Processing) scenarios.

For blockchain ledger storage, the distributed database provides good support for ledger storage, access, and expansion, and is suitable for the underlying blockchain system of the commercial environment. HNB blockchain's ledger storage, not only supports RDBMS and NoSQL databases, but also supports distributed HTAP databases to provide better data management support for HNB's business operations.



HNB's HTAP database technology will be based on the latest NewSQL open source database TiDB (HTAP Database for TiDB), supporting both online transaction processing (OLTP) and online analytical processing (OLAP) business types of distributed database products; also supports strong consistency Distributed transactions, online elastic expansion, remote and automatic failure recovery; compatible with MySQL database.

For light nodes (see 11.4 SPV nodes for details), the HNB blockchain uses Level DB. LevelDB is a very efficient Key-Value database implemented by Google. The key values are binary and currently support a billion-level data volume. LevelDB uses the LSM (Log Structured Merge) algorithm to maintain very high performance at large data volumes. Log-Structured Merge-tree (LSM-tree) is a disk-based data structure that provides efficient indexing of files that require high-speed record insertion (and deletion) for lower system resource overhead. LSM_Tree delays and batches index changes and efficiently migrates updates to disk in a manner similar to merge sorting, reducing index insertion overhead.

7.1.2 Distributed File Storage

IPFS (Inter-Planetary File System) is a protocol that defines: a content-addressed file system; coordinated content distribution; combined with Kademlia + BitTorrent + Git.

IPFS file system, each file and all blocks in it are given a unique fingerprint of the encrypted hash. IPFS deletes files with the same hash value through the network. It can be used to determine which files are redundant and duplicated. And keep track of the version history of each file. Each network node only stores the content it is interested in, as well as some index information, which helps to figure out who is storing what. When looking for a file, the hash value of the file can be used to find the node where the file is stored in the network and find the desired file.

The HNB blockchain system serves the real business activities, and the community should provide high-availability and high-security file storage services for the community merchants. The IPFS file system features multiple copies and automatic redundancy management, making it ideal for autonomous organizations to provide file services. HNB's underlying file storage will be integrated with the IPFS file system to provide file management services to community merchants.

7.2 Data Sharding

Sharding is a process used extensively in databases for enhancing their efficiency. A shard represents a horizontal section of a database, and separate servers store each shard. The result is a uniform spread of the load, which in turn, increases database efficiency.

The data access efficiency and scalability of blockchain are important considerations for data storage solutions. HNB's data storage solutions use data sharding to meet efficiency and scalability requirements.

The distributed HTAP database provides the capability to support data sharding.

For data write operations, the HTAP database automatically fragments the underlying

data according to the range of Key (sharded data), with each shard being an interval. The Key-Value trigger rule setting in the shard (for example, the amount of data exceeds a certain threshold) will be automatically fragmented to support horizontal expansion of data.

For the data read operation, the HTAP database uses the load balancer to schedule the load of the cluster according to the state of the storage cluster. The scheduling is based on the shard, and the strategy configured by the load balancer is the scheduling logic, and all the processes are automatically completed to ensure that the data reading can be completed efficiently.

7.3 Pluggable Storage

The storage of the blockchain should be designed with different storage tools and storage schemes according to different application scenarios. Therefore, the blockchain storage should not be strongly bound to the storage media. HNB's data storage supports different types of databases for RDBMS, NoSQL, HTAP, and IPFS file systems.

For blockchain data storage, it is recommended to use the database for the storage solution, which can be selected between the RDBMS, NoSQL and HTAP according to the specific requirements. HNB will also provide pluggable storage media, and community participants can switch data storage media based on actual operating conditions.

7.4 Data Encryption

The blockchain can be tamper-proof, decentralized, and run in an untrusted network environment, but the user's accounting book including transaction information is transparent to the participating nodes, and any participants can have access to these data.

If the user and business privacy data are all put on the chain, this will increase the risk of data privacy breaches. Currently in the public chain system such as Bitcoin, all transaction element information is public (such as transaction amount). In the real business activities, the transaction element information is sensitive data, and non- related parties cannot have access to it.

HNB blockchain’s data encryption solution:

- **Ledge Data Encryption**: digital asset transaction information is encrypted and stored in the blockchain
- **Account ID Masking:** the account ID is masked, and only the HNB-ID info is stored on the chain. The actual identity information corresponding to the HNB-ID is managed off-chain and stored with encryption
- **Commercial Data Storage:** the merchant's transaction data contains business sensitive information, these commercial data is stored independently of the chain with

encryption and access control

8 Multi-Asset Ledger

8.1 Multi-Asset Ledger Architecture

8.1.1 Multi-Chain Approach

Multi-chain, that is, abandoning the traditional scheme of “one chain and all”, adopts the “one chain one contract” scheme to ensure that each contract can operate independently. This innovation greatly simplifies the architecture, reduces the pressure on data processing, and ensures that the surge in traffic on one chain does not affect the efficiency of the other chain. Any business carried out on the chain is not interfered by other services, effectively implementing resource isolation.

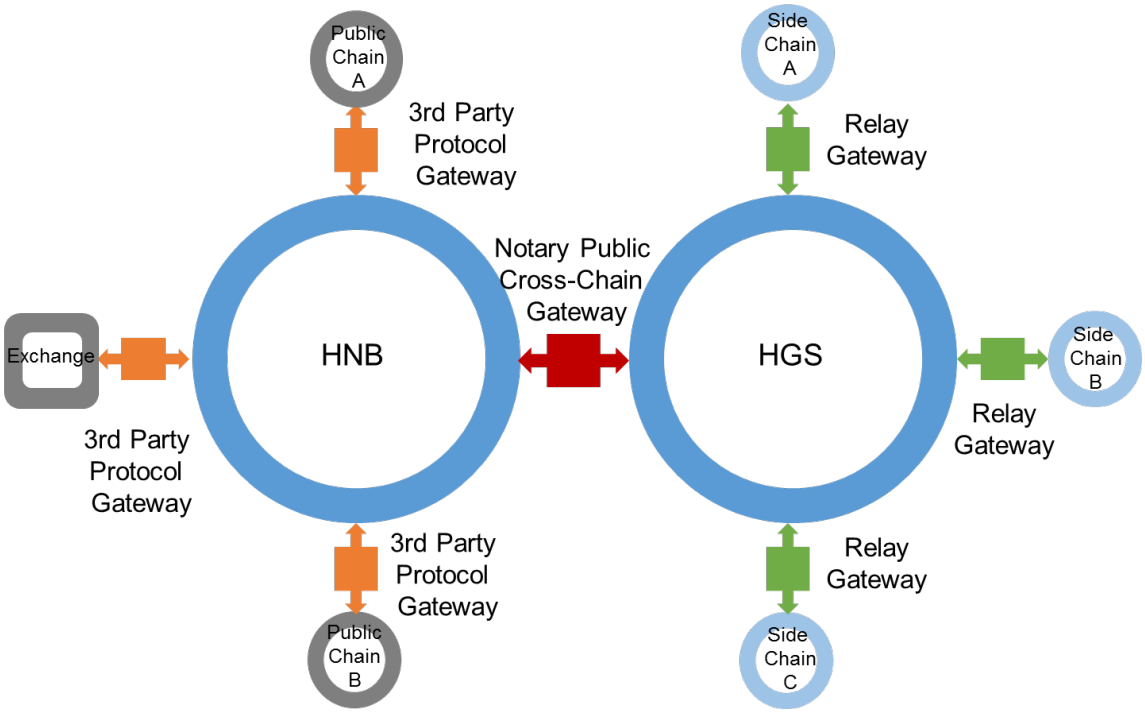


Figure 10 HNB Multi-Chain Structure

HNB's core blockchain operation adopts the "one chain and one token" architecture approach. HGS and HNB operate in parallel based on the double-chain architecture, without affecting each other. The multi-chain architecture ensures improved performance, isolation, scalability, and ecosystem integration to support HNB community’s dual-token economic growth.

In terms of performance, the multi-chain architecture ensures that the consensus model state machines are independent of each other during the operation of the dual native digital token (HGS and HNB). The time-consuming operations, such as transaction checking, voting waiting, and database access, are converted from serial execution to parallel execution, by expanding the technical resources. System latency can be greatly optimized, IO blocking and execution time can be reduced, and system throughput can be increased.

In terms of isolation, the multi-chain architecture will share the channel and transaction processing between HNB and HGS. HNB and HGS chains cannot obtain transaction-related information, nor can they read or modify the other party's ledger data. Physically separate the isolation of dual native token to ensure the isolation of HNB and HGS, and protect the privacy of users and transactions.

As for scalability, when the HNB underlying chain needs to be upgraded to HNB or HGS, the multi-chain architecture will allow a single chain to be upgraded separately without affecting other chain service execution. When the HNB underlying chain has increased chain requirements, the multi-chain architecture will easily add new chains, use cross-chain interaction technology to communicate with the original asset chain, realize the flow between digital assets, and ensure the high scalability of HNB.

8.1.2 Heterogeneous Ledger Architecture

HNB blockchain supports the heterogeneous ledger architecture, and designs a unique dual-token ledger structure based on the currency characteristics of HNB and HGS. The heterogeneous ledger architecture greatly improves the availability of the ledger module and provides various ledger query services.

HNB token as digital asset token that exchange with the outside world, needs to record information about transactions with external digital assets in the ledger, such as: source digital asset identification, target digital asset identification, and exchange identification.

As a stable token in HNB community, HGS token ledger needs to record relevant information within the community, such as transaction side chain identification, derivative digital asset identification, merchant identification, and commodity order summary, etc

8.1.3 Merkle Tree

HNB block header contains the Merkle Tree root value for quick verification of intra-block transactions.

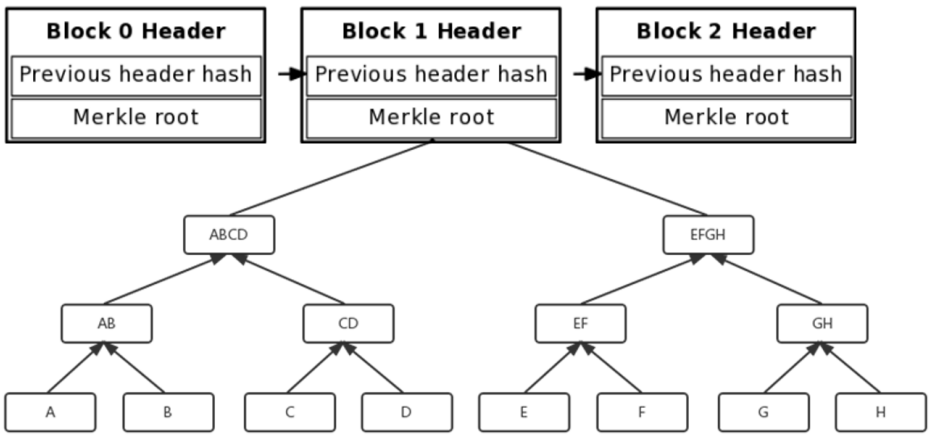


Figure 11 Merkle Tree

A hash value is calculate for each transaction in the blockchain. A Merkle Tree is constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains, the Merkle Root. Each leaf node is a hash of a block of data, and each non-leaf node is a hash of its children. Typically, Merkle trees have a branching factor of 2, meaning that each node has up to 2 children.

The key advantage of the Merkel tree transaction verification algorithm is that each transaction can be deleted directly, leaving only the hash value of the transaction. For the entire block, it does not change his cryptographic security and integrity, but the amount of data can be greatly reduced. The light node only saves the block header to ensure that the transaction cannot be falsified or forged.

For Example: If we need to prove that the "F" exists and has not been tampered with, we just need to provide "E", "GH", "ABCD", Merkel root can prove that the "F" transaction has not been forged and falsified.

8.2 Ledger Model

8.2.1 UTXO Model

The full name of UTXO is Unspent Transaction Output. The UTXO model organizes the input and output of all transactions in a chained manner, and the input of each transaction must be the output of a transaction. The UTXO model was originally created and cited by Nakamoto in the bitcoin model, but it is not unique to Bitcoin and is not necessarily related to the blockchain.

Advantages of the UTXO model:

- **Scalability:** as multiple UTXOs can be processed simultaneously, parallel transactions can be implemented and scalability innovations can be encouraged.

- **Privacy:** Bitcoin is not a completely anonymous system, but UTXO can provide a higher level of privacy as long as the user uses a new address for each transaction.

- **Double-spending Prevention:** Each transaction input of UTXO must be an unsuccessful transaction output with the ability to naturally prevent double-spending.

8.2.2 Balance Model

The balance model is the traditional accounting model, which requires account system support and the balance is associated with the account. Each transaction only needs to verify that the sending account has sufficient balance to pay for the transaction.

One disadvantage of the balance model is exposure to double spending attacks. Incremental random numbers can be implemented to counteract this type of attack. In Ethereum, each account has a publicly visible random number, and each time a transaction is made, the random number is incremented by one. This prevents the same transaction from being committed multiple times.

Advantages of the balance model:

- **Simplicity:** UTXO's stateless model forces transactions to contain state information, which unnecessarily complicates contract design.

- **Efficiency:** besides simplicity, the account/balance model is more efficient because each transaction only needs to verify that the sending account has sufficient balance to pay for the transaction.

8.2.3 HNB Hybrid Ledger Model

The HNB underlying chain adopts a hybrid ledger model strategy. HGS and HNB native token use the UTXO model to improve the underlying concurrency.

The digital assets generated through smart contracts adopt the balance mode, which is simple to develop, reduces the difficulty of developing smart contracts, and reduces the resulting security problems.

9 Transaction Management

9.1 Transaction Verification

HNB as a public blockchain system that supports the business of the HNB economic community, the effectiveness of the transaction verification is the top priority. The verification of the transaction is mainly divided into two parts: transaction integrity verification and transaction validity verification.

Transaction Integrity Verification refers to the verification of transaction signature, ensuring that the transaction message has not been tampered.

Transaction Validity Verification, including the validation of the transaction field format and the legitimacy of the transaction content. Among them, the transaction field format check is used to verify that the format length of all fields must be correct; the legitimacy check of the transaction content (including but not limited to the amount, double-spending). Different check processes are applied for different HNB node types.

For Full Node, the blockchain can be independently established and verified, from the first block (Genesis Block) to the latest block in the network. The full node can independently verify any transaction information without resorting to any other nodes or other sources of information. The node maintains a UTXO pool to verify the validity of the transaction.

For SPV Node, there is no need to store a complete blockchain ledger, a simplified payment verification (SPV) approach is adopted. This type of client is called an SPV client or a lightweight client. The SPV node only needs to download the block header instead of downloading the transaction information contained in each block. The resulting blockchain without transaction information is only 1/1000 of the full blockchain. Since the SPV node does not have the ability to construct all of the ledger information, its approach is different. The main steps are as follows:

1) Determine the block number where the transaction is located. Generally, the screening of the ange is carried out by the time of the transaction, and then the specific positioning

2) Determine the Merkel tree path for the requested transaction and perform verification

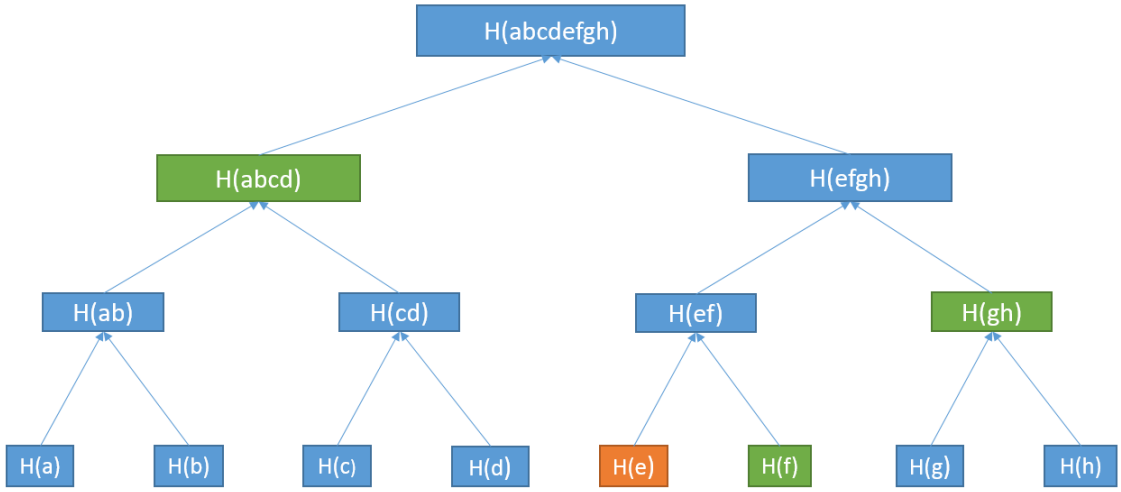


Figure 12 Merkle Tree Verification Path

As shown in the Figure above, to verify the existence of H(e) transactions, only H(f) transactions, H(gh) Merkel nodes, H(abcd) Merkel nodes, and H(abcdefgh) Merck are required. The root node of the tree can construct an audit path, and then calculate the root hash and the root hash of the SPV node to verify the validity of the transaction.

9.2 Transaction Pool

After completing the transaction verification, the system will broadcast the transaction to the whole blockchain, and put the transaction into the transaction pool management for next step processing.

If the transaction volume is large, the memory may become the bottleneck of the transaction pool management. HNB will expand and maintain an extended transaction pool, which is greater than the transaction volume of the memory transaction pool. HNB will choose to persist the transaction and use DB or file for storage.

When a transaction is confirmed to be packaged into a block, the system will delete the transaction from the transaction pool. And the transactions in the extended trading pool will be moved into the in-memory transaction pool.

9.3 Sorting Strategy

After the transaction is put into the transaction pool, HNB system will use the preset policy to determine which transactions are taken out of the pool and packaged into the block.

The traditional transaction fee-based sorting strategy has the disadvantage that if the transaction fee for a submitted transaction is too low, and the transaction can never be confirmed. This will cause the effect of "freezing" transaction due to double-spending

verification.

For scenario of the HNB services for real economic activities, the FIFO mode is adopted for transaction queuing without transaction fee charge. HNB's FIFO sorting strategy fully reflects the fairness of first come first served, the earlier the transaction is submitted, the easier it is to be packaged into blocks.

10 Smart Contract

10.1 HNB Smart Contract Design

The smart contract of the HNB blockchain system supports: golang and JavaScript languages. So that developers of smart contracts do not need to learn new languages and lower the barriers for merchant access. Using golang as the execution environment for smart contracts, smart control logic can be implemented for the HNB application layer framework. The golang virtual machine has Turing completeness, can implement arbitrary logic, and is highly deterministic, which is very suitable for commercial business scenarios with high deterministic requirements.

At smart contract layer, the programming languages support contract parsing and conversion, and flexibly supports the basic application of the virtual machine. The external interface of the virtual machine is realized through customized API operations, which can flexibly interact with the ledger account data and external data.

10.2 Smart Contract Management

10.2.1 Smart Contract Release

HNB smart contract development platform can be used by HNB community users to develop customized smart contracts and complete testing. HNB smart contract development platform provides a complete tool set, including development, testing, debugging, API description, etc., efficiently locates smart contract bugs, detects contract write exceptions, and effectively reduces security issues caused by vulnerabilities in smart contracts.

The contract development platform makes it easy to publish contracts for one-click deployment.

10.2.2 Smart Contract Access

As a public chain, the query operation of HNB smart contract does not need to be authorized. The write operation of the smart contract can only be executed after the ID

verification process via cryptographic algorithm. HNB provides two smart contract access methods: external access, inter-contract access.

External Access: HNB supports sending transaction information through wallet, and querying block information and transaction information through wallet and browser. It also supports for accessing contract related information via gRPC or RESTful interfaces.

Inter-contract Access: HNB fully considers the interaction and collaboration between various business domains, thus supports data interaction between smart contracts, opening up industry barriers and achieving data interconnection.

10.3 Smart Contract Running Environment

HNB adopts the self-developed HNB-VM virtual machine platform. HNB-VM performs deep optimization on the execution of instruction code parsing, which improves the execution efficiency of instructions while ensuring security. HNB-VM features are as follows:

- **High Speed Cache Access:** HNB-VM creates a multi-layer cache and designs a cache structure according to the probability of occurrence of the instruction code, which improves the cache hit probability and reduces the number of times of calculating and reading the object address data.

- **Instant compilation:** The traditional virtual machine platform is interpreted as a series of operations such as stacking, popping, reading and writing variable areas, and reading method areas. These operations could seriously affect the efficiency of virtual machine interpretation. HNB will compile the contract hotspot program in real time, reducing the operation steps of the registers.

- **Storage structure optimization:** Optimize the object storage structure, establish the mapping relationship between the object identifier and the storage address, and realize efficient retrieval of the object storage address. At the same time, optimize the attribute information in the object structure to improve the memory resource utilization.

10.4 Formal verification

In 2016, hackers successfully exploited 3.6 million Ethereums through The DAO, exploiting vulnerabilities in smart contracts. After the DAO incident, Ethereum founder Vitalik Buterin proposed to modify the Ethereum code, implement a hard fork on the Ethereum blockchain, and roll back the transaction record of hacking funds, while also receiving support from most miners in the community. The strong opposition from a few people eventually led to the split of the Ethereum community.

Strengthening smart contract audit is an important guarantee for improving the security

of blockchain. Formal verification is an effective method for smart contract audit. Formal verification of HNB is based on the established formal specifications, and the relevant characteristics of the specified system are analyzed and verified to judge whether the system meets the desired characteristics.

Formal verification does not completely ensure that the performance of the system is correct, but it can maximize the understanding and analysis of the system, and try to find errors such as inconsistency, ambiguity, incompleteness and so on.

11 Node Management

11.1 Node Type

HNB nodes are classified into full-node, SPV nodes, and accounting nodes according to their functionality. HNB users can choose different types of nodes to participate in the HNB ecosystem according to their own conditions (hardware resources, participation identities).

- **Full node** has full ledger accounting data and provides external query information and block information services
- **SPV node** can determine whether a transaction exists in the blockchain system
- **Accounting node** implements consensus process with other nodes, generates and broadcasts the ledger data

11.2 Full Node

Full node is the node with the complete HNB ledger information. Full node needs to synchronize the accounting data with the accounting node, can independently verify the block data validity and broadcast the transaction, and provide the block query service for the SPV node. .

HNB full node records all the ledger information. It is very convenient to verify the transaction balance, double-spending, transaction signature, etc., and can also determine the validity of the transaction without relying on other node data.

However, the node storage request shall be addressed. With the increasing volume of transaction and data, the database read and write operations will be a key concern. HNB supports scalable data storage such as data sharing to support efficient storage of full nodes.

11.3 Accouting Node

Accounting node is the blockchain node participating in the consensus process via

running the Algorand consensus protocol to ensure the stable operation of the HNB blockchain system. Accounting node writes the block information after consensus process into the HNB ledger, and broadcasts it to the entire chain. As accounting node performs the Algorand consensus, the validity of the transaction will be verified.

Accounting node is selected from validated full nodes in the community based on DPoS algorithm, and accounting node is a subset of full node. Accounting node has the same function as the full node.

Based on the HNB consensus algorithm mechanism, there will be 99 incumbent accounting nodes in the community at any moment. In the new accounting cycle, the entire network will select 99 nodes as accounting nodes through DPoS algorithm. The accounting cycle is based on the number of generated blocks as a batch. One batch is 3,000,000 blocks (approximately 6 months period).A new election will be triggered for each new batch.

11.4 SPV Node

SPV stands for "**S**implified **P**ayment **V**erification". SPV node only stores the header information of the ledger, and determines whether a transaction exists with the blockchain system based the stored block header information and the Merkel tree of the transaction structure.

SPV node can verify the existence of the transaction but cannot verify the validity of the transaction. SPV node requests transaction information from full node and verifies whether the transaction information returned by the full node has been tampered or not. However, the SPV node has no other transaction information, and it is impossible to determine whether the transaction output is double-spending, so the transaction validity cannot be verified.

11.5 Node Authorization Management

Since the HNB blockchain system serves the HNB ecosystem, non-community members are not allowed to use the HNB ecosystem service. Therefore, the HNB blockchain system is designed with a strict node admission mechanism to perform strict identity verification on nodes joining the HNB blockchain network.

The 99 committee members selected from the HNB community will participate in the DPoS + Algorand consensus process. Committee members will be required to apply to the community as HNB accounting nodes, and the community will issue corresponding authorization certificates to the committee members. When the node joins the blockchain network, it actively presents the authorization certificate to the neighbor node, indicating the identity of its accounting node.

The members of the committee are elected by full nodes in the community. The community members who intend to become the full node initiate the application to the community. The community management committee checks whether the hardware resources and network conditions provided by the representative meet the basic requirements of the full node. The community issues a full-node authorization certificate to the node that meets the requirements.

Any community members can apply for SPV node. The community management committee will verify the application and issue the corresponding certification

11.6 Node Summary

Based on the introduction of the above three types of nodes, the following table summarizes the nodes as follows:

Node Type	Consensus Process	Ledger Booking	Synchronization	Transaction Verification	Transaction Query	Smart Contract
Full Node	✔	✔	✔	✔	✔	✔
Accounting Node	✔	✔	✔	✔	✔	✔
SPV Node	—	—	—	✔	—	—

Table 2 Node Type & Function Summary

12 Network Communication

12.1 Services Interface

There are two main types of HNB services: command line and API interface.

The command line interface is based on urfave. It can provide rich command parameters and provide enough default parameters. You can complete all operations by simply entering simple commands on the console interface. HNB also supports an interactive command line for a more friendly experience when entering commands.

HNB's API interface adopts gRPC and RESTful, which provide user friendly access experience:

gRPC is a high-performance, open source, and general-purpose RPC framework based

on the HTTP/2 standard. It provides features such as bidirectional flow, flow control, header compression, multiplexing over a single TCP connection, and request waiting. The efficiency has been greatly improved.

RESTful API also supports HTTP/2, which provides powerful routing and reflection capabilities. It optimizes the transmission of key-value pairs when processing URL GET parameters, thus reduces the transmission of network packets to existing blocks, and provides user-friendly access to blockchain systems

12.2 P2P Network

HNB P2P network adopts Kademlia algorithm, which is suitable for the networking of distributed massive node networks. Its node relationship management mechanism can effectively resist DDoS attacks.

The algorithm is based on the distance between two nodes, which is the exclusive OR of the ID numbers of the two network nodes, and the result of the calculation is finally returned as an integer value (eg 01000000). Then, the network is grouped according to the distance, and the grouping is based on the fact that all the previous digits are the same, starting from the nth digit of the countdown (for example, 00000000, 00000001 is a group, 00000000 and 00000010, which are different groups). Based on the grouping, the node can quickly find a node that it has not connected to, facilitating resource discovery and fast access and exit of the node.

The communication between the nodes uses the gRPC communication protocol.

12.3 Node Communication

Communication between nodes is through gRPC. The communication protocol between nodes uses the Protobuf code base. The Protobuf code base supports high performance transcoding and supports multiple development languages. There are four ways to exchange data between various nodes of HNB:

- One-way communication (full synchronization)
- Streaming server
- Streaming client
- Bidirectional flow (completely asynchronous)

HNB also supports encrypted transmission between nodes, and uses TLS protocol to ensure the security of communication between nodes. There are inadequacies in general TLS communication, such as: before TLS is started, it has been attacked, and TLS protection is invalid.

The HNB blockchain also provides a hardware-based solution that ensures end-to-end communication security between blockchain nodes.

13 Cryptographic Algorithm

13.1 Signature Algorithm

Signature algorithm is also referred as digital signature algorithm. Digital signature is a non-forgeable digital string generated only by sender through mathematic scheme on sender's digital messages or documents. It is also a valid proof of sender's authenticity and non-repudiation of messages. Digital signature is a digital string generated through one-way function to process the sending messages for authenticating the sender and validating any alters in the messages during transit.

HNB has taken the signature algorithm of Secp256k1 elliptic curve. Secp256k1 is the parameter in ECDSA(Elliptic Curve Digital Signature Algorithm) used by Bitcoin, defined in the standard of high efficient cryptography. Secp256k1 forms elliptic curve based on Fp over finite field. Given the peculiarity of its structure, the post-optimization of its implementation produces better performance than other curves by 30%. Main distinctive advantage points are

- Fewer bandwidth and storage resources needed, short private key.
- Same field operation by all users.

13.2 Digest Algorithm

The main characteristics of message digest algorithm are to encrypt the messages without keys and the encrypted messages can not be decrypted. The same encrypted message can only be obtained by applying the same algorithm to exact the same original messages. Key management and distribution are not needed therefore it fits for in use on decentralized network.

HNB supports Hash256, Hash512 various Hash algorithms. Users can select the optimal Hash algorithm in accordance to their business scenario.

13.3 Switch of Cryptographic Algorithm

With technological evolution, more and more cryptographic algorithms get decoded. Algorithms like MD5, SHA1, DES became obsolete over time. The emergence of the new technologies on high computing area such as quantum computing, are crushing those once invincible cryptographic algorithms. In order to safeguard from the attacks of quantum computing, HNB security layer supports switch among multiple algorithms such as ECDSA256、ECDSA384、ECDSA521. The longer length of the keys cryptographic algorithm deploys, the stronger defending capability it has and higher level of security it produces.

13.4 Zero-Knowledge Proof

Zero-knowledge proof was put forward by S.Goldwasser, S.Micali and C.Rackoff in early eighties of 20th century. Early zero-knowledge proof was required to have message interaction between prover and verifier to validate the proof which is called “interactive zero-knowledge proof”. In late eighties of 20^{th} century, the use of short random string proposed by Mr. Blum and others, was adopted to fulfill zero-knowledge proof. This approach is called “non-interactive zero-knowledge proof” which is only one time message sending from prover is required for verifier to validate the proof without two-ways of interactions between prover and verifier.

HNB blockchain provides zero-knowledge proof capability and protect data privacy for user and merchant to reduce the risk of merchant data privacy breach. HNB optimizes and upgrades zk-SNARK library for zero-knowledge proof and encapsulate it to highly reusable interfaces with rich functionalities. Zk-SNARK stands for zero-knowledge Succinct Non-interactive Arguments of Knowledge and is one of non-interactive zero-knowledge proof methods who embodies features of short proof message and fast validation.

Zero-Knowledge proof key process -- 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}

Zero-Knowledge proof key process – 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, τ)
b' := Verify(vk_POUR, x, π_POUR)
return  b ^ b'
```

13.5 Blind Signature

Conventional digital signature approach is the signer signs the clear text data. When the signer and data owner belong to different entities, the data privacy is not protected from the signer.

HNB implemented blind signature algorithm to realize the following privacy protectioin:

- **Blindness**: the content of a message to be signed is unkown to the signer.
- **Unlinkability**: The final signature S(m) cannot be linked to blinded message m' the signer keeps.
- **Unforgeability:** Anybody who doesn't know the private key of the signer is not able to effectively generate the digital signature that is certified by the digital signature validation formula. Only the signer itself can generate the effective digital signature in its name.

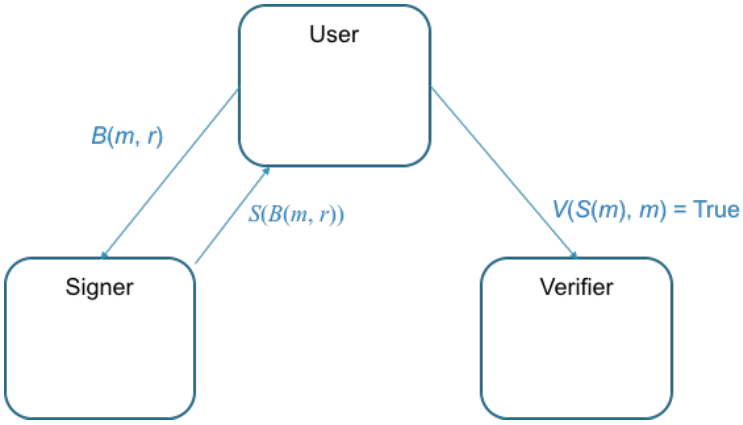


Figure 13 Blind Signature Process

Blind signature mathematic formula proof:

- Signature and message pair: ((S(m), m))
- Encryption process of the content of a blind-signed message：U(S(B(m,r)), r) = S(m)
- Verification process: V(S(m), m) = True

Blind signature process as following:

- The signer only knows the identity of signature requester without seeing the content

of a document when it is signed.

- The signer put the signature against document black box. The digital signature value can be printed on the document through carbon paper of the black box and become effective.
- When the signer links the content of a document and the respective signature, the signer cannot validate the signature belonged to self.

13.6 Ring Signature

Ring signature realizes unconditional anonymity of the signer and the identity of the signer cannot be traced. Ring signature is performed by the signer selecting a group of members including the potential signer and generating digital signature by the signer's private key and the public keys of the members. The group of members is called Ring and signature as result is Ring Signature. The recipient of the signature can prove the signer comes from a member of the Ring and does not know the identity of the signer.

The Theory of Ring Signature

- N members and each has a pair of public and private keys.
- Signer uses N public keys and his private key to sign the clear text of a message.
- Verifier uses N public keys to validate the digital signature. If private key of the signer belongs to one of the members, the validation succeeds. Or the validation fails.

The chance of hitting the identity of the signer is 1/N in the process of signature validation by verifier. The goal of hiding the identity of signer is attained.

Ring Signature Process

1. Randomly generate a v as initial parameter, $v \in (0,1)^b$， e.g: (011010…1100101).

2. Generate r number of $x_i \in (0,1)^b$ for r number of users. x_i of r-1 number of users are generated randomly. x_s is not randomly generated.

3. Use $y=g_i(x)$ to solve respective y_i. y_s is temporarily empty because x_s is still empty.

Given r number of users, RSA encryption algorithm is applied. Each user has a key pair （P_i,S_i）. RSA encryption algorithm has two functions:

$y=g_i(x)$——g_i function must use P_i
$x=g_i^{-1}(y)$——g_i^{-1}function must use S_i

4. To use $z=C_{k,v}=v$， quickly calculate y_s.

V is the initial parameter generated in step one, k is H(M)，E is symmetric encryption: Ck,v(y1, y2,…, yr) = Ek(yr ⊕ Ek(yr-1 ⊕ Ek(yr-2 ⊕ Ek(…⊕ Ek(y1 ⊕ v) …))))

5. Given known y_s, based on $x=g_i^{-1}(y)$，to solve x_s. This step is the core, as only s owns the private key s_s，only s can compute out x_s. The security of ring signature is founded on the security of asymmetric encryption algorithm.

6. A ring has been created. Only r number of users can create a complete ring. Attacker can calculate out y_s，but will never get to x_s.

13.7 Homomorphic Encryption

In many data exchanging scenarios, data provider only provides the right of use of data as opposed to data ownership transfer to the seller. To provide privacy protection of merchant data is the essential security requirement to which the homomorphic encryption is a sound solution. The merchant at first uses its public key to encrypt data m through homomorphic encryption algorithm. The encrypted data C is then provided to algorithm provider to conduct arithmetic computation, the result is sent back to the merchant. The merchant can use its private key to restore clear text result of f(m).

HNB supports homomorphic encryption on addition. Homomorphic encryption is also called privacy homomorphism. Homomorphic encryption is to allow computation on encrypted data as if it were on plain text data. Homomorphic encryption on multiplication supports multiplication operation on encrypted data. Homomorphic encryption on addition supports plus operation on encrypted data. The encrypted data by homomorphic algorithm can be performed addition or multiplication operation without decryption of the data. The computing results of encrypted data and clear text data are identical.

Assume clear text space is P，encrypted text space is C，homomorphic algorithm is Enc，decryption algorithm is Dec.To use * to present computation on clear text space, # to present computation on encrypted text space. For two clear text messages $\mathrm{m}_1, \mathrm{m}_2 \in \mathrm{P}$，two encrypted messages $\mathrm{c}_1, \mathrm{c}_2 \in \mathrm{C}$，$\mathrm{c}_1 = \mathrm{Enc}(\mathrm{m}_1)$，$\mathrm{c}_2 = \mathrm{Enc}(\mathrm{m}_2)$, homomorphic encryption can satisfy the formula：$\mathrm{c}_1 \# \mathrm{c}_{21} = \mathrm{Enc}(\mathrm{m}_1 * \mathrm{m}_2)$.

14 Application in Real Economy

14.1 DApp – Decentralized Application

14.1.1 DApp Architecture in HNB

HNB blockchain layer supports DApp (Decentralized Application) as one of the key features in HNB decentralized economic community.

DAapp backend codes run in HNB decentralized P2P network. In HNB blockchain layer, DApp backend codes are implemented by smart contracts. The frontend can be any user interface applications such as HTML5, mobile app, and is communicated with backend smart contracts via API to realize specific business functions.

HNB DApp can viewed as the combination of user API and smart contract.

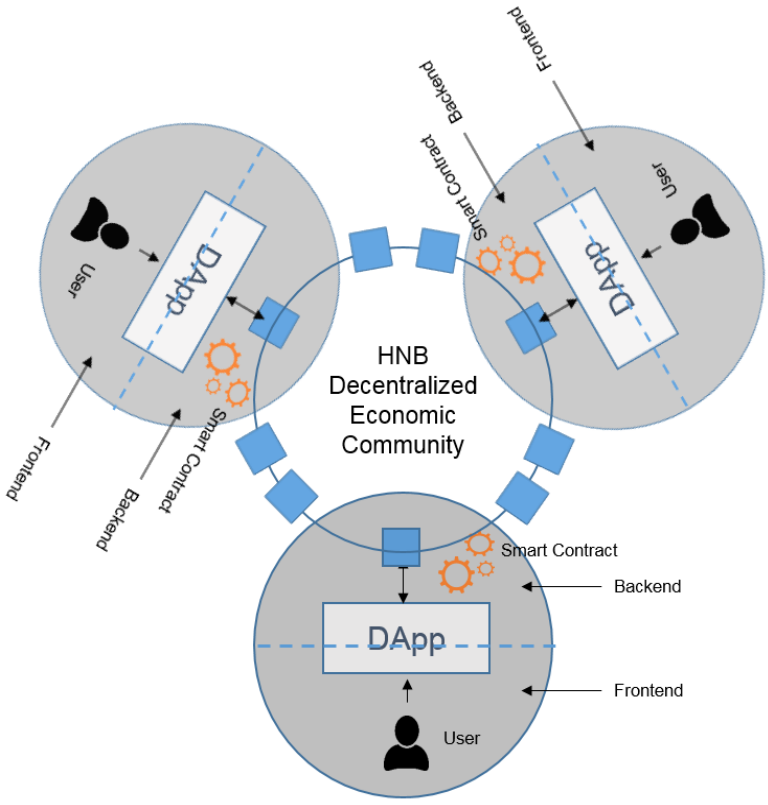


Figure14 HNB DApp Strcuture

Two methods can be taken for DApp integration:

1. **API/SDK Integration** —— Lightweight SDK integration allows participants of HNB ecosystem to quickly translate the existing business applications to DApp and release assets and payments that matter in HNB blockchain.

2. **HNB Frontend Integration** —— Integration into HNB ecosystem through HNB predefined protocols such as: gRPC, RESTful etc.

14.1.2 DApp Functionality

HNB DApp includes the follow feature:

- **Open Source:** Source codes make available to public for anyone and are governed by participants of other communities.
- **Decentralization:** Run on HNB blockchain network, collective participation, maintenance and governance.
- **Incentive:** HGS as incentive for node/user is helpful to DApp execution.
- **Consensus Algorithm:** Based on HNB blockchain consensus mechanism, DApp ensures single input and output in its logic execution and the consisten execution result across entire network**.**

14.1.3 DApp Architecture Description

To further the level of granularity, HNB DApp can be viewed as two layers architecture of frontend API and backend smart contract.

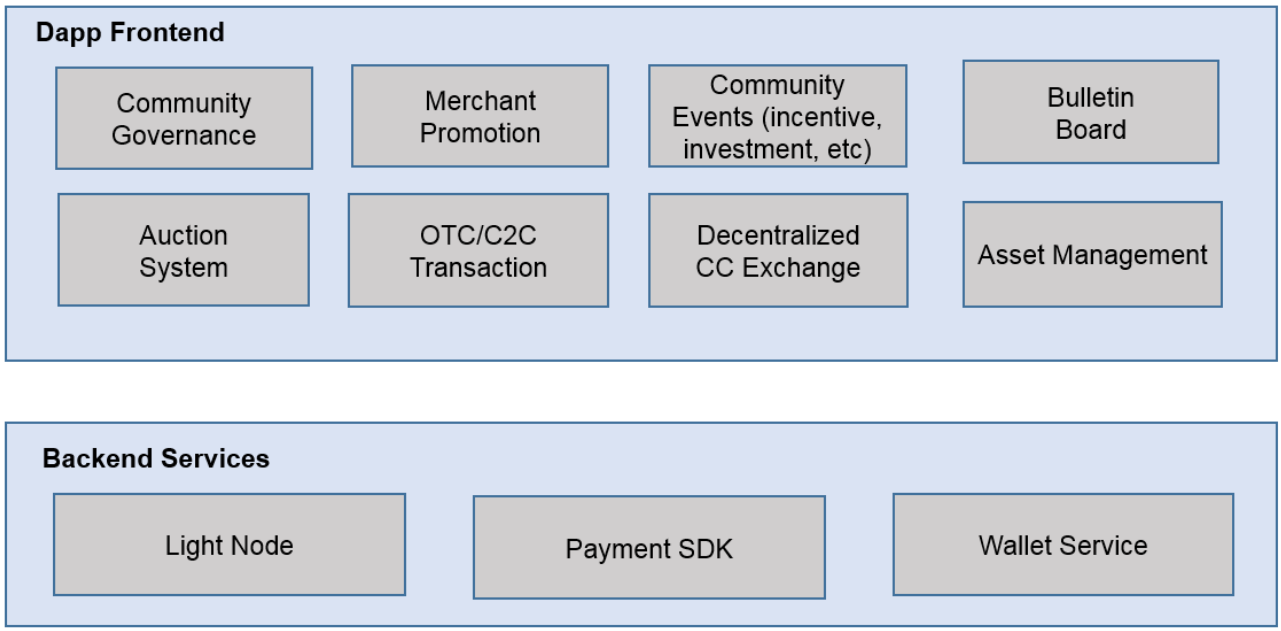


Figure 15 HNB DApp Architecture

14.1.3.1 DApp Frontend

HNB provides a series of organic and robust DApp in the community. DApp frontend services include, not limited to, community governance, auction system, merchant promotion,

OTC/C2C transaction, decentralized token trading, community events, bulletin board, asset management.

Above some functionalities described in user layer and business layer are fulfilled by the embedded business logics in DApps and backend smart contracts.

14.1.3.2 DApp Backend

DApp backend services include lightweight or SPV (Simplified Payment Verification) nodes, wallet service and payment SDK. The goal of SDK is to provide API interface for 3rd party application to facilitate easy integration or reorientation to HNB system in payment process.

14.2 Business Privacy Protection

HNB provides privacy protection solution to business data based on encryption algorithm and data segregation.

Firstly, any purchase orders and confidential information related to HNB merchants are encrypted by symmetric encryption algorithm to ensure data are stored and transmitted in non-clear text fashion. Key is managed by HNB community.

Secondly, core confidential user data is encrypted by asymmetric algorithm. Users encrypt the data with public key and store them in HNB community only visible to the owners. When a requestor needs to access the data, it must submit access application. Upon approval from the data owner, the owner will negotiate the conversational key with the requestor. After the negotiation is completed, the owner will decrypt the data with private key and send it to requestor protected by conversational key to ensure interactively oriented data transmission and no leakage of confidential data in the process.

Thirdly, due to general ledge is shared in blockchain, HNB segregates ledge data and business data. Ledge data is used for storing transaction information, mainly in digital asset transfer functions, with user identifier as HNB-ID excluding order and merchandise information such as ordering, merchandise items, identification, to ensure the privacy of business data is strongly protected.

14.3 Cross Chain Interaction

Ethereum founder Vitalik Buterin published a report for bank union on chain interoperability. There are three categories of strategies proposed in the report. They are Notary schemes, Sidechains/relays and Hash-locking.

In conjunction with HNB application scenario, HNB blockchain layer adopts notary schemes, sidechains/relays and extensible 3rd party protocol, three types of cross chain

interaction methods.

Notary Schemes: Support cross chain transactions between HNB dual token digital assets. Notary mechanism is leaning toward centralized process mechanism. The notary respectively detects cross chain transaction request of one asset type and submits respective process on another digital asset. HNB and HGS two digital assets are managed by HNB community and cross chain transactions are conducted by notary mechanism. HNB community is playing the role of notary and effectively managing exchange trading between two digital assets. Meanwhile, this approach can support high throughput of cross chain transaction.

Sidechains/Relays: To be used in cross chain interaction between HGS digital asset and side chain. HGS and sidechain HGS initiate cross chain interaction requests which are routed to counterparts through relay gateway in the form of events to achieve cross chain transaction integrity between two chains. HNB foundational platform provides relay gateway module for connecting HGS chain and side chain.

Extensible 3rd protocol: To be used in asset trading transactions between HNB digital asset and 3rd party blockchain assets. Future blockchain architecture is the chain network structure composed of multiple blockchains. Cross chain interaction is inevitable trend. Therefore, HNB blockchain layer reserves protocol interface for 3rd party cross chain to allow for future integration with other blockchains.

14.4 Off Chain Lighting Network

It has been a while that blockchain community sparked debates and experiments on the solutions to Blockchain scalability. Now the main solutions include block size expansion, consensus algorithm enhancement, security hardware assistance, segregated witness, lighting network, transaction sharding, state sharding, multiple subchains. But none can concurrently meet three critical requirements of decentralization, expandability and security. Blockchain is correlated to specific application thus the key is to strike the balance among many requirements according to the specific scenario.

The goal of HNB is to serve real economy. In massive scale DApp applications, small payment requests make up majority of transactions. Yet small payment does not need the timely confirmation from main chain. For example such as small payment scenario that is widely existing in real economy HNB, if massive volumes of high frequent transactions of small amount are taken to off chain lighting network for processing without real time interaction with main chain except when the transactions are closed or exited, they connect to main chain to record final state, this will greatly ease off processing pressure from main chain. This is the design philosophy of off chain micro payment for which the typical applications are Lighting Network under Bitcoin and Raiden Network under Ethereum.

14.5 Archive of Ledger

Data in business environment is required of continuous availability thus data storage for a period time is needed. For storage requirements of blockchain general ledger, the transaction history data will be less frequently visited with time moving on so it is costly to store them in actively ledger. Therefore, a solution to archive historic ledger is necessary to ensure the integrity of ledger yet reduce resource consumption by ledger management.

HNB deploys ledger archiving technology. It defines the ledger based on database as hot ledger HNB and the one based on other storage media as cold ledger. It archives the historic ledger and moves to cold ledger according to predefined rules. Latest transaction data remain on hot ledger.

Ledger archiving function is conducted by full node. When it completes archiving operation, the cold ledger is stored locally. If transaction history data are needed, the client node must access the full node.

For SPV and consensus nodes, majority of the transactions and validations are done through hot ledger so there is no need to store cold ledger. If there is a demand related to dormant accounts that are needed from cold ledger, it can be done by accessing full node to retrieve transaction history data

Ledger archiving high level operations:

- The rules in full node trigger archiving operation. The starting point of hot ledger is chosen. (All transaction data before the starting point need to be moved to cold ledger)
- All transaction data before the starting point are packaged in the format of filesystem and stored in cold ledger；
- The index is created for cold ledger for the sake of convenient search；
- The block size of hot ledger remains unchanged and a new block is added when existing one is full。

14.6 Parallel Accounting

HNB adopts multi-chain architecture. Each chain has independent network for its own consensus so to ensure no cross impact from consensus operation by each chain. The bursting concurrent peak transaction volumes in one chain will not affect others. Meanwhile, consensus parallel execution by multiple chains make sure high efficiency of consensus for entire network.

14.7 Algorithm Bank

HGS is a digital asset pegged to FIAT currency. HGS token supply is determined by the number of users/currency circulation/consumption scale/real world inflation. Algorithm bank is used to issue or reclaim HGS to ensure HGS value is pegged to FIAT currency and maintain

stable currency value.

• **HGS as stable currency**

HGS as stable currency is different from conventional currencies like USDT, TUSD backed by assets, accepted by a centralized platform which proved either high audit costs or non-transparent. HGS is unlike bitcny and DAI guaranteed by encrypted assets which proved very volatile as a result of constant black swan events. It is also unlike BASIS, CARBON totally dependent on algorithm and financial derivatives which proved difficult to avoid trust crisis that spawn downward spiral of value collapse.

• **HGS more meaningful and vigorous stable mechanism backed by real economy**

HNB basic regulation approach is based on currency quantitative theory and implanted by algorithm bank through machine learning. Based on many factors in HNB ecosystem such as the number of user/currency supply/currency circulation rate, model is created through enhanced learning. When abnormal result is detected, currency supply will be regulated by issuing or reclaiming.

In addition, HNB also designs a mechanism to stabilize HGS through backing the acceptance by community merchants. First of all, goods and services in HNB community are leveraged to support HGS value. Due to HNB is closed tied to real economy, merchants in HNB community could be selling popular products. Let us say, a merchant sets the price of $10USD for its rice product in the store and tags 100HGS in HNB community. If for some reasons HGS drops to $0.095 USD, the rice product becomes cheaper in HNB ecosystem so more HGS will be bought to purchase it. The increasing buy demands in HGS will push its price to go up and get closer to $0.1 USD. So HGS is actually backed by all the merchandises in HNB ecosystem. Another mechanism is auction. DAO will periodically auction HNB tokens and users can only buy HNB by HGS. Theoretically the auction price of HNB token is lower than secondary market so HGS is more valuable than FIAT. As a result, HGS value is naturally pegged to FIAT currency.

15 Reference

[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.