REAP CHAIN Co., Ltd.

REAP CHAIN Yellow Paper

Ver 0.7

REAP CHAIN

Table of Contents

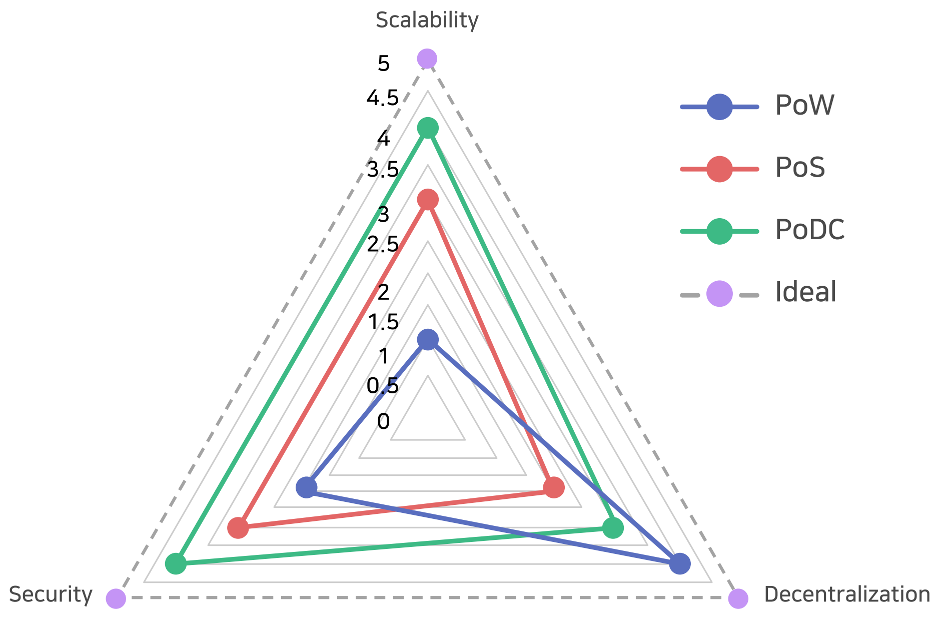
Fundamental Design Objectives of REAP CHAIN

REAP CHAIN proposes improved ways for the blockchain ecosystem with objectives shown in the following section to innovatively solve various problems associated with existing blockchain platforms.

REAP CHAIN intends to improve speed, stability, and security, all at the same time by employing consensus processes that consist of nodes selected by using quantum random numbers from authorized as well as general nodes to overcome disadvantages of previously proposed consensus algorithms in various blockchain platforms.

Solutions for Trilemma

The three dilemmas of blockchain namely Blockchain Trilemma are scalability, security, and decentralization. Now, The Blockchain Trilemma says that these three issues cannot be resolved at the same time. In other words, an attempt to resolve one issue leads to the problem in which others are compromised. For example, PoS or DPoS employees proof of stake mechanism to improve scalability and security, are considered as the weaknesses of PoW, and because of this issue of centralization arises.



Comparison of Blockchain Trilemma Solutions

Scalability Aspect:
In general, as the number of users increases, the number of transactions increases as well and it results in problems associated with handling increased traffic. REAP CHAIN maintains consistent internal configuration throughout the entire stages except in the Pre-prepare stage during which the proposal blocks are propagated. Therefore, it does not experience such problem as increased traffic even when the network is expanded. Because only certain number of nodes are allowed to participate in the consensus process even when the number of nodes in network increases, the processing speed can be guaranteed even after the network is expanded.

Decentralization Aspect:
In the REAP CHAIN, all communication processes of a blockchain network are voluntarily and autonomously connected. The interconnections are unconventionally established by selecting the coordinator and the random candidate groups through a dynamic selection method when each block is generated. Also, primary entities participating in the consensus processes became different. REAP CHAIN has no central or external involvement because the designation and the finalization of the candidate groups are performed separately by an algorithm in both autonomous and voluntary ways throughout the block generation as well as confirmation process without any external intervention.

Security Aspect:
In REAP CHAIN, access is controlled through public-key encryption so unauthorized users cannot access any data or programs within the blockchain. And the blocks are being generated to maintain confidentiality and thus finalized by limiting the access to those who have been selected through quantum random numbers to maintain the integrity of the consensus processes. It also provides availability to ensure that the service is promptly accessed only by the participants of block generation by prohibiting information from being freely accessed on the Network.

In particular, computing power attacks generally executes by increasing the number of nodes that can be maliciously controlled by taking over the control of nodes. In the case of REAP CHAIN, At the time of block generation nodes are equally distributed for block finalization to candidate groups, and its stability increases as the number of nodes increases. REAP CHAIN is also more secure and safe as the consensus derived by combining with the decision from a consensus group that consists of reliable and continuously monitored nodes. For computing power attacks, if nodes have selected as candidate groups, the availability of such nodes will always be ensured because the decision of the final voting group is made separately by using unpredictable quantum random numbers. Different member nodes have given different authorities, which in turn can use to register in the governance and to climb up the status hierarchy to secure reputation and authority.

To participate in the consensus process, one needs to register as a candidate node and is required to pay a registration fee and have to place different levels of identification regulation on standing and steering committee candidates for assigning duties, responsibilities, and authority. Before participating, Identity verification is done by the Qmanager and coordinator levels depending on each role during consensus.

Therefore, in the REAP CHAIN, a highly secured balance can be achieved between external and internal nodes. Such nodes shall get select without interference through an algorithm on which security-enhanced decentralized quantum random numbers have applied. The consensus derives through direct voting with a minimum number of messages delivered to a coordinator who has been selected by a highly scalable quantum random number. All the paid transaction fees are collected and distributed legally to all the participants who hold coins.

Introduction of Quantum Random Number Selection Mechanism

In general, there are several ways to generate random numbers. One way to generate pseudorandom numbers is by using some mathematical functions in computer software. Once generated, they have been further use for general industrial sectors such as the financial, aviation, and security industries. A subsequent random number, however, could be statistically predicted if pseudorandom numbers are repeatedly used. In some consensus algorithms, nodes have authorities to participate in voting by using random numbers that are mathematically validated. But, such validation process requires much time. Besides, this consensus algorithm requires more than 1,000 consensus nodes, which means that this network would not be able to work unless it expanded at the very initial stage.

Another approach that has been attempted is to generate random numbers in hardware. When utilizing properties based on natural phenomena, for example, it is said that elemental motion of light protons or isotopes are completely random and therefore unpredictable. It is also said that such method of generating random numbers by digitally quantifying such analog movement is the most unpredictable method. Unlike other consensus algorithms that use pseudorandom numbers, REAP CHAIN introduces a reliable quantum random number generator. Random numbers obtained by using this are encrypted with public keys and used with robust security.

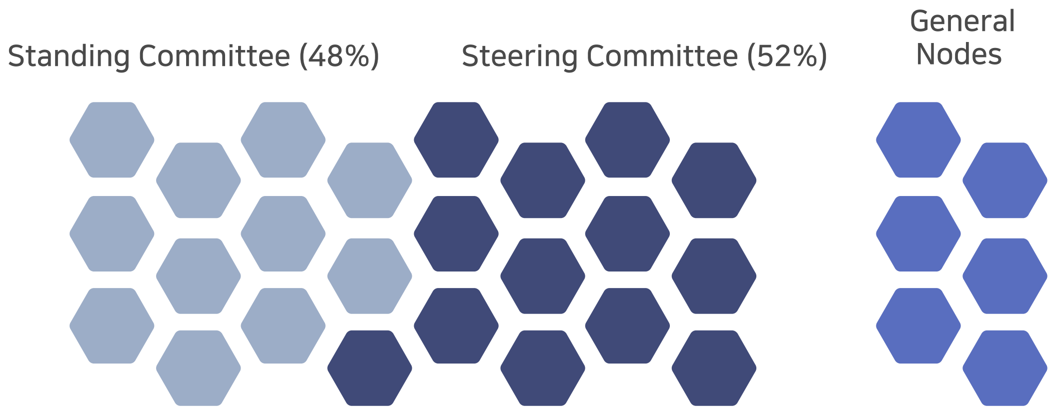
Introduction of PoDC (Proof of Double Committee) Consensus Algorithm

REAP CHAIN operates standing committee nodes and steering committee nodes to balance between the internal and external nodes and forms a double committee to conduct the consensus process. The standing committee nodes consists of an internal group separately constructed to maintain the REAP CHAIN blockchain network operating at all times. The standing committee nodes should possess 2% of the total number of REAP coins issued. In contrast, the steering committee nodes which consist of an external group authorized through the registration process of governance while satisfying a condition of holding 100,000 REAP coins among other general nodes. Anyone who satisfies these criteria can register as a steering committee node. Such registered groups are called steering committee candidates in the actual consensus process.

PoDC aims for decentralization in constructing the standing and steering committees. The standing committee nodes initially operates in the governance community. It can be further selected by those candidates that meet the particular criteria of standing committee amongst the steering committee nodes. And if any of the existing standing committee nodes are suspended or fail to meet the criteria, then those nodes must go through a separate voting process of the governance to be promoted from steering to standing committee node status.

REAP CHAIN consists of 14 standing committee nodes that operated at all times as well as 15 steering committee nodes that had selected from general nodes through unpredictable quantum random numbers comprises of robust security. These 29 committees participate in the consensus process for generating blocks. In order to ensure whether the standing committee that operated at all times is still working legally, the steering committee must account for at least 52% of all nodes that participate in the consensus process.

<Figure – Construction of PoDC Doble Committee Nodes>



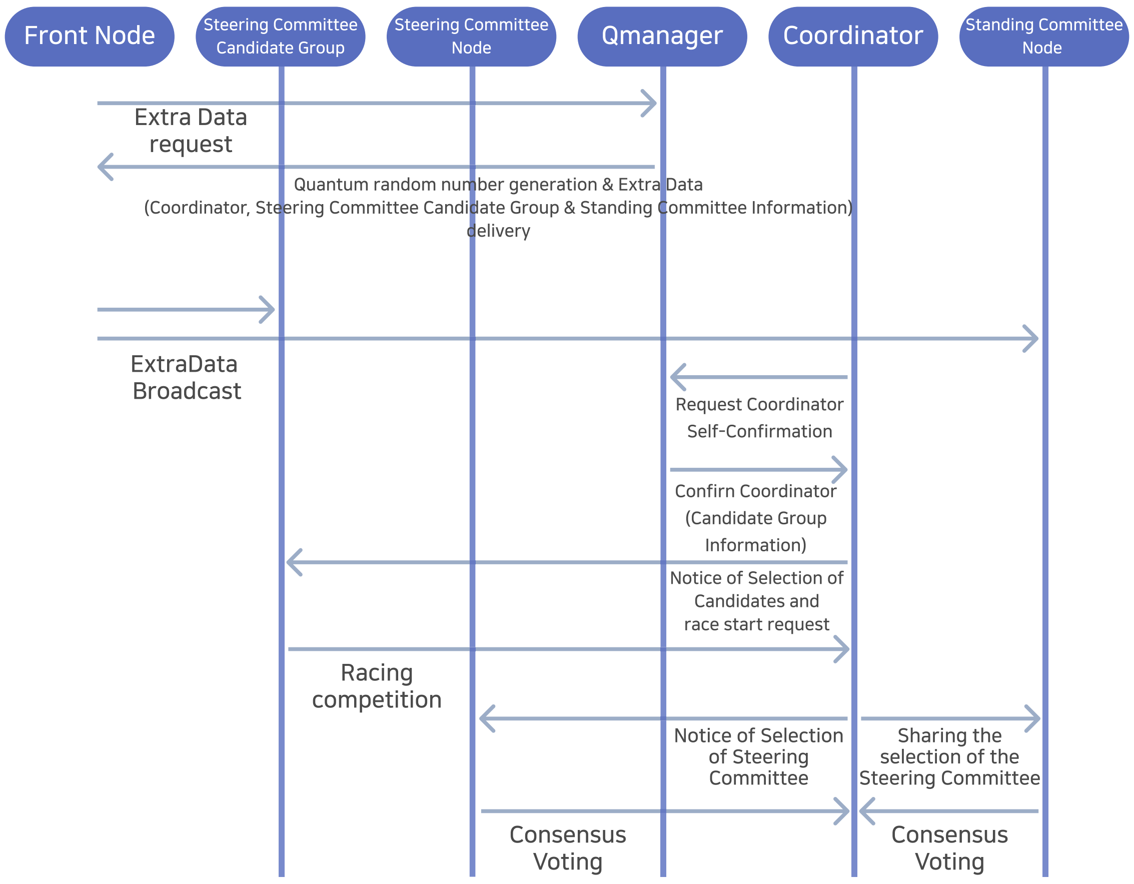
▣ Consensus Process and Block Generation

REAP CHAIN has a basic block generation cycle of 2 seconds. But in the REAP Middle Chain, that corresponds to the temporary ledger; approvals are notified in less than 1 second to create the same effect as if consumers are receiving credit card payment through authorized notifications. In reality, block generates within 2 seconds in REAP CHAIN, so the consumers rarely notice any payment approval delays, and It thus makes the REAP CHAIN suitable for commercial transaction deployment. The REAP CHAIN can quickly reach a consensus within 2 seconds by selecting a coordinator from a standing committee that operates at all times. It is possible because it reaches the consensus with the minimum amount of network traffic under direct communication among 28 nodes surrounding the selected coordinator during the process. Mainly, time delays for a while and inevitably network loads occur in collecting validation messages that broadcasts in P2P networks; A coordinator exchanges message only once in the internal network organization. Such a coordinator ultimately becomes a single block producer to generate a block. If a block does not generate within 2 seconds, the block generation of the corresponding cycle skips.

In the REAP CHAIN, It may require more time to reach the finality of every block, even though the block generates within brief seconds. Therefore, considerations can make to finalize the transactions after accumulating a certain number of blocks by installing checkpoints at every 20 blocks like in Ethereum Casper protocol.

REAP CHAIN creates one round of tables for block generation. In one round, 15 steering committee nodes have selected among the general nodes through security enhanced random selection mechanism, and 14 standing committee nodes participated together in the validation process. The reliability of the consensus that is achieved within the REAP CHAIN's QSN (Quantum Safety Net) have insured here.

Block Proposal	A random front node that has Tx pool that stores continuously incoming transaction Tx receives the last Tx to be included in a new block to generate a block Bx and this is when it makes a request for generation of a quantum random number by connecting to the quantum random number managing Qmanager
↓	
Qmanager Notification	Qmanager receives requests from front node, encrypts the coordinator and steering committee candidates for block generation and consensus, and quantum random information for recognition of such, and sends them to the front node. Steering committee candidate information is sent to selected coordinator
↓	
Platform-wide Notification of Front Nodes	Front node configures extra data in block header with encrypted information received from the Qmanager and broadcasts it to all nodes
↓	
Coordinator and Candidate Confirmation	Each node verifies its own hash from the extra data of the received block and decrypts only if they are their own information to confirm the coordinator and candidate group
↓	
Steering Committee Registration Racing	Nodes that confirmed as the steering committee candidates participate in first come first served based racing for registration in the steering committee
↓	
Steering Committee Finalization	The coordinator finalizes the steering committee with 15 nodes that register first and notifies the result to the standing and steering committees
↓	
Consensus Process	Standing committee and steering committee conducts consensus
↓	
Block Generation	Coordinator generates blocks according to the consensus


Front Node
Steering Committee Candidate Group
Steering Committee Node
Qmanager
Coordinator
Standing Committee Node
Extra Data request
Quantum random number generation & Extra Data (Coordinator, Steering Committee Candidate Group & Standing Committee Information) delivery
ExtraData Broadcast
Request Coordinator Self-Confirmation
Confirn Coordinator (Candidate Group Information)
Notice of Selection of Candidates and race start request
Racing competition
Notice of Selection of Steering Committee
Sharing the selection of the Steering Committee
Consensus Voting
Consensus Voting

In the REAP CHAIN, the Consensus process of generating blocks always establishes in-network, even if fewer nodes are participating due to some failure process. In particular, the reliable standing committee nodes that are constantly operated by the governance continuously perform a health check, and the coordinator selected from the standing committee nodes leads the consensus process. Considerations have made for a mechanism in which 2f+1 coordinators are selected just in case that there can be a chance of momentary failure even after the coordinator is selected.

▣ Selection of Coordinator and Executive Committee

The REAP CHAIN uses unpredictable, unbiased, and numerically unrelated quantum random numbers to fairly determine the standing committee coordinator and the steering committee candidates amongst the general nodes. Qmanager uses quantum random numbers to provide a candidate group for determining the steering committee candidates, by unknowing which nodes become the actual candidates. The steering committee is determined every moment on the network through the racing of candidates to ensure its integrity

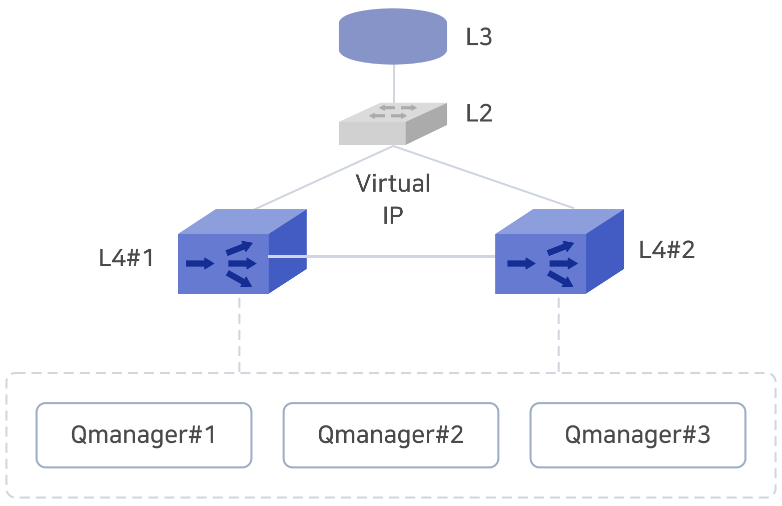
Quantum random numbers encrypted by using the public key of each node in the steering committee candidate group selected through the quantum random number and distributed on the blockchain network, but only the nodes selected as the steering committee candidate group can decrypt and verify the quantum random numbers by using the private key.

The decrypted contents include the public key and information of the coordinator selected by using the quantum random number amongst the standing nodes. The steering committee candidates use this information and start racing to the coordinator. The coordinator compares the quantum random numbers for each node of the steering committee candidate group, previously received from Qmanager, to check the integrity of the selected candidate group.

▣ Stability of Selection Mechanism by Qmanager

REAP CHAIN introduces Qmanager that generates more secure quantum random numbers than pseudorandom numbers to implement strong security-enhanced random selection. By using hardware generating quantum random numbers, Qmanager is basically safer than a system that uses software consists of algorithms to generate pseudorandom numbers. Qmanager mainly creates and manages quantum random numbers, and also manages node information provided by the governance while selecting candidates for the steering committee and the coordinator.

Qmanager operates with up to 2f + 1 systems by applying redundancy structure for protection against DOS attacks and more stable operation. When Qmanager needs af = 1 node, 3 units are applied by applying 2f + 1 to complete the dual backup structure.



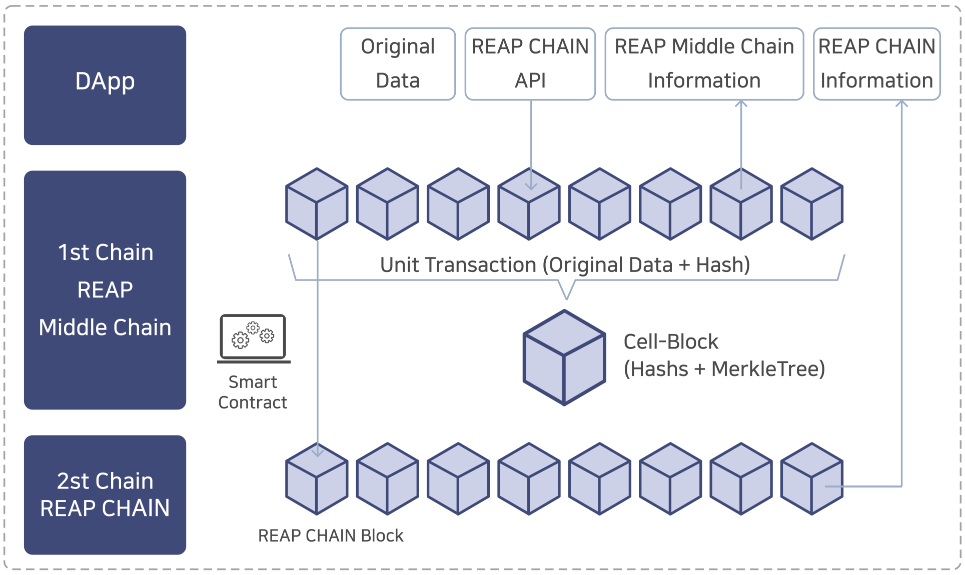
To judge if the information regarding coordinator and the steering committee candidates, appropriately selected by the Qmanager, whether or not Qmanager's signature that encrypts in a block's extra header or the candidate groups that are to participate in the vote transmits information to the Qmanager for additional verification after the coordinator is selected. Therefore, it confirms that the current coordinator may or may not get selected in the current round by the right Qmanager. And the information regarding the steering committee candidate group received from the Qmanager and the racing signals gets transmitted to the steering committee candidates. Here, each node verifies the information through the coordinator's public key in their information within the previously received block's extra header. Besides, the steering committee candidates that have completed the racing are required to get verified by transmitting information to the coordinator.

The node's information must reach out together for the coordinator and the Qmanager, which includes Omanager's signature, and gets verified to check the possibility of tampering before the subsequent processes. It thus ensures that the Qmanager's information reaches all nodes without being compromised. The subsequent processes do not take place until and unless the step gets completed, and the Qmanager's signature verified without being exposed externally. So another Qmanager cannot be generated. Also, random selections are signed off at the time of consensus to save the block, so whether or not, everyone can see and verify all the signed off authenticated blocks. The governance manages Qmanager's stability by conducting health check-ups so that whether or not the selection of the coordinator and steering committee nodes by Qmanager occurs in the same pattern with open block records. The Qmanager requires any supplementation or upgrade is conducted by obtaining approval of the governance by following voting procedures.

Therefore, REAP CHAIN selects a coordinator by using an unpredictable way using quantum random numbers and a steering committee group by internal competition system. Such as racing from an unspecified number of candidates to construct a new voting group every time a new block generates. The election can be justified through such mechanisms because it provides a popular method in which the randomness is maximized at the time of block confirmation even when the candidates only consist of small numbers of standing committee nodes and steering committee nodes. It is proven that EOS to be a successful case with a small number of voting nodes that is 21, but it tends to become more centralized by depending on the private community to construct the nodes.

Double Chain Structure and Middleware

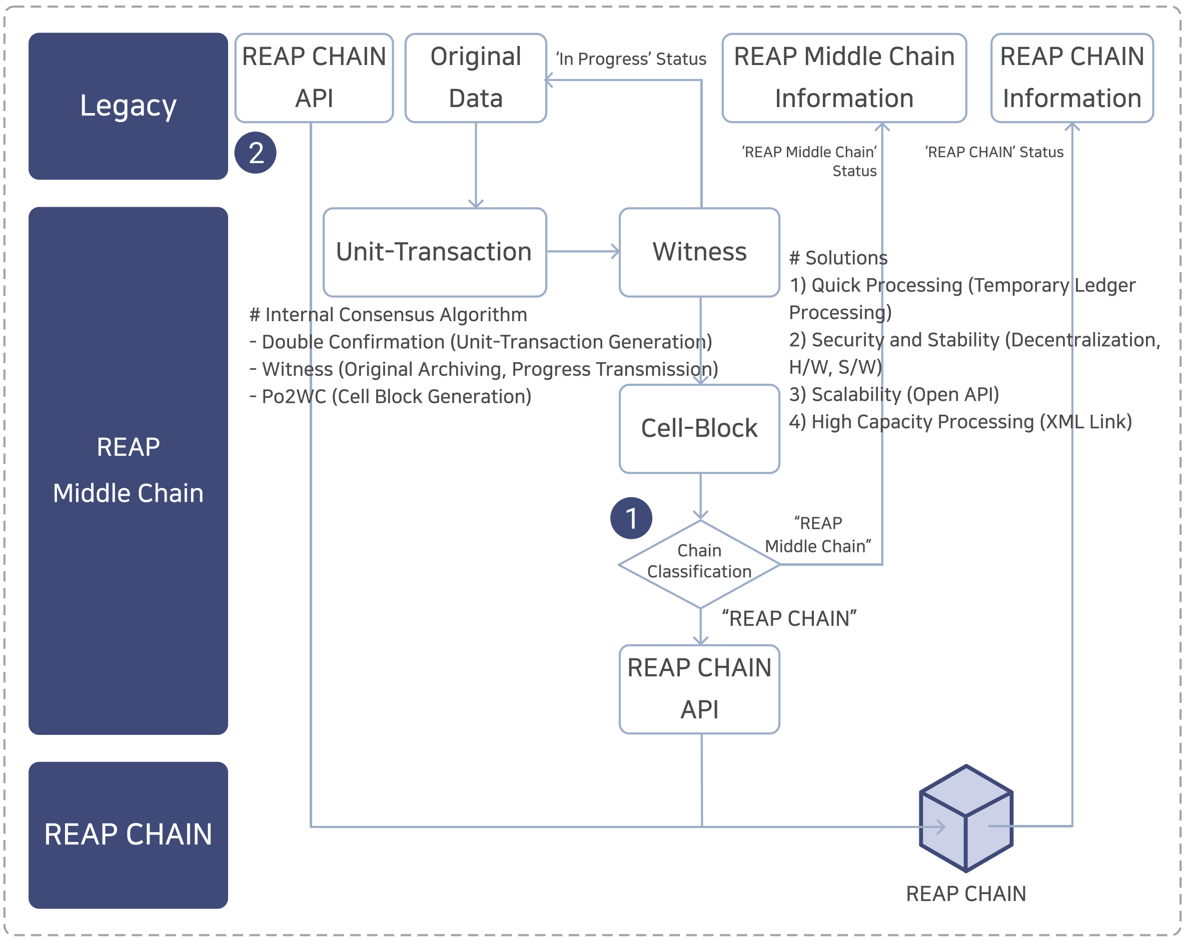
In the REAP CHAIN, the concept of a temporary and permanent ledger that uses in the financial system has technically implemented. The data has stored in the REAP Middle Chain to use as the temporary ledger first, and then the block data has recorded in REAP CHAIN is managed as the permanent ledger.


DApp
1st Chain REAP Middle Chain
2st Chain REAP CHAIN
Original Data
REAP CHAIN API
REAP Middle Chain Information
REAP CHAIN Information
Unit Transaction (Original Data + Hash)
Smart Contract
Cell-Block (Hashs + MerkleTree)
REAP CHAIN Block

When a transaction takes place, REAP Middle Chain records the raw data to manage the original copy. In the security sector of financial industries, the importance of the original copies emphasizes more as it manages from a point where an order is made for correction, cancellation, and completion. At the same time, transactions have managed in a form that can be inserted into blocks to implement the blockchain processing and, the data, hash values of the transaction, as well as the Merkle trees, are also managed.

Transactions have recorded by transforming into blockchains in the form of mini blocks called Cell Blocks, and then it transferred to the REAP CHAIN platform through APIs with end-to-end security. The final REAP CHAIN blockchains have completed after the confirmation that there has been no tampering by referencing the transaction data, hash value, and Merkle tree.

The completion of Double chains has been done by managing mini blockchains at the middleware stage by saving blockchains in REAP CHAIN with more certainty and managing data conformity to check for the existence of tampering between the double chains. The blocks that have recorded in the REAP Middle Chain have no fees, but the blocks that have recorded in the REAP CHAIN blockchains have fees. Overall, REAP CHAIN manages the raw data, transaction Cell Blocks, and the entire REAP CHAIN blockchains through a double-chain structure. All of these data are written into secure hardware storage, making it highly secure.

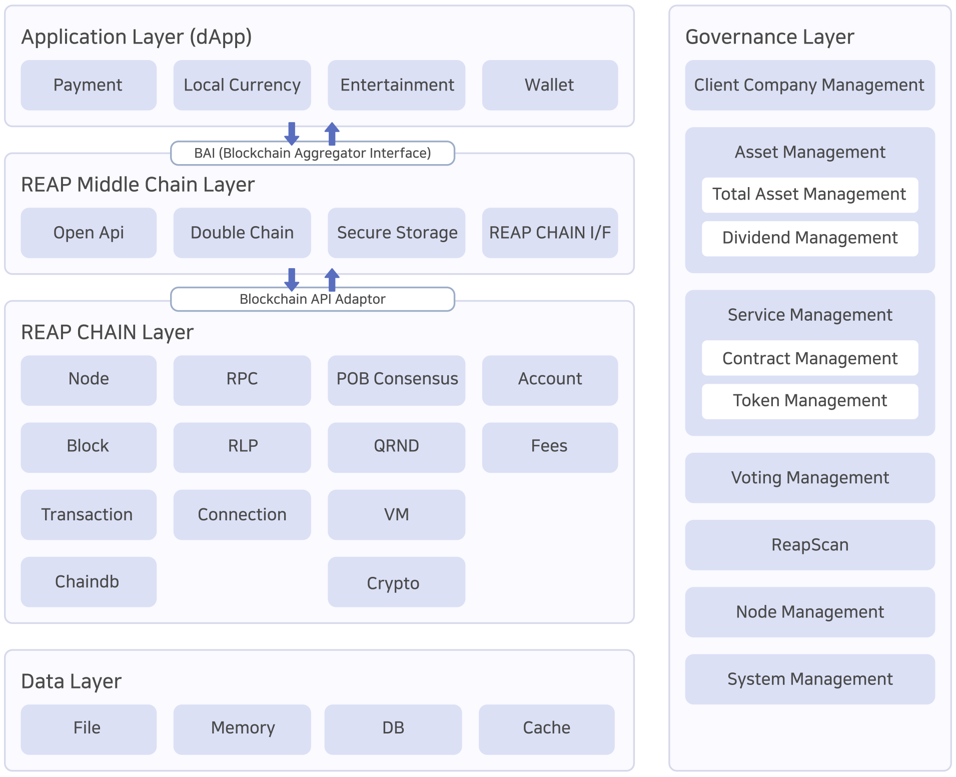

Legacy
REAP CHAIN API
Original Data
'In Progress' Status
REAP Middle Chain Information
REAP CHAIN Information
2
'REAP Middle Chain' Status
'REAP CHAIN' Status
REAP Middle Chain
Unit-Transaction
Witness
# Solutions
1) Quick Processing (Temporary Ledger Processing)
2) Security and Stability (Decentralization, H/W, S/W)
3) Scalability (Open API)
4) High Capacity Processing (XML Link)
# Internal Consensus Algorithm
- Double Confirmation (Unit-Transaction Generation)
- Witness (Original Archiving, Progress Transmission)
- Po2WC (Cell Block Generation)
Cell-Block
1
Chain Classification
"REAP Middle Chain"
"REAP CHAIN"
REAP CHAIN API
REAP CHAIN
REAP CHAIN

REAP CHAIN has a middleware that handles lightweight accelerated Cell Blocks between the legacy of DApp that provides business services and the mainnet that records the blocks, where DApp independently maintains the service to minimize the load of the mainnet.

The biggest advantage of a hierarchical architecture is that it is easy to make improvements on functionalities or performances of a particular tier without modifying the entire system.

REAP CHAIN provides the following hierarchical architecture for effective interworking with various applications.

<Figure – Hierarchical Architecture of REAP CHAIN>



● DApp Layer: This is a layer in which various kinds of business applications have provided that utilize REAP CHAIN.

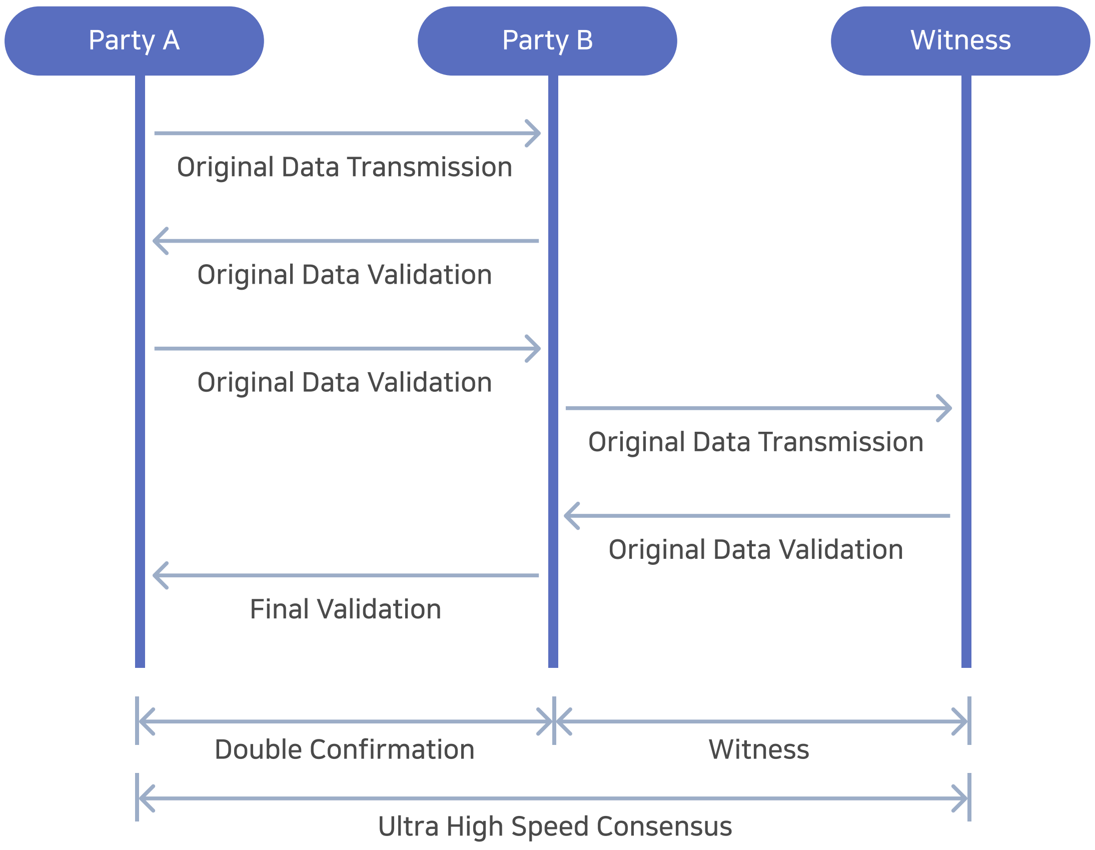
● REAP Middle Chain Layer: This is a layer which is supported by the multilayer transaction architecture, and the transaction processes into Cell Blocks in ultra-high-speed. It enables the middle layer to process transactions efficiently and quickly, by providing the upper layer with a front end interface that is conveniently interconnected with the existing systems and provides the bottom layer with an efficient interface by interworking with the mainnet.

- BAI (Blockchain Aggregator Interface): It provides an environment that allows applications to work seamlessly and thus effortlessly interwork with the REAP CHAIN.
- Blockchain API Adaptor: It enables interworking with REAP CHAIN core functions.

● REAP CHAIN MainNet Layer

- Core Layer: It consists of a few core functions of REAP CHAIN as consensus processes, block generation, ledger management, and token management.
- Governance: This is a particular layer that provides management functions, such as fees allocation policy, random user node selection policy, and all kinds of node management policies required for the maintenance of the REAP CHAIN.

Ultra High Speed Consensus in REAP Middleware Chain



When a transaction takes place, both parties first go through a double confirmation process in which the correctness of a deal has checked twice, followed by a third party consent process that involves a notarization as a witness. As the key participants, parties of the transaction and a notary public ensure that consent has established quickly. Since the deal conforms to the international standard XML structure, the grammar of the protocol, as well as the validation of contents, especially the numerical conformity, are checked through XML parsing processes. Upon verification of contents and the hash value of the transaction, consents between REAP Middle Chain nodes take place to generate a temporary ledger. Like this, continuous operations have made in Cell Blocks. Such transactions and blocks are stored in security-enhanced storage to ensure proper security.

Consensus Model

REAP CHAIN's consensus algorithm quickly generates blocks by conducting votes in the standing and steering committee that use quantum random numbers (QRN).

- In REAP CHAIN, 48% of the operating standing committee nodes and 52% of the general nodes have selected, and each of the nodes acts as a subject and verifier of the block generation that participates in the consensus process.

- The nodes that participate in the consensus; the percentage of the steering committee nodes is maintained at 52% or higher to monitor whether or not the continually operating standing committee nodes have produced relatively.

- As for the selection of the consensus participant, quantum random numbers are used to relatively select and distributed to the validator nodes, and then quantum random numbers and public keys are used again to assign encrypted random number tickets. A library of hardware-dependent call functions in which an unhackable message delivery system and fast authentication and consensus are implemented and named as Quantum Random Number Management Server (Qmanager).

- Because of the hardware-dependent characteristic of quantum random number generation, Qmanager can either exist separately from the nodes.

- The quantum random function mechanism has adopted for the role of Qmanager and the differentiating features of the REAP CHAIN contained in the QRF mechanism.

QRF Mechanism for Consensus

● QRF Mechanism: Consensus is established in the REAP CHAIN by utilizing quantum random numbers, public-key encryption, and public key hash for fast recognition. A series of authentication and information transfer systems has named Quantum Random Function Mechanism.

● QRF provides a functional element by using a quantum random number server (Qmanager). Qmanager holds network information and public keys of all nodes that participate in the blockchain network. The network address of Qmanager is open to all nodes. In contrast, nodes participating in the chain generate public and private keys of applicable nodes by using public key encryption specified by REAP CHAIN, and then retain the private keys and register the public keys in Qmanager. The Qmanager has an encrypted node master table in the form of an in-memory DB and stores node ID, node network information, public key, and an assigned quantum random number.

<node master table>

NODETYPE	TRNO	NODEID	NETINFO	PUBKEY	QRND
Standing Committee, General Classification	Applicable Transaction	ID	Network Address	Public Key	Assigned Random Number

QRF Mechanism Function Configuration

(1) QRF_REGISTER(NODEID, PUBKEY): A node that receives an entry node or a public key update request accesses Qmanager to register the public key.

Input: Node ID, Node Public Key

Output: Resulting Integer Value

(2) QRF_FINDKEY(NODEID): Uses node ID to obtain the public key.

Input: Node ID

Output: Node Public Key

(3) QRF_GENQ(NODEID): Assigns a quantum random number to a node.

Input: Node ID

Output: Quantum Random Number

(4) QRF_SLIST(): Returns an array of node identification information nni containing ID, the public key, and network information of all standing committees currently active in the blockchain network.

Input: None

Output: IDs of all active standing committee, network information

(5) QRF_QRLIST(): Returns the node identification information nni of the steering committee candidates corresponding to the number of inputs by using a probabilistic selection rule that uses quantum random numbers from all general nodes currently active in the blockchain network.

Input: Seed, Number

Output: Ids of all active steering committee

(6) **QRF_RID** $(PK^i,\ PK^c)$: It Means a mechanism in which a random node is assigned a quantum random number through a public key and recognizes the public key and network information of a specific node and message generation for such a mechanism. The message is named as node I identifier and has expressed in a structure shown below.

Input: PK^i : Public Key of Node i , PK^c : Public Key of Node c

Output: $\left[Q^i_{pk^i} \mid\ N^c_{pk^i} \mid h_{pk^i}\right]$ ~ [Node i Identifier]

$Q^i_{pk^i}$: Value in which the quantum random number of node i is encrypted as the public key of node i .

$N^c_{pk^i}$: Public key of node c, Value in which the network information is encrypted as the public key of node i.

h_{pk^i} : Public key hash value of node i.

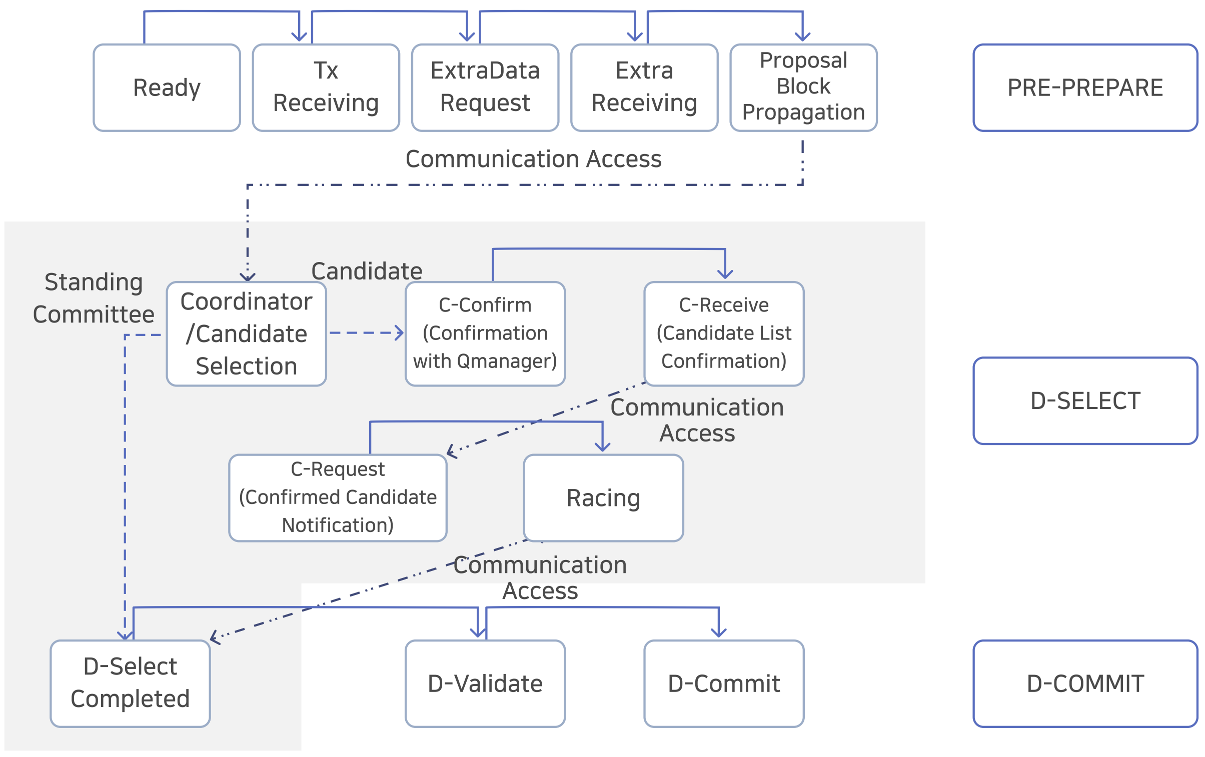
The fact that it has been assigned to oneself can be immediately confirmed first by comparing the hash value through h_{pk^i}. Then, a quantum random number Q^i allocated through decryption using a private key and the network information (node-net-info, nni) of the coordinator node are obtained.

(7) **QRF_CID** $(Q^1, \cdots, Q^n,\ PK^c)$: A function used by node c to calculate coalescence between Q_(pk^(c))^1,⋯,Q_(pk^(c))^n, a value obtained by encrypting Q^1,⋯,Q^n, a quantum random number value assigned to n number of node 1,2,⋯n, with public key of node c and h_(pk^c), public key hash value of node c. This is a function used by node c to find coalesce between $Q^1_{pk^c}, \cdots, Q^n_{pk^c}$, a value obtained by encrypting $Q^1, \cdots, Q^n$, quantum random number values assigned to n number of nodes 1,2,⋯ n as public key of node c. This means a structure in which the coordinator node recognizes quantum random number of each steering committee candidate group.

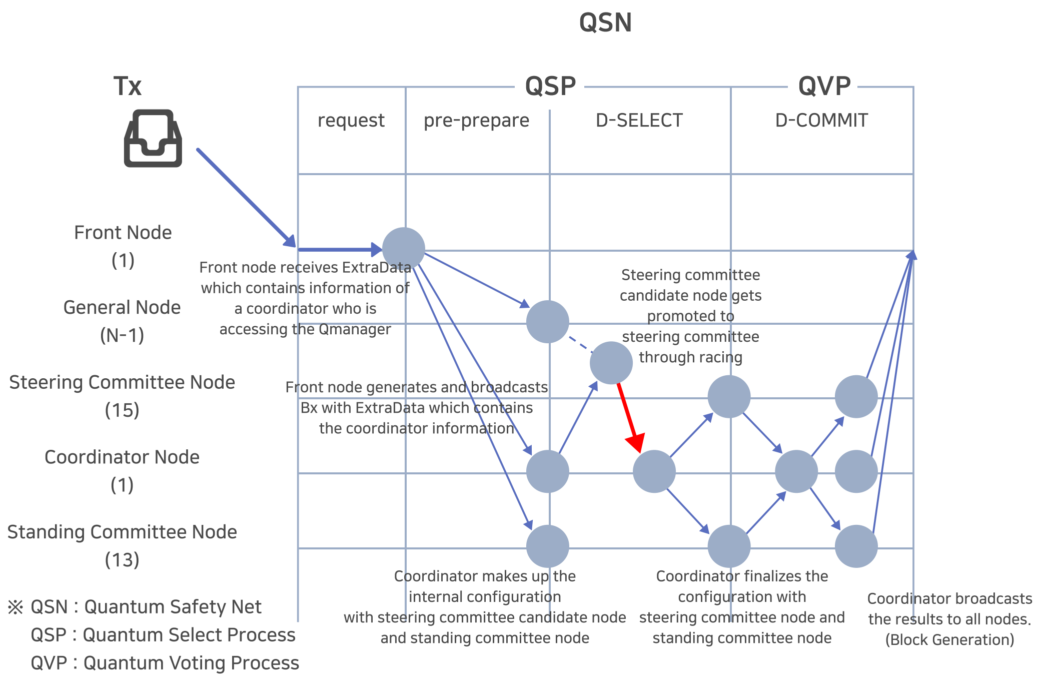
Input: Quantum random number value $Q^1, \cdots, Q^n$ that has been assigned to node1,2,⋯ n ,

Public key of $c\ PK^c$

Output: $[Q^1_{pk_c}|Q^2_{pk_c}| \cdots |Q^n_{pk_c}|h_{pk_c}]$~ [Coordinator Identifier] Consists of a list of quantum random numbers in which he quantum random number of each steering committee candidate conjugates a value that is encrypted as the coordinator's public key and the public key hash value of the selected coordinator.


Ready
Tx Receiving
ExtraData Request
Extra Receiving
Proposal Block Propagation
PRE-PREPARE
Communication Access
Standing Committee
Coordinator /Candidate Selection
Candidate
C-Confirm (Confirmation with Qmanager)
C-Receive (Candidate List Confirmation)
Communication Access
D-SELECT
C-Request (Confirmed Candidate Notification)
Racing
Communication Access
D-Select Completed
D-Validate
D-Commit
D-COMMIT

<Figure - Consenssus Process Over Time>


QSN
Tx
QSP
QVP
request
pre-prepare
D-SELECT
D-COMMIT
Front Node (1)
General Node (N-1)
Steering Committee Node (15)
Coordinator Node (1)
Standing Committee Node (13)
Front node receives ExtraData which contains information of a coordinator who is accessing the Qmanager
Steering committee candidate node gets promoted to steering committee through racing
Front node generates and broadcasts Bx with ExtraData which contains the coordinator information
Coordinator makes up the internal configuration with steering committee candidate node and standing committee node
Coordinator finalizes the configuration with steering committee node and standing committee node
Coordinator broadcasts the results to all nodes. (Block Generation)
※ QSN : Quantum Safety Net
QSP : Quantum Select Process
QVP : Quantum Voting Process

Detailed Explanation of Consensus Algorithm

▣ Consensus algorithm of REAP CHAIN

● Terminology and Technical Symbols

- **QM Server:** QM Server has a piece of network information and public keys of all nodes, currently connected to the blockchain network; it is responsible for the rule-based selection of nodes using a distributional property of quantum random number. Encryption and allocation of public key after generating quantum random number to selected nodes, and generation of hash function of public key. It has physical, functional, and role-related characteristics.

  Physical Characteristic: It is a server in which quantum random number generator with a specific address is installed and has disclosed to all nodes in advance.

  Functional Characteristic:
  ① Generation of quantam random number
  ② Generation of encryption key
  ③ Probabilistic Selection, and
  ④ Implementation of Block Validation Function has defined as the working functions of the server.

Role-related Characteristic: The working functions are combined to implement
(1) Selection of coordinator and
(2) The role of taking care of selecting steering committee candidates.

If quantum random number generators are installed or implemented on all standing committee nodes, a node is selected from the standing committee nodes to assume the role of the QM server.

- QRND: It Means quantum random numbers that have generated from the quantum random number generator. It is possible to generate a random byte number in the desired size under 4,9151 bytes at once by using a library function GETQRNARR() of Qmanager. An even distribution within a vast natural number N has obtained by applying Bitwise operation.

- Transaction: $tx_i,\ i = 0,1,\cdots$
- Block: $Bx = (\cdots, (tx_0, tx_1, \cdots), \cdots)$
- Node n_* : Front (Single) n_F, Coordinator (Single) n_C, Standing Committee (Multiple) n_S, Steering Committee Candidate Group (Multiple) n_{RR}, Steering Committee n_R , General (Multiple, standing committee excluded) n_G

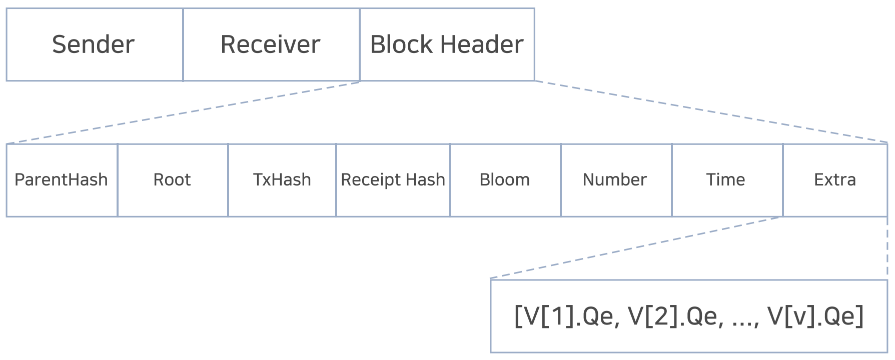
- PK^i : Public key of node i
- pk^j : Means that applicable object is encrypted as public key of node j by being used as a subscript of the object.
- Q^i : Quantum random number assigned to node i.
- $Q^i_{pk^j}$: Character string encrypted as public key of node j of quantum random number Q^i, assigned to i
- h_{pk^j} : Public key has value of node j
- N^i : Recognition information (node-net-info nni): ID of node (public key) of node i, network information
- $N^i_{pk^j}$: A value in which recognition information of node i are encrypted as public key of node j
  A superscript c in place of i or j means the node is a coordinator node.

● Definition of Terms and Status of PoDC Algorithm

- Environment: E is QM server, n (Node: Front, coordinator, standing committee, steering committee candidate, steering committee, and general)

- Action(Action, A): Means all active or passive activities related to environment with respect to generation of blocks that occur within the blockchain network.
  There are 1) Actor: QM, n_F, n_C, n_S, n_{RR}, n_R, n_G
  2) Receiver: Bx, QM, n_F, n_C, n_S, n_{RR}, n_R, n_G
  Status of receiver changes in accordance with action. Expressed as a natural number depth containing zero in an order of progress.

- Message Structure: ExtraData item of a block that needs to be updated is propagated with information of nodes that require action.
  ■ Sender: Information of sending node.
  ■ Receiver: Information of receiving node
  ■ Block Header: Header of a block to be generated going forward
    ◆ ParentHash: Hash value of parent block header
    ◆ Root: Hash value for root node of the Merkle Patricia tree that contains account status information
    ◆ TxHash: Hash value for root nodes of Merkle trees of all transactions within the current block
    ◆ ReceiptHash: Hash value for root nodes of Merkle trees of all transaction receipts within the current block
    ◆ Bloom: 32-byte bloom filter used to retrieve log information
    ◆ Number: Number of current block (The genesis block has a block number of 0 and then increasing by an increment of 1 thereafter.)
    ◆ Time: The time of initial generation of current block is recorded and used as a reference for chronological order when the blocks are connected to the chain. The time of block generation cannot be randomly manipulated.
    ◆ Extra: Other additional information related to the current block.
    ● V[1],V[2], ⋯ , V[v]: Steering committee candidate consisting of v persons.
    ● V[i].Qe: QRND and coordinator information of steering committee candidate i which is encrypted as public key of steering committee candidate i, a member variable containing public key hash value of steering committee candidate i



Example of Stage 4 Message

- Status: S, Changes of status are classified into changes of block and changes of environment (QMserve, node).
  Status Function 1: $S(QM,\ n_F,\ n_C,\ n_S,\ n_{RR},\ n_R,\ n_G)$ For each variable, the environment status becomes the input and the result is a status in an integer value that is equal to or greater than zero expressed in depth.
  Status Function 2: $S(Bx)$, Function in accordance with changes of status of blocks.
  There are three states; (1) Block proposal, (2) Addition of ExtraData, and (3) Block update.

- (Stage) or (Depth): Means all statuses that take place until status change of one block is completed.

Depth	Actor	Receiver	Environment Status	Block Status
0	-	all	Stanby Status	
1	-	n_F	Status change proposal of front node block	Block proposal
2	n_F	QM	Block transmission to Qmanager by front node	
3	QM	-	Selection of coordinator and steering committee candidate group of Qmanager $\Rightarrow$ ExtraData generation	
4	QM	n_F	ExtraData transmitted to front node by,	ExtraData update by block
5	n_F	all	Broadcasting of updated block by front node	
6	-	n_C, n_{RR}	Recognition of primary selector (coordinator, steering committee candidates)	
7	n_C	QM	Coordinator registration (Selection)	
8	n_C	n_S, n_{RR}	Coordinator: Internal network configuration	
9	n_C	n_{RR}	Coordinator: Racing notification	
10	n_C	n_{RR}	Coordinator: Steering Committee Registration	
11	n_C	n_S, n_R	Coordinator: Steering Committee Finalization	
12	n_S, n_{RR}	n_C	Block Validation	
13	n_C	all	Block Propagation	Block Update

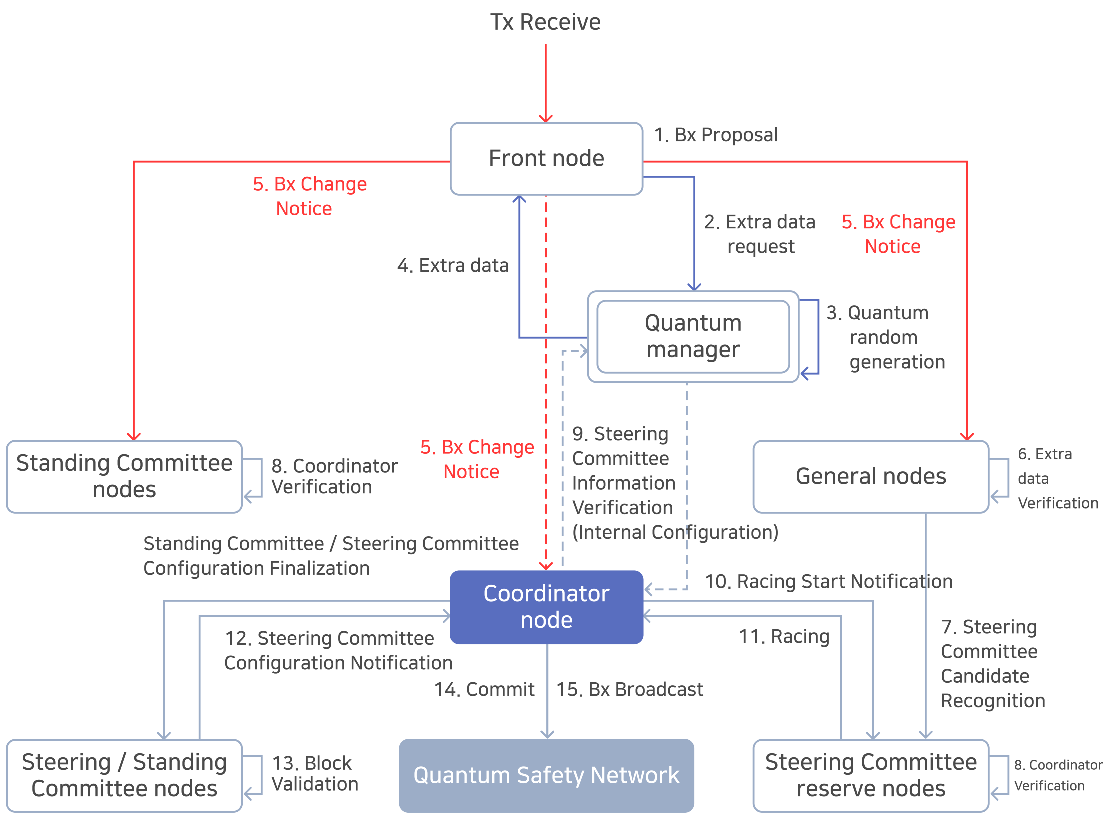
- Status Transition Figure of Environment Objects

QM	nf	nc	ns	nrr	nr	ng
-	-	-	-	-	-	-
-	1	-	-	-	-	-
2	2	-	-	-	-	-
3	-	-	-	-	-	-
4	4	-	-	-	-	-
0	5	5	5	5	5	5
-1	-1	6	-	6	-	-
-1	-1	7	-	-	-	-
-	-1	8	8	8	-	-
-	-1	9	-	9	-	-
-	-1	10	-	-1	10	-
-1	-1	11	11	-1	11	-
-1	-1	12	12	-1	12	-
-1	-1	13	13	-1	13	13

-: Stanby status, 1,…,11: Each status described above,

-1: Termination status Detailed Transitions of Environment Processing Phases:

Detailed Transitions of Environment Processing Phases:


Tx Receive
Front node
1. Bx Proposal
5. Bx Change Notice
2. Extra data request
5. Bx Change Notice
4. Extra data
Quantum manager
3. Quantum random generation
Standing Committee nodes
8. Coordinator Verification
5. Bx Change Notice
9. Steering Committee Information Verification (Internal Configuration)
General nodes
6. Extra data Verification
Standing Committee / Steering Committee Configuration Finalization
Coordinator node
10. Racing Start Notification
12. Steering Committee Configuration Notification
11. Racing
7. Steering Committee Candidate Recognition
14. Commit
15. Bx Broadcast
Steering / Standing Committee nodes
13. Block Validation
Quantum Safety Network
Steering Committee reserve nodes
8. Coordinator Verification

1. Standby Status: Waits for status change proposal of new block.

2. Block Status Change Request: Proposal (T-Proposal): Front node receives transaction(Tx) which requests for status change from proposer.

3. Front node accesses Qmanager before block generation to make request (with its signature included) for ExtraData generation for consensus and waits for response.

4. Qmanager (Q-Work): Qmanager works on ExtraData generation for consensus. In particular, it encrypts coordinator and steering committee candidates information, as well as quantum random numbers fore their recognition, transform them into ExtraData, and transmit them to awating front nodes.

   Qmanager Notification (Q-Request): Qmanager uses public key of front node to transmit the encrypted Extra Date information after generating Extra Data and the front node receiving the information validates the Extra Data by validating its signature by using its own private key.

5. Front Notification (F-Request): Front node adds and updates the Extra Data item of a proposed block and transmits new Bx to all nodes after validating to make sure that there is no abnormality.

6. Selection of Coordinator and Steering Committee Candidate Group:
   - Selection of Coordinator: Each standing committee node first decodes a quantum random number list identifier, the last item of the Extra Data to verify the public key hash value, and to check whether he/she is a coordinator or not. A verified coordinator decodes the item contents with a public key to obtain a quantum random number list of the steering committee candidates.

   Selection of Steering Committee Candidate Group: Each general node decodes receiving ExtraData to check whether the network information (node net-info, nni) of the coordinator and the applicable node belong to the candidate group. Thus, preparing those nodes that are verified as steering committee candidates for registration into the steering committee with the coordinator.

7. Coordinator Confirmation (C-Confirm): A verified coordinator has access to Qmanager to notify and confirm that it has registered as a coordinator.

8. Coordinator Notification (C- Request): A coordinator uses public keys of subject nodes (standing committee + steering committee candidates) to notify that it has been selected as the coordinator and configures the internal network.

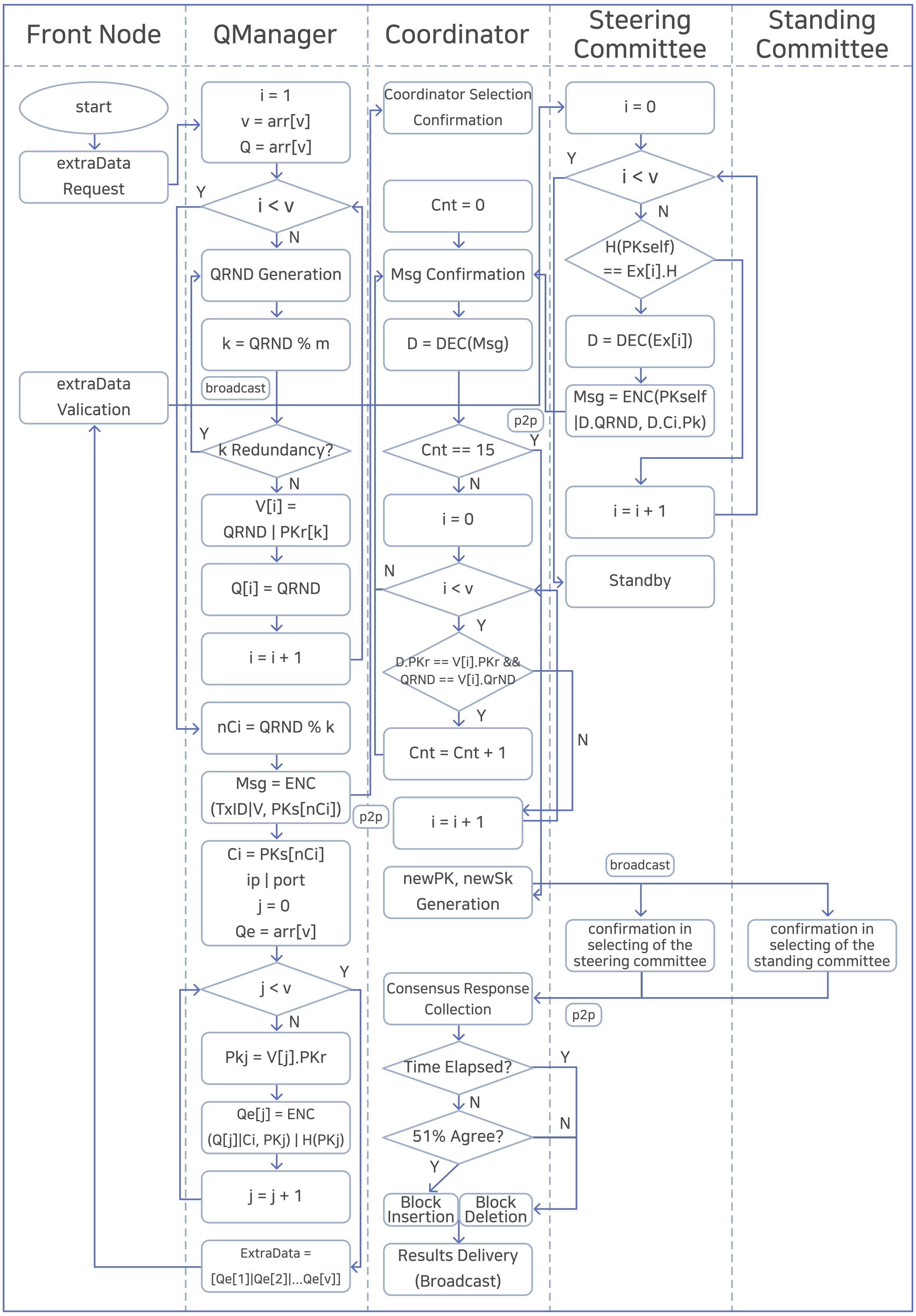
9. Racing: The steering committee candidate nodes, receiving notification of registration from the coordinator, start the registration process (racing) on a first-come, first-serve basis.

10. Steering Committee Finalization (D-Select Completed): The coordinator validated the registered nodes and finalized the steering committee with the first 15 nodes that had registered.

11. Block Validation (D-Validate): Validator nodes, i.e. (standing committee + steering committee), upon validation, notify the coordinator with the validation message, and the coordinator waits until 2/3 are validated.

12. Validation (D-Commit): The coordinator propagated the result of block validation (i.e., hash validation from the Genesis block to the current block) to all nodes when the verification node reported as usual.

<Consensus Algorithm Flowchart>


Front Node
QManager
Coordinator
Steering Committee
Standing Committee
start
extraData Request
extraData Valication
i = 1
v = arr[v]
Q = arr[v]
i < v
Y
N
QRND Generation
k = QRND % m
broadcast
k Redundancy?
V[i] = QRND | PKr[k]
Q[i] = QRND
i = i + 1
nCi = QRND % k
Msg = ENC (TxID|V, PKs[nCi])
p2p
Ci = PKs[nCi]
ip | port
j = 0
Qe = arr[v]
j < v
Pkj = V[j].PKr
Qe[j] = ENC (Q[j]|Ci, PKj) | H(PKj)
j = j + 1
ExtraData = [Qe[1]|Qe[2]|...Qe[v]]
Coordinator Selection Confirmation
Cnt = 0
Msg Confirmation
D = DEC(Msg)
p2p
Cnt == 15
i = 0
i < v
D.PKr == V[i].PKr && QRND == V[i].QrND
Cnt = Cnt + 1
i = i + 1
newPK, newSk Generation
Consensus Response Collection
Time Elapsed?
51% Agree?
Block Insertion
Block Deletion
Results Delivery (Broadcast)
i = 0
i < v
H(PKself) == Ex[i].H
D = DEC(Ex[i])
Msg = ENC(PKself |D.QRND, D.Ci.Pk)
i = i + 1
Standby
broadcast
confirmation in selecting of the steering committee
confirmation in selecting of the standing committee
p2p

Transaction and Block Building

REAP CHAIN implements a double chain structure to give finality to the transactions, where the final confirmation has made by generating Cell Blocks in a temporary ledger. Thus, followed by block generation in a permanent ledger where blocks containing transactions are stored in secured hardware storage and can't be changed any further.

The size of blocks can be changed dynamically, and TPS, which represents the expectation of a transaction, can achieve high TPS through the improvement of performing nodes.

< Table – Comparison of Block Generation Cycle and Finalization Frequency of Other Blockchains >

	Consensus	Block Generation Cycle (Seconds)	Finalization Frequency	Finalization Time (Seconds)
EOS	DPoS	3	15	45
STEEM	DPoS	3	15	45
NEO	dBFT	15 ~ 20	1	15 ~ 20
Ethereum	PoW	14	12	180
Bitcoin	PoW	600	6	3,600
REAP CHAIN	PoDC	2	12	24

Since the consistently operated nodes of REAP CHAIN account for 48% of all consensus nodes and the general nodes using quantum random numbers have randomly selected without prediction, chances that a ratio of malicious user nodes is at 52% continuously is statistically meager.

▣ REAP CHAIN Node Composition

REAP CHAIN consists of various types of nodes as per their roles and authorities. Some of those nodes are Some of those nodes have elaborated below:

1. **Standing Committee Nodes**: These nodes are operated continuously in REAP CHAIN and have authorities to generate blocks by participating in the consensus processes.

2. **General Nodes**: These nodes can install by anyone and may participate in the transaction validation processes through a random selection method using quantum random numbers. Nodes that participate in the transaction validation defined the steering committee as a validator.

3. **Governance Nodes:** These are particular nodes that control fees allocation policy, quantum random selection policy, and all kinds of node management policy features.

4. **Qmanager**: It generates and manages quantum random numbers and hence utilizes the node information provided by the governance to support the coordinator. And the selection of steering committee group processes with quantum random numbers.

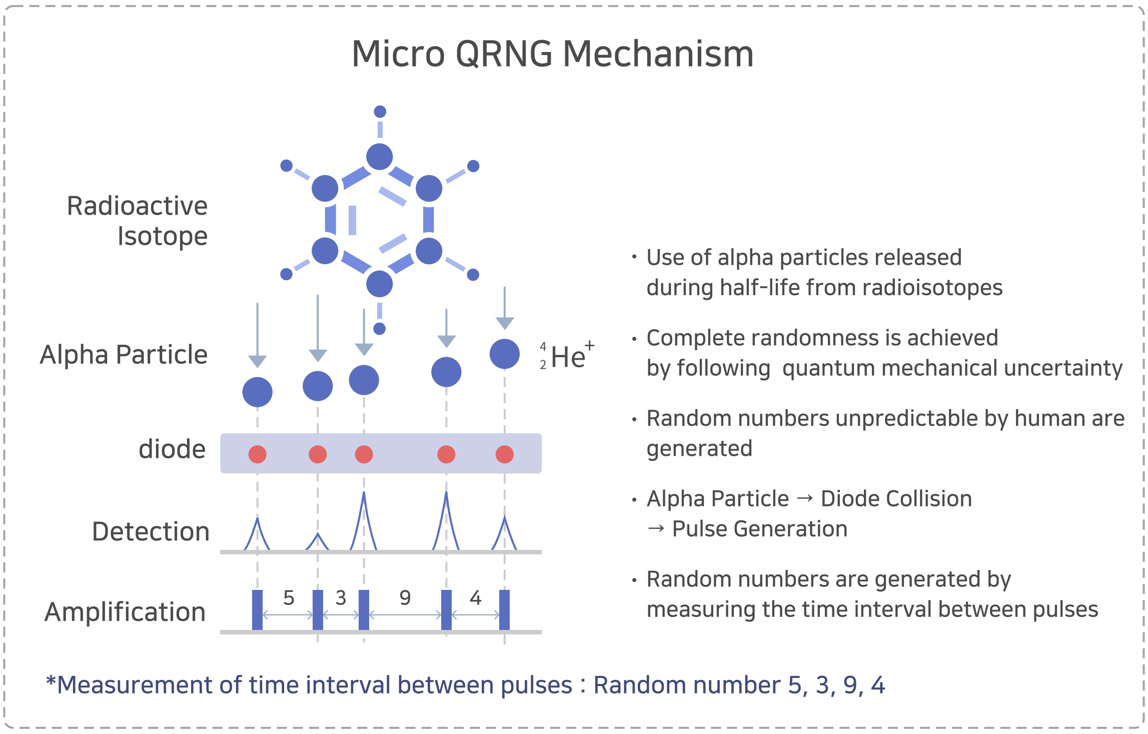
< Table – Roles and Authorities by Node Types >

Authority	Qmanager	Governance Node	Standing Committee Node	General Node (Validator)	General Node
Node Management	X	○	X	X	X
Block Generation Authority	X	X	○	X	X
Consensus Participation	X	X	○	○	X
Transaction Processing	X	X	○	○	○
Block Inspection	X	X	○	○	○

Security Enhanced Randomness through Quantum Random Number

Nodes that are participating in consensus processes of REAP CHAIN has selected through an unpredictable mechanism through Quantum Random Number Generators (QRNG).

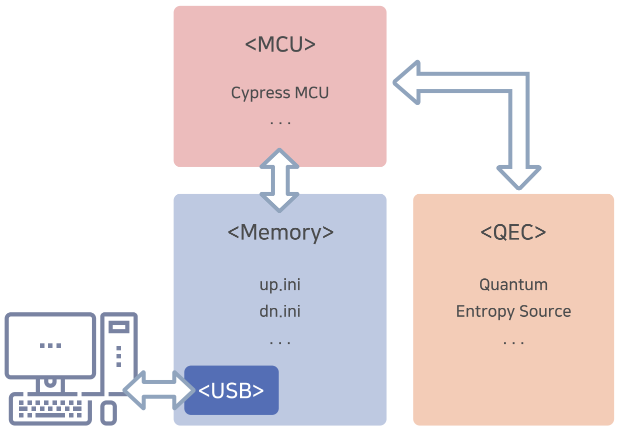
Since a predictable pattern of random numbers generated by random number generators can have a detrimental impact on security industries in which encryption technologies applied, the roles of stable random number generators are highly critical. Such processes, an encryption key generation, and validation must not be easily predicted, especially in open systems like public blockchains.


Micro QRNG Mechanism
Radioactive Isotope
Alpha Particle
diode
Detection
Amplification
$^{4}_{2}He^{+}$
5
3
9
4
• Use of alpha particles released during half-life from radioisotopes
• Complete randomness is achieved by following quantum mechanical uncertainty
• Random numbers unpredictable by human are generated
• Alpha Particle → Diode Collision → Pulse Generation
• Random numbers are generated by measuring the time interval between pulses
*Measurement of time interval between pulses : Random number 5, 3, 9, 4

When an encryption system constructed, three factors of the condition must warranty: unpredictable, unbiased, and uncorrelated. Pseudo-random numbers generated by mathematical algorithms are now widely used as advanced computing technologies, but encryption technologies that use these pseudo-random numbers are vulnerable to hacking attacks that identify patterns of encryption through monitoring the random numbers.

In the case of existing pseudorandom number generators (PRNG), the seed value can be traced with encryption patterns shown below.

- Use blockhash value of old block
- Use blockhash value of current block
- Use blackhash value of the last block
- Use block variables



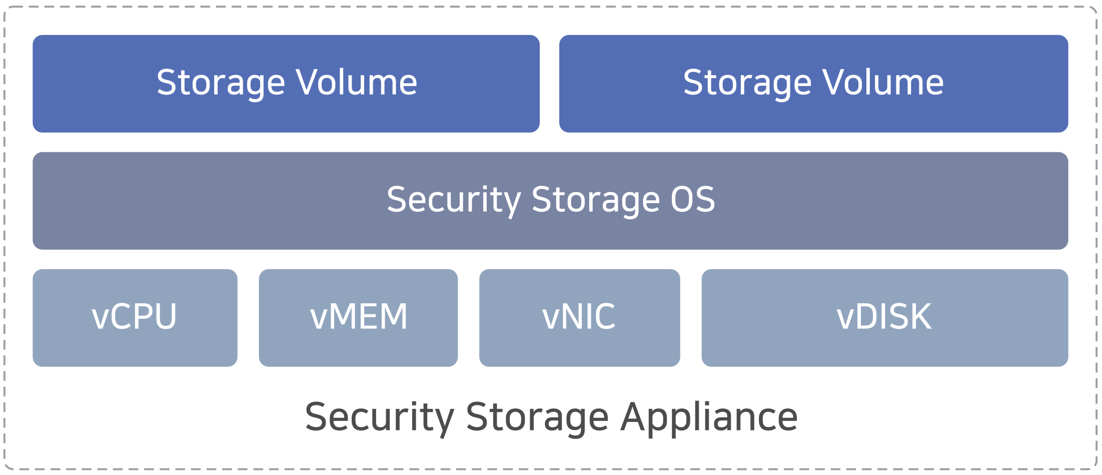
Compared with pseudo-random number generation mechanism, QRNG that uses quantum mechanical uncertainty can be regarded as more legitimate RNG, and therefore REAP CHAIN uses such QRNG to enhance the security required during its consensus processes. When the technologies make further advancements, consensus processes on blockchain may require Proof of Randomness (PoR), to perform someday.

While quantum random numbers have encrypted by using public keys of each candidate node selected through the quantum random number before being distributed on a blockchain network,
only those nodes that have chosen as the steering committee candidate group can decrypt and view the quantum random number by using their private keys.

The decrypted content includes public key and coordinator information that has selected through the use of the quantum random number. By this information, the quantum random number is encrypted and delivered to the coordinator through racing. The coordinator compares quantum random numbers for each node of the steering committee candidate group previously received from Qmanager to verify the integrity of the selected candidate group.

Data Saved in Secured Hardware Storage

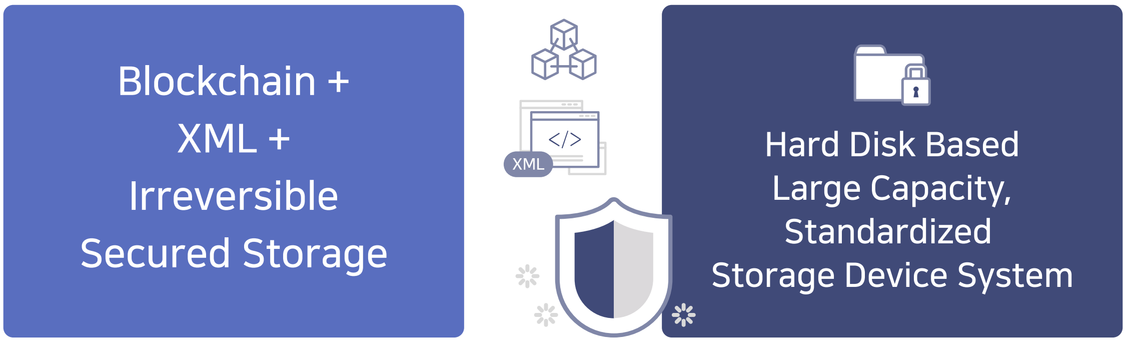
REAP CHAIN stores data in secured storage specially designed for blockchains. Just as data recorded on DVD has not erased likewise, REAP CHAIN data recorded on secure storage cannot be tampered and deleted. Also, access to the stored data is impossible without appropriate permission. For doing this, optimal security technology for blockchain data has applied to the REAP CHAIN.



A secured storage technology applied to the REAP CHAIN is similar to a concept of an optical disk, which is a read-only storage medium. Mainly, OS level security technology that performs computing resource management and authority management is applied. In the end, it combines the software concept of blockchain with hardware of secured storage to accomplish stronger protection from stronger tampering attempts.

Arguments made by existing blockchains are tampering and would not be permissible as long as solid links between data have established by using block hashes between blocks containing data. Tampering can be possible if a hacker has an ability of super-strong computing; additionally, data blocks can get deleted, while delaying or disabling the services.

REAP CHAIN's secured data management technology stores block hashes and encrypted data in proper storage to absolutely prevent access control and hacking. By applying the blockchain data block hash, encrypting the data, and storing on physical hardware, absolutely irreversible triple security processing has been accomplished.



REAP CHAIN can safely protect original data of national historical records, financial transactions, data securities, specific student records, and graduation certificates from uncontrollable events like war, natural disasters, and hacking attacks. Additionally, in regards to vulnerabilities to ransomware hacking attempts, the access-controlled thoroughly so that control is never taken over by the hackers.

Revenue and Dividend

In general, nodes that mine blocks receive from transaction fees in PoW (Proof-of-Work) method and deposits of cryptocurrencies are required to receive compensations in PoS (Proof-of Stake) method. On the other hand, in the REAP CHAIN, fees are allocated simply by holding REAP coins.

The following are the transactions that may take place in services depending on the types of service providers.

- l Reap pay transactions.
- l Used item transactions for marketplace platforms.
- l Transactions for Reap Bank.
- l Reap Chain universal voting system transactions.
- l All other transactions in REAP CHAIN that uses blockchains.

The total fees consists of standard fees, additional fees, and contract fees, as shown below.

{Total Fees = 0.02 Reap + (0.02Reap × OpCode) + 10,000 Reap}

REAP CHAIN pays dividends to Reap holders, standing committee, steering committee, and candidate groups to vitalize the ecosystem. Dividends are gross commission income and are paid in Reap coins when the specific terms of dividends, which may include fees reserve amount and block generation cycle, are satisfied. If dividends exceed the upper limit at the time of payment, all payments will get lump sum.

Total dividend ratio is 70% for all Reap holders, 20% for the steering committee and candidate group, and 10% for the standing committee and matching dividends are allowed.

Following is the calculation of dividends:

▣ All Reap Coil Holders

$$Dividend(n) = \sum_{i=1}^{m} D_i \times 0.7 \times \frac{B(n)}{R_{Issue}}$$

$\mathrm{Dividend}(n) : Dividend\ of\ user\ n$

$D_i : Total\ Dividedn\ of\ i - th\ block$

$\mathrm{B(n)} : Balance\ of\ user\ n(The\ total\ reap\ amount\ of\ user\ n)$

$\mathrm{R}_{Issue} : The\ amount\ of\ Reap\ Issue\ Amount$

$\mathrm{m} : Block\ number\ per\ dividend\ cycle$

▣ 14 Standing Committee Nodes

$$Dividend(cn) = \frac{\sum_{i=1}^{m} D_i \times 0.1}{\sum_{i=1}^{14} C_i}$$

$\text{Dividend}(cn) : Dividend\ of\ standing\ committee\ n$

$D_i : Total\ Dividedn\ of\ i - th\ block$

$C_i : Number\ of\ standing\ committee\ i = 1 \cdots 14$

$\text{m} : Block\ number\ per\ dividend\ cycle$

▣ 15 Steering Committee Nodes and Candidate Group

$$Dividend(scn) = \frac{\sum_{i=1}^{m} D_i \times 0.2}{\sum_{i=1}^{m} SC_i}$$

$\text{Dividend}(scn) : Dividend\ of\ steering\ committee\ n$

$D_i : Total\ Dividedn\ of\ i - th\ block$

$SC_i : Total\ Number\ of\ standing\ committee\ i = 1 \cdots m$

$\text{m} : Block\ number\ per\ dividend\ cycle$

In addition to dividends on total commission income, 10% of all Reap coins issued are paid to all Reap coin holders as rewards for participation in the ecosystem at times of certain block cycles in proportion to individual Reap holdings. The formula used for such supplementary dividend is as follows.

$$\text{A Supplementary Dividend(n)} = R_i \times \frac{B(n)}{R_{Issue}}$$

$\text{A Supplementary Dividend(n)} : A\ Supplementary\ Dividend\ of\ user\ n$

$\text{R}_i : Total\ Reap(Reap\ Chain\ Coin)\ Issue\ Amont\ of\ i - th\ block$

$\text{B(n)} : Balance\ of\ user\ n(The\ total\ reap\ amount\ of\ user\ n)$

$\text{R}_{Issue} : The\ amount\ of\ Reap\ Issue\ Amount$

$$R_i = \sum_{n=1}^{i} RB_n$$

$$RB_n = \frac{0.1 \times \sum_{i=1}^{n} RB_i}{Total\ number\ of\ Block}$$

$$\text{Total number of Block} = \frac{10\ years}{Block\ generation\ Cycle}$$

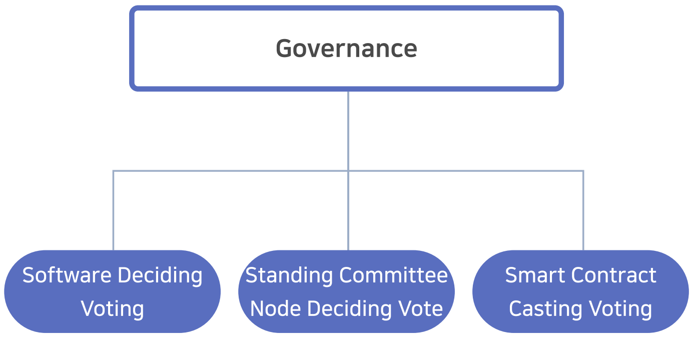
RB_n: $Reap\ Issue\ Amount\ of\ per\ block$

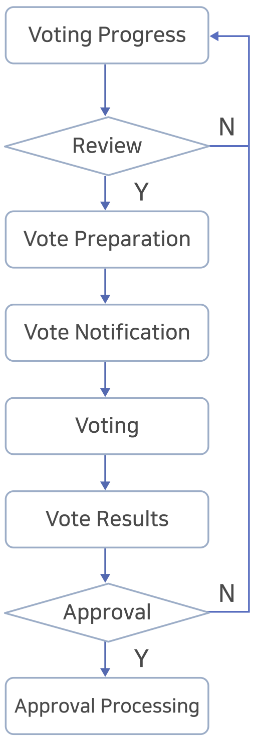
RB_i: $Reap\ Issue\ Amount\ of\ block\ i = 1 \cdots n$

Governance

In the REAP CHAIN, changes have authorized by consensus of standing committee nodes which, operated at all times.

Persons running the standing committee nodes have selected through voting software provided by REAP CHAIN and are authorized to change the underlying protocol or updating defective software. Such a person can even decide whether to approve Smart Contracts.


Governance
Software Deciding Voting
Standing Committee Node Deciding Vote
Smart Contract Casting Voting



Votes make decisions on software changes of 11 nodes out of 15 nodes. Change the underlying protocol or update defective software is determined by votes of 21 nodes out of all standing and steering committee nodes.

In any event, where a person who operates the standing committee node refuses to make governance decisions, such person may get removed by voting of participants holding REAP coins.

A person operating a standing committee node can participate in decision makings related to the issuance of tokens and approval of token burning.

REAP CHAIN Cryptography Algorithm Technical Summary

REAP CHAIN's cryptographic algorithm provides an ability to generate, sign, and update encryption keys between devices and servers without having to have a full key on a single machine.

The structure of REAP CHAIN's cryptographic algorithm has shown below.

- The encryption key does not have the complete key at any point during the generation or use of the key. The key has shared through communication, and the entire key has not exposed during such communication.
- Only typical message delivery contains information related to what is currently processed.
- Definition of actions, such as key signatures, are made by contexts that
- Works for contexts include the sharing of keys, notes, and numerous messages.

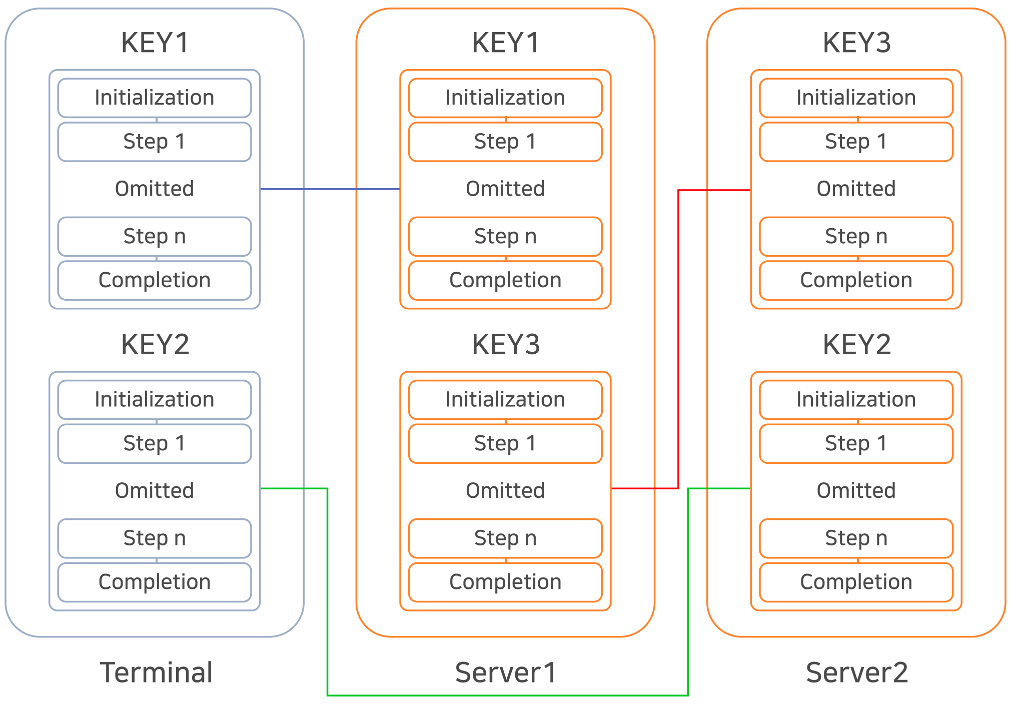
REAP CHAIN's cryptography algorithm provides the following operational functions.

- ECDSA secp256k1 generation and signing between the parties.
- EdDSA ed25519 generation and signing between the parties.
- BIP32 generation between the parties.
- Key sharing and storage between the parties.
- Zero-knowledge back up storage and recovery between the parties.

It includes content that is related to handling serialization, deserialization, and memory management for sharing messages and context structures.

The operational flow in the system has defined in a way that the first step is an initialization phase, where necessary information of actions entered, and the contents have generated, where each target performs its initialization. A series of steps exist within contexts of each stage, where each step receives input messages, perform specific tasks, generate output, and sends this output out again, where a party is receiving it and hence associated with the context. The whole system transitions into a completed final status and frees up the memory and copies the shared key information to the transition.

As per the end-to-end role, the requester has generated a key, where the signer performs a step in which it asks the requester to initiate the process for signing, upon completion. The requester sends the results to the signer and performs validation.



- Requester launches initialization function.
- Requestor enters the input parameters and performs internal Processing to generate output messages.
- The Requestor sends details and output messages to Signer.
- Signer checks details and performs execution consent.
- Signer launches initialization function.
- Signer launches step function with a message received from a Requestor as input messages.
- Signer sends output message generated from the step function to Requestor.
- Repeats processes of launching step-functions through input messages between the Requestor and the Signer and delivering and receiving the mutual output messages.

● Decisions have executed when both the requestor and the signer complete the step functions.
- fgREAP CHAIN_MPC_PROTOCOL_FINISHED flag means the last step of the requestor and the signer.
- When fgREAP CHAIN_MPC_SHARE_CHANGED flag has included, the local key share gets changed as in the New Change Action. Key sharing must be retrieved from the context using getShare function and hence saved for later use.
- A message must be sent to the other party unless the output message of the last step is empty.
- The requestor and the signer may need to launch getResult function depending on the type of operation. For example, only the requestor involves in sign action, but a new key for the signer has shared in refresh action.

The same contexts must be used throughout the entire processes. When it is necessary to save the context, it can be read again by using serialization feature followed by use of deserialization feature.

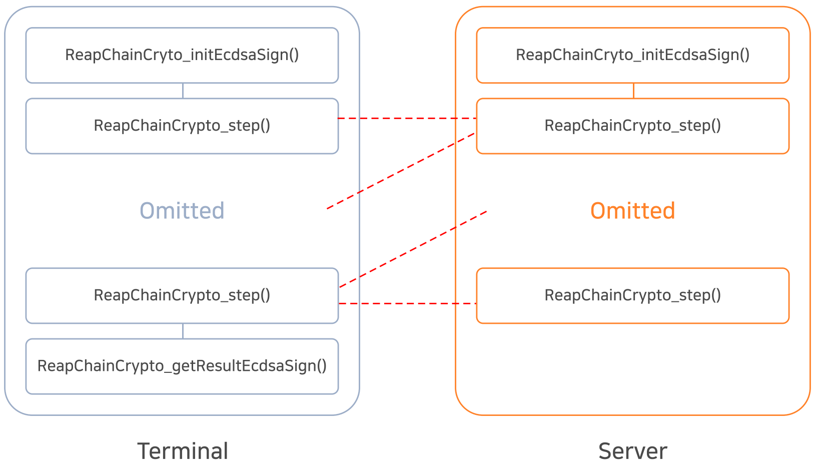
Actions of REAP CHAIN Algorithm

An example of ECDSA signing action in REAP CHAIN is shown below.

● The requestor and signer start by launching ReapChainCrypto_initEcdsaSign () function for initialization.

● After the initialization, the requestor and the signer launch ReapChainCrypto _step () function over and over until the signing process is completed.

● A signature which is the result of signing process is received by launching the final function, ReapChainCrypto_finalEcdsaSign () upon completion of signing.



REAP CHAIN Encryption Solution Summary

REAP CHAIN's cryptography algorithm is not based on a single device with a private key used to generate signatures and transfer funds.

Because a private key to be stored is shared between a device and the server, no one gets the information about the actual key, even when signatures are being generated, a secure bilateral calculation protocol that make signatures without sharing keys to each other. An account used in each terminal is not limited to the same person but can be configured in various forms.

REAP CHAIN's cryptography solution supports BIP32 / BIP44 key generation and when a key is generated by BIP key, a BIP master seed or key is used. Keys made by BIP key are executed by REAP CHAIN's cryptographic algorithm between the terminal and the server which is integrated through

signatures, and receives only a part of the key while the complete key is not disclosed.

Reason for applying BIP is it can be crucially used for recovery of keys even if some of the keys are tampered. In any event where a private key is tampered, a RSA key pair gets generated to perform the backup. The private RSA key is stored in cold backup, and the ECDSA/EdDSA private keys are encrypted with the RSA code backup public key. This encryption is performed individually by each party for its own share, and each party generates a publicly verifiable zero proof of knowledge which shows that a correct percentage of the private key is encrypted.

This method works in the same way as keys generated by using BIP or other key generation methods. The backup is a real key, so when using BIP generation, it is passed as a root private key instead of a master seed, and all generated BIP keys can be recovered if necessary. Since more efficient zero knowledge proof exists for the root node private key, a method of directly backing up the BIP master seed itself is not provided.

REAP CHAIN's cryptography solution consists of an MPC protocol that operates between two parties. However, by using the same protocol, it is possible to achieve an MPC that sets up 2-out-of-n parties (if n number of entities are present enough to sign and perform all other operations). The expansion is then executed by the first key generation by both parties, thus, sharing to all the pairs between n number of parties.

It is common and supports ECDSA on the secp256k1 curve and EdDSA oned25519 curve (or Schnorr). EdDSA uses Schnorr's Threshold Signing Protocol, which additionally shares a private key.

The zero-knowledge proof used for key generation is more efficient where keys generated through BIP and integrate MPC with ECDSA key generation by including key sharing updates.

Supported Features

REAP CHAIN's cryptography algorithm supports the following.

- **General Secret Generation:** It has used to generate the first secret to create private keys through BIP.

- **BIP Generation**: It generates keys in MPC by using the first secret as per under the BIP32 standards. The results of this step are keys that can be used by ECDSA, and the two parties thus receive the generated public key and retain a random share of the associated private keys. MPC is use, so no party knows the full private keys.

- **ECDSA / EdDSA Key Generation**: Use MPCs for two parties to directly generate ECDSA or EdDSA keys. The results are random sharing of public and quasi-private keys known to both parties.

- **Backup Mechanism**: This procedure can use zero proof of knowledge to verify the accuracy, but to ensure that nothing has disclosed about private key sharing. The backup validation validates zero evidence of knowledge. It is used by all entities to verify that the backup is valid (this feature receives a public ECDSA/EdDSA key to verify whether or not the reserve is the private key of the provided public key). Finally, the backup recovery method brings the information, and the RSA backup, private key outputs ECDSA/EdDSA private keys.

- **ECDSA / EdDSA Signatures:** This method performs a login to MPC from the message "m," which has been approved by both by using a previously generated key. Since MPC has been used, neither party can create such messages or trick the other party into signing, that is different from an approved message by other party.

- **Key Share Update**: It is used by parties to jointly generate a new random share of an existing shared key by making the previous stock obsolete. Also, if an attacker tries to steal the sharing information on one device and tries to move onto another device to take the other info after refreshing, no information will come out about the private keys.

Encryption Technique Used

① ECDSA Signature

Definition of DeECDSA signature algorithms are as follows:

Alice generates a key pair(dA, QA)

- ● l d is a private key integer randomly selected from numbers between 1 to n-1.
- ● lQ is a public key integer that satisfies Q=dg.
- ● l Alice follows the following steps to sign a message m.

1) Calculate e=H(m), where H is a cryptographic has function.
2) Let z be the Ln leftmost bits of e, where Ln is the bit length of the group order n.
3) Select a cryptographically secure random integer k from [1, n-1].
4) Calculate the curve point (x1, y1) = k * g
5) Calculate r = x1 (mod n). If r=0, go back to step 3 and repeat by selecting another k.
6) Calculate s = k-1(z + rdA) (mod n). If s=0, go back to step 3 and repeat by selecting another k.

- ● l The completed signature is the pair (r, s).

② EdDSA Signature

EdDSA signature algorithm is a Schnorr version for the Edwards curve. For a purpose of simplicity, Schnorr's signature is explained here-

Let G be an elliptic curve group of order q with a reference point of(generator) G. The private key is a random value x ∈ Zq and the public key is Q = x·G. The message signature m is as Follows-

1) Select random r ∈ Z q * (generated from private keys and messages that use pseudorandom number function but randomly selected in standard EdDSA.)
2) E ← H (R, Q, m)
3) s ← r + x·e mod q
4) Output (R, s)

③ Paillier Encryption

Paillier cryptography is a public key-based cryptosystem which is similar to RSA that relies on difficulties, associated with a large number of factorization like.

1) Paillier's public and private keys generated by selecting two random prime numbers p and q.
   For a private key (p, q), the public key is N = pq.
2) To encrypt a message m, sample a random value R and compute
   ENCpk (m, R) = (1 + N)^X * R^N mod N^2

An essential characteristic of Paillier encryption is that it is additionally homogeneous. Given cipher text c1, c2, which is an encryption of plaintext m1, m2, (c1) v mod N2 is an encryption of m·v mod N and c1· c2 mod N2 is the encryption of m1 + m2 values, and the encrypted value can be summed up and multiplied by known scalars.

④ Commitment Schemes

Commitment is an ability to reveal later encrypted values, a fundamental element of cryptography that hides the information to others while committing selected values (or sentences). The commitment scheme has been designed so that a party cannot change its value or statement after committing. In other words, the committee system is bound to change. Commitment schemes include secure coin flips, zero proof of knowledge, and other secure calculations, and are thus, applied in many cryptographic protocols.

A way of visualizing, commitment scheme is to think of a sender sending a message in a locked box and giving the box to its recipient. The message in the box is hidden from the receiver and the lock can't be disengaged directly. Because the recipient has the box, the internal message cannot be altered if the sender chooses to provide a key later.

Interactions of the commitment scheme consist of steps shown below:

- A commit phase in which the value is selected and designated.
- A publishing phase consists of value that is disclosed and confirmed in a simple protocol the commit phase consists of a single message from a sender to a recipient, and this message is called a promise.

● Hidden attribute:

At this time, the recipient must not be able to know the selected specific value. A simple publishing phase consists of a single message, opening from the sender to the recipient, and a confirmation performed by the recipient.

● Binding attribute:

A value selected during the commit phase must only be calculable by the sender and can be of unique value, which can validate during the publishing phase.

The commitment scheme is implemented by using cryptographic hash functions.

In this case, a sender can select a character string r (128 bits in length) to commit to messaging "m." Send H (m || r) to the receiver. Then, the commitment is sent by the sender (m, r), and the recipient checks to see whether or not the hash of the two values are the assigned values. It gets bound by the hash function's anti-collision properties and is hidden under
an assumption on the hash function's properties. Notably, in a random Oracle model where the hash function, modeled as a random function, the value H (m || r) cannot be identified by guessing. This way, the commitment scheme implemented by using cryptographic hash functions.

⑤ Zero Proof of Knowledge

Zero proof of knowledge is a protocol between a prover and a verifier, and the prover proves to the verifier that the statement is true without disclosing information. Zero proof of knowledge must satisfy three characteristics.

● (Soundness). The verifier cannot prove that the statement is true,
● (Completeness) the statement made by the verifier is true if the statement is accurate, and
● (Zero-Knowledge) the prover should not be able to learn information from the evidence beyond the fact that it is true. A zero-knowledge of proof enables one to prove a statement only if he or she knows the actual NP witness for a given statement. REAP CHAIN protocol uses zero-knowledge of evidence.

1) An evidence showing that the committed and encrypted values are the same.
2) An evidence showing that the committed and encrypted values are the same.
3) An evidence showing that the player knows the discrete logarithm of the elliptic curve point (ZK-DL).
4) Prove that the Paillier public key is appropriately generated (ZK-PPK).(ZK-DL).
5) Prove that the decryption of a given Paillier cipher text is a discontinuous log of a given elliptic curve point (ZK-PDL).
6) An evidence that the RSA cipher ci encrypts the value xi so that Qi becomes known xi = xi·G, used for cold backups.

In any event, where the zero-knowledge proof is none interactive, it has a publicly verifiable attribute which allows anyone to read the proof and verify that the statements are genuinely correct. It can be used to confirm that the backup is valid before transferring funds to an address.

⑥ Oblivious Transfer

The manifested transmission has used in the Garbled Circuits protocol and oblivious transmission and character strings sent between senders and receivers in the following way.

- A sender has two character strings, S0 and S1
- A recipient chooses i ∈ {0, 1} and the sender sends Si together with oblivious transfer protocol.
- The recipient cannot obtain information on S(1-i).
- The value of i is not exposed to the zero-knowledge sender.

Oblivious transmissions implemented using asymmetric encryption, such as the RSA encryption system.

Oblivious Transfer (OT) requires the use of public keys, which allows users to obtain more real OTs by only using hashes after executing OT for a fixed number of repetitions based on public keys through use OT extension.

Overall, this protocol requires three rounds of interactions in the configuration and two rounds to perform the OT operation (the first round of actual OT is sent with the third round of configuration, so overall, four rounds are required).

⑦ Garbled Circuits

Garbled Circuit means cryptographic protocols that allow two mutually untrusting parties to jointly evaluate the functionality of personal input and enable reliable, mutually, supplemented, secure calculations. The function of garbled circuit protocols shall be described with Boolean circuits. It is an encryption algorithm used in secure two-party calculation protocols.

If an evaluator is given access to a wrong circuit by using encryption of the input, one can know the output at the given input without providing any other information. It is used for safe, secured calculations between two parties by requiring garbled circuits, in which one party is later evaluated by the other for creating.

⑧ Dual Execution

If the two players are honest (meaning that they can execute specified protocols), secured calculations can be implemented even with a single broken garbled circuit. In case of malicious intent is involved; however, there is a possibility that a false garbled circuit may be used to leak the information through the output.

Achieving security when the parties are malicious is an important task and involves expenses. One of the methods used to prevent the parties from cheating is called "dual execution."

The work continues by requiring each party to execute the calculation once for the garbled circuit. And once as an evaluator of the circuit.

After two evaluations, each party compares the results without disclosing it, and both circuits produce the same output by knowing it correctly. Since each party generates one of the circuits, they understand that the channels and calculations are correct and honest.

It is because there is an interruption created in the information of dual executions that are leaked to an attacker. In other words, a single bit can leak if the comparison of the results is "not equal." For example, a compromised party can generate an incorrect circuit that calculates garbage if the first bit of the key is zero or can calculate the correct output.

The circuit is broken and cannot be detected. Then, it leaks a simple bit of whether or not there was an interruption.

However, it is possible to understand that the leak is minimized because an attack is detected when the results were not the same in the user instruction for key generation (i.e., on average, 2 bits of the key is disclosed). Thus, if an attack is detected, it is wise to remove the attacker and transfer the funds to a new address.

Based on a new equivalence test, 3 rounds of communications were constructed (using ElGamal in-the-exponent additionally homogeneous encryption). The design of the test is made to enable proof of security in the ideal/real model simulation paradigm of secured calculations and to provide verified output to both parties.

With the use of two rounds of OT based on scaling, the full dual execution has five communications, including an equivalence test.

Our dual execution protocol has the property that if a party cheats, then it can guess a single bit, but at the cost of being caught with approximately 1/2 of probability. Thus, if an attack is detected, the use of the key must be stopped.

One of the strategies used when one party tries to cheat has to claim that there is a "crash" or network error instead of providing the honest party with the results of the equivalence test (when this result is first received). If this is not processed, the other party can know all bits of the secret key. Therefore, all executions should be completed (i.e., messages required to finish have been saved in the disk, and the execution continues after crashes or failures.), or it should be presumed that there is some cheating involved.

Also, since an attacker does know a little through each parallel execution, it is not possible to execute multiple parallel operations on the same input,
If this is necessary, it is possible to change the codes so that a single equivalence test is run for every execution and this is a safe way.

Block Crypto MPC

REAP CHAIN's cryptographic algorithm adopts Secure MPC (Multiparty Computing) and employs technology designed by Professor Yehuda Lindell and Dr. Samuel Ranellucci for protection of secrets of encrypted signature key/seed.

The Secure MPC technology used by REAP CHAIN has an advantage to use without depending on a ledger. Unlike a multi-sig type in which multiple users create their keys and use a master key obtained by combining such keys for encryption and protection of secrets.

Shown below are the 3 advantages:

● l Properties of Separation: The secret of private keys is maintained and stored in two separate physical spaces. And contained information is not mutually shared.
● l Properties of Confidentiality: Private keys stored in a separate critical space Where all cryptographic operations are executed without being associated/combined and are known to uncompromisable unless two devices attack at the same time. Additionally, since the keys themselves are physically separated, necessary information cannot be obtained even if one is compromised.
● l Properties of Unpredictability: Keys are shared by continually changing values as they are transmitted, which prevent the information leaked unless the two physically separated systems are compromised simultaneously.

Key algorithms used in REAP CHAIN include the following.

● l ECDSA secp256k1 generation and signing between the parties.
● l EdDSA ed25519 generation and signing between the parties.
● l BIP32 generation between the parties.
● l Key sharing distribution and storage between the parties.
● l Provisioning of zero-knowledge backup between the parties.

The fundamental algorithms are encryption methods that are deployed and used on two or more separate systems to provide robust security.

REAP CHAIN's security algorithm can is used to provide security in apps on the blockchain.

① How to use between an individual terminal and the server

REAP CHAIN users can communicate with the server by using mobile wallets of individual terminals and receive services of the encryption algorithm.

BIP32 seed value and all signature values do not exist in one terminal but two or more separate devices. Therefore, it is required to execute an encryption operation through communication.

1) Typical Standards
- Mnemonic code words, based on BIP-39
- HD wallets, based on BIP-32
- Multipurpose HD wallet structure, based on BIP-43
- Multicurrency and multi account wallets, based on BIP-44

2) The standards can be changed or abrogated but can use as the de facto wallet standards for REAP CHAIN and cryptocurrencies.

3) Each standard makes it possible to interoperate with a wide range of software and hardware wallets. Users can create mnemonics in one of their portfolios, put them in another wallet, and recover all the keys and addresses.

The characteristics, advantages, and disadvantages between individual terminals and servers are as follows:

● At the time of signing a transaction, the service is not provided unless the terminal and the server sign together, providing effects of preventing denials or disapprovals.
● In order to steal or damage the key, both the terminal and server must be attacked and compromised simultaneously, providing effects of the fail-safe device.
● After generating Information related to the keys should not leak outside, ensuring confidentiality, which is one of the three elements of security.
● It provides robust security against auxiliary channel attacks on the system in addition to direct password attacks.
● The protection of crypto assets with encryption methods, provide stable services to both the terminal and the server.

② How to use between a terminal and another terminal

It can be applied if a user has multiple terminals; to secure blockchain wallet and signature transaction safeties of users with various terminals.

Encryption performed through communication for BIP32 seed and all signature keys between a device and another.

The characteristics, advantages, and disadvantages between individual terminals and servers are as follows:

- Transactions cannot be approved on a single device. Instead, it must be approved on two different devices.
- In order to compromise with the critical contents of transactions, two different devices must be damaged through attacks simultaneously.
- The confidentiality is ensured because no seed key or private key is exposed during encryption.
- It provides robust protection against attacks on auxiliary channels in addition to protection on direct attacks on the keys.
- Safe service is provided for secure transactions between terminals.

Cryptography Algorithm Backup

It is essential to have plans for performing backups to ensure reliability, which is given below:

An encrypted cold backup that is allowed to be publicly validated provided as one of the default features. It will enable different participants to verify the accuracy of the reserve without having to perform decryption separately. It provides fundamental protection against incorrect generation and storage of backups through verification.

① Management of User Information when Backup is Used

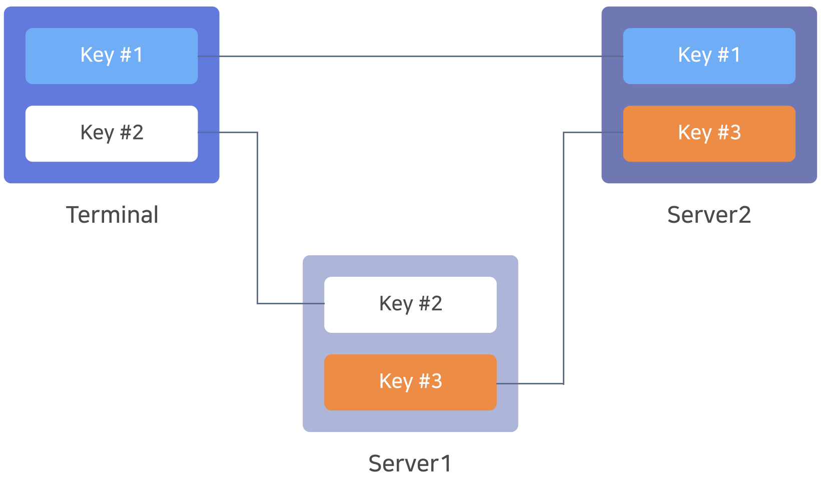
Typically, this is a standard backup format that serves as backup management for end-users.

An encrypted backup of a wallet can be stored in multiple locations for redundancy (e.g., it can be saved by the service provider as described in mobile/personal computer/server usage). The private key of this backup must individually, owns by the user as a cold backup. The backup recovery process should be used for disaster recoveries.

Encrypted backups of a wallet can be held by different users for redundancy or stored on a server. The private key for this backup must be individually owned by the user and retained in the form of cold backup. The backup recovery process should only be accessible if the key is compromised.

② Backup Using a Separate Management Server

A separate backup server can be used between users and terminals or in the server to enable uses when their public keys are lost.



Duplicate recovery information, maintained in this case, and information for making recoveries between terminals and servers, between terminals and a backup server and between the backup server and server is maintained.

References

[1] kblock, "[Kebly] #48. Understanding Consensus Algorithm - PBFT Consensus Algorithm," 2018. [online]. Available: https://steemit.com/consensus/@kblock/48-pbft-consensus-algorithm.

[2] vasa, "ConsensusPedia: An Encyclopedia of 30+ Consensus Algorithms," 2 7 2018. [online]. Available: https://hackernoon.com/consensuspedia-an-encyclopedia-of-29-consensus-algorithms-e9c4b4b7d08f.

[3] MabelOza12, "Hyperledger Consensus Algorithms," 7 4 2018. [online].
Available: https://www.slideshare.net/MabelOza12/hyperledger-consensus-algorithms.

[4] G. Konstantopoulos, "Understanding Blockchain Fundamentals, Part 1: Byzantine Fault Tolerance," 1 12 2017. [online]. Available: https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419.

[5] N. S, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [online].
Available: https://bitcoin.org/bitcoin.pdf. [access: 2019].

[6] M. &. G.-E. J. C. Herrero-Collantes, "Quantum Random Number Generators. Reviews of Modern Physics. 89. 10.1103/RevModPhys.89.015004.," 2016.

[7] A. D. Adamk , "Internet Tools and Services," 2014. [online].
Available: https://gyires.inf.unideb.hu/GyBITT/08/ch04.html.

[8] Kim S.M, Im J.C, Yoo H.K, Kwak J.Y, Blockchain and Consensus Algorithm, ETRI, 2018.

[9]A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari and D. Yakira, "Helix Consensus Algorithm," 4 2018. [online].
Available: https://www.orbs.com/white-papers/helix-consensus-whitepaper/. [access: 2019].

[10] NEM Foundation, "NEM Technical Reference," 2018.

[11] FUTUREPIA, "Whitepaper V1.2," 2019.

[12] SYMVERSE INC., "Whitepaper," 2018.

[13] Juan Carlos Garcia, "Quantum Random Number Generators," 2016. [online].

[14] Choi N.J., Lee J.E., Kim K.J., "Case Study on Auxiliary Channel Attack on True Random Number Generator and Analysis of Randomness," 2018.

[15] Ministry of Science and ICT and Korea Institute of S&T Evaluation and Planning, "The Future of Blockchain" 2019.

[16] PegaSys, "Scaling Consensus for Enterprise: Explaining the IBFT Algorithm," 6 2018. [online]. Available: https://media.consensys.net/scaling-consensus-for-enterprise-explaining-the-ibft-algorithm-ba86182ea668.