

# FINL

Technical Whitepaper V0.9

**Written by FINL team.**

1.	Philosophy -----	5 -
2.	Architecture -----	7 -
2.1	Overall Network Architecture -----	7 -
2.2	FVM-----	10 -
2.2.1	FVM Structure-----	10 -
2.2.2	FVM Roles -----	10 -
2.2.3	FVM Usages-----	11 -
2.2.3.1	LUA Script-----	11 -
2.2.3.2	nodeJS Addon-----	12 -
2.3	Block Explorer-----	12 -
2.4	Full Block Node (FBN) -----	12 -
2.5	Index Server (IS) -----	12 -
2.6	Inter-change Server Agency Global (ISAG)-----	13 -
2.7	Network Node (NN)-----	13 -
2.7.1	Smart Contract Agency (SCA) -----	13 -
2.7.2	Network Node Agency (NNA)-----	13 -
3.	Consensus Algorithm-----	14 -
3.1	Consensus Group -----	14 -
3.2	P2P Network Layer -----	15 -
3.3	Consensus Layer -----	15 -
3.4	Consensus Group Configuration Scenario -----	16 -
4.	Wallet -----	17 -
4.1	Key Generation -----	17 -
4.1.1	Concepts of Bitcoin Key Generation -----	17 -
4.1.2	Concepts of FINL Key Generation -----	19 -
4.2	DSA Algorithm -----	22 -

4.2.1	EdDSA (ED25519)	22
4.2.2	ECDSA (Secp256k1 and Secp256r1)	23
4.3	Private Key encryption and decryption	23
4.4	Wallet Address	24
5.	Token	25
5.1	Platform Token	25
5.2	Utility Token	26
6.	Smart Contract	27
6.1	Creation Rules of Account Number	27
6.1.1	User Account	27
6.1.2	Token Account	27
6.2	NFT	28
6.3	Contract	28
6.3.1	Create Time	28
6.3.2	Fintech	29
6.3.3	Privacy	29
6.3.4	Fee	29
6.3.5	From Account	29
6.3.6	To Account	29
6.3.7	Action	29
6.3.8	Contents (Blob)	29
6.3.9	Memo	30
6.3.10	Signature	30
6.3.11	Signed Public Key	30
7.	Block	31
7.1	Block Description	31

7.1.1	Block Number-----	31 -
7.1.2	P2P Address-----	31 -
7.1.3	Block Generation Time -----	31 -
7.1.4	Previous Block Hash -----	32 -
7.1.5	Transaction Count -----	32 -
7.1.6	Block Hash-----	32 -
7.1.7	Signature-----	32 -
7.1.8	Public Key-----	32 -
7.1.9	Block Confirm Time -----	32 -
7.2	Group XOR-Hash Cipher based on time arriving -----	33 -
8.	istribute Database -----	35 -
8.1	Replication -----	35 -
8.2	Shard -----	35 -
8.3	Query-----	35 -
9.	Governance -----	36 -
9.1	Currency and Issuance -----	36 -
9.2	Reward Policy-----	38 -
9.3	Fee Policy -----	38 -
10.	Inter-operability-----	40 -
11.	Reference -----	41 -

# 1. Philosophy

FINL is designed to enable digital trust-guaranteed transactions using high-frequency transactions and smart contracts based on Delegated Proof of Reputation (DPoR).

These digital trust processors are largely classified into relationship trust, information trust, process trust, and transaction trust. First, the trust of the relationship is established through the globally connected blockchain delegated reputation proof network. Second, it establishes trust in contract information through oracle problem solving. Third, trust in the process is achieved through securing transparency and regular and periodic confirmation of the transaction verification process. And Fourth, the trust of the transaction is achieved by securing the stability and reliability of the transaction.

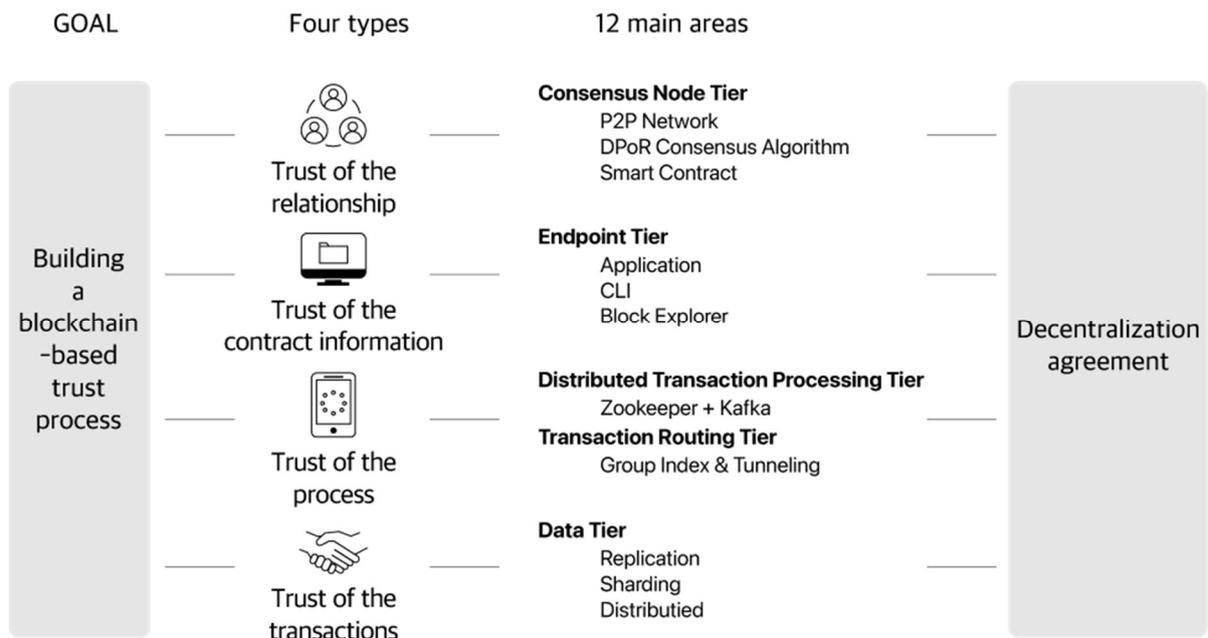
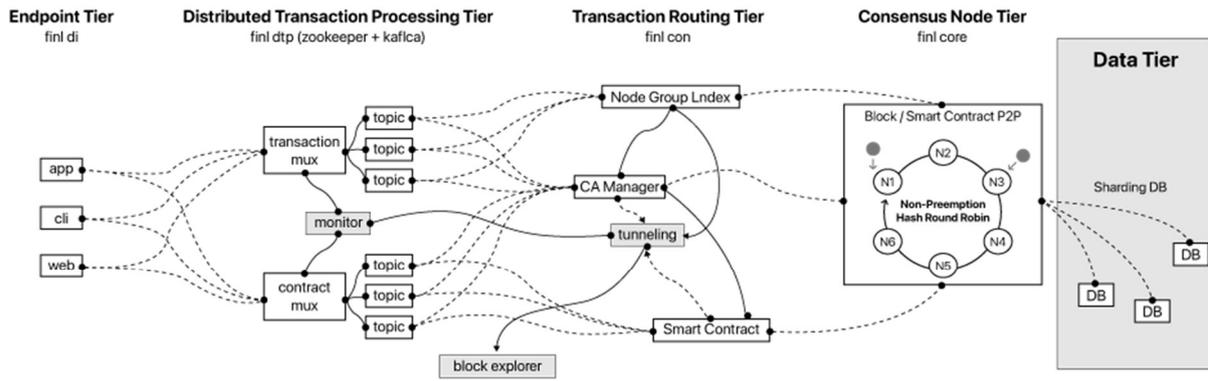


Figure 1. Trust Process

In other words, the digital trust process must maintain a fair and transparent process flow for the value and transaction of information. To this end, first, the governance structure of the decentralized network is effectively established and transparently operated. Second, in order to secure the trust of the relationship, a Consensus Node Tier consisting of a P2P main network and a P2P sub-network is operated. Third, establish the Wuhan user interface group (Endpoint Tier) to secure the trust of information. Fourth, to secure the trust of the process, a distributed transaction routing group (Distributed Transaction Processing Tier and Transaction Routing Tier) is operated. Fifth, we operate a distributed data storage group (Data Tier) where anyone can participate in order to secure trust in transactions.



Role	Description
Endpoint Tier	User application access section (application, command line interface, web application)
Distributed Transaction Processing Tier	Distributed transactions for high-speed processing as a gateway for domain access and monitoring section in real time
Transaction Routing Tier	High-speed identification network processing section of target crypto-address system for optimal exchange with distributed routing service
Consensus Node Tier	Block reliability process operation section through hash round robin and super-fast delegated proof of stake

Figure 2. Tiers

FINL consists of a P2P main network and a P2P sub-network structure based on the independently developed delegated reputation proof consensus algorithm. Each P2P subnetwork is an independent consensus group that generates blocks in parallel, and the P2P main network serially connects blocks generated by each P2P subnetwork. This serialization is called Hash Round Robin (HRR). The maximum speed of this blockchain platform increases according to the number of independent consensus groups, and the minimum speed of the independent consensus group, P2P subnetwork, is aimed at 5000 TPS. In addition, a decentralized decision-making device (DGOS, Distributed Governance Organization Syndicate) exists. All policies are decided by votes of members through DGOS, and the process of policy proposals, initiatives, and voting is transparently disclosed.

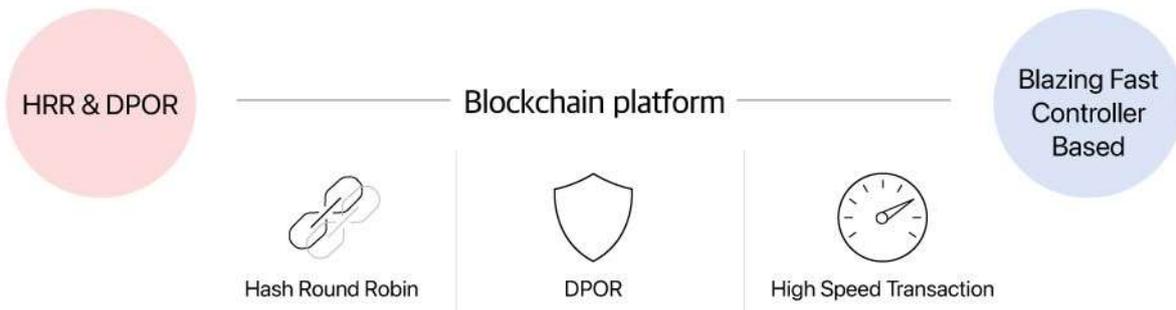


Figure 3. Main features of Final Chain

# 2. Architecture

## 2.1 Overall Network Architecture

The FINL network is largely divided into Main Chain and Sub Chain. The Main Chain acts as a Public Network, and the Sub Chain can be used as a Private Network. It is used as Globalization, and Sub Chain is used as Localization. Main Chain and Sub Chain actually function the same and are connected by Net Connector. This paper describes the Public Network.

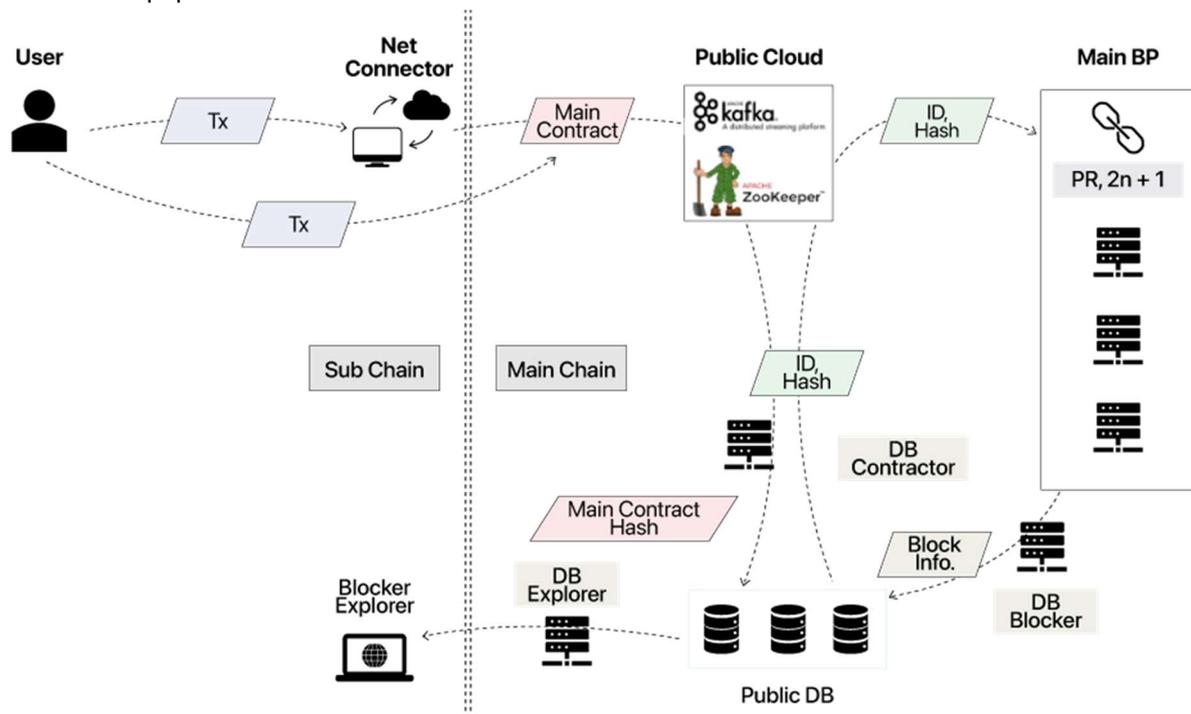


Figure 4. Overall flow of Main Chain

Net Connector or User uses Kafka/Zookeeper messaging cloud system to send transaction to chain.

The chain that receives the transaction through the messaging cloud system checks the validation of the transaction, turns it into a smart contract, gives a unique ID along with the hashed value of the contract, and stores it in the DB through the DB contractor.

Block Producer (BP) is composed of odd numbers, and each BP composes its own Cluster, that is, a P2P Sub Network. This BP records the contract's unique ID and hash value in its own block through the DB Contractor, and stores the block information in the DB through the DB Blocker.

Block Explorer can search all information such as Transaction information, Contract information, and BI

Block information stored in the Chain DB through DB Explorer.

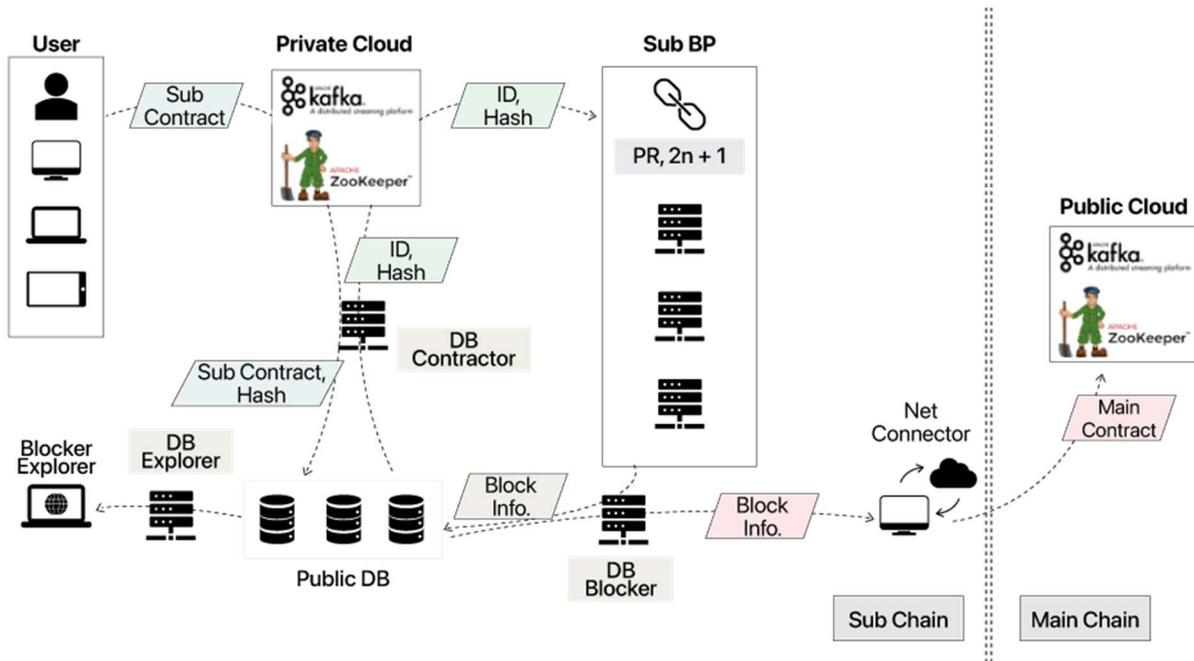


Figure 5. Overall flow of Sub Chain

The core functions of Chain can be divided into Control Path and Data Path. Control Path consists of IS, ISAG, and ISA. IS and ISAG act as independent nodes, and ISA operates as an application in Network Node (NN). Data Path consists of SCA and NNA, and both operate as applications in NN.

In addition, for the purpose of providing information to users, a Node with Block Explorer or similar functions can be operated, and these Nodes can request information necessary for a Full Block Node (FBN).

The DGOS Foundation operates the Index Server (IS), collects information on nodes that want to participate in the chain ecosystem, and can form a cluster based on this information. ISAG replicates and stores IS information and plays a role in distributing the information. In addition, ISAG replicates all information stored in NNs of each cluster. The ISA serves as a medium for various control information transmitted from the IS to the NN.

Chain is a set of Clusters with BPs as the main axis. Each BP operates NN for each cluster and plays a role in collecting transactions, making smart contracts, and creating blocks. Blocks created in parallel in this way are serially connected between clusters. At this time, replication occurs for all information stored in each cluster between NNs. In addition, the generated block is transmitted to the NN that will generate the next block through a network frame.



## 2.2 FVM

As a core element of FINL, it is used everywhere, including User, Block Explorer, and Block Chain Core.

### 2.2.1 FVM Structure

The core of FVM is implemented in C/C++. Core Function can be accessed through LUA Script, Node JS, or C/C++. Based on this structure, the same standard API can be applied anywhere.

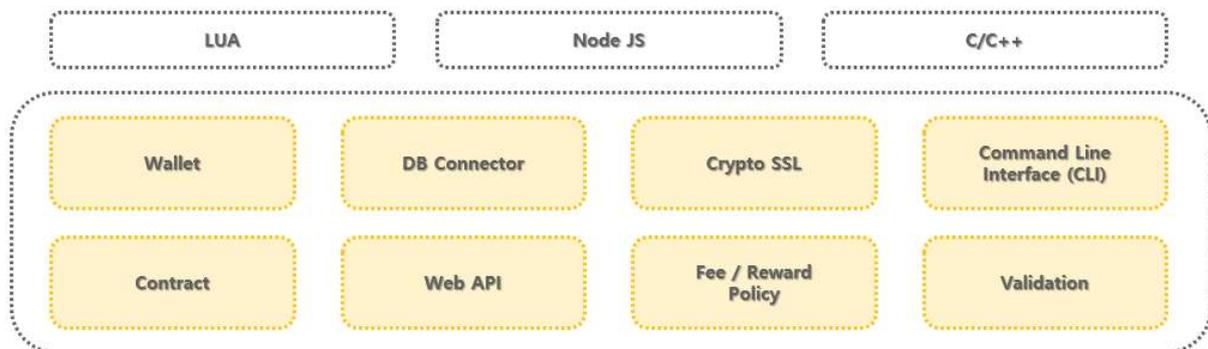


Figure 8. FVM 구조

### 2.2.2 FVM Roles

#### [Wallet]

It provides its own wallet that supports the Secp256r1 and Secp256k1 algorithms and applies the ED25519 algorithm with enhanced security. Key generation and recovery are possible using this wallet.

#### [Crypto SSL]

By providing OpenSSL-based Crypto SSL, various Cryptography functions can be used. In particular, it supports Signature Algorithms such as ECDSA and ED25519 and Encryption/Decryption Algorithms such as ECIES and X25519.

#### [Contract]

It provides its own smart contract using a script language with excellent readability and extensibility.

#### [Fee/Reward Policy]

Provides API to calculate Fee for Contract. In addition, it is possible to provide a reward policy of the node.

[Command Line Interface]

Provides API to receive commands as Arguments and execute them. Through this API, LUA, Node JS, or Functions that can be executed with the C/C++ API can be directly executed.

[Web API]

Provides a function that allows the client to communicate with the server using cURL (Client URL). In particular, HTTP GET and HTTP POST are mainly used.

[DB Connector]

Provides API to access database.

[Validation]

Provides API that can determine whether Contract and Transaction are valid.

## 2.2.3 FVM Usages

### 2.2.3.1 LUA Script

Currently, FVM is available for standalone operation using LUA Script on Linux and Windows systems. At this time, the Makefile automatically determines whether it is Windows-based or Linux-based and compiles, and the executable file can be linked with the LUA Script file.

[ lua\_addon.cpp ]

It is a language based on C/C++ and includes a main function. The lua\_addon function implemented in this file binds various C/C++ functions that can be used in LUA Script and registers them in LUA. Also, the "luaConn" function of "main.lua" is executed, and if there is an argument value during execution, the argument value is transferred to the "luaConn" function.

[ main.lua ]

It is called from the FVM executable file and there is a "luaConn" function that can execute LUA Script.

[ init.lua ]

Register the LUA Core Script files and Configuration files used in FVM.

Details related to interworking between LUA Script and FVM are provided in the manual.

## 2.2.3.2 NodeJS Addon

FVM can be used as an Addon of nodeJS on Linux basis.

[ addon.cpp ]

It is a C/C++ based language and includes Initialize function for nodeJS addon. The Initialize function implemented in this file is registered by binding various C/C++ functions that can be used in nodeJS.

Details related to interworking between nodeJS and FVM are provided in the manual.

## 2.3 Block Explorer

Live TPS check function is provided. It provides various information related to Chain, such as General Information, Account, Information, Token Information, Block List, and Transaction List.

Provides key generation and recovery functions on the web, and provides functions to create Encryption/Decryption-based Contracts.

## 2.4 Full Block Node (FBN)

Replicate ISAG information. Replication information can be distributed to other nodes.

## 2.5 Index Server (IS)

IS manages information about the entire mainnet system and can control the system.

It provides the function of whether to participate in the consensus group and the selection of the node role. It provides functions such as network initialization, save, update, and restart. Provides node startup and shutdown functions. Provides functions such as adding, fine-tuning, and deleting topics by cluster to the Kafka/Zookeeper cloud network. Genesis contract creation function is provided. Provides functions to start, stop, or restart block generation.

## 2.6 Inter-change Server Agency Global (ISAG)

It has a duplicate of the information stored in the IS.

If an external node wants to check the information stored in the IS, it can acquire the necessary information through the ISAG instead of directly accessing the IS.

## 2.7 Network Node (NN)

### 2.7.1 Smart Contract Agency (SCA)

[Contract reception, verification, storage function]

User contract reception, verification, and storage functions entered through the distributed scheduler C ontract reception, verification, and storage functions through ISA.

[Transaction creation and storage function]

The non-duplicate key value given to the contract and the hashed value of the contract are bundled together and called a transaction. Create and store these transactions.

[Block information reception function]

Receive block information generated by the local consensus node.

### 2.7.2 Network Node Agency (NNA)

[ Consensus group network information update and configuration function ]

P2P main network update function. P2P subnetwork update function P2P subnetwork and P2P main network configuration function according to node information.

[Node security update function]

Automatic firewall setting function according to node and network information.

[Block creation start, reception, verification, delivery function]

Function to create and verify blocks Function to deliver the created block to other consensus groups Function to deliver the created block to the local SCA.

[Transaction processing function]

Transaction reception function from Local SCA.

# 3. Consensus Algorithm

## 3.1 Consensus Group

A consensus group refers to a set of NNAs that participate in the process of creating, exchanging, storing, and confirming blocks between clusters, authenticating transactions and making them irreversible.

Each NNA belonging to the consensus group is logically composed of a P2P Network Layer and a Consensus Layer developed by itself on the TCP/IP Layer basis. The Security Layer supports the security functions of the P2P Network Layer and Consensus Layer such as ECIES, X25519, ECDSA, and EDDSA

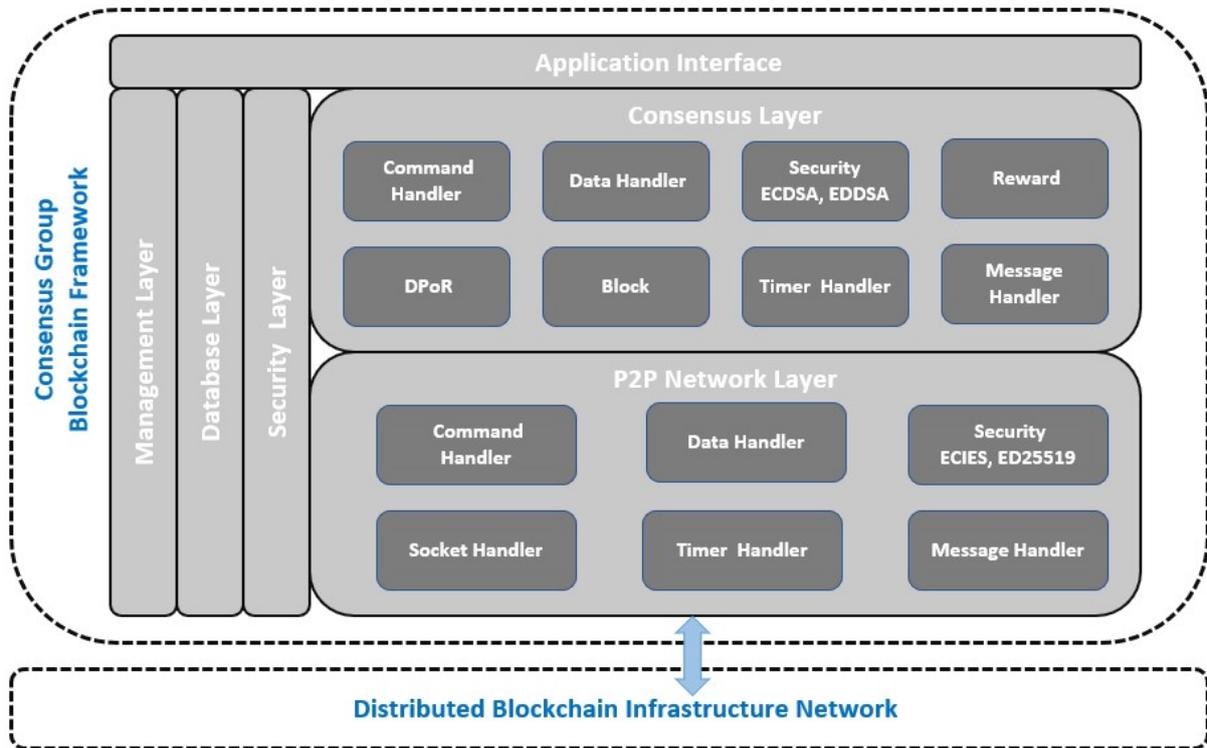


Figure 9. Logical layer of NNA

## 3.2 P2P Network Layer

8 Bytes	8 Bytes	8 Bytes	4 Bytes	2 Bytes	2 Bytes	Variable	4 Bytes
Dest Addr	Src Addr	Timestamp	Info	Seq. Num.	Total Length	TLV	CRC32

Figure 10. P2P Header Description

The P2P address system, which is the basis of the P2P network, consists of a total of 64 bits.

The upper 32 bits contain GPS information and consist of 16 bits of latitude and 16 bits of longitude, if latitude and the longitude is "37.5207176, 127.0539982", use only "37.52, 127.05", which is the lower two digits of the decimal point. In decimal, it is "37 52 127 05" and in hexadecimal it is "25 34 7F 05". Therefore, it becomes possible to display as "0x25347F05". At this time, the GPS maximum error range is 2.88 km, which is two multiples of 1.44 km.

The lower 32 bits are the chain code, continent code, country code, national hub network code, hub network subnet code, subnet node address, etc. indicating whether this blockchain platform operates as a public or private blockchain, is used as.

Total Length	64 bits	0x	FFFF	FFFF	FFFF	FFFF
GPS Information	32 bits	0x	FFFF	FFFF	0000	0000
Unique Information	16 bits	0x	0000	0000	FFFF	0000
Chain	1 bit	0x			8000	
Continents	3 bits	0x			7000	
Countries	4 bits	0x			0F00	
Hub	4 bits	0x			00FC	
SubNET	4 bits	0x			0003	
SubNET Address	16 bits	0x				FFFF

Figure 11. P2P Address system

## 3.3 Consensus Layer

Each local consensus group can independently generate a block, and in each round, the entire consensus group generates a block individually.

One round period is called Block Generation Interval (BGI) or Round Trip Time (RTT). One round minimum round period (BGI) is the minimum time required for the entire local consensus group to generate blocks in parallel and connect them in series. At this time, the minimum round period (BGI) of one round is affected according to the time taken for serial connection between the local consensus groups ( $t_1$ ) and the time between one round and the next round ( $t_2$ ).

The minimum delay time of  $t_1$  is 500 msecs. This is the minimum time to ensure that the block of one consensus group and all transaction contract information accordingly are replicated to the other consensus

us group. Therefore, if a block is created in 4 clusters, a BGI of at least 2 sec must be guaranteed. In other words, the block generation cycle of one cluster is 2 sec.

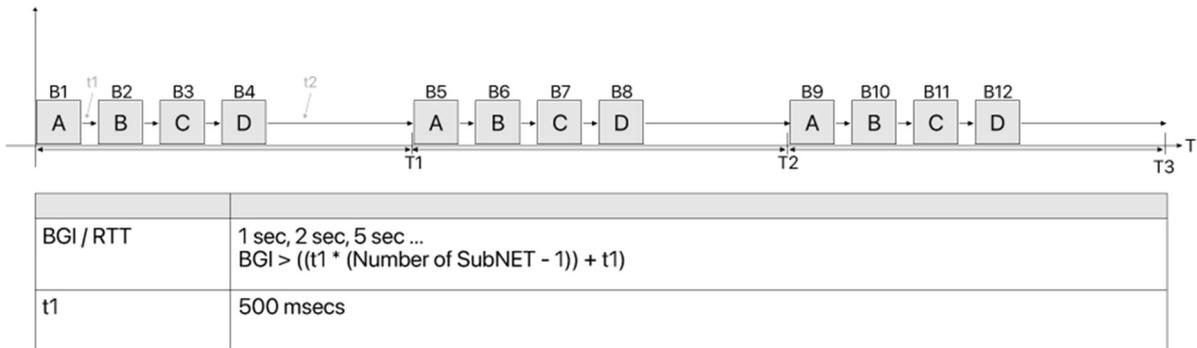


Figure 12. Block Generation Period Vs. Number of Blocks

### 3.4 Consensus Group Configuration Scenario

Each node in the system transmits hardware information, public key necessary for consensus, and various node information through ISA.

Send to The IS checks whether the node is a valid node based on information registered through DGOS and information collected during system startup. Based on this information, the IS creates a list for configuring the P2P main network and creates a list of P2P sub-networks within each P2P main network.

The IS transmits a P2P main network list and a P2P sub-network configuration list to each node according to the functions and roles of the nodes in the system. After receiving information for system operation, each node starts the system for each role. After confirming that all nodes are operating, the IS sends a block generation command to the P2P consensus group to generate a Genesis block. The P2P consensus group that has received the block generation command starts block generation.

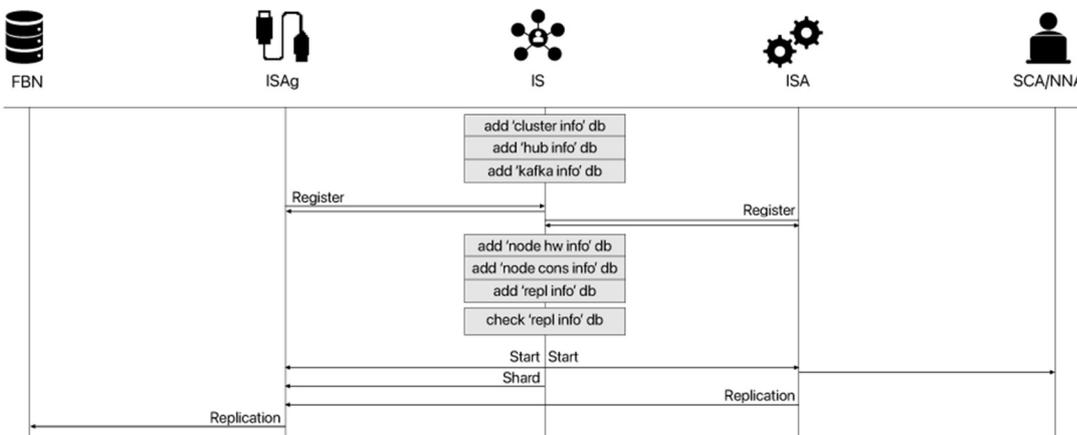


Figure 13. Control Sequence for Consensus Group Configuration

# 4. Wallet

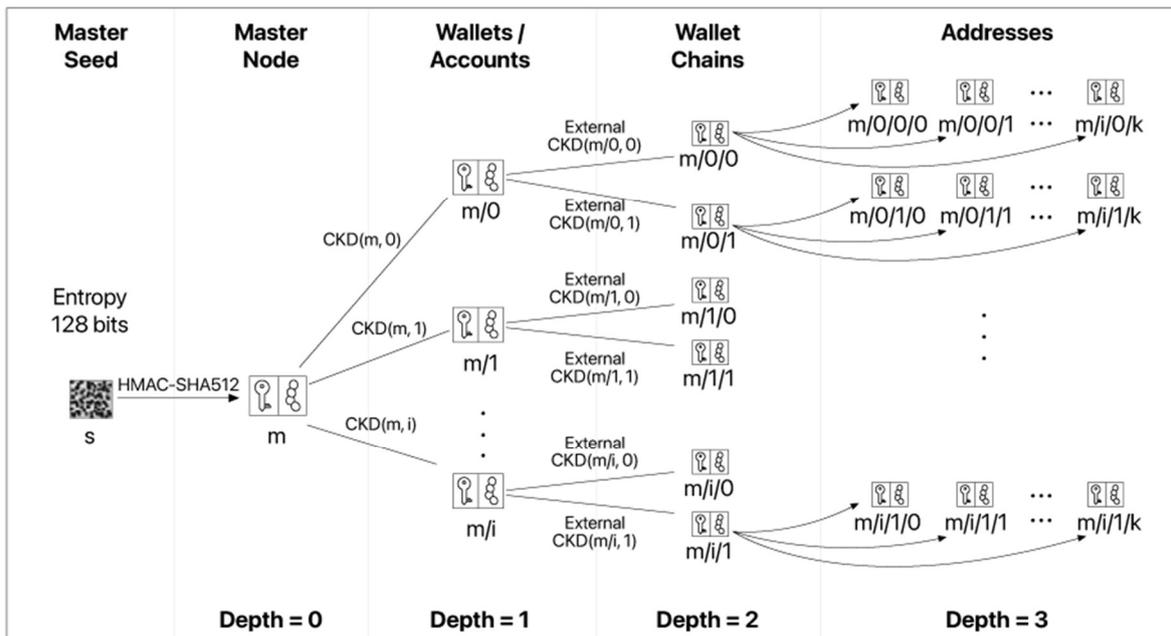
## 4.1 Key Generation

### 4.1.1 Concepts of Bitcoin Key Generation

[BIP 32]

BIP 32 is a Bitcoin standard that can construct Hierarchical Deterministic (HD) Wallets from Seed. HD Wallets can manage keys and wallet addresses in a hierarchical structure. The key standard uses secp25k1. Create a Master Private Key and Public Key with Master Seed, and use the Child Key Derivation (CKD) function with the Master Key to generate the key and address of the lower layer. In other words, if only the Master Seed value is kept, all the keys and addresses of the lower layer can be restored to their original state.

BIP 32 - Hierarchical Deterministic Wallets



Child Key Derivation Function ~  $CKD(x, n) = HMAC-SHA512(X_{Chain}, X_{PubKey} || n)$

Figure 14. HD Wallets

[BIP 39]

In order to effectively and intuitively manage this Master Seed, BIP39 has been proposed. BIP39 uses Mnemonic Code to replace Master Seed with natural language. Mnemonic Code consists of words in a total of 2048 languages of each country and is used as a seed to derive a Master Seed.

The number of words to be used as seeds for deriving Master Seed is determined according to Entropy.

Entropy (bits)	Checksum (bits)	Entropy + Checksum (bits)	Mnemonic (Word)
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

Table 15. Entropy Vs. Number of Word

### Mnemonic Words 128-bit entropy/12-word example

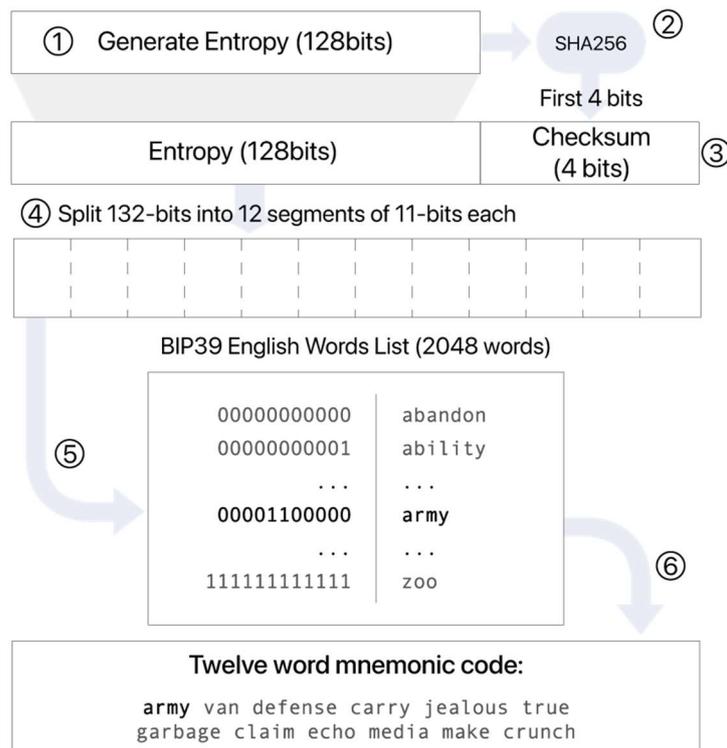


Figure 16. Mnemonic Code Generation

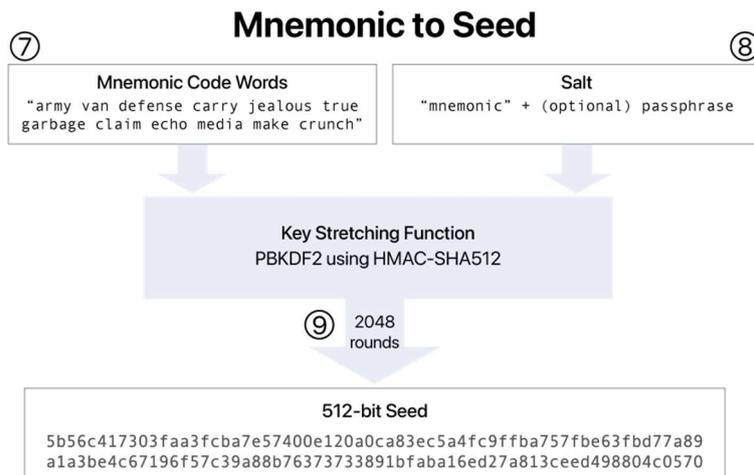


Figure 17. Mnemonic Code to Seed

[BIP 44]

BIP44 is a logical hierarchy for deterministic wallets based on the purpose scheme of BIP43 and the algorithm of BIP32. BIP44 consists of five predefined tree levels.

m / purpose' / coin\_type' / account' / change / address\_index

level-1: the purpose is fixed at 44

level-2: uses SLIP0044 ex. ETH: m/44'/60', BIT: m/44'/0'

level-3: account, The root of its own subtree

level-4: HD wallet has two subtrees(deposit address 0, change address 1)

level-5: usable address index, a child of level-4 derived from an HD wallet.

## 4.1.2 Concepts of FINL Key Generation

FIP39 provides two ways to create and restore wallets.

In both methods, you can create a wallet by preparing a single sentence and password.

For the first method, the password must be at least 10 bytes in length. Each sentence must have a minimum length of 20 bytes or more and a maximum length of 500 bytes or less. Each password and sentence are converted into Multi-bytes. At this time, in the case of a sentence, it is hashed.

The converted password and hash values are subjected to XOR processing using randomly generated 2b

ytes unsigned integer values. The output value obtained through XOR processing is hashed in PBKDF2, and the password is used as the password argument value, and the password is used as the salt argument input value. PBKDF2 is password and Hash the salt 2048 times to generate a seed value of 512 bits.

Optionally, the sentence part may use a mnemonic word of BIP39. However, because Hash Processing and XOR Processing are used internally, the value is different from the value of BIP39.

The second method provides the same method as BIP39. Mnemonic derived from entropy can be listed in one sentence. In this case, Passphrase is optional. Each sentence and passphrase are converted into Multi bytes.

FIP32 can be regarded as the same as actual BIP32. The Random Number and Master Node Keys values generated during the wallet creation process are provided as outputs. At this time, the Hash Left 256 bits value of the Master Node Keys is used as the Master Private Key, and the Master Public Key is derived. At this time, the key standard uses ed25519 instead of secp256k1 for signature. In addition, X25519 standard is used for encryption/decryption.

Password, Sentences 1, and Sentences 2 used as inputs when creating a wallet, and Random Number values provided as outputs are used when recovering the wallet. When recovering a wallet, after undergoing multi-bytes conversion and HASH concatenation as in the wallet creation step, the key value of the wallet can be recovered by using the random number provided by the wallet user instead of creating a new random number.

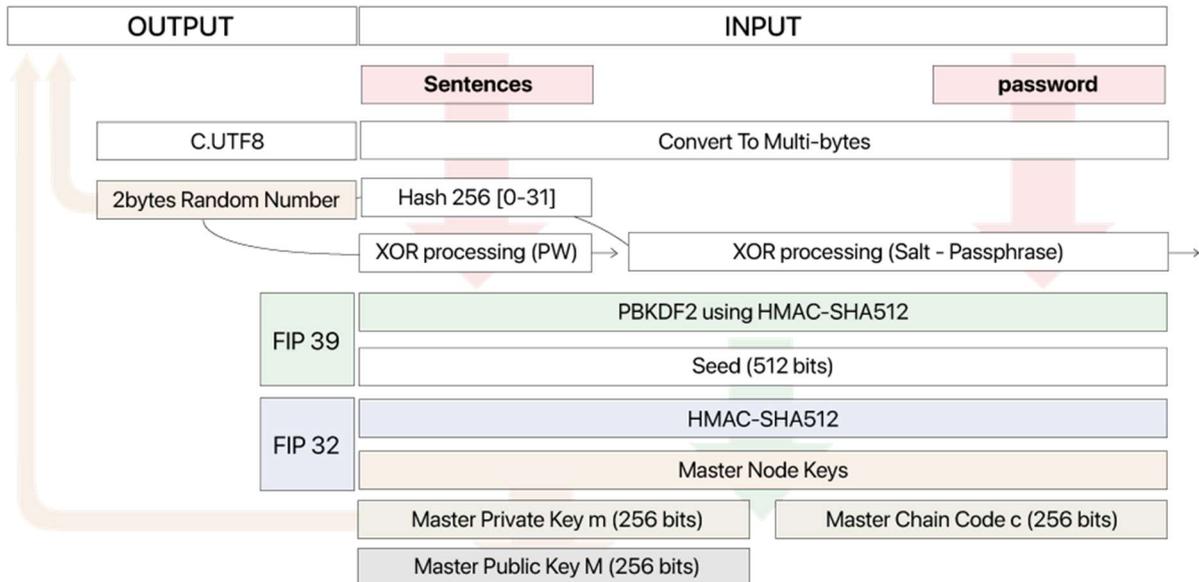


Figure 18. FIP 39 Method 1 & FIP 32

When creating a wallet, a random number is generated. This random number consists of 2 bytes and its

range is 0x1 FF or more and 0xFFFF or less. This Random Number and Password or Sentences 1 and XOR the first 2 bytes of the hashed value of Sentences 2. XOR the value derived in this way with the next 2 bytes of Password or Hashed value. In this way, XOR is performed up to the length of the password or hashed value. At this time, if the length of the password or hashed value is odd, 1 byte with a value of 0x 00 is added to the end of the sentence to make it even.

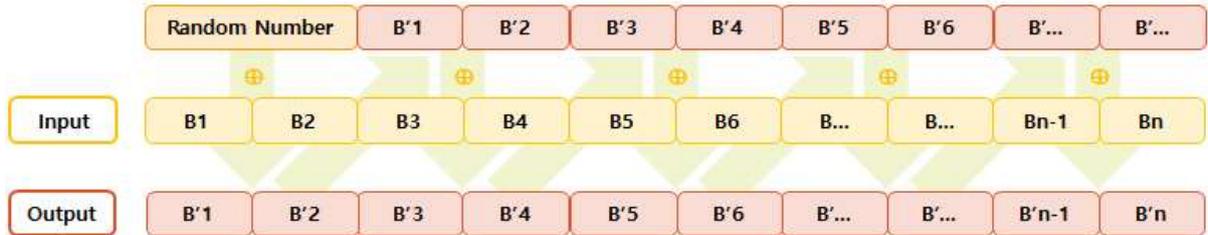


Figure 19. XOR Processing of Password and Salt used by PBKDF2 for FIP 39 Method 1

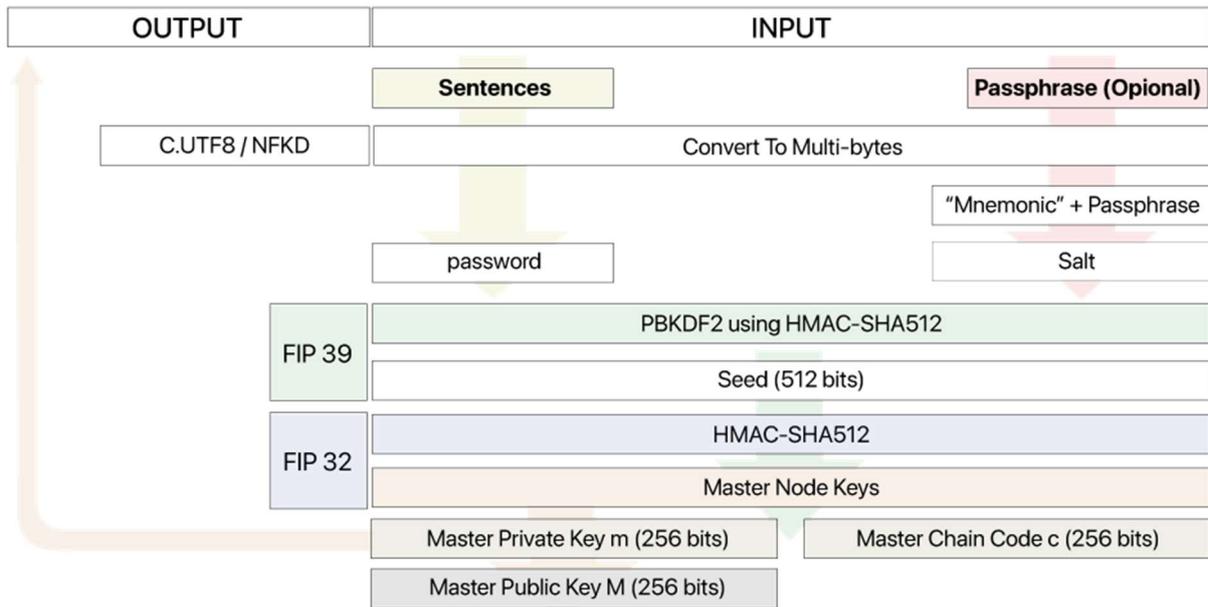


Figure 20. FIP 39 Method 2 & FIP 32

## 4.2 DSA Algorithm

This mainnet uses ED25519 by default. In addition, it supports secp256k1 used by most mainnets including Bitcoin, and supports secp256r1, which is officially recommended by NIST.

### 4.2.1 EdDSA (ED25519)

EdDSA generates signatures using variants of Schnorr signatures based on SHA-512 and Curve25519, a Twisted Edward Curve. EdDSA is documented in the RFC 8032 standard and has better performance than RSA or DSA.

The advantages of EdDSA are as follows.

- Fast single-signature verification
- Even faster batch verification
- Fast key generation
- High security level
- Foolproof session keys
- Collision resilience
- No secret array indices
- No secret branch conditions
- Small signatures
- Small keys

In addition, the advantages described in the RFC 8032 standard are as follows.

- It provides high performance on various platforms.
- The use of a unique random number for each signature is not required.
- Against side channel attacks, it can be better protected.
- Small public key size and signature size (In case of Ed25519, public key size is 32 bytes (256-bits), signature size is 64 bytes)
- For all points on the curve, the formulas are valid and no exceptions occur.
- Collision resilience (no has-function collision)

Protocol	Key length	Create AttrCert	Verify AttrCert
RSA	1024 bits	4.85 s	1.91 ms
RSA	2018 bits	24.06 s	8.33 ms
RSA	4096 bits	189.07 s	30.91 ms
DSA	512 bits	1.01 s	7.86 ms
DSA	1024 bits	1.34 s	10.36 ms
ED25519	256 bits	25.79 ms	29.34 ms

Figure 21. RSA vs. DSA vs. ED25519

This mainnet uses ED25519 by default, and the length of both the private key and public key is 256 bits. In the case of Public Key, 0x05 is defined and attached as a prefix by itself to confirm that it is ED25519.

[ Prefix (1 Byte : '0x05') + Public Key (32 Bytes) ]

## 4.2.2 ECDSA (Secp256k1 and Secp256r1)

This mainnet ECDSA is provided as an option. The length of the private key is 256 bits, and the compressed public key is used for the public key.

In case of public key, there is a standard prefix, and the contents are as follows.

[ Prefix 1 Byte ('0x02' or '0x03') + Compressed (32 Bytes) ] or Prefix 1 Byte ('0x04') + Uncompressed (64 Bytes) ]

## 4.3 Private Key encryption and decryption

The extension name of the standard key file is 'pem'. Among them, the private key is an important file and should not be exposed to anyone other than the creator. This private key file is encrypted with 'key seed' and the extension name is 'fin'. In this case, 'key seed' is the password created by the creator himself.

This encrypted private key file can be used by decrypting it with 'key seed'. Encrypted Private Key Files must be safely isolated from external factors such as hacking and stored.

## 4.4 Wallet Address

After encoding the public key value with base58, prefix is added. Prefixes are “FINL” for mainnet, “FINT” for testnet, and “FIND” for DEVnet.

# 5. Token

Tokens are largely divided into Platform Tokens and Utility Tokens. Platform Token is issued and used to construct and develop the Mainnet ecosystem. Utility Token can be used by anyone using this Mainnet. It can be freely published under the permission of DGOS.

Groups	Actions
Platform Token	0x00000000
Utility Token Platinum	0x00000001 ~ 0x0000FFFF
Utility Token Gold	0x00010000 ~ 0x000FFFFF
Utility Token	0x00100000 ~ 0x7FFFFFFF
Smart Contract Default	0x80000000 ~ 0x800000FF
Smart Contract Reserved 01	0x80000100 ~ 0x8000FFFF
Smart Contract General	0x80100000 ~ 0x801000FF
Smart Contract	0x80101000 ~ 0x9FFFFFFF
Smart Contract NFT	0xA0000000 ~ 0xBFFFFFFF
Smart Contract Reserved 02	0xC0000000 ~ 0xEFFFFFFF
Smart Contract Reserved 03	0xF0000000 ~ 0xF000FFFF
Notice	0xF0010000 ~ 0xFFFFFFFF

Figure 22. Token and Smart Contract

The unique number of Token and Smart Contract is designated as Action and is used when creating a token contract and transmitting tokens.

## 5.1 Platform Token

The unique action number of the platform token is 0x00000000, and the platform token is issued on the blockchain mainnet and cannot be used on other blockchain mainnets. Platform tokens are used throughout the blockchain ecosystem.

Block generation  
reward, transaction fee,  
network usage fee, storage usage  
fee, CPU and RAM usage fee, etc.

In addition, platform tokens can be used in various ways such as swap and staking in connection with ut

ility tokens issued in the blockchain ecosystem.

## 5.2 Utility Token

Utility tokens are divided into Platinum, Gold, and General tokens. Platinum tokens have values from 0x00000001 to 0x0000FFFF. Gold tokens range from 0x00010000 to 0x000FFFFF. Normal tokens have values from 0x00100000 to 0x7FFFFFFF.

Utility tokens are created in the form of smart contracts on various blockchain mainnets. The size and shape of utility tokens can be freely issued according to the type of product, but they can be used to purchase goods or services suitable for the purpose at the time the token is issued.

# 6. Smart Contract

## 6.1 Creation Rules of Account Number

Account Number consists of a total of 64 bits and is largely divided into Token Account and User Account.

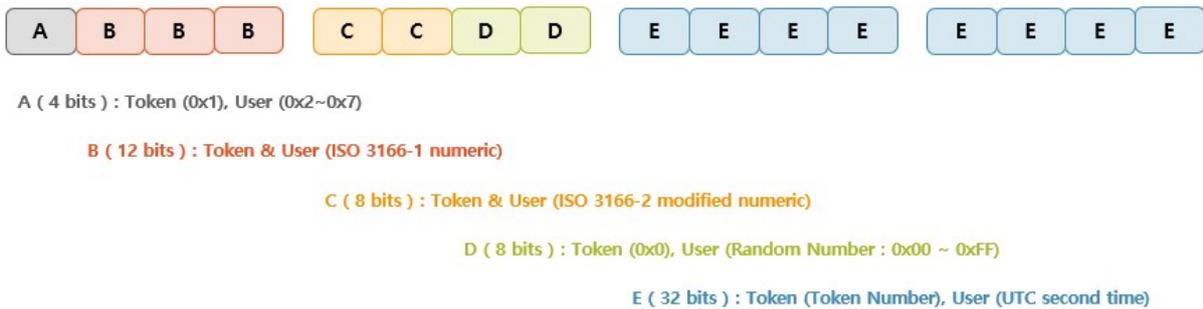


Figure 23. Account Number

### 6.1.1 User Account

User Account settings are largely divided into MSB 32 bits and LSB 32 bits.

In the case of MSB 32 bits, it is divided into values that can determine Token Account and User Account, country code, area code, and random number.

If the MSB 4 bits are between 0x2 and 0x7 for User Account, the next MSB 12 bits are set for each country according to the ISO 3166-1 numeric code, a country code table. The next MSB 8 bits is set as the area code according to the country previously set according to the local numeric code table that maps the ISO 3166-2 code, which is the area code table, to numbers. For the next MSB 8 bits, a random number is used. In this case, the range of the random number is 0x00 or more and 0xFF or less.

In the case of LSB 32 bits, it is set to UTC second time.

### 6.1.2 Token Account

Token Account settings are largely divided into MSB 32 bits and LSB 32 bits.

In the case of MSB 32 bits, it is divided into values that can determine Token Account and User Account, country code, area code, and random number.

If the MSB 4 bits are between 0x2 and 0x7 for User Account, the next MSB 12 bits are set for each country according to the ISO 3166-1 numeric code, a country code table. The following MSB 8 bits are regional codes according to the country set above according to the local numeric code table that maps the ISO 3166-2 code, which is an area code table, to numbers.

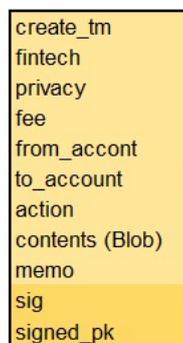
set to draw In the case of the next MSB 8 bits, it is set to 0x0.

In case of LSB 32 bits, Token Number is set to that value.

## 6.2 NFT

This mainnet supports non-fungible tokens through smart contracts.

## 6.3 Contract



create_tm
fintech
privacy
fee
from_accont
to_account
action
contents (Blob)
memo
sig
signed_pk

Figure 24. Contract Contents

### 6.3.1 Create Time

Indicates the time when the Contract was created in the past or when it will be executed in the future. UTC milliseconds are used.

## 6.3.2 Fintech

Indicates whether it is a financial transaction. The status value has true / false.

## 6.3.3 Privacy

Indicates whether the contract is public or private. The status value has true / false.

## 6.3.4 Fee

Network fee for contract transmission.

## 6.3.5 From Account

This is the account that sends the contract. Value is Default Account (0x00000000), Token Account, or User. It can be an Account.

## 6.3.6 To Account

This is the account that receives the contract. Value is Default Account (0x00000000), Token Account, or User. It can be an Account.

## 6.3.7 Action

It is a unique value for the actual operation. Token Action value, Action value according to contract type, and It is divided into Smart Contract Action values

## 6.3.8 Contents (Blob)

These are the detailed items included in the contract. Each contract has different details.

### 6.3.9 Memo

A contract can optionally make a memo about the contract.

### 6.3.10 Signature

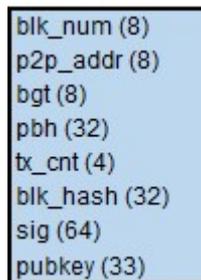
It is the signature of the contract sender.

### 6.3.11 Signed Public Key

It is a public key used as the wallet of the contract sender.

# 7. Block

## 7.1 Block Description



blk_num (8)
p2p_addr (8)
bgt (8)
pbh (32)
tx_cnt (4)
blk_hash (32)
sig (64)
pubkey (33)

Figure 25. Block Contents

Blocks are created in the consensus layer of the NNA. Block Fields are Block Number, NNA's P2P Address that creates Block, Block Generation Time (BGT), Previous Block Hash (PBH), Transaction Count, Block Hash, Signature, Public Key matching the Private Key used for Signature, and It consists of Block Confirm Time (BCT).

### 7.1.1 Block Number

Block Number starts with 1 in Genesis Block and increases by 1 for each Block.

### 7.1.2 P2P Address

This is the P2P address of the NN that creates the block.

### 7.1.3 Block Generation Time

Block Generation Time is the time when a block is created and is expressed in UTC milliseconds time.

## 7.1.4 Previous Block Hash

Previous Block Hash is the Block Hash of the previous block, and in the case of Genesis Block, Previous Block Hash Field is set to 0x0.

## 7.1.5 Transaction Count

It is the total of new transactions entered into the cluster when the block is created.

## 7.1.6 Block Hash

Block Number, P2P Address, Block Generation Time, Previous Block Hash, Transaction Count, and the value obtained by concatenating the XOR values of Transactions are input values to SHA256 and outputted.

## 7.1.7 Signature

Issue the signature for the previously derived Block Hash value. At this time, to generate the signature, the ed25519 private key of the NN that creates the block is used.

## 7.1.8 Public Key

It is NN's ed25519 public key to verify the signature for the previously issued Block Hash.

## 7.1.9 Block Confirm Time

Block Confirm Time is the time to confirm that the created block is irreversible.

## 7.2 Group XOR-Hash Cipher based on time arriving

The independently designed Group XOR-Hash Cipher based on time arriving algorithm is as follows, and it is planned to be supported in the future.

Bitcoin manages the transaction details included in the block in the form of a Merkle tree, and finally stores the Merkle Root in the block.

Ethereum manages the state machine using the Modified Merkle Patricia tree, and stores the state in the form of a key-value pair. Taking this one step further, we proposed a new format called Verkle tree.

It focuses on how to minimize the vectors required for verification.

This mainnet is a grouping method rather than a tree type of Group XOR-Hash Cipher on time arriving.

use the expression

Assuming that the maximum number of transactions in a group is 256, check the number of all transactions that arrived at the time the block is created and group them by 256 in order of arrival time. It is okay even if the number of transactions in the last group does not reach 256.

Hash each transaction by group.  $C[m]$  is derived by XORing the hashed values for each group. After hashing the  $C[m]$  values derived for each group, XOR them all to derive the C value. Hashing this C value becomes the root hash value.

At this time, it is assumed that the value of C is the same as the value obtained by XORing the value obtained by hashing a specific transaction  $M[k]$  and  $H(C[m])$  of the group and then XORing the unknown constant value,  $\alpha[k]$ . Through this assumption,  $\alpha[k]$  is the value obtained by XORing the values of  $M[k]$ ,  $H(C[m])$ , and C.

In other words, to verify a specific transaction  $M[k]$ , only the values of  $\alpha[k]$ ,  $H(C[m])$ , and  $H(M[k])$  need to be known. This is because the Root Hash can be checked at all nodes.

If a specific transaction is changed from the original value, the hash value is changed, so it is easy to check whether it is forged or tampered with.

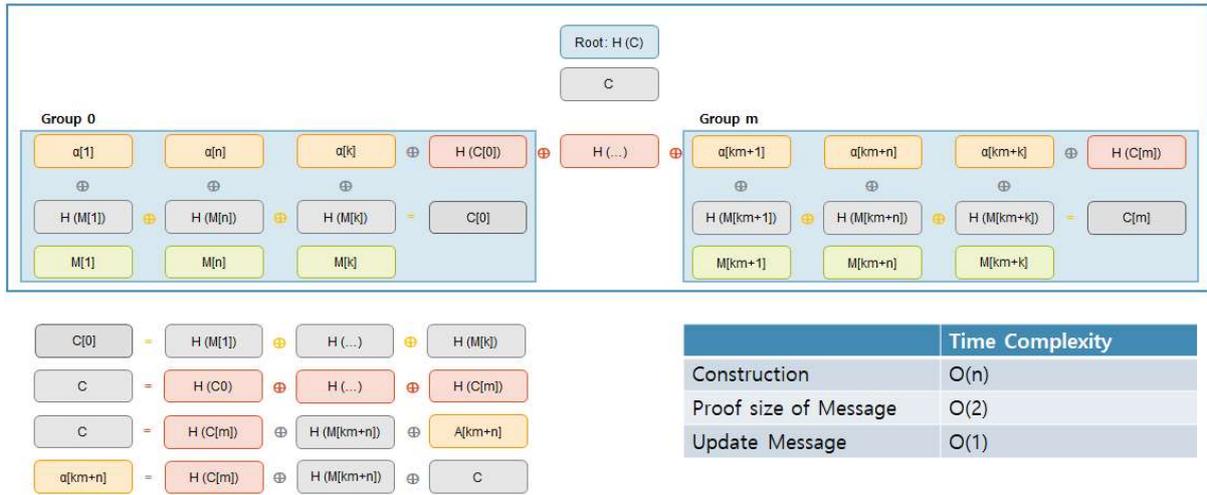


Figure 26. Group XOR-Hash Cipher on time arriving

# 8. Distribute Database

## 8.1 Replication

Replication takes place between cluster NNs in the data path, between NNs and ISAGs within a cluster, between NNs and FBNs within a cluster, and between FBNs and external FBNs. In addition, replication is performed between IS and ISAG of each cluster in the control path.

Synchronization is made with each other through replication between major nodes that make up the mainnet ecosystem, and As the number of FBNs increases, the ecosystem expands and becomes richer.

## 8.2 Shard

ISAG can shard is as an option.

## 8.3 Query

You can request information from FBN to check the data of the mainnet's Data Path. Control Path information is requested from ISAG.

# 9. Governance

## 9.1 Currency and Issuance

[unit of currency]

The platform coin, Fin, consists of 9 decimal places. For each three decimal places, the name of each unit is zoned, redo.

1 kitty = 0.001 Fin

1 goofy = 0.000001 Fin

1 micky = 0.000000001 Fin

[Currency Issuance]

The basic supply for each year is 60 million, and based on the inflation standard supply of 600 million for each group by year. Inflation is generated by applying an inflation weight to The authority for issuance by year and group is It is at the DGOS altar.

### Weighted filter formula

$$Z_k = A \cdot \exp(-B \cdot i) + C$$

The basic inflation rate (IR) for each number of nodes is 8%, and the inflation weight ( $Z_k$ ) for each node is applied to A 86.43% to 167.19% inflation weight ( $N_k$ ) is given. Accordingly, from node 1 to node 70, It has an initial inflation rate of 6.91% ( $8\% \cdot 86.43\%$ ) and 13.38% from the 911 node to the 980 node. It has an initial inflation rate of ( $8\% \cdot 167.19\%$ ).

### Inflation rate by number of nodes ( $N_k$ )

$$\text{Basic Inflation Rate (IR)} = 8\%$$

$$\begin{aligned} \text{Inflation weight per node } (Z_k) &= A \cdot \exp(-B \cdot i) + C \\ ; \text{ where } A &= 138.06306475, B = 0.06764022, C = 29.12741104, \\ &1 \leq k < 14, \text{ and } 13 \geq i \geq 0 \end{aligned}$$

$$N_k = IR \cdot Z_k$$

<i>k</i>	<i>n (number of nodes)</i>	<i>i</i>
1	1 ≤ <i>n</i> ≤ 70	13
2	71 ≤ <i>n</i> ≤ 140	12
3	141 ≤ <i>n</i> ≤ 210	11
4	211 ≤ <i>n</i> ≤ 280	10
5	281 ≤ <i>n</i> ≤ 350	9
6	351 ≤ <i>n</i> ≤ 420	8
7	421 ≤ <i>n</i> ≤ 490	7
8	491 ≤ <i>n</i> ≤ 560	6
9	561 ≤ <i>n</i> ≤ 630	5
10	631 ≤ <i>n</i> ≤ 700	4
11	701 ≤ <i>n</i> ≤ 770	3
12	771 ≤ <i>n</i> ≤ 840	2
13	841 ≤ <i>n</i> ≤ 910	1
14	911 ≤ <i>n</i> ≤ 980	0

Table 1. *n vs i* (inflation weight by number of nodes)

The annual inflation weight ( $Z_y$ ) is 90%, and in the first year, the initial inflation rate by number of node  $s$  ( $IR_{y=1}$ ) of 100% ( $Z_{y=1}$ ) is supplied as inflation. Thereafter, the inflation rate decreases by 90% each year. i.e. 1 year. Primary inflation rate ( $IR_{y=1}$ ) is 100% of the initial inflation rate for each group, the second year inflation rate ( $IR_{y=2}$ ) is the group 90% of initial inflation rate (100%\*90%), third year inflation rate ( $IR_{y=3}$ ) is the initial inflation by group. Decrease to 81% of the rate (100%\*90%\*90%) and the 12-year inflation rate ( $IR_{y=12}$ ) is the initial inflation by group reduced to 28% of the rate.

### Yearly Inflation Rate ( $IR_y$ )

$$Z_{y=1} = 100\% \text{ and } Z_{y \geq 2} = Z_{y-1} * 90\%$$

$$IR_y = Z_{y=1} * N_k; \text{ where } y = 1,$$

$$IR_y = Z_y * N_k; \text{ where } y \geq 2$$

Accordingly, the total supply by Year 12 fluctuates from approximately 1 billion to 1.27 billion depending on the number of nodes.

Inflation-based supply	600000000	inflation	6.91%	6.22%	5.60%	4.98%	4.43%	3.94%	3.53%	3.11%	2.77%	2.49%	2.21%	1.94%	Nodes (70)
Base supply by year	600000000	8.00%	13.38%	12.04%	10.83%	9.63%	8.56%	7.62%	6.82%	6.02%	5.35%	4.82%	4.28%	3.75%	Nodes (980)
inflationary increase	90.00%	100.00%	90.00%	81.000%	72.000%	64.000%	57.000%	51.000%	45.000%	40.000%	36.000%	32.000%	28.000%		
	1	2	3	4	5	6	7	8	9	10	11	12	Year		
Nodes	inflationary increase	Supply	Supply	Supply	Supply	Supply	Total Supply								
70	86.43%	6.91%	101487553	97338798	93604918	89871038	86552034	83647905	81158652	78669399	76595021	74935519	73276017	71616515	1008753369
140	90.44%	7.24%	103412459	99071213	95164091	91256970	87783973	84745101	82140354	79535606	77364983	75628485	73891986	72155488	1022150709
210	94.73%	7.58%	105472069	100924862	96832376	92839890	8919079	8519079	83190755	80462431	78188827	76369945	74551062	72732179	1036485599
280	99.32%	7.95%	107675812	102908231	98617408	94326585	90512520	87175213	84314664	81454115	79070325	77163292	75256260	73349227	1051823652
350	104.24%	8.34%	110033773	105030396	100527356	96024317	92021615	88519251	85517224	82515198	80013509	78012158	76010807	74009456	1068235060
420	109.49%	8.76%	112556746	107301071	102570964	97840857	93636317	8957345	86803940	83650535	81022698	78920428	76818158	74715888	1085794947
490	115.12%	9.21%	115256276	109730648	104757583	99784519	95364016	91496077	88180701	84865324	82102510	79892259	77682008	75471757	1104587250
560	121.13%	9.69%	118144720	112330248	107097223	101864199	97212621	93142490	89653807	86165124	83257888	80932099	78606310	76280521	1124687250
630	127.57%	10.21%	121235299	115111769	109600592	104089415	99190591	94904120	91230002	87555884	84494119	82044707	79595295	77145883	1146197676
700	134.46%	10.76%	124542157	118087941	112279147	10647053	101306980	96789029	92916500	89043970	85816862	83235176	80652490	78071804	1169213409
770	141.83%	11.35%	128080430	121272387	115145148	109017909	103571475	98805945	94721019	90636193	87232172	84508954	81785737	79062520	1193839789
840	149.72%	11.98%	131866312	124679681	118311713	111743744	105994439	100963798	96651819	92339840	88746524	85871872	82997219	80122567	1220189528
910	158.16%	12.65%	135917131	128325418	121492876	114660334	108586964	103272765	98717737	94162709	90366852	87330167	84293482	81256796	1248383231
980	167.19%	13.38%	140251428	132226285	125003655	117781028	111360914	105743314	100928228	96113142	92100571	88890514	85680457	82470399	1278549936

Figure 27. supply by group/year

## 9.2 Reward Policy

The currency supplied by year is diverse for the expansion of the Final Chain ecosystem, reserve, team, marketing, sales, etc. used. In addition, a certain amount may be incinerated at the discretion of the DG OS Altar to maintain its value.

## 9.3 Fee Policy

Fees are used for deflationary purposes. In other words, all fees are incinerated. Ecosystem members must submit a fee each time a contract is generated.

[Transaction model]

Transactions consume network resources. Costs are incurred to consume these network resources. In the case of network resources, since they are public goods, the minimum cost is reasonable in actual use. In order to minimize these costs, 5,000 network points are paid per day at the time a transaction occurs. After one day, the remaining network points cannot be used and a new 5,000 network points will be paid. A day is from 00:00:00 to 24:00:00 UTC time. 1 network point equals 1 goofy. However, this network point cannot be changed to a token.

By staking Fins, the cost of network resource consumption can also be reduced. If Fins are staked for 3 days, 10 additional network points can be used per 1 fin per day after 3 days (72 hours) have elapsed. This network point cannot be changed with a token (Fin). After one day, the existing network points cannot be used and are recharged at 10 network points per Fin.

In order to consume network resources, network points can also be purchased using Fin. At this time, 1 Fin = 1,000,000 network points are charged, and all used Fins are burned. Charged network points cannot be changed to tokens.

The transaction fee rate can be obtained according to the first byte unit amount. The unit of Fee is goofy, and it is expressed as 1 Byte = 10 goofy. For example, if a transaction with a size of 100 bytes is generated, the fee will be 1,000 goofy.

In addition, when a transaction between clusters occurs, the fee rate is weighted. Each cluster receives user registrations according to local distances. Transactions that occur between clusters have strong long-distance transactions.

When a transaction occurs, check whether there are enough network points to transmit the transaction according to the following priorities.

- 1) Check 5,000 network points that are automatically recharged per day.
- 2) Check the network points generated by staking.
- 3) Check if there are network points purchased from the transaction originating wallet.

If all are not enough, the transaction transfer fails.

[Contract model]

The contract consumes CPU and Memory of mainnet participating nodes. In the case of CPU, resource consumption occurs whenever a contract is executed, and in the case of memory, consumption occurs when a contract is saved. CPU and In the case of memory, since it is a consumer product, it must be handled separately from transactions. The fee for such consumption is expressed as Power. 1 Power equals 10 goofy. CPU consumption is expressed as Execution Power, and Memory consumption is expressed as Stack Power.

Power is not automatically recharged like network points and must be purchased using Fin. After purchasing 100,000 Power with 1 Fin, you can use it immediately. Power can be converted into tokens (Fin) as needed. However, the conversion from Power to Fin takes 3 days (72 hours).

[Memory consumption]

Since the contract once stored in the memory must be stored permanently, the corresponding fee must also be quite large. This fee is incinerated in its entirety and is a fixed value as shown below.

1. Token Generation: 1000 Fin (100,000,000 Stack Power) 2. Account Creation: 1 Fin (Activation Fee, 100,000 Stack Power) - Minimum Maintenance Cost

[CPU consumption]

Each time the contract is executed, 100 Execution Power is used. At this time, all the power used is incinerated.

# 10. Inter-operability

Inter-operability will be supported in the future.

# 11. Reference

<https://bitcoin.org>

<https://ethereum.org>

<https://eos.io>

<http://wiki.hash.kr/index.php/BIP39>

[https://ko.wikipedia.org/wiki/ISO\\_3166-1](https://ko.wikipedia.org/wiki/ISO_3166-1)

[https://en.wikipedia.org/wiki/ISO\\_3166-2](https://en.wikipedia.org/wiki/ISO_3166-2)

<https://www.ip2location.com/free/iso3166-2>