

한 번에 끝내는 블록체인 개발 A to Z

Chapter 2

Blockchain 2.0 - Ethereum

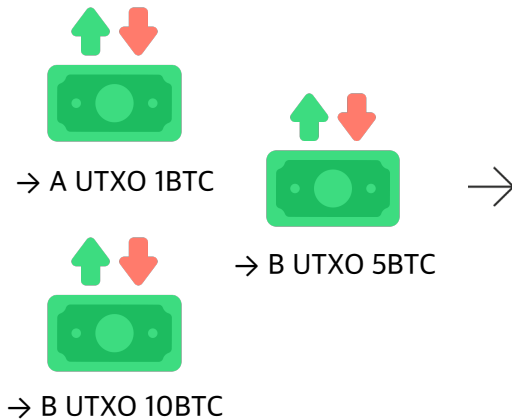
Chapter 2

Blockchain 2.0 - Ethereum

Account Structure

State 기반 Account

- Bitcoin 블록체인의 경우 UTXO 기반으로 상태값이 관리된다. 블록이 새로 생길 때마다 UTXO 사용여부를 업데이트 하는 식으로 사용자의 잔액이 관리된다.
- Ethereum 블록체인의 경우 State 기반으로 Account(계정)에 잔액, 데이터 상태값이 저장되어 블록이 생길 때 마다 잔액을 업데이트 하는 방식이다.



Bitcoin(UTXO)



Balance : 10ETH

A->B : 5ETH

A->B : 10ETH

Ethereum(State)

Account 종류

- Etheruem의 경우 사용자가 직접 관리하는 EOA(External Owned Account)와 Smart Contract 코드를 네트워크에 배포시에 생기는 CA(Contract Account) 두 가지 종류가 존재한다.

EOA(External Owned Account)

- EOA는 Private Key를 소유한 사람이 관리하는 방식으로 Bitcoin의 PKI와 동일한 방식으로 관리된다.
- Eth 전송, Contract 호출 등 다양한 활동을 할 수 있다.
- 계정 생성에 비용이 발생하지 않는다.

CA(Contract Account)

- CA는 생성한 사용자의 정보에 의해서 네트워크에서 Account를 생성한다.
- 사용자의 호출에 따라 State 변경, 함수 호출, ETH 전송 등 다양한 활동을 할 수 있다.
- 계정 생성에 코드의 크기에 따른 비용이 발생한다. 이 비용은 생성자가 지불한다.

Account의 state는 어떻게 저장될까?

nonce

계정에서 전송한 Transaction의 수를 기록한다. 0부터 시작하여 1회 전송 시 마다 +1 씩 증가한다.

balance

계정의 잔고(잔액)을 표시한다. Wei단위로 표기하며, $1e+18$ wei 가 1ETH 이다.
($1e+9 = 1$ gwei)
 $1e+12 = 1$ Twei(Szabo)
 $1e+15 = 1$ Pwei(finney))

codeHash

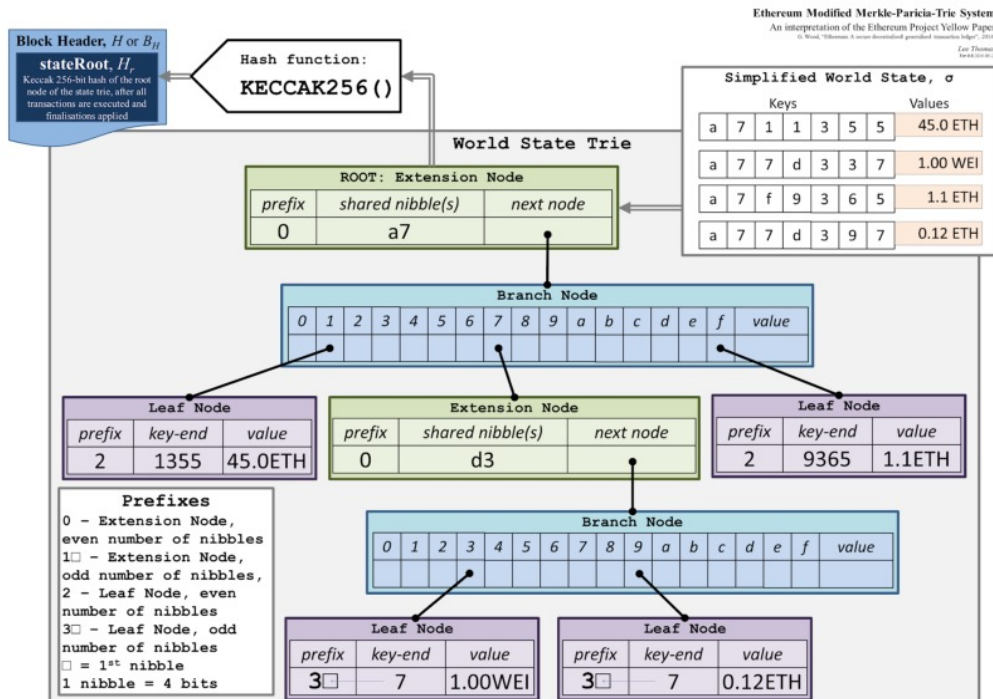
CA만 가진 데이터이며, EOA는 빈 공간으로 가지고 있다. Contract Code의 Hash 정보가 들어가게 된다. EVM code와 같이 사용된다.

storageRoot

계정 Merkle Patricia Trie의 root node의 Hash 값이다. 기본값은 비어있는 상태이다. Account Storage와 같이 사용된다.

Patricia Merkle Tree

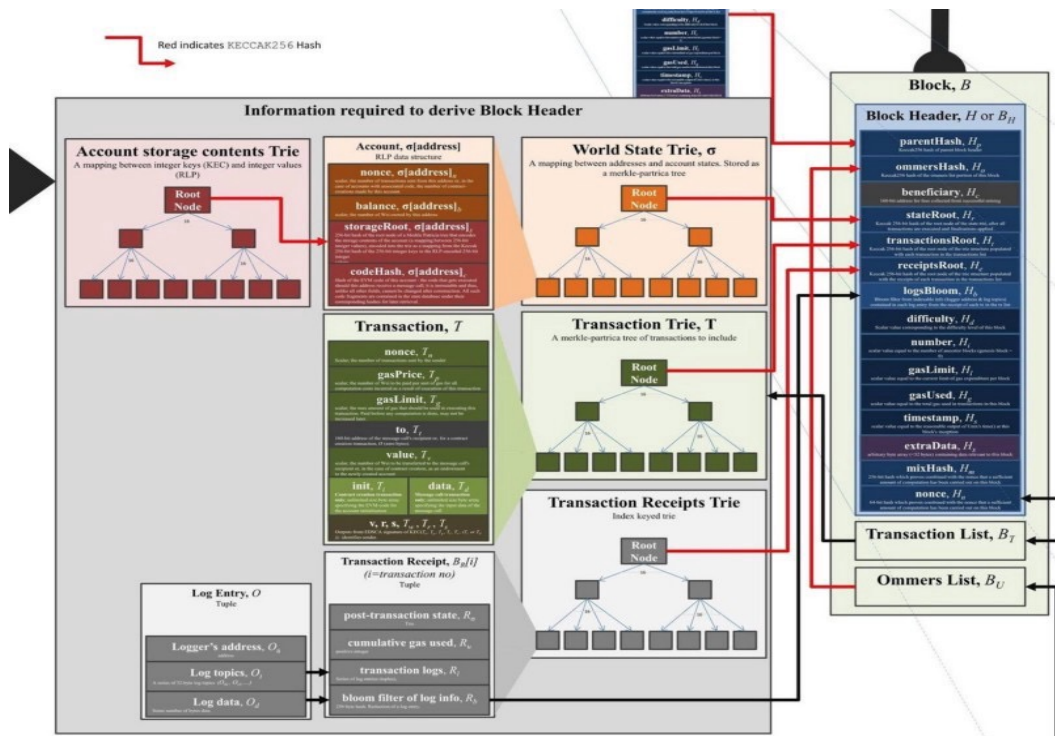
- Ethereum은 Key-Value 기반의 DB(golang-ethereum은 leveldb)를 사용하고 있다.
- Address를 Hash한 값을 Key로 하여 오른쪽 그림과 같이 Address 주소를 Path로 하여 Account의 State 값을 알 수 있다.



(출처 : an interpretation of the ethereum project yellow paper)

Ethereum의 tries

- Ethereum에서 사용하고 있는 Trie는 Storage Trie, State Trie, Transaction Trie, Receipts Trie 총 4개의 Trie가 존재한다.
- 이 중 Block Header에 Root Hash값이 저장되는 것은 State Trie, Transaction Trie, Receipts Trie 총 3 가지이다.



(출처 : an interpretation of the ethereum project yellow paper, Lee Tomas)

EOA 생성

Private Key

[illegible]

Key Conversion(secp256k1)

Public Key

0479be667ef9dcbba55a06295ce870b07029bfcdb2dce28d959f2815b16f81798483a
da7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

Keccak-256 Hash
(결과값 하위 20byte)

0x7E5F4552091A69125d5DfCb7b8C2659029395Bdf

CA 생성

① Network 배포

EOA가 CA를 생성하거나 Contract 상에서 CREATE를 사용하는 경우 네트워크에서 아래 코드와 같은 방식으로 Address를 생성하여 돌려준다.

```
def mk_contract_address(sender, nonce):  
    return sha3(rlp.encode([normalize_address(sender), nonce]))[12:]
```

② CREATE2 이용

CREATE2 를 이용하면 네트워크 배포 전에 미리 Contract Address 미리 알고 네트워크 배포 할 수 있다. CREATE 또한 CREATE 문과 마찬가지로 Contract 상에서 실행하는 함수이다.

```
keccak256( 0xff ++ senderAddress ++ salt ++ keccak256(init_code))[12:]
```

Address Checksum

- Ethereum Address의 경우 Bitcoin과 달리 생성 과정에 Checksum이 들어가지 않기 때문에 정확한 주소인지 확인하기 어렵다. 이를 해결하기 위해 EIP-55를 통해 Address의 대소문자 변환을 통해 Checksum 기능을 제공한다.

```
# Treat the hex address as ascii/utf-8 for keccak256 hashing
hashed_address = eth_utils.keccak(text=hex_addr).hex()

# Iterate over each character in the hex address
for nibble_index, character in enumerate(hex_addr):

    if character in "0123456789":
        # We can't upper-case the decimal digits
        checksummed_buffer += character
    elif character in "abcdef":
        # Check if the corresponding hex digit (nibble) in the hash is 8 or higher
        hashed_address_nibble = int(hashed_address[nibble_index], 16)
        if hashed_address_nibble > 7:
            checksummed_buffer += character.upper()
        else:
            checksummed_buffer += character
    else:
        raise eth_utils.ValidationError(
            f"Unrecognized hex character {character!r} at position {nibble_index}"
        )

return "0x" + checksummed_buffer
```

(출처 : <https://github.com/ethereum/EIPs/blob/master/EIPS/eip-55.md>)






ENS

- Ethereum Naming Service(ENS)는 DNS와 같이 사용자가 이해하기 쉬운 언어로 등록하여 사용하는 것이다. Smart Contract로 해당 명칭을 사서 구매하면 지원을 하는 서비스(Wallet)등에서는 해당 명칭을 수신인에 입력하게 되면 자동으로 변환되어 해당 주소로 전달되는 기능이다.

Result for **vitalik.eth**

[Ethereum Name Lookup](#) / Search Results

Overview

Resolved Address:	 0xd8da6bf26964af9d7eed9e03e53415d37aa96045 
Expiration Date:	 2032.05.03 at 21:05
Registrant:	0xd8da6bf26964af9d7eed9e03e53415d37aa96045 Lookup names 
Controller:	0xd8da6bf26964af9d7eed9e03e53415d37aa96045 Lookup names 
Token ID:	79233663829379634837589865448569342784712482819484549289560981379859480642508
Content:	QmbKu58pyq3WRgWNDv9Zat39QzB7jpzgZ2iSzaXjwas4MB
Text Records:	url https://vitalik.ca
	avatar eip155:1/erc1155:0xb32979486938aa9694bfc898f35dbed459f44424/10063

[Click to see less](#) 

(출처 : etherscan.io)