Blockchain Sturcture

Ch 1 : About Bitcoin Network

비트코인 네트워크의 개략적인 흐름을 살펴본다

비트코인의 탄생

2008.10) 나가모토 사토시 – 탈 중앙화 화폐 시스템 논문

주요 내용

중앙 통제 시스템이 없는 **탈중앙화 P2P 네트워크**

(전에 해결 못한) **이중지급 문제 해결**

거래 원장 (블록체인) 이 모든 참여자(노드, 코인 사용자)에게 공개

참여자들이 규칙에 맞게 작성됐는지 **검증 가능**

블록체인 데이터를 신뢰할 수 있는 **합의 시스템(PoW)**이 있다

이중지급: 계좌 잔액이 100원인데 이를 동시에 보낼 시 어디서 관리하는지 문제 (검증할 주체 없음 – 조작 위험성)

암호화폐 역사

1980년대 : 사이퍼펑크 중심으로 연구 해 왔다

David Chaum : Ecash 소개 (1984) , 전자화폐 발행 & 관리 주체 Digicash 설립 (1990)
- 은닉 서명(Blind Signature) 및 비밀키 공유(Secret Sharing – 이중지급 방지) 기능 존재

Cynthia Dwork & Moni Naor (1992) : 스펨메일 & DoS 공격 방지 수학적 퍼즐 풀이 제안
- 1997 Adam Back은 이를 기반으로 HashCash 제안

Wei Dai (1998) : B-Money 시스템 제안 (수학적 퍼즐 → 암호화폐 생성 → 푼 사람에게 보상) *비트코인 기초 아이디어

Nick Szabo (2005) : BitGold 제안 → 가장 유사했지만 실현 못함
Hal Finney → 종합한 암호화폐 시스템 만들었지만 탈중앙화는 실패

2009.01.03 : 나카모토 사토시 논문 토대로 50BTC Genesis Block 생성

기존에도 여러 시도가 있었지만, P2P 시스템이 기본적으로

- **데이터의 일관성 (Consistency)**
- **네트워크 가용성 (Availability)**
- **분할내성 (Partition Tolerance)**

을 동시에 만족할 방안이 없어서 비트코인 전에는 탈중앙화 시스템 구현이 힘들었다

암호화폐 주요 이슈 - 문제점(Problem)

해결되야 할 문제점

- 내가 받은 암호화폐가 **진본인지 위조본인지 확인**할 수 있는가?
- 내가 받은 암호화폐가 **이중으로 지급된 것인지 아닌지 확인**할 수 있는가?
- 내가 보유한 암호화폐 **소유권을 다른 사람들도 인정**할 수 있는가?

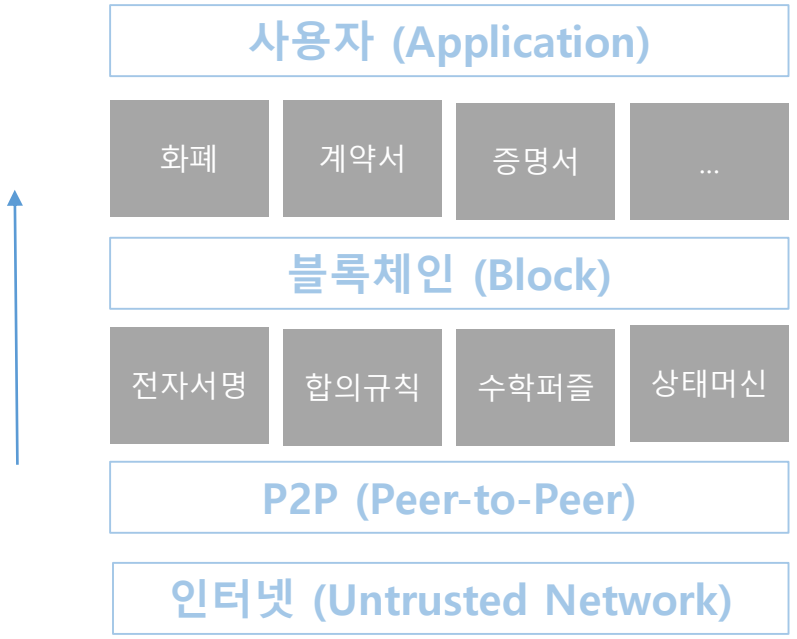
기존에는 중앙 시트템이 보증을 해줌 (단, 여기 공격 당할 시 취약)

암호화폐 주요 이슈 - 해결책(Solution)

Ecash schemes	P2P에서 자신이 암호화폐 소유자임을 증명하고, 다른 사람들이 검증하기 위해 전자 서명 (Digital Signature) 사용
State Machine Relication	노드 간 데이터 일관성 보장. 블록체인 상태(State) – 시간에 따라 변함 (Transaction). 모든 노드에서 동일하게 관리 필요
Consensus Mechanism	개별 거래 원장부터 블록체인 상태까지 모든 과정에 대해 동의한 합의 규칙. 이로 인해 전체 상태가 일관되게 유지된다
HashCash computational Puzzle	규칙에 따라 수학적 퍼즐 풀고, 이긴 채굴자가 블록 생성 (거래 승인) 권한 받음. 이 과정서 암호화폐 생성되고, 일관성이 유지된다
Peer-to-Peer	서버–클라이언트 구조가 아닌 P2P 구조로 탈중앙화 (Decentralization)를 이룸

블록체인 기술의 의의

안전하지 않은 인터넷에, **중재자 없이 가치를 자유롭게 교환**할 수 있는 기술



인터넷을 통해 디지털 정보 교환은 편리하지만, 화폐, 증권, 계약서를 안전하게 교환은 어렵다.
따라서 은행과 증권사 같은 중재를 통했다. 블록체인 시스템은 이들을 필요로 하지 않는다.
비트코인인 1세대 블록체인은 화폐만 교환 가능하다.
이더리움은 계약서나 증명서 등 스마트 컨트렉트 기능이 추가된 2세대 블록체인이다.

블록체인 기술의 의의

화폐, 계약서, 증명서 등 **디지털 대상의 다양화**

인터넷 상에서도 모든 대상을 안전하게 거래 (부동산, 계약서, 지적 재산권 등)

사용자 증가에 따른 **새로운 가치 창출**

신뢰와 보안이 중요한 4차 산업혁명 시대 기술
디지털 가치 교환의 Starting Point

인터넷 상에 교환하면서 새로운 가치 창출될 것
신뢰성의 담보는 곳 자유롭게 서비스를 탄생시키며, 우리 삶을 전환시킬 것

블록체인이 가져올 비즈니스 혁신

블록체인 특징 | 탈중앙화 | 투명성 | 익명성 | 불변성

제 3자의 개입 없이 구성원들만 협업으로 작동하는 더 안전하고 자유로운 인터넷

블록체인의 비즈니스 활용

금융산업에서 제일 관심 갖고 있다 (거래절차 간소화, 보안성 증대)
제조업에서도 유통정보 관리로 주목
공공서비스도 각종 정보의 보안과 투명성으로 인해 도입 검토

스마트 컨트랙트와 비즈니스의 결합

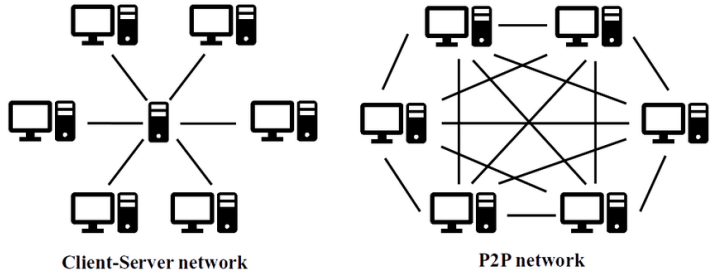
이더리움 Smart Contract → 거래 자동화 시스템 구축 (위조, 불이행 방지)
계약이 자동으로 실행되어 신뢰성 보장
비즈니스, 지능형 로봇이나 자율주행차 등에 도입되어 여러 산업에 사용 가능

P2P 네트워크

중앙집중시스템 없이 각 노드가 동등한 계층에서 통신하는 네트워크

서버 – 클라이언트 네트워크

서버가 전체 네트워크 관리, 통제, 유지 → 고객들에게 서비스 제공
원장 관리 쉽고, 업그레이드 편리하지만 안정성 보장 어려움 (서버 공격으로 모두 Down) – 유지비용 큼



P2P 네트워크

서로가 서비스하면서 자율적으로 운영. 데이터가 분산돼 있다
통제/업그레이드 어렵지만, 중앙시스템 문제가 없어서 보안 강하고 복원 용이하다

비트코인 P2P와 차별 – 가치부여 대상 교환 여부 (신뢰성 존재)
원장관리, 거래 승인, 거래내역 검증 등 엄무를 각 노드들이 합의 규칙(Consensus)에 따라 수행한다
데이터 불일치 시 다수결에 따라 많은 노드 가진 데이터 기준으로 복귀
50% 이상이 정상이면 일관성(Consistency)이 유지된다

비트코인 네트워크의 구성원

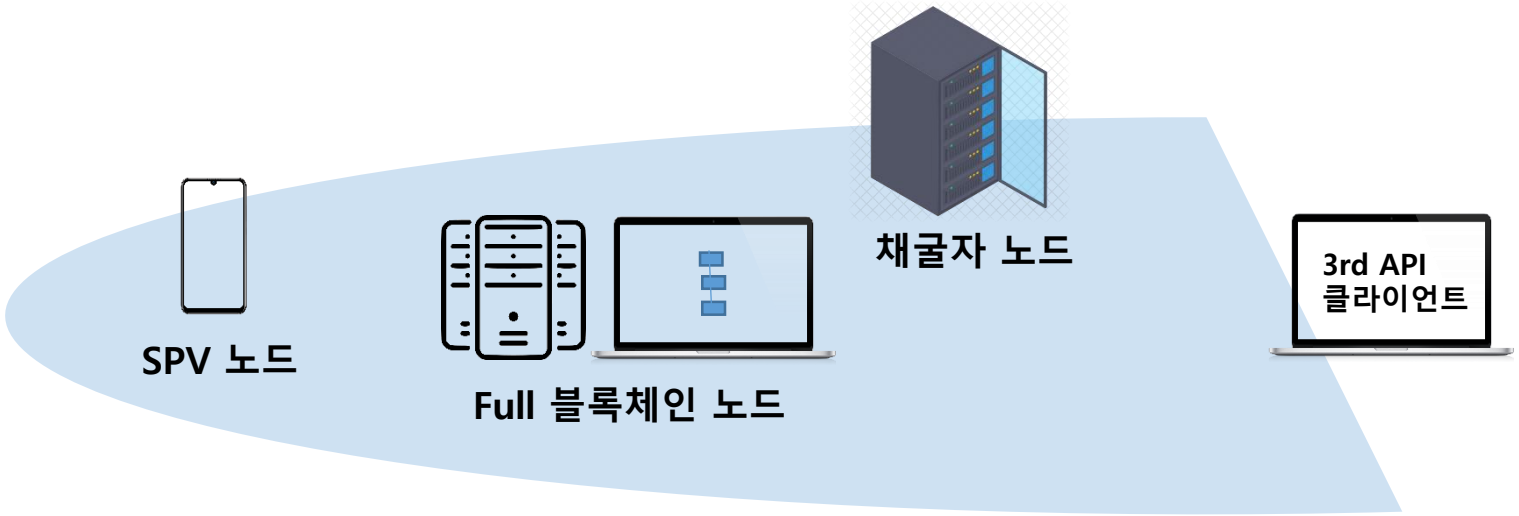
합의 규칙만 따른다면 누구나 비트코인 네트워크에 참여 가능하다

기본 구성원

1. Full 블록체인 노드
2. Lightweight 노드 (Simplified Payment Verification)
3. 채굴자(miner) 노드

추가 구성원

+ 3rd Party API Client (안전성 문제로 기본 구성원은 아님) – 실습에서 활용 예정



비트코인 네트워크의 구성원

풀 노드 & 채굴자 : 중 대형 컴퓨터 사용
SPV : 스마트 기기
서드파티 API 클라이언트: 일반 컴퓨터나 웹

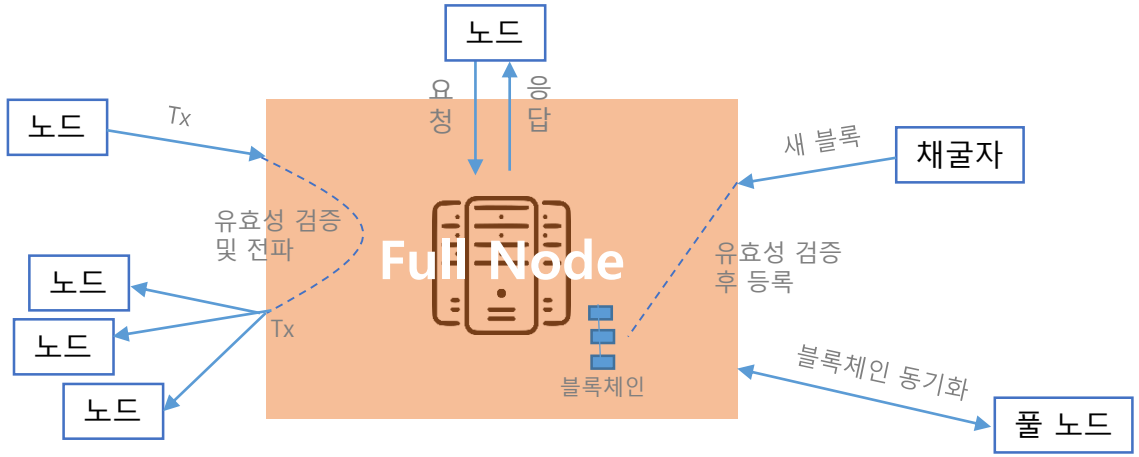
모든 노드에서 기본적 수행 작업
1. 송금 (거래 생성, 트랜잭션 생성)
2. 송금 내용 확인 (트랜잭션 검증)

송금 내역을 모두 모아 승인 절차 (블록 생성)
채굴자 노드가 수행
승인 완료된 Transaction들 모아서 블록 생성 후 네트워크로 전파

기존 블록체인에 연결해서 관리
Full 노드가 수행
승인 완료된 Transaction들 모아서 블록 생성 후 네트워크로 전파
(소프트웨어에 따라 역할 상이)

Full 노드 관리하는 블록체인에는 모든 거래 내역(Transaction)이 저장된다
블록체인 = 거래 내역
따라서 공공 거래 원장 (Public Ledger) 라고도 한다
모든 Full 노드들이 갖고 있으므로 분산 저장된다 (하나에서 잘못되도 복원 가능)

각 노드의 기능 Full Blockchain Node (풀 노드)



블록체인 전체를 관리하는 노드

1. 신규 블록 받기

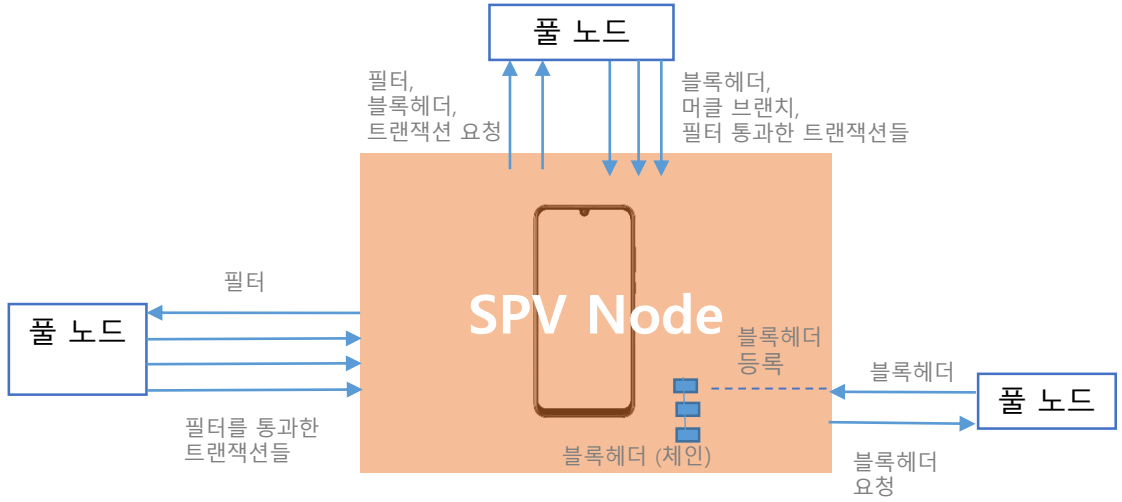
- 채굴자 노드한테 최근 트랜잭션(Tx) 담긴 블록 받음
- 받은 노드를 합의 규칙에 따라 유효성 검증 → 맞으면 블록체인에 연결
- 신규 블록의 헤더 정보(Block Header Hash)를 전파한다
- 만일 정보 받은 노드가 세부 내용을 요청하면 보내준다 (Block Relay)
- 다른 풀 노드들과 블록체인 데이터를 동기화하여 신뢰성과 일관성 유지

2. 개별 트랜잭션 받을 시

- 합의 규칙에 따라 유효성 검증 후 자신의 Memory Pool에 저장
- 그 후 다른 노드로 전파한다 (Transaction Relay)
- 블록 전파와 마찬가지로 해시를 먼저 보낸 후 요청하는 노드들에 세부 내용을 보낸다

블록 헤더 해시 : 블록의 body는 요약되어 헤더에 저장되고, 이는 다시 헤더 해시로 요약된다 (1MB 크기에서 80바이트, 32바이트 크기까지 요약된다)
메모리 풀 : 각 노드의 임시저장소로, 승인(채굴)되지 않은 트랜잭션들이 저장된다. 채굴자들도 자신의 메모리 풀 속 저장된 것들을 모아 블록 생성한다

각 노드의 기능 Lightweight Node (SPV 노드)



블록체인의 헤더 정보만 가지고 있다 (헤더체인)

새 블록 생성 시 풀 노드로부터 블록 헤더를 받아 헤더 체인에 연결
스마트 기기 등 소형 장비에 사용. 송금과 수금만 필요할 시에 사용한다

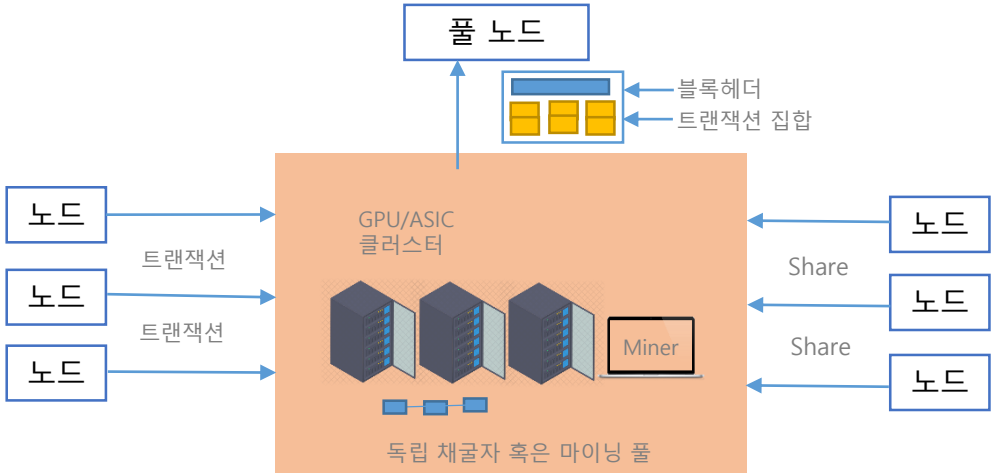
Bitcoin 지갑 애플리케이션은 주로 SPV 노드
자신의 송금/수금 관련 트랜잭션 유효성을 검증
검증을 위해서 풀 노드의 도움 받아야 한다 – 필터를 통해 연관된 것만 선택적으로 받는다

자신의 가진 블록 헤더 정보와 풀 노드에서 받게 될 머클 플록을 이용해 자신 트랜잭션이 특정 블록에 포함되어 있는지 판다 + 그 블록 이후 몇 개 블록이 추가되었는지도 확인 가능 (Block depth)
P2P 프로토콜 설명에서 더 자세히 다룰 예정

풀 노드에 비해 안전하지 못하다 (의존해야 하므로 자신 주소 노출시 공격 대상 될 수도) → 편리성을 위해 안전성 희생

머클 블록 (Merkle Block): 머클 트리 중 자신 트랜잭션을 검증하는데 필요한 부분 데이터
Block Depth : 1일시 트랜잭션이 승인(채굴) 됐다고 하고, Block Depth > 6 이상이어야 d이 트랜잭션이 최종 승인(Confirmation)됐다고 판단한다.

각 노드의 기능 Miner Node (채굴자 노드)



트랜잭션 승인 과정 참여

각 노드에서 보내는 트랜잭션들 **유효성을 검증**하고, 이들을 모아 **새 블록으로 묶어 생성** 후 풀 노드로 전송
풀 노드에서 다시 검증하고, 통과시 블록체인에 추가된다 (2 단계를 거쳐야 유효한 거래)

트랜잭션 승인 과정

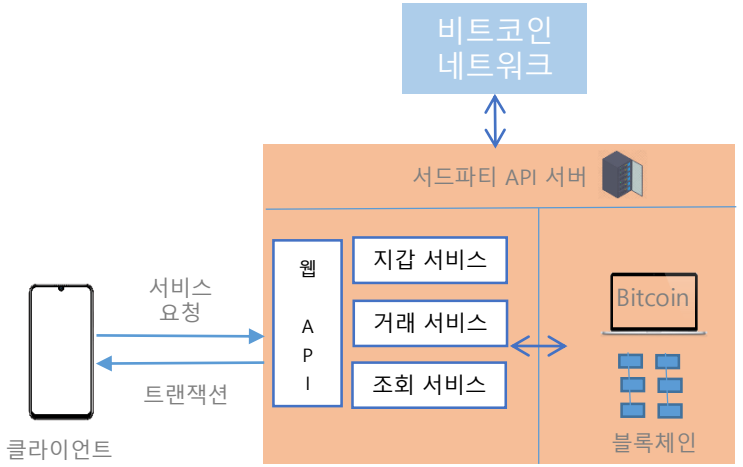
경쟁적으로 수학적 퍼즐 풀고 블록 헤더에 그 해답을 제시. 먼저 제시한 채굴자 블록이 풀 노드로 먼저 보내짐
풀 노드에서 해답이 옳바른지 확인 (유효성 검증) → 이긴 채굴자는 코인 받음
경쟁을 통해 쉽게 채굴할 수 없기에 신뢰성이 높아진다

채굴자는 블록 생성시 발생되는 코인과 수수료를 받는다
유효한 트랜잭션도 수수료 미지급시 블록에 포함되지 않는다

채굴자가 마이닝 풀일 경우, 서브 채굴자들이 해답을 찾아 마이닝 풀에 보내고 보상을 분배받는다 (이길 확률 높힘)

* 해답은 블록헤더의 Nonce 필드에 기록된다
Block Depth : 1일시 트랜잭션이 승인(채굴) 됐다고 하고, Block Depth > 6 이상이어야 d이 트랜잭션이 최종 승인(Confirmation)됐다고 판단한다.

각 노드의 기능 3rd Party API Client



비트코인 네트워크 기본 구성 요소는 아니다

블록체인 데이터 전혀 없고, 전적으로 3rd party에 의존한다 (보안 매우 취약)
클라이언트는 2가지 경우가 존재한다

1. 자신이 지갑을 소유한 경우 : 블록데이터나 헤더도 없고 오직 자기 지갑 키만 갖는다. 이를 통해 트랜잭션을 생성한다. 사용 가능한 잔액, 잔액의 블록체인에서 위치, 인증 여부 등은 서드파티 API 서버를 이용해 확인한다

2. 지갑 자체를 서드파디 서버에 위탁하는 경우: 서드파티 서버에 가입해 지갑을 발급받고, 서버를 통해 거래가 이뤄진다. 지갑이 서버에 저장되어서 웹을 통해 잔액을 확인하거나 거래한다. (거래소 등)

탈중앙화 개념과 어긋난다. 서드파티의 보안을 신뢰해야한다는 문제점

참여자 노드 현황 관찰

[실습] 노드 수 변화 관찰

참여자 노드 현황 사이트의 API 서비스를 이용해 최근 네트웨크에서 활동중인 서비스를 관찰한다
스냅샷 데이터 100개를 받아 기간에 따른 노드 수 변화 측정 (7일 정도 분량)

```
import requests
import time
import matplotlib.pyplot as plt

# 100 페이지까지만 조회한다. 이 사이트는 최근 60일까지 데이터를 제공함
nPage = 100
if nPage > 100:
    print("요청 페이지가 너무 많아 시간이 오래 걸립니다.")
else:
    t = []
    n = []
    for page in range(1, nPage):
        # 페이지 당 100개씩 요청한다. (Max = 100)
        url = 'https://bitnodes.earn.com/api/v1/snapshots/?limit=100&page=' + str(page)
        resp = requests.get(url=url)
        data = resp.json()
        print("page %d loaded." % page)

        for i in range(len(data['results'])):
            ts = time.gmtime(data['results'][i]['timestamp'])
            t.append(time.strftime("%Y-%m-%d %H:%M:%S", ts))
            n.append(data['results'][i]['total_nodes'])
t=t[::-1]
n=n[::-1]

# 최근 노드수의 변화를 확인한다
plt.figure(figsize=(8,6))
plt.plot(n, color='red', linewidth=0.7)
plt.title('BTC Nodes₩n' + t[0] + '~' + t[-1])
plt.grid(color='green', alpha=0.2)
plt.show()
```

블록체인의 구조

블록들은 아래부터 위로 가면서 순차적으로 번호 부여
크게 바디(Body)와 헤더(Header)로 구성된다
[맨 아래 0번부터 위로 쌓인다]

바디에 트랜잭션들이 저장되어 있고

블록헤더에는 바디 정보가 요약
+ 이전 헤더를 포인터로 가리키고 있다

이를 통해 서로 체인으로 연결되어 있다
이 구조를 **블록체인**이라한다

[예시] 블록 구조 - 헤더

블록 번호 : 557416
블록 해시 : 00000000000000000000025d1a02a5b429464...

Version	4 bytes	20400000
Previous Block Hash	32 bytes	00000000000000000037fb8b5b05787887a6391eb66...
Merkle Root	32 bytes	5093d58359e203f2c296f0307f5f399f6c9e514381e4d1...
Timestamp	4 bytes	5c3303b7
Bits	4 bytes	173218a5
Nonce	4 bytes	c539cb45

블록 구조 - 바디

Transaction Count	4 bytes	0536
Coinbase transaction		
Transaction #1		
Transaction #2		
...		
Transaction #1333		

2019/1/7 07:45:59
실제 생성 블록 예시

1334(0x0536)개 트랜잭션 담김
1개 코인베이스 트랜잭션
1333 일반 트랜잭션

블록헤더에 요약정보 담김
헤더는 총 80 byte

블록체인의 구조 (헤더)

Version	4 bytes	20400000
Previous Block Hash	32 bytes	00000000000000000037fb8b5b05787887a6391eb66...
Merkle Root	32 bytes	5093d58359e203f2c296f0307f5f399f6c9e514381e4d1...
Timestamp	4 bytes	5c3303b7
Bits	4 bytes	173218a5
Nonce	4 bytes	c539cb45

Version

비트코인 소프트웨어 버전 정보 :보완되면서 추가된 기능 버전으로 표시
제네시스 버전 1이고 2,3,4 순으로 발전 BIP-9(Digital Signature) (현재)
소프트포크시 채굴자들이 해당 기능 지원할지 여부를 표시할 목적으로 사용

Previous block Header Hash

이전 블록 헤더의 해시 값 : 해시 포인터 역할 (체인으로 연결)
내용이 바뀌면 해시도 변해서 연결 끊어짐 – 위조 불가

Merkle Root

머클 트리: 트랜잭션 데이터 보호. 트리의 일부 바뀌면 머클 루트 노드 값도 바뀜
→ 헤더 해시도 바뀜 → 블록이 체인에서 끊어짐 – 위조 불가
특정 트랜잭션이 이 블록에 포함되어 있는지 확인 시 사용된다 (SPV 노드 등)

Timestamp

블록이 생성된 날짜와 시각(UTC)
유닉스 시간 기준 1970/1/1 0시부터 현재까지 경과 시간

- 소프트포크 : 이전 기능과 호환되는 방식으로 소프트웨어 보완 (반대 개념은 하드 포크 – 이전 지원 불가)
- Bitcoin Improvement proposal : 기능 향상을 위한 제안 문서. 여러 사람들이 검토하고 타당하면 비트코인 소프트웨어에 반영

블록체인의 구조

Bits & Nonce

채굴 난이도(비츠) 와 해답(논스) 과 연관
비츠 = 채굴 성공 여부 판단 정보 // 논스 = 해답
채굴자는 임의의 논스 값을 계속 대입하면서 헤더 해시를 계산 후, 비츠와 비교
(채굴자 목표 : 비츠가 가리키는 조건 맞는 논스 값 찾기가 관건)
비츠는 채굴 시간이 평균 10분 정도 걸리도록 조절된다. (비츠=채굴 난이도)

Block Header Hash

80byte 헤터를 요약하여 32byte 해시 값으로 만듦
해시 계산 방식 : double SHA-256 알고리즘
다만 해당 블록 해세는 헤더에 저장되지 않는다
각 노드에서 필요하면 계산하여 사용 (2. 암호기술에서 자세히 기술)

Coinbase Transaction

채굴자가 블록 생성 보상을 자기 지갑으로 송금하는 트랜잭션
보상 : 12.5 BTC (현재) + 1,333개 트랜잭션에 포함된 수수료

Transaction

코인을 주고 받는 거래
거래가 유효하려면 채굴자에 의해 선택되어 블록체인에 포함돼야한다
포함되도 취소될 가능성이 있어 가 승인 상태
최종 승인되려면 6개 이상 블록 더 채굴돼야한다 (1시간 정도 소요 - 느림)

Transaction Count	4 bytes	0536
Coinbase transaction		
Transaction #1		
Transaction #2		
...		
Transaction #1333		

- 이 외에 데이터를 보여주는 서드 파티 사이트들이 있지만, 체인 위에 직접적으로는 없다

블록체인 데이터 확인 (사이트 변경으로 죽은 코드)

[실습] 블록 생성 시간 분포 관찰

최근 10일동안 생성된 블록 생성 시간 – 헤더와 타임스탬프 이용해 시간 간격 분포 구함
몇 분만에 생성될 지 확률 추정 가능 (이 코드 서는 5분내 승인될 확률 구함)

```
# 채굴자에 의해 블록이 생성되는 시간 간격을 관찰한다
# 참조: https://Blockchain.info/blocks
# 블록 생성 간격은 지수 분포를 따라 내 거래가
# 몇 분 이내 채굴될지 계산 가능하다
import requests
import time
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

print("금일 생성된 블록을 읽어옵니다")
url = 'https://blockchain.info/blocks?format=json'
resp = requests.get(url=url)
data = resp.json()

header = []
block = data['blocks']
for n in range(len(block)):
    height = block[n]['height']
    btime = block[n]['time']
    bhash = block[n]['hash']
    header.append([height, btime, bhash])

# 어제 생성된 블록을 읽어온다.
stime = btime - 24 * 60 * 60

# 이전 10일 동안 생성된 블록 정보를 읽어온다.
for nDay in range(0, 10):
    ts = time.gmtime(stime)
    date = time.strftime("%Y-%m-%d %H:%M:%S", ts)
    print("%s 생성된 블록을 읽어옵니다" % date)

    url = 'https://blockchain.info/blocks/' + str(stime) + '000?format=json'
    resp = requests.get(url=url)
    data = resp.json()

    block = data['blocks']
    for n in range(len(block)):
        height = block[n]['height']
        btime = block[n]['time']
        bhash = block[n]['hash']
        header.append([height, btime, bhash])

    stime = block[0]['time'] - 24*60*60

df = pd.DataFrame(header, columns=['Height', 'Time', 'Hash'])
sdf = df.sort_values('Time')
sdf = sdf.reset_index()
print('총 %d 개 블록 헤더를 읽어왔습니다.' %len(df))

# 블록 생성에 소요되는 시간 분포를 관찰
mtime = sdf['Time'].diff().values
mtime = mtime[np.logical_not(np.isnan(mtime))]
print("평균 Mining 시간 = %d (초)" % np.mean(mtime))
print("표준편차 = %d (초)" % np.std(mtime))

plt.figure(figsize=(8,4))
n, bins, patches = plt.hist(mtime, 30, facecolor='red',
edgecolor='black', linewidth=0.5, alpha=0.5)
plt.title("Mining Time Distribution")
plt.show()

# 5분 이내에 내 거래가 채굴될 확률
s = 60 * 5
p = 1 - np.exp(-s / np.mean(mtime))
print("5분 이내에 내 거래가 Mining 될 확률 = %.2f (%s)" % (p * 100, '%'))
```

실행 결과

평균 10분으로 유지되고 있다.
다만 몇초만에 풀린 문제도 있는 등 표준편차가 크다.
5분 이내 나올 확률은 42%
초당 평균 2~3개의 트랜잭션이 처리되고 있다.
평균적으로 1800개 트랜잭션을 포함한다

블록의 크기 제한과 확장성

블록의 크기는 1MB로 제한
한 블록에 들어갈 트랜잭션의 개수도 제한 받는다
= 비트코인의 최대 처리 용량과 확장성(scalability) 문제

비트코인의 확장성 문제

각 블록의 크기는 1MB를 초과할 수 없다
초기 DoS 공격 등의 가능성 때문에 1MB로 제한함
현재는 크기 제한 영향 없으나 향후 거래량 증가 시 문제
처리 수요가 더 많아져서, 더 큰 거래 수수료 지불한 것만 승인 – 확장성 문제
블록의 크기를 늘려 트랜잭션 개수 늘려야 한다 – 합의 규칙 변경은 전체 노드 SW 업데이트 필요
중앙통제가 아니므로 합의에 이뤄야 한다 – 아니면 분리된다 (하드포크)

세그윗(segregated witness, SegWit)과 비트코인 캐시(BCH) 탄생

거래 데이터 조작 가능성(Transaction Malleability) 해결 위해 2017/8 세그윗 기능 추가
원래는 소프트포크를 하려고 함 – 하드포코는 사용자가 분리되어 코인 가치 떨어짐
트랜잭션 데이터 → 송금/수금자 지갑 정보 & 금액, 전자서명(비중 큼)으로 구성
전자서명 없을 시 더 많은 트랜잭션 넣을 수 있다 → 세그윗 (분리 전자서명)

전자서명을 분리해서 블록 내 특정 영역에 저장하고 (거래 데이터 조작 문제 해결)
크기 제한을 바이트 단위에서 중량(Weight)로 바꿈 (평균 1.8MB 크기)

일부 메이저 채굴자들 ASIC 장비와 호환성 문제로 인해 반대 → 블록 크기 늘리는 방향 제안
비트코인은 결국 업데이트 진행되었고, 반대자들은 비트코인 캐시로 하드포크 되었다

비트코인 지갑과 지갑 주소

지갑 – 주소와 키를 관리하고, 트랜잭션을 생성하거나 검증하는 기능을 수행한다

일반 사용자의 지갑

기능: 키 관리, 잔액 관리, 거래 생성, 거래 확인, 거래 내역 관리 수행
주소로 받은 코인은 지갑이 아닌 블록체인에 보관된다 (블록체인에 있는 것만 사용 가능)
지갑 비밀키 분실시 영원히 찾을 수 없다, 해킹으로 권한 뺏겨도 끝 (탈중앙화)

대부분 지갑은 SPV 노드로 동작
블록의 헤더 체인을 저장하고 있어 잔액 확인 가능하고, 거래 승인 여부도 확인할 수 있다
다만 헤더 체인을 관리하는 것도 어려움 (풀 노드 도움 받아 지속적으로 관리 필요)
따라서 일부는 지속적 관리를 안 하는데, 이러면 보안에 배우 취약 – 조심 필요

개발자, 전문가 측면의 지갑

개인키(private key), 공개키(public key), 지갑 주소(address)로 구성

개인키: 블록체인 저장된 BTC가 자기 소유임을 증명할 때 사용 (전자서명)
송금 – 블록체인 상에 있는 코인이 내 소유라는 것을 모든 노드에게 증명해야한다
공개키는 다른 노드들이 전자서명 검증에 사용, 개인키는 나만이 알고 있어야 한다
공개키와 주소만 공개해야한다

지갑은 소프트웨어이므로 전자 저장 장치에 위치하는데, 장비가 손상되거나 분실하면
코인(의 소유권)도 유실된다. 따라서 개인키를 주가적으로 백업하는 관리가 필요하다 (적어둠)

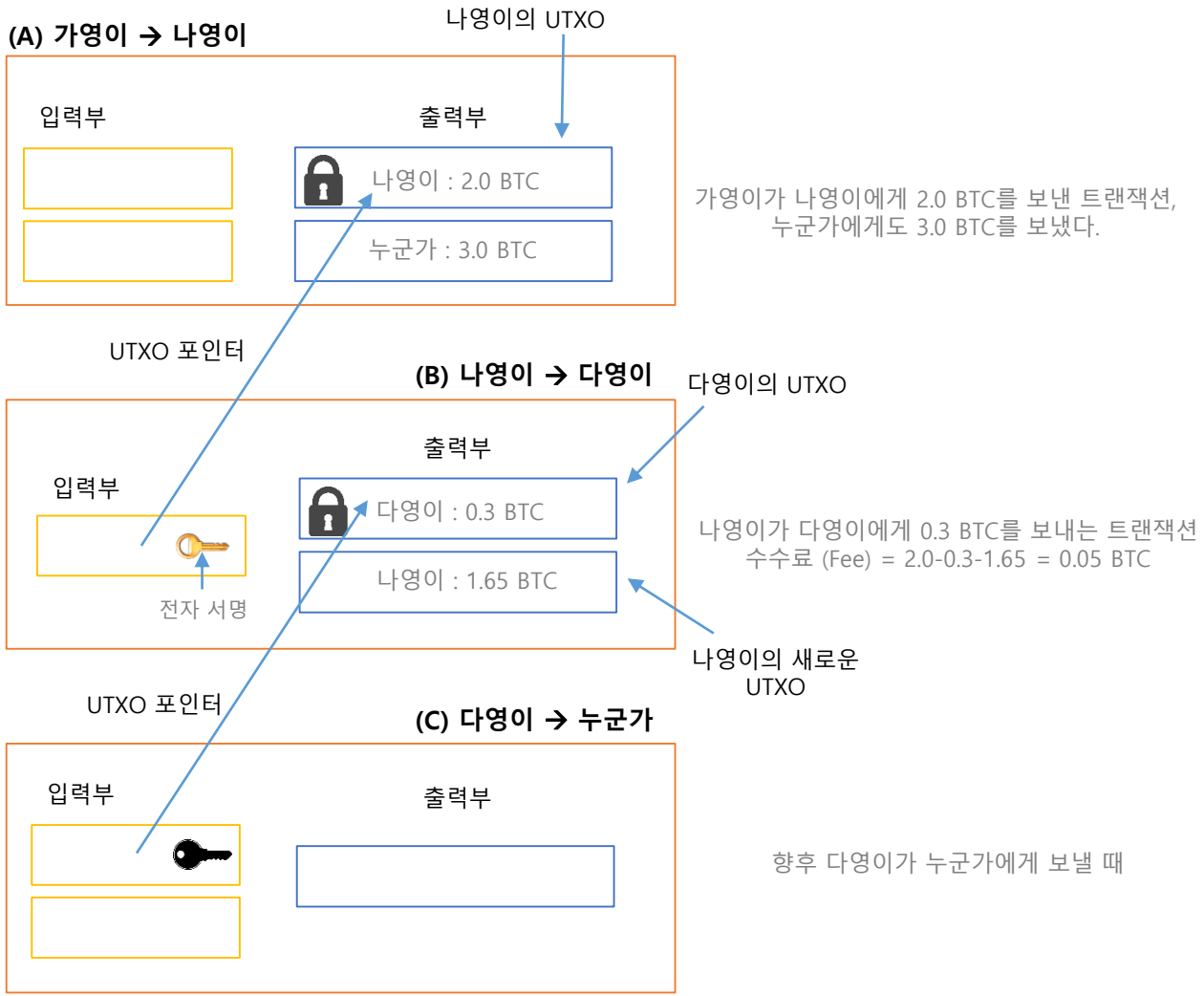
개인키 : 256비트의 매우 큰 수 (현재는 유출되지만 않으면 안전하다)
개인키 통해 공개키 만들고, 공개키 통해 지갑 주소 만든다. 반대로 알아내기는 어렵다
개인키만 알면 언제든지 다시 만들 수 있다
안전상 이유로 여러 개인키를 만들거나 거래마다 바꿔 나가기도 한다. 다만 관리 어려움

트랜잭션(Transaction) 생성

Transaction = 화폐교환절차 (비트코인의 가장 중요한 작업)
나머지는 이를 안전하게 만들기 위한 보조 수단이다

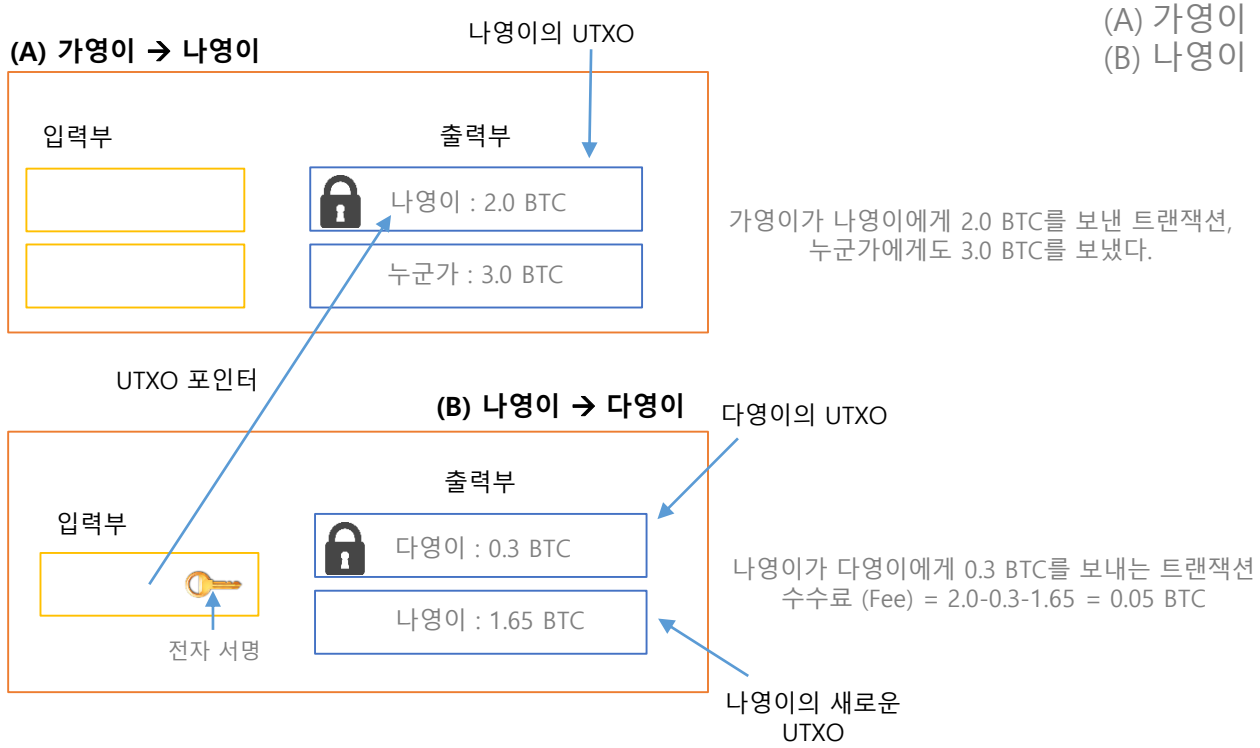
트랜잭션 생성 예시

총 3개의 트랜잭션이 있다 (우선 2개부터)
(A) 가영이 → 나영이 (2.0 BTC)
(B) 나영이 → 다영이 (0.3 BTC)



트랜잭션(Transaction) 과정 (B)

트랜잭션 생성 예시
총 3개의 트랜잭션이 있다 (우선 2개부터)
(A) 가영이 → 나영이 (2.0 BTC)
(B) 나영이 → 다영이 (0.3 BTC)

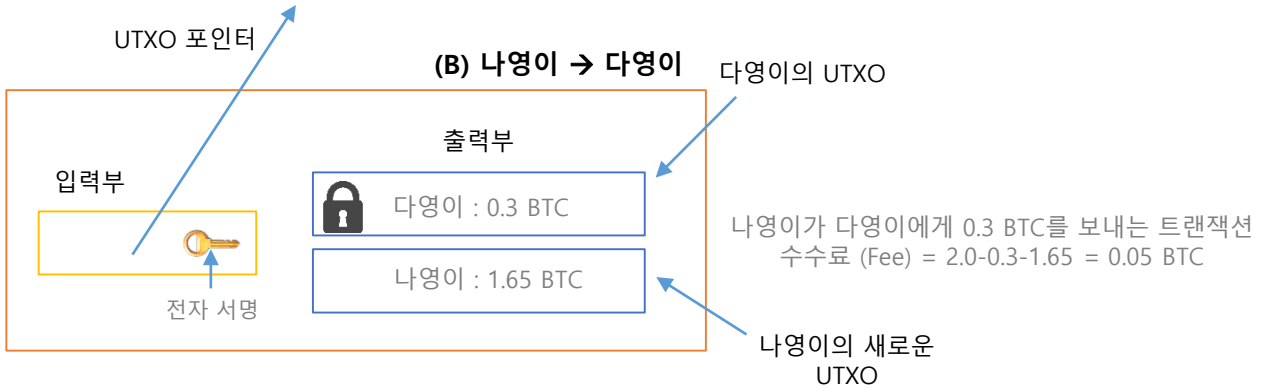


Transaction: 입력부와 출력부로 구성된다
입력부 : 송금자 사용가능 잔액 & 전자서명 기록
출력부 : 수금자 주소와 송금할 금액

나영이가 (B)를 하려고 한다. 우선 트랜잭션 A를 찾고, A 출력부 자신이 사용할 수 있는 잔액(2.0)이 기록된 위치를 찾는다
그 위치를 Unspent Transaction Output(UTXO)라 한다 = 아직 사용되지 않은 잔액
나영이의 **UTXO는 (A)의 0번째 출력부**다. 나영이는 트랜잭션 (B) 입력부에 (**A-0**)인 UTXO 포인터를 기록한다

가영이는 2 BTC를 가영이 공개키를 통해 (A-0)을 잠궜다 (=나영이만 이 돈을 사용할 수 있다)
나영이의 개인키만이 이 자물쇠를 풀 수 있다 = 전자서명
나영이는 자신의 전자 서명을 트랜잭션 (B) 입력부에 기록 → 입력부에는 UTXO와 전자 서명이 기록된다

트랜잭션(Transaction) 과정 (B)



나영이는 다영이에게 0.3BTC를 보내기 위해 출력부를 만든다.
트랜잭션(B)의 첫번째 출력부에 다영이의 주소와 송금할 금액을 기록한다.
그리고 나중에 다영이가 이용할 수 있도록 잠궈 놓는다 (주소 대신 공개키 해시 기록)
이 위치는 다영이의 향후 UTXO가 된다

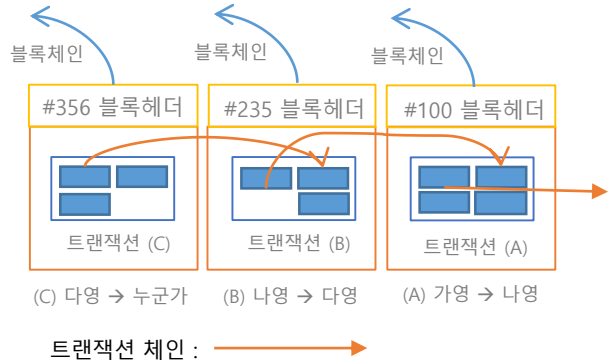
다영이에게 0.3BTC를 보내기 위해 출력부를 만든다.
트랜잭션(B)의 첫번째 출력부에 다영이의 주소와 송금할 금액을 기록한다.
그리고 나중에 다영이가 이용할 자신의 UTXO가 된다

나영이가 자신의 UTXO를 포인팅해서 다영이에게 0.3BTC를 보내도 UTXO (A-0)는 'Spent' 상태로 바뀌고 사용 불가능
나머지 1.7BTC는? → 트랜잭션 (B) 두번째 출력부를 이용해 자신에게 송금
이 출력부가 나영이의 새로운 UTXO가 된다 (B-0)는 다영이의 UTXO, (B-1)은 나영이의 새 UTXO

But, 이러면 채굴자에게 지급할 수수료가 없다. 나영이가 수수료를 지급하려면 수수료를 뺀 만큼 자신에게 재송금하면 된다
1.7 BTC – 0.05 BTC = 1.65 BTC 를 자신에게 재송금하면 된다. 수수료는 트랜잭션에 기록되지 않음
수수료는 UTXO 총합에서 출력부 총 금액을 빼는 방식으로 계산한다 (2.0 – 0.3 – 1.65 = 0.05BTC)
만일 나영이가 자신에게 재송금하지 않으면 출력부에 0.3만 기록되므로 채굴자가 1.7BTC를 다 가져간다
지갑을 직접 만들어 수행할 때는 재송금 절차를 반드시 수행해야 한다.

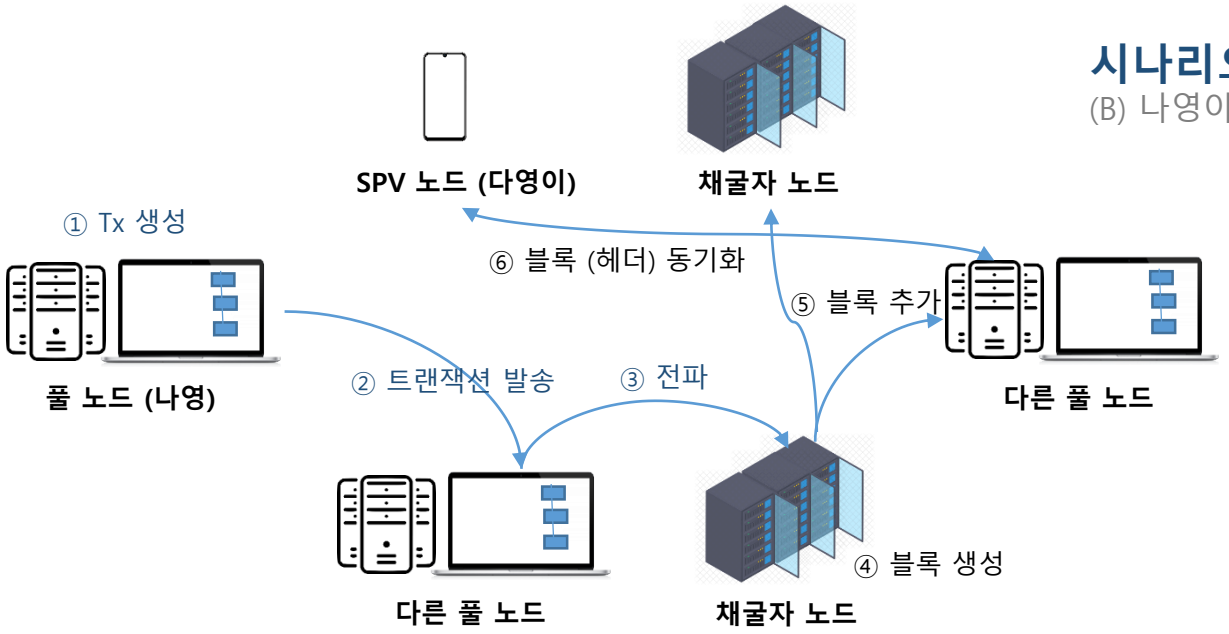
이렇게 (B)가 만들어져 비트코인 네트워크로 전송되고, 노드들은 (B) 입력부 전자서명 열쇠로 UTXO (A-0) 풀리는지 검증
검증되면 트랜잭션 (B)는 유효하다고 판단하고 블록체인에 등록 (송금 완료)

Block 체인과 트랜잭션 체인



각 트랜잭션은 서로 다른 블록에 포함될 수 있다
모든 트랜잭션은 UTXO 포인터로 연결돼 있다 (마치 블록체인처럼)
모든 트랜잭션은 연결돼 있기 때문에 (코인베이스 트랜잭션 제외) 모든 거래 내역은 추적이 가능하다
공공거래 장부이므로 신뢰할 수 있다.

트랜잭션 전송 (풀 노드일시)



시나리오
(B) 나영이 → 다영이 (0.3 BTC)

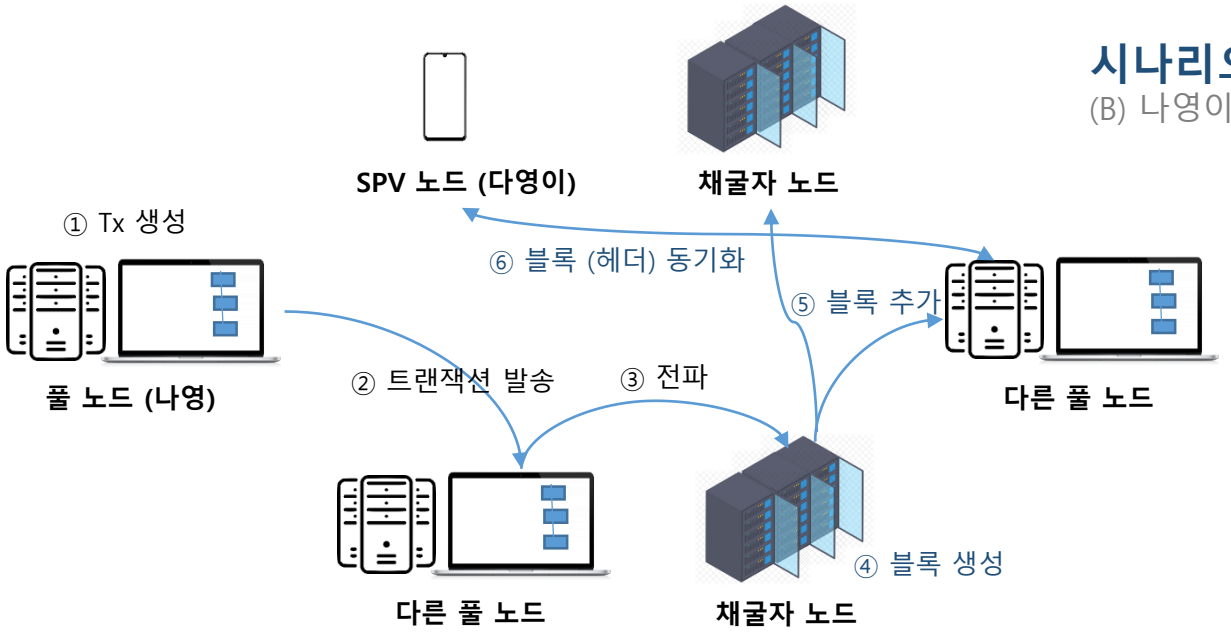
나영이는 풀 노드이므로 트랜잭션 (B)를 생성하고 승인 여부도 확인할 수 있다

1. Tx 생성 : 트랜잭션 생성시 가장 먼저 자신의 UTXO를 찾아야한다. (어느 출력부에 기록되어 있는지). 나영이는 별도의 UTXO DB를 이용해서 (A-0)를 찾는다. 이를 입력부에 자신의 개인키를 이용한 전자서명과 함께 기록. 출력부는 송금 금액과 (지갑 주소로 변환한) 다영이의 공개 키 해시 값 + 수수료 차감한 자신 재송금 추가

2. 트랜잭션 발송 : (B)를 발송한다. 풀 노드는 현재 접속된 다른 주소들 IP 주소를 통해 인근 노드들로 보낸다

3. 전파 : 받은 노드들은 올바르게 작성 되었는지 유효성 검증하고, 올바르면 추가 전파, 아니면 Reject
   검증 절차 – 입력부 UTXO가 Unspent인지, 잔액이 송금액보다 큰지, 전자서명 값이 옳바른지
   다영이 노드가 (B)를 받으면 송금한 트랜잭션이 네트워크에 전파 완료 (단, 승인은 아직)

트랜잭션 전송 (풀 노드일시)



시나리오
(B) 나영이 → 다영이 (0.3 BTC)

4. 블록 생성 : 채굴자 노드가 트랜잭션 (B0를 받으면 다른 트랜잭션과 합쳐서 새 블록을 만들고, 인근 볼록에 보낸다. 단, 통째로 보내면 네트워크에 부하가 많이 걸리므로 별도 절차(추후 서술)에 따른다.

5. 블록 추가 : 풀 노드가 새 블록을 받으면 블록 안에 있는 트랜잭션과 해시를 합의 규칙에 따라 전부 검증한다 블록에 옳바르면 체인에 연결한다

6. 블록 동기화 : 풀 노드가 블록을 추가하면 인근 노드들에게 사실을 알린다. 다른 노드들도 자체적으로 블록을 추가할 수 있다. 풀 노드들은 블록 높이(개수)를 맞추는 절차로 블록체인을 동기화한다. SPV 노드는 블록 헤더체인만 갖고 있다. 자신 관련 트랜잭션이 있다고 통보받으면 보내 달라고 풀 노드에게 요청. 이를 받으면 0.3BTC가 블록체인에 등록됐음을 확인할 수 있다(가승인 상태). 이후 6개 추가 생성 시 최종승인된다.

(B)가 승인되면 풀 노드들은 (B)가 사용한 UTXO (A-0)를 Spent 상태로 바꾼다. 이 상태는 풀 노드의 UTXO set(DB)에 기록대 있으므로 이 DB를 사용되었다고 업데이트한다. 이를 통해 이중 지급을 방지한다.

트랜잭션 전송 (그 외)

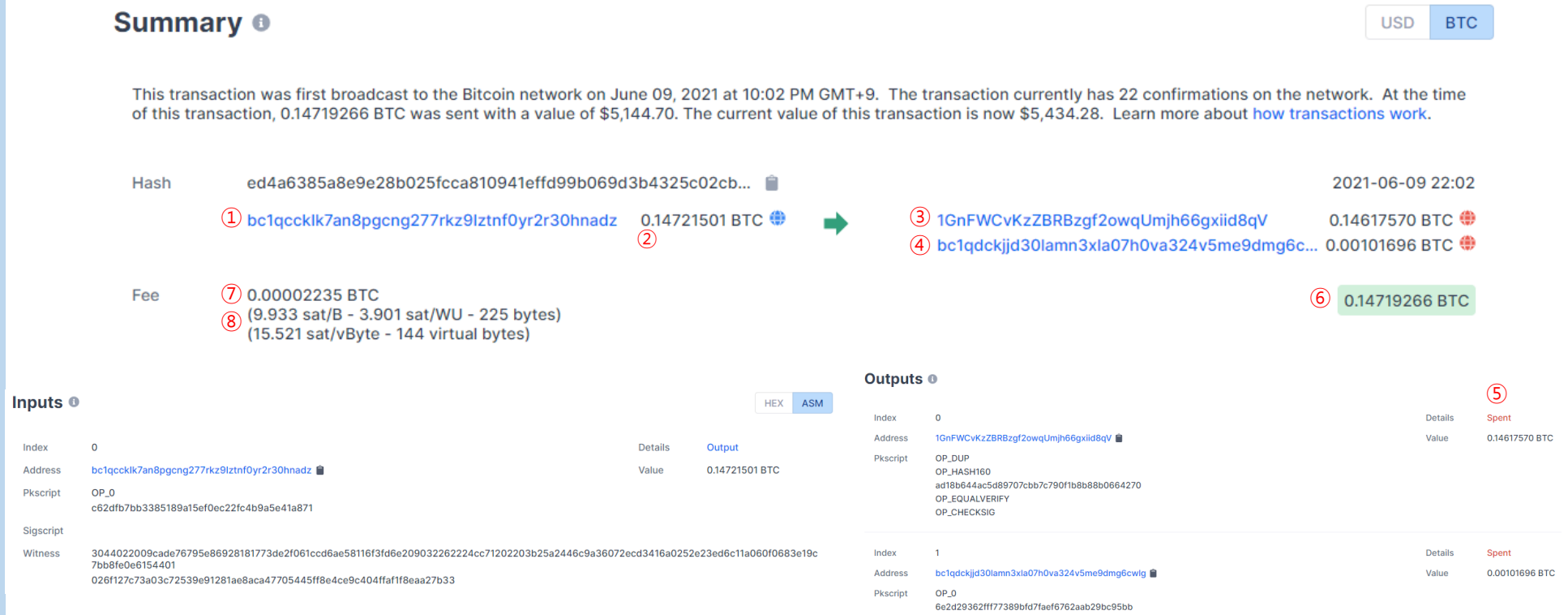
나영이가 SPV 노드일 때

지갑 앱은 블록 데이터는 없지만, 자신의 UTXO는 스스로 관리해야한다.
자신의 UTXO들이 어느 트랜잭션에 기록돼 있는지 자체적으로 기록하고 관리한다.
(A)에서 가영이가 보낸 것을 받고 승인이 확인되면 (A) 트랜잭션 해시 값과 출력부 번호를 기록해분다
그 이후에는 앞선 그림처럼 작동한다

나영이가 서드파티 API 클라이언트일 때

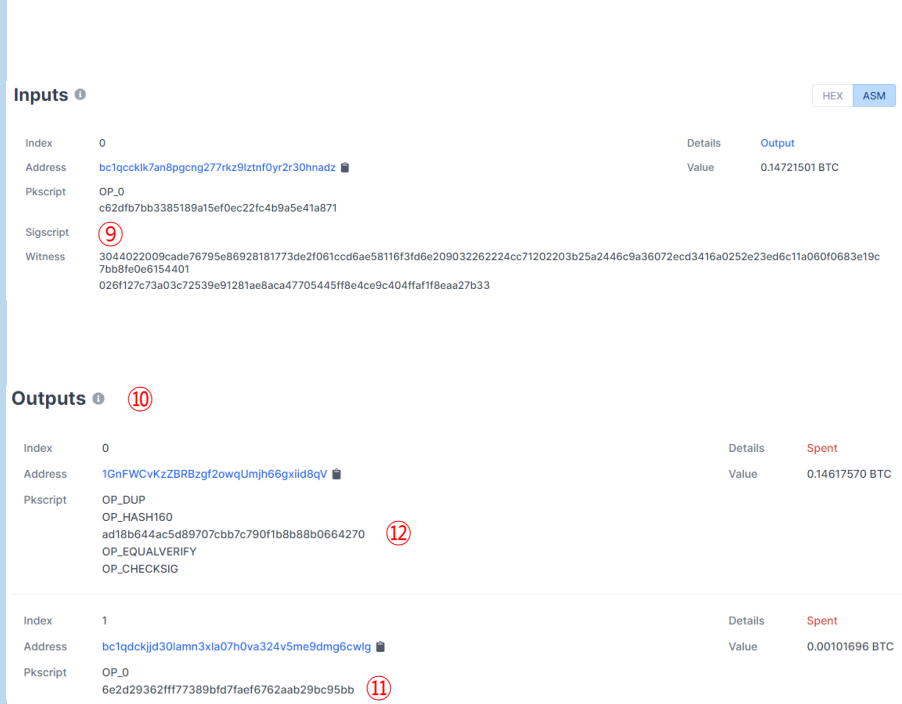
지갑 앱은 오직 자신의 키(key) 만을 갖고 있다. 블록체인 데이터도 없고, UTXO set도 없다.
트랜잭션(B)를 스스로 만들 수 없다. 따라서 서드 파티에게 자신의 지갑 주소를 알려주고 자신의 UTXO를 찾아달라고 요청한다. 서드파티가 알려준 UTXO를 통해 트랜잭션(B)를 만들어 네트워크로 송출한다.
비트코인 네트워크에 직접 연결된 것이 아니므로 이 트랜잭션을 릴레이 받을 수 없다.
(정상적으로 전파, 승인되었는지 알 수 없다)
확인하려면 자신이 보낸 (B)의 해시 값을 이용해 서드파티에게 승인 여부를 요청해야한다
(트랜잭션 편에서 파이썬을 이용해 이 형태에 지갑을 만들 예정)

트랜잭션 사례 (예시)



(1) : 송금자의 지갑 주소. 직접 기록이 아닌 입력부 UTXO 가리키는 트랜잭션 출력부 공개키 해시 형태로 기록
(2) : 송금자가 선택한 UTXO의 잔액 (0.1472...BTC)
(3) : 수금자의 지갑 주소. 직접 기록된 것이 아닌 수금자의 공개키 해시 형태로 기록돼 있다. 수금자에게 약 0.146BTC 전송
(4) : 송금자가 자신에게 수수료를 제외한 0.001BTC를 재송금했다. 지갑 주소가 송금자와 동일한 것을 알 수 있다
(5) : Spent 상태로 이미 사용했다. 아직 사용안한 새로운 UTXO를 추적해야한다.
(6) : 출력부에 기록된 금액의 합계다
(7) : 입력부 합계 – 출력부 합계 = **채굴자에게 지급한 수수료**
(8) : 트랜잭션에서 입력부 출력부가 여러 개 있을 수 있다. 한 UTXO 잔액이 부족하면 여러 개 UTXO 입력부에 기록해야한다. 따라서 트랜잭션은 개수에 따라 크기가 다르다. 트랜잭션이 클수록 처리 시간이 더 걸려 수수료를 더 많이 내야한다. 수수료는 송금할 금액과는 무관하다. **채굴자는 바이트 당 수수료가 만족할 수준이면 승인**한다.

트랜잭션 사례 (예시)



Inputs

HEX | ASM

Index	0	Details	Output
Address	bc1qcckk7an8pgcng277rkz9lztnf0yr2r30hnadz	Value	0.14721501 BTC
Pkscript	OP_0 c62dfb7bb3385189a15ef0ec22fc4b9a5e41a871		
Sigscript	⑨		
Witness	3044022009cade76795e86928181773de2f061ccd6ae58116f3fd6e2090322622 24cc71202203b25a2446c9a36072ecd3416a0252e23ed6c11a060f0683e19c7bb8fe0e6154401 026f127c73a03c72539e91281ae8aca47705445ff8e4ce9c404ffaf1f8eaa27b33		

Outputs ⑩

Index	0	Details	Spent
Address	1GnFWCvKzZBRBzgf2owqUmjh66gxiid8qV	Value	0.14617570 BTC
Pkscript	OP_DUP OP_HASH160 ad18b644ac5d89707cbb7c790f1b8b88b0664270 ⑫ OP_EQUALVERIFY OP_CHECKSIG		
Index	1	Details	Spent
Address	bc1qdckjjd30lamn3xla07h0va324v5me9dmg6cwlg	Value	0.00101696 BTC
Pkscript	OP_0 6e2d29362fff77389bfd7faef6762aab29bc95bb ⑪		

Details

Hash	ed4a6385a8e9e28b025fcca810941effd99b069d3b4325c02cb8492c6e096243
Status	Confirmed
Received Time	2021-06-09 22:02
Size	225 bytes
Weight	573
Included in Block	686979
Confirmations ⑬	26
Total Input	0.14721501 BTC
Total Output	0.14719266 BTC
Fees	0.00002235 BTC
Fee per byte	9.933 sat/B
Fee per vbyte	15.521 sat/vByte
Fee per weight unit	3.901 sat/WU
Value when transacted	$5,144.70

(9) : 입력부에 기록된 전자서명 값 → 이 값으로 UTXO에 걸려 있는 자물쇠를 푼다.
(10) : 출력부에 기록된 수금자의 공개키 해시 값. 수금자가 향후 이 출력부를 UTXO로 사용할 수 있도록 값 잠궈둠
(11) : 0.146 BTC를 받은 수금자의 공개키 해시 값이다
(12) : 송금자가 자신에게 재송금한 0.001BTC를 나중에 사용할 수 있도록 자신의 공개키 해시에 기록해둔다?
(13) : 이 트랜잭션은 승인되어 블록 686,979번에 등록되었고, 26개 블록이 추가되어 최종 승인 상태 (6 이전은 가승인)

채굴

채굴자 : 트랜잭션들 모아 블록 생성
(각 트랜잭션 유효성 검증 → 수수료 집계 → Tx 모아서 블록바디 구성)

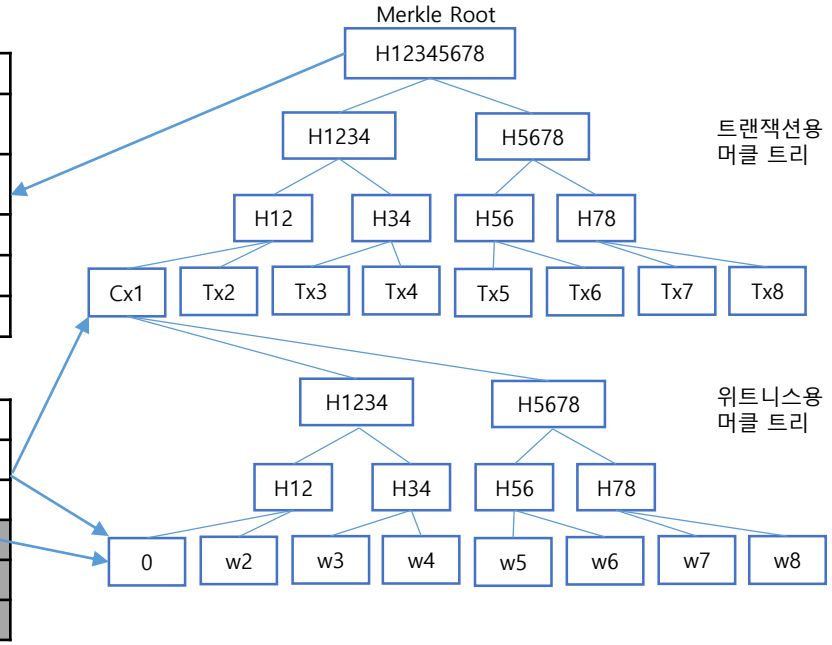
Tx의 해시값 계산으로 머클 트리를 만들고 머클 루트를 블록 헤더에 기록한다.
또한, 이 블록이 연결될 이전 블록의 해시 값을 계산해서 블록 헤더에 기록한다.
+ 이 블록이 연결될 이전 블록 해시 값을 계산해서 블록 헤더에 기록한다
이 상태에서 현재 Bits 값이 제시하는 조건에 맞는 논스 값을 찾아 블록 헤더에 기록하면 블록 완성

블록 데이터 구조

블록 바디에는 채굴자 선택 트랜잭션과 코인베이스 트랜잭션이 들어있다.
트랜잭션의 전자서명 부분은 분리되어 witness 영역에 따로 저장된다.

Version	4 bytes	20400000
Previous Block Hash	32 bytes	00000000000000000037fb8b5b05787887a6391eb66...
Merkle Root	32 bytes	5093d58359e203f2c296f0307f5f399f6c9e514381e4d1...
Timestamp	4 bytes	5c3303b7
Bits	4 bytes	173218a5
Nonce	4 bytes	c539cb45

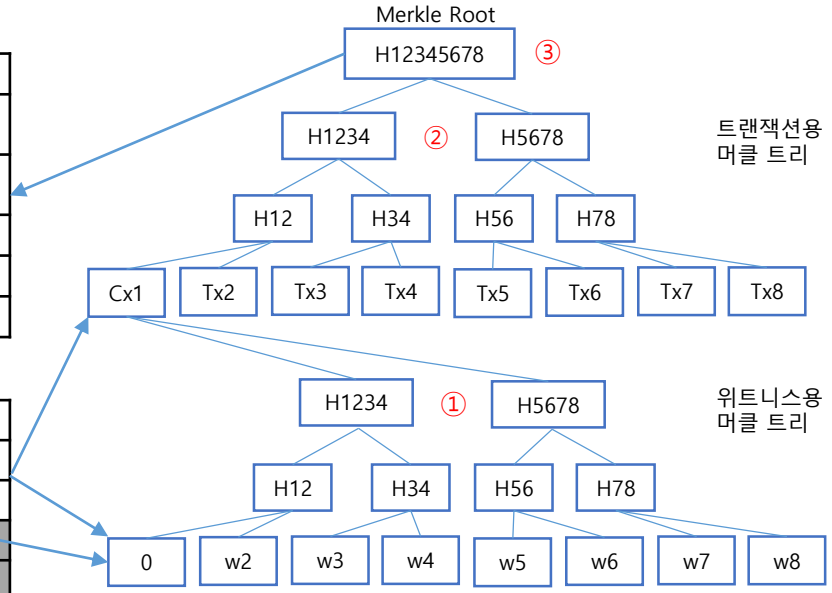
Transaction Count	4 bytes	0536
Coinbase transaction		
Transactions		
Witness		
...		
Witness		



채굴 – 블록 데이터 구조

Version	4 bytes	20400000
Previous Block Hash	32 bytes	00000000000000000037fb8b5b05787887a6391eb66...
Merkle Root	32 bytes	5093d58359e203f2c296f0307f5f399f6c9e514381e4d1...
Timestamp	4 bytes	5c3303b7
Bits	4 bytes	173218a5
Nonce	4 bytes	c539cb45

Transaction Count	4 bytes	0536
Coinbase transaction		
Transactions		
Witness		
...		
Witness		

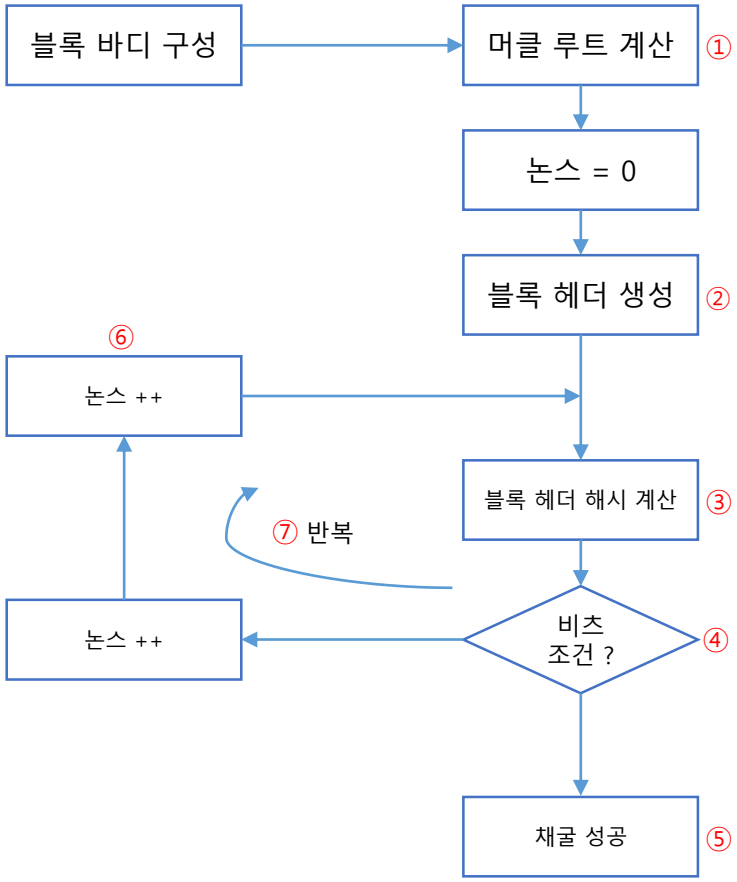


1. **트랜잭션들의 전자서명**인 위트니스의 해시 값을 계산하여 **위트니스용 머클트리**를 만든다. 맨 아래 노드의 왼쪽부터 (w2부터) 오른쪽으로 해시 값을 기록한다. 맨 왼쪽 값은 '0'이다. 이것은 코인베이스 트랜잭션 자리이며, 코인베이스는 전자서명이 없으므로 '0'을 채워 넣는다. 머클 트리는 **이진 트리**이므로 **맨 아래 노드는 짝수 개**여야 한다. 만약 트랜잭션이 **홀수 개이면 마지막 노드를 복제해서 짝수 개**로 만든다. 맨 아래 노드를 두 개씩 합쳐서 또 해시를 계산하면서 위의 노드를 채워 나간다.

2. 트랜잭션들의 해시 값을 계산해서 **트랜잭션용 머클 트리**를 만든다. 만드는 절차는 위트니스용 머클 트리와 동일하다. 맨 첫 번째 노드인 Cx1은 위트니스용 머클 트리의 루트 노드와 코인베이스 트랜잭션을 합쳐서 만든 해시 값이다.

3. 머클 트리의 맨 위에 있는 노드가 머클 루트이고 이 값을 블록 헤더의 머클 루트 자리에 기록한다. 블록에 있는 모든 트랜잭션은 **머클 트리에 요약**돼 있으므로 블록이 생성된 이후에는 변경이 불가하다. 일부 트랜잭션의 **일부 데이터라도 바뀌면 머클 루트 값이 완전히 바뀐다**.

채굴 과정



1) 선택한 트랜잭션들을 모아 블록 바디를 구성하고, 머클 루트를 계산한다.

2) 블록 헤더를 생성한다. 버전, 이전 블록 해시, 머클 루트, 타임스탬프, Bits를 기록한다. 논스 값은 아직 모르므로 0을 기록한다.

3) 블록 헤더의 해시 값을 계산한다.

4) 해시 값이 **Bits 조건**에 맞는지 확인한다.

5) 해시 값이 Bits 조건에 맞으면 채굴에 성공한 것이다.

6) 해시 값이 Bits 조건에 맞지 않으면 논스 값을 1만큼 증가시킨 후 블록 헤더를 변경한다.

7) 3,4 과정을 반복한다.

수많은 반복을 위해 해시 계산 능력이 빨라야 한다.
이를 위해 메이저급 채굴자들은 ASIC 장비를 동원한다.
ASIC 3~7 과정에 해당하는 프로그램을 전용 칩으로 만든 것이다.
논스는 4바이트로 최대값이 약 42.9억 정도다.
만약 모든 논스를 다 적용해도 Bits 조건을 만족하지 못하면
다른 방법으로 논스 값을 찾아야한다(추후 다룸).

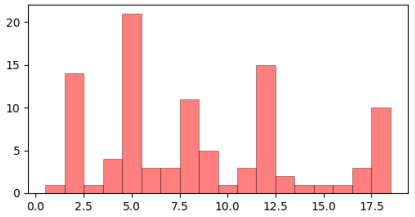
[실습] 채굴자 지갑 분포 확인

Blockchain.info API 서비스를 통해 최근 생성된 100개 블록을 읽고, 첫번째 트랜잭션인 코인베이스 트랜잭션을 읽는다.
코인베이스 출력부에 기록된 채굴자 지갑 주소를 수집하고, 주소 분포를 확인하여 어떤 지갑이 가장 많이 채굴했는지 확인 가능하다.

```
# 서드파티 API 서버로부터 코인베이스 트랜잭션을 수집해서
# 채굴자의 지갑을 관찰한다. 최근 몇 개의 블록만 조회해 본다.
import requests
import time
import pandas as pd
import matplotlib.pyplot as plt

# 마지막 블록 번호를 조회한다
resp = requests.get(url='https://Blockchain.info/latestblock')
data = resp.json()
nHeight = data['height']

# 마지막으로부터 몇 개 블록을 읽어서
# 코인베이스 트랜잭션의 지갑 주소를 수집한다
mining = []

for n in range(nHeight, nHeight-100, -1):
    url = 'https://Blockchain.info/block-height/' + str(n) + '?format=json'
    resp = requests.get(url=url)
    data = resp.json()
    block = data['blocks'][0]

    stime = block['time']
    addr = block['tx'][0]['out'][0]['addr']
    value = block['tx'][0]['out'][0]['value']

    ts = time.gmtime(stime)
    date = time.strftime("%Y-%m-%d %H:%M:%S", ts)

    # 결과를 리스트에 저장한다
    mining.append([date, addr, value])

    # 중간 결과를 표시한다
    print("#%d : %s₩t%s₩t%f" %(n, date, addr, value*1e-8))
```

```
# 결과를 데이터프레임에 저장한다
df = pd.DataFrame(mining, columns=['Date', 'Address', 'Reward'])

# 같은 지갑끼리 묶어본다
grp = df.groupby('Address').Address.count()
print()
print(grp)

# Histogram
plt.figure(figsize=(6,3))
plt.title=("Miner's Address")
x = list(range(1, len(grp.values)+1))
plt.bar(x, grp.values, width=1, color="red", edgecolor='black', linewidth=0.5, alpha=0.5)
plt.show()
```

실행 결과




```
#687032 : 2021-06-10 08:05:19   18cBEMRxXHqzWCxZNtU91F5sbUNKhL5PX      6.444627
#687031 : 2021-06-10 08:04:09   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY      6.587541
#687030 : 2021-06-10 07:59:20   1Bf9sZvBHPFGVPX71WX2njhd1NXKv5y7v5      6.441692
#687029 : 2021-06-10 07:57:25   18cBEMRxXHqzWCxZNtU91F5sbUNKhL5PX      6.587762
#687028 : 2021-06-10 07:53:38   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY      6.461234
#687027 : 2021-06-10 07:52:27   1PQwtwajfHWyAkedss5utwBvULqbGocRpu      6.585347
#687026 : 2021-06-10 07:50:34   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY      6.711318
#687025 : 2021-06-10 07:28:51   18cBEMRxXHqzWCxZNtU91F5sbUNKhL5PX      6.653559
#687024 : 2021-06-10 07:11:40   1CK6KHY6MHgYvmRQ4PAafKYDrg1ejbH1cE      6.629245
#687023 : 2021-06-10 07:02:40   bc1qx9t2l3pyny2spqpqlye8svce70npppwtaxwdrp4      6.491192
#687022 : 2021-06-10 06:58:28   12dRugNcdxK39288NjcDV4GX7rMsKCGn6B      6.556589
#687021 : 2021-06-10 06:48:47   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY      6.550312
#687020 : 2021-06-10 06:39:06   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY      6.572294
#687019 : 2021-06-10 06:29:12   12dRugNcdxK39288NjcDV4GX7rMsKCGn6B      6.541919
#687018 : 2021-06-10 06:22:06   1Bf9sZvBHPFGVPX71WX2njhd1NXKv5y7v5      6.689980
#687017 : 2021-06-10 06:05:53   1Bf9sZvBHPFGVPX71WX2njhd1NXKv5y7v5      6.462541
#687016 : 2021-06-10 06:05:20   1EepjXgvWUoRyVNvuLSAxjiq21QqKGDANLW      6.586701
#687015 : 2021-06-10 06:03:14   12QVFmJH2b4455YUHkMpEnWLeRY3e74Jb5      6.569835

Address
12QVFmJH2b4455YUHkMpEnWLeRY3e74Jb5            1
12dRugNcdxK39288NjcDV4GX7rMsKCGn6B           14
147SwRQdpCfj5p8PnfsXV2SsVVpVcz3aPq            1
17HzvxMhoke5K3oavLzPGkLy4zd3SKjWEb            4
18cBEMRxXHqzWCxZNtU91F5sbUNKhL5PX           21
191sNkkTGBpziUsNgZYXo7DH2odg3XDAGo            3
19dENFt4WvWos6XtgwStA6n8bbA57WCS58            3
1Bf9sZvBHPFGVPX71WX2njhd1NXKv5y7v5           11
1CK6KHY6MHgYvmRQ4PAafKYDrg1ejbH1cE            5
1E6v7Be9KLChSZqEkLoV2fH5syXMhKkna            1
1EepjXgvWUoRyVNvuLSAxjiq21QqKGDANLW            3
1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY           15
1PQwtwajfHWyAkedss5utwBvULqbGocRpu            2
35y82tEPDa2wm6tzkEacMG8GPPa7zbMj83            1
3D72db1KMCnj7FL7M8smxTw81c2bVu4UN5            1
3DPNFXGoe8QGiEXEApQ3QtHbBwM15VCQU3            1
bc1qf274x7penhcd8hsv3jcmwa5xxzjl2a6pa9pxwm     3
bc1qx9t2l3pyny2spqpqlye8svce70nppwtaxwdrp4    10
Name: Address, dtype: int64
```

가장 큰 지갑이 20% 정도를 차지한다. 상위 5개 지갑이 70% 정도 차지한다. 이들은 메이저급 마이닝 풀일 것이비다. 코인 채굴을 일부가 독점하고 있다. (문제)