



블록체인 기술의 이해

<http://bit.ly/2Vv390B>

1. 화폐, 비트코인, 블록체인
2. 블록체인 패러다임
3. 비트코인으로 알아보는 블록체인 원리
4. 스마트 컨트랙트, 이더리움, 하이퍼레저

The background features a gradient from green at the top to blue at the bottom. It is decorated with faint, semi-transparent circular patterns, some of which include concentric circles and arrows, suggesting a technical or scientific theme. A vertical scale with numerical markings is visible on the left side of the image.

시스템을 구축하기 위해 필요한 것들?

소프트웨어 – 서버, 개발도구 ...

데이터베이스가 필요하다.

- ① 데이터베이스를 만든다.
- ② 데이터베이스를 구매한다.

블록체인이 필요하다.

- ① 블록체인을 만든다.
- ② 블록체인을 구매한다.

블록체인 도입

- ① 기존 블록체인 이용
- ② 자체 블록체인 구축
- ③ 클라우드 서비스 이용 – BaaS

화폐 vs 암호화폐

• 거래의 기록

물물교환 → 귀금속 → 금화 → 지폐 → 전자화폐 → ???

현금통화 → 종이 장부의 기록 → 예금통화, 전산 처리, 트랜잭션만 발생
현금 없는 사회(cashless society)
화폐, 지불수단의 다양화



화폐 vs 암호화폐

- 비트코인의 등장

2008.10.31 사토시 나카모토라는 이름으로 암호학 메일링 리스트에 게시

<http://www.metzdowd.com/pipermail/cryptography/2008-October/014810.html>

“Bitcoin: A Peer-to-Peer Electronic Cash System”

2009.01.03 비트코인 네트워크 오픈, 블록 생성으로 사토시 나카모토 50 BTC 보상

할 피니, 사토시 나카모토와 이메일을 주고 받으면서 비트코인 베타 테스트
첫 번째 비트코인 거래 테스트 - 사토시 나카모토가 할 피니에게 10 BTC를 전송

<https://bitcointalk.org/index.php?topic=155054.0>

화폐 vs 암호화폐

Bitcoin and me (Hal Finney)

March 19, 2013, 08:40:02 PM

“When Satoshi announced Bitcoin on the cryptography mailing list, he got a skeptical reception at best. Cryptographers have seen too many grand schemes by clueless noobs. They tend to have a knee jerk reaction.”

“사토시가 암호학 메일링 리스트를 통해 비트코인을 발표했을 때 회의적인 반응이 전반적이었다. 그 동안 암호 전문가들은 아무것도 모르는 신참의 허황된 구상을 많이 경험했기 때문이다. 그래서 처음에는 무시하는 경향이 있다.”

“When Satoshi announced the first release of the software, I grabbed it right away. I think I was the first person besides Satoshi to run bitcoin. I mined block 70-something, and I was the recipient of the first bitcoin transaction, when Satoshi sent ten coins to me as a test. I carried on an email conversation with Satoshi over the next few days, mostly me reporting bugs and him fixing them.”

“사토시가 첫 소프트웨어 릴리즈를 발표하자마자 나는 즉시 그것을 받았다. 내 생각에 사토시가 비트코인을 시작할 때 곁에 있던 사람은 내가 처음이 아닐까 싶다. 나는 70 몇 개 되는 블록을 채굴했고, 사토시가 테스트를 위해 10 코인을 내게 보내면서 나는 비트코인 첫 번째 거래의 수취인이 되었다. 다음 며칠 동안 사토시와 이메일을 주고 받았고, 내가 버그를 알려주면 그가 수정했다.”

화폐 vs 암호화폐

“After a few days, bitcoin was running pretty stably, so I left it running. Those were the days when difficulty was 1, and you could find blocks with a CPU, not even a GPU. I mined several blocks over the next days. But I turned it off because it made my computer run hot, and the fan noise bothered me. In retrospect, I wish I had kept it up longer, but on the other hand I was extraordinarily lucky to be there at the beginning. It's one of those glass half full half empty things.”

“며칠이 지난 후 비트코인은 안정화 되었고 나는 그냥 지켜보기로 했다. 시작할 때 난이도는 1이었으며 GPU 없이도, CPU만으로 블록을 추가할 수 있었다. 그 다음 수 일 동안 몇 개의 블록을 더 채굴했다. 그러나 컴퓨터가 뜨거워지고, 또 팬 소리가 너무 거슬려서 꺼버렸다. 돌이켜보면 좀 더 채굴했었다더라면 하는 아쉬움이 남기도 하지만, 다른 한편으로는 그 출발점에 내가 있었다는 것은 굉장한 행운이었다. 그것은 생각하기 나름인 것이다.”

“The next I heard of Bitcoin was late 2010, when I was surprised to find that it was not only still going, bitcoins actually had monetary value. I dusted off my old wallet, and was relieved to discover that my bitcoins were still there. As the price climbed up to real money, I transferred the coins into an offline wallet, where hopefully they'll be worth something to my heirs.”

“비트코인 소식을 다시 들은 것은 2010년 말이었다. 아직도 돌아가고 있다는 사실 뿐만 아니라, 비트코인이 실제 화폐로서 가치를 가질 수 있다는 것에 놀랐다. 나는 오래된 지갑의 먼지를 털어낸 후, 내 비트코인이 아직도 그대로 남아 있는 것을 확인하고서는 안도했다. 가격은 계속 올랐기 때문에 나는 내 아이들에게 물려줄 만한 것이 되기를 바라면서 코인을 오프라인 지갑으로 옮겼다.”

화폐 vs 암호화폐

• 비트코인의 등장



“AFTER 10 YEARS, BITCOIN HAS CHANGED EVERYTHING – AND NOTHING”

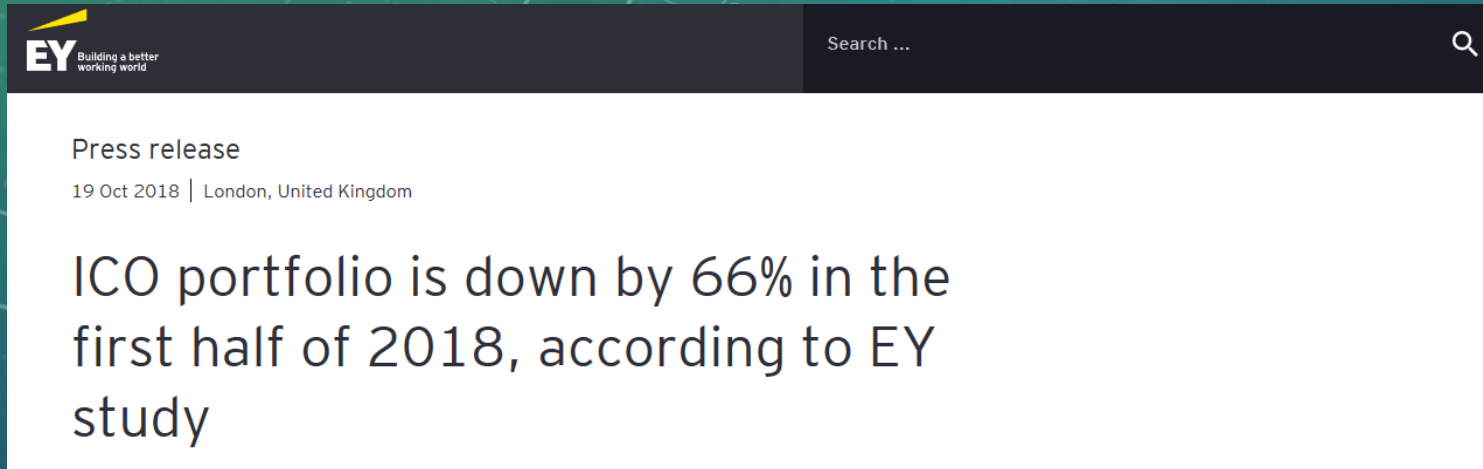
<https://www.wired.com/story/after-10-years-bitcoin-changed-everything-nothing/>

<https://coinmarketcap.com/currencies/bitcoin/>

암호화폐투자 – ICO vs STO vs IEO

- ICO = Initial Coin Offering
- STO = Security Token Offering
- IEO = Initial Exchange Offering

언스트앤영 회계법인에 따르면...



The screenshot shows the EY logo with the tagline 'Building a better working world' on the left. To the right is a search bar with the text 'Search ...' and a magnifying glass icon. Below the header, the text 'Press release' is followed by the date and location '19 Oct 2018 | London, United Kingdom'. The main headline reads 'ICO portfolio is down by 66% in the first half of 2018, according to EY study'.

- 2017년 ICO들 중 86% 정도가 가치 하락
- 초기 가치 대비 66% 수준으로 하락
- Ethereum platform remains dominant

https://www.ey.com/en_gl/news/2018/10/i-c-o-portfolio-is-down-by-sixty-six-percent-in-the-first-half-according-to-ey-study

비트코인

• 비트코인의 등장 배경

기존 중앙은행,
금융시스템에
대한 불신

인터넷, 모바일
정보기술의
발전

일상적인
지불수단의
변화

투자심리
(전무후무한
차익실현)



비트코인, 암호화폐에 주목

블록체인

- 블록체인, 분산 공유 시스템, 분산원장기술(DLT)

Bitcoin is just one application of Blockchain

Internet of Data(Information) → Internet of Value

“정보의 교환처럼 가치를 자유롭게 주고 받을 수 있는 기반을 제공”

거래의 신뢰성

유동성의 확대, 자산의 유동화

금융 인프라의 탈중앙화 → “De-Fi”

정보의 대중화 → 정보화 시대

금융의 대중화 → 글로벌 금융 서비스 시대

블록체인

여러 명의 이해관계자가 얽혀 절차가 복잡한 업무

(예) 수출입과정에서 각종 서류 작업에 10일 이상 소요, 신용장(letter of credit) 개설, 보험, 송장 송부, 대금 결제, 물품 반출 등

→ 블록체인을 도입하여 서로 신뢰할 수 있는 정보를 공유하고 시간과 비용 절감

→ 암호화폐를 통한 결제로 수수료 절감

환전 수수료 없는 글로벌 화폐의 기능

→ 다양한 혜택을 제공하는 결제 서비스 가능

자산의 유동성 증가(가치의 거래, 공유)

→ 보유한 부동산이나 주식, 만기전의 채권, 미술품 등을 은행을 거치지 않고 소유권 변동 없이 현금화하여 일정 기간 사용

→ 토큰화를 통해 모든 숨어있는 시장가치를 글로벌 디지털 거래소에서 거래할 수 있는 기반

→ "토큰 이코노미"

블록체인

- **블록체인 “Players”**

암호화폐 거래(Exchange): 거래소, 코인 분석, 투자상담, 브로커, 간접투자, ...

블록체인 솔루션: 블록체인 컨설팅, 솔루션 제공

블록체인 인프라: 퍼블릭 블록체인 플랫폼 개발(이더리움, EOS, ...)

기업 블록체인: 산업 전반에 블록체인 기술을 적용하려는 기업들의 커뮤니티
(Hyperledger, Enterprise Ethereum Alliance)

클라우드 서비스(BaaS)

공유경제를 위한 플랫폼: 잉여 자원, 소비의 공유, 공익 추구, 지역화폐

블록체인

• 블록체인 응용 분야

금융, 물류, 유통, 의료, IoT, 저작권, 게임, 신원인증, 결제, ...
→ 다양한 분야에서 접점을 찾는 것이 생각만큼 쉽지 않음.

IT가 적용될 수 있는 모든 비즈니스 분야?! (서로를 신뢰하지 않는 비즈니스 환경)

잠재력은 있지만 다양한 위험 요소들이 존재하고 어떤 비즈니스 프로세스에는 적합하지 않을 수 있음.

한국처럼 정부 중심의 전자정부시스템이나 금융 인프라가 잘 갖춰진 환경에서는 블록체인의 도입이 불필요하거나 경제성과 효율성이 떨어질 수 있음.

~(ツ)/~

블록체인

- Public vs Private

Public blockchain

자발적이고 건전한 참여를 유도하는 보상 메커니즘, 누구나 참여 가능
탈중앙화(Decentralization), 기득권, 검열, 통제 등으로부터 자유
관리자가 존재하지 않는 “trustless” 시스템(재단, 커뮤니티 등은 존재)

블록체인 자체는 탈중앙화를 지향하지만 외부 요인에 민감
→ 정부 또는 금융기업의 정책에 의존

이론적으로 “Finality”를 보장할 수 없음

블록체인

- Public vs Private

Private blockchain

“Permissioned”, “Enterprise”, “Consortium”

관리 주체가 존재하는 기업 컨소시엄 형태

특정 허가된 노드만이 블록을 검증

일반적으로 화폐기능이 필요하지 않음

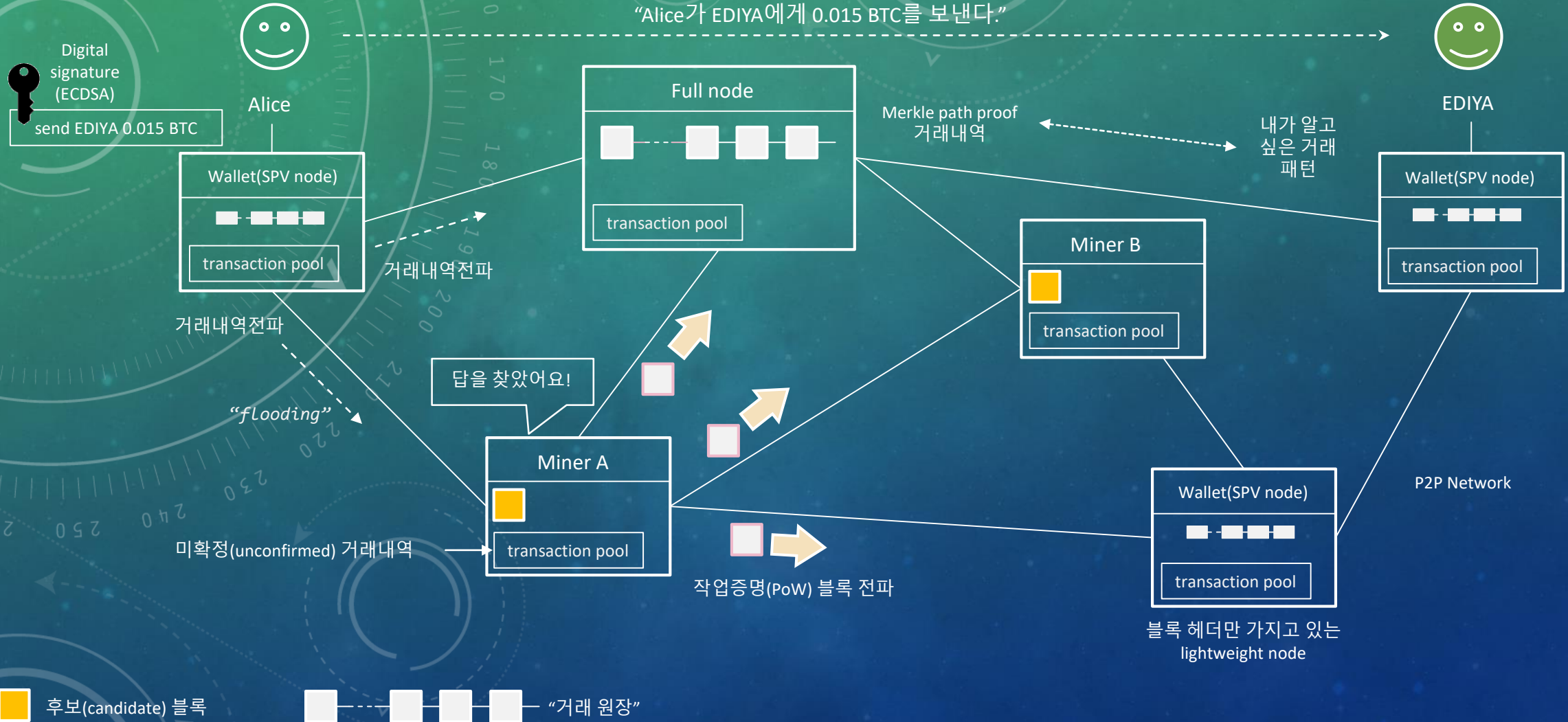
확실한 “Finality” 보장



HYPERLEDGER

비트코인

"Alice가 EDIYA에게 0.015 BTC를 보낸다."



비트코인

- ① 노드의 기능(역할)에 따라 크게 full, miner, SPV(wallet) 노드로 구분된다(하나 이상의 기능을 수행할 수도 있다).
- ② Alice는 0.015BTC를 EDIYA에게 보낸다는 거래에 자신의 전자서명을 첨부하여 비트코인 네트워크에 전송한다.
- ③ 거래 메시지는 P2P 네트워크를 통해 모든 노드에 전파되고(flooding) 기본적인 유효성을 검증한다.
- ④ 각 노드는 transaction pool에 거래 메시지를 풀링한다(아직 미확정 상태인 거래들이 모여있다).
- ⑤ miner 노드들은 transaction pool에서 거래 내역들을 후보 블록에 담고, 이 블록이 비트코인 블록체인에 추가될 수 있도록 정해진 조건에 만족하는 값을 찾을 때까지 연산을 수행한다(작업증명).
- ⑥ miner 노드들 중 만족하는 값을 가장 먼저 찾은 miner 노드는 후보 블록을 P2P 네트워크에 전파한다(작업증명에 성공하면 보상이 따른다).
- ⑦ 각 노드는 전파된 블록의 유효성을 검증한다. 다른 miner 노드들은 이미 새 블록을 받았으므로 그 뒤를 이을 새로운 후보 블록을 만드는 작업을 계속 수행한다.
- ⑧ full 노드는 원장에 해당하는 비트코인 블록체인에 유효성이 검증된 새 블록을 추가한다.
- ⑨ EDIYA의 wallet 노드는 원장을 가지고 있지 않으므로 Alice가 0.015BTC를 자신에게 전송했다는 거래 사실을 확인하려면 full 노드에게 원장에 해당 거래가 있는지 확인해야 한다.
- ⑩ full 노드는 EDIYA wallet 노드의 확인 요청에 대한 응답으로 해당 거래를 검증할 수 있는 정보(merkle path, 거래정보)를 리턴 한다.
- ⑪ wallet 노드는 리턴 받은 정보를 사용하여 해당 거래의 유효성을 검증한다(merkle path proof).

비트코인

해쉬(Hash) 함수

```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import hashlib
>>> a = hashlib.sha256("Alice gave 0.015BTC to EDIYA")
>>> print(a.hexdigest())
81f35d72f061f88260e0c84f276306ec7673297b6c8142140894be0b0c854ad9
>>> a = hashlib.sha256("Alice gave 0.016BTC to EDIYA")
>>> print(a.hexdigest())
B4777beb9a4dbe34da64984012e09d2ff9767102e3f91c7c237ca142b285dd16
>>>
```

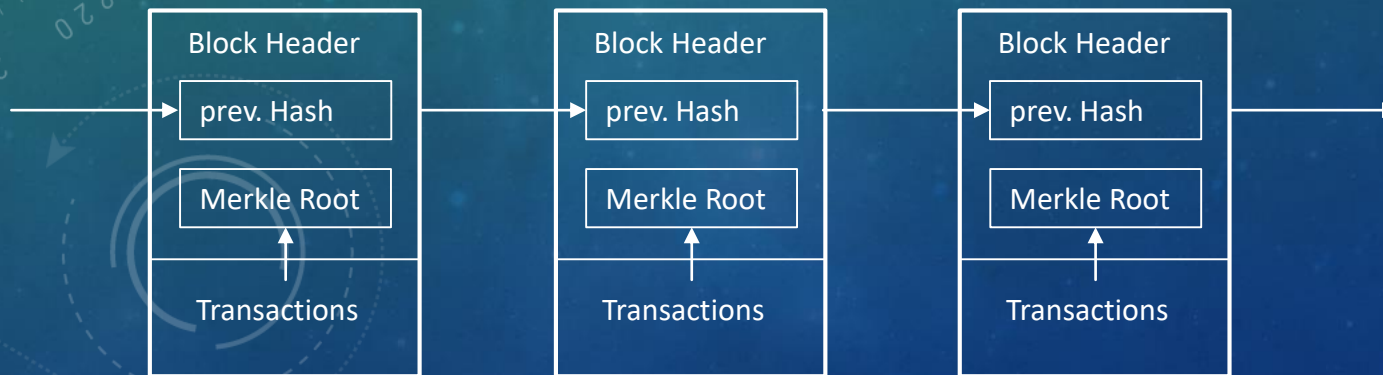
비트코인



$\text{SHA256}(\text{SHA256}(\text{[Block Header + Transactions]})) = \text{block hash 32 bytes (16진수 64자리)}$

000000000019d6689c085ae165831e934fff763ae46a2a6c172b3f1b60a8ce26f

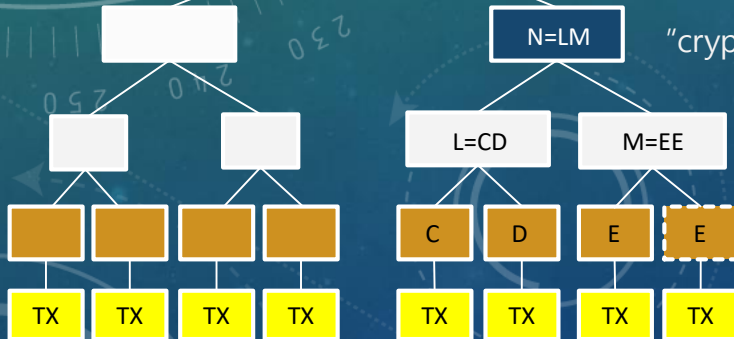
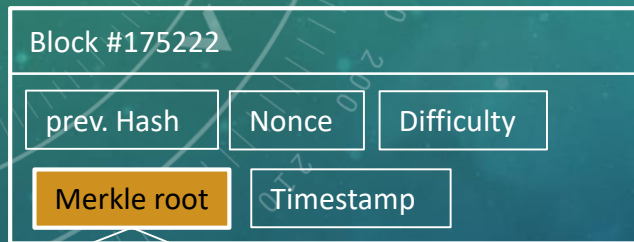
블록 해쉬 값은 헤더에 저장되지 않고 노드에 인덱스 형태로 저장
각 노드들은 블록 해쉬를 계산하여 블록의 유효성을 검증



비트코인

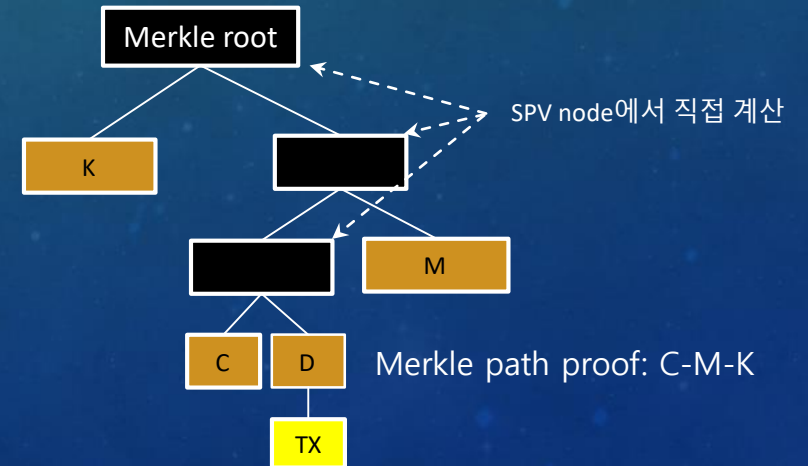
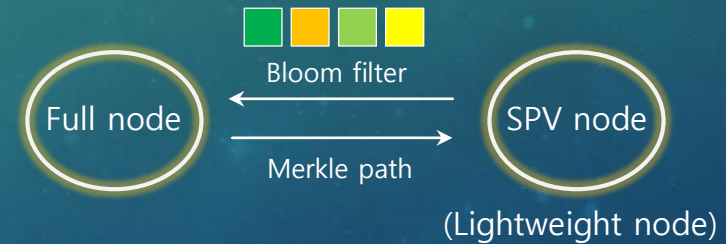
머클트리

- Merkle tree: 블록이 저장하고 있는 모든 거래 내역을 Merkle tree(binary hash tree)를 만들고 Merkle root를 헤더에 저장한다. 거래의 존재 여부는 Merkle path 정보를 이용하여 해쉬 값을 계산한 후 Merkle root와 일치하는지 비교한다.
- Lightweight 노드(SPV 노드, Simplified Payment Verification)는 블록 헤더만 가지고 있지만 Merkle path를 통해 거래의 존재 여부를 확인할 수 있다.



"cryptographically authenticated data structure"

거래내역을 해쉬한 leaf 노드로 부터 2개씩 짝을 지어 double SHA256으로 해쉬하여 노드가 하나(root) 남을 때까지 반복한다.

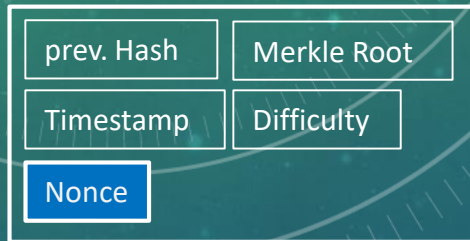


비트코인

채굴과 합의(Mining, Consensus)

채굴(Mining)이란 작업증명(Proof of Work, PoW)이라는 합의를(consensus) 통하여, 거래 내역들을 확정(confirmation)하고 이를 메인 블록체인에 추가하는 것을 말한다.

비트코인 합의 알고리즘 PoW = 조건에 맞는 숫자를 찾는 일 "Brute Force algorithm"

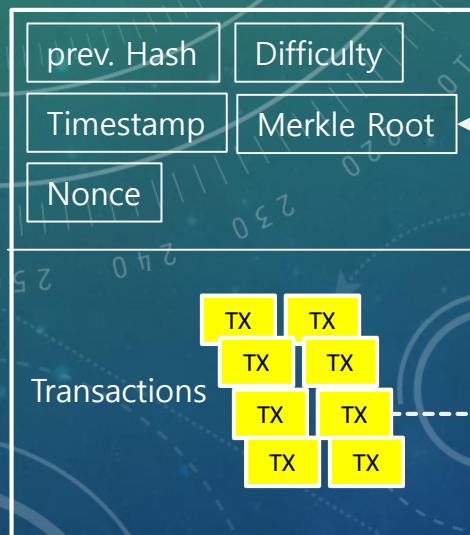
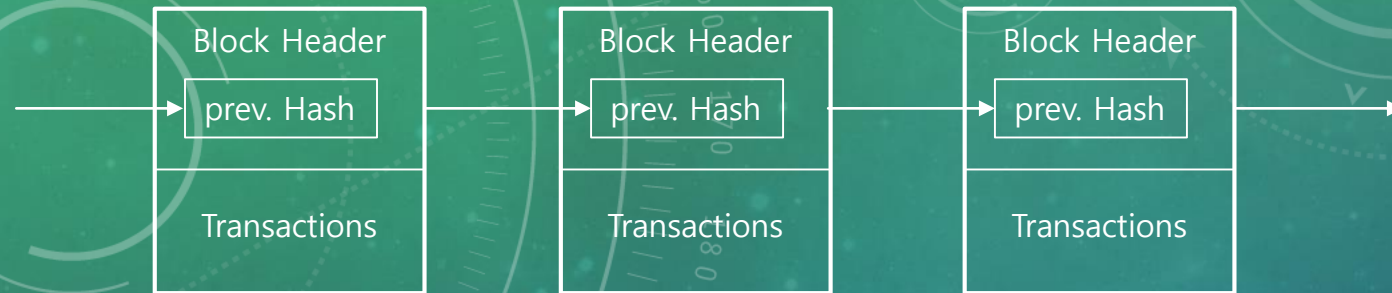


Nonce를 제외한 나머지 값들은 상수이므로 Nonce를 바꾸면서 목표 값을 찾는다.
찾아낸 값이 그 블록의 블록 해시가 된다.

$\text{SHA256}(\text{SHA256}(\text{Nonce})) = \text{block hash 32 bytes (16진수 64자리)} \leq \text{난이도 (Difficulty target)}$

hash power, hash rate = 초당 해쉬 값을 계산하는 속도

(예) 비트코인 네트워크 해시율 51,792,981,409 GH/s → 과도한 전력 소모



$$\text{SHA256}(\text{SHA256}(\text{ })) = \text{block hash} \leq \text{난이도(Difficulty target)}$$

"Merkling"

블록정보확인

<https://www.blockchain.com/>

#577004

버전 = 0x20000000

이전 블록의 블록해쉬 = 00000000000000000001113e97e93bd818554382f16d089a81371c6873a0b92b6

머클루트 = f765e73e351ef8ed6c0f09c2b2c48b87fee449ac41582193f757f0327a1d65e3

타임스탬프 = 2019-05-21 00:36:35 = 1558398995

난이도(Bits) = 388627269 = 1729FB45

Nonce = 3231819810 = C0A1A822

https://en.bitcoin.it/wiki/Block_hashing_algorithm

<https://github.com/codefoo/Blockchain/blob/master/blockhash.py>

#577004

난이도(Bits) = 388627269 = 1729FB45

난이도(Bits) = 388627269 = 1729FB45

Difficulty target = 29FB45 x 2^{8x(23-3)}

= 29FB45 x 2¹⁶⁰ = 29FB45 x 16⁴⁰

Difficulty target

[illegible]

40개의 0

Block hash

```
00000000000000000000000023a2e5875e10f0a18117192169f26855a8eb937a366afb
```

$$1 \times 2^4 = 10000$$

#577004

```
00000000FFFF00000000000000000000000000000000000000000000000000000  
0000000000000000000029FB45000000000000000000000000000000000000000
```

0x00ffff x 2²⁰⁸
0x29FB45 x 2¹⁶⁰

$$\frac{65535}{2751301} \times 2^{48} \approx 6704632680587.42$$

→ 난이도 1에 비하여 약 6조 7천억배 어렵다.

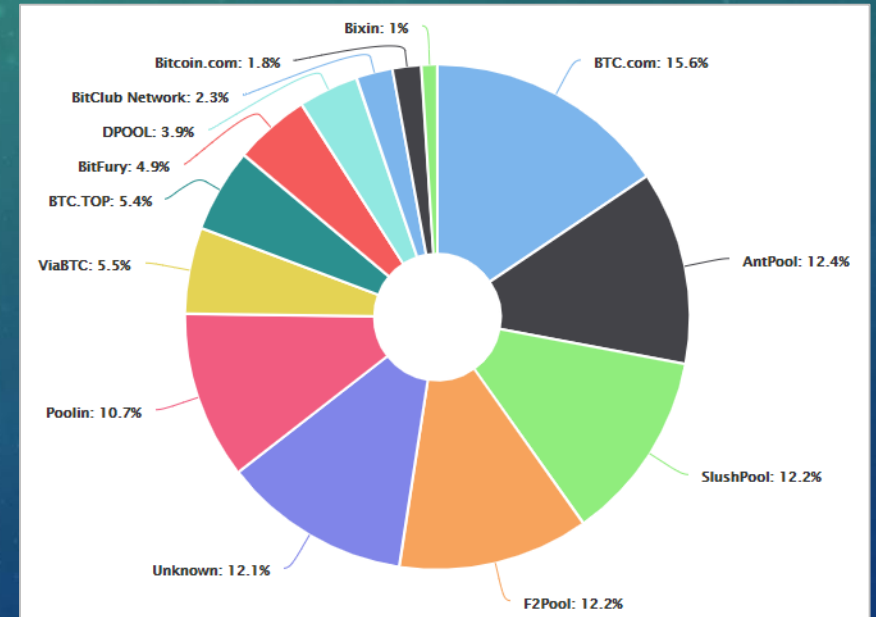
#492972

비트코인

비트코인과 같은 퍼블릭 블록체인은 보상(incentive)이 제공되어야 네트워크가 유지될 수 있다.
→ 채굴 보상 12.5 BTC, 블록 생성 주기는 약 10분



Bitcoin.com



비트코인의 동작

UTXO

- 비트코인의 거래는 거래되는 비트코인의 값만큼 소유자 변경 내역을 기록한다.

거래ID: 25			
INPUT	FROM	OUTPUT	TO
Joe	[10#2] 0.1000 BTC	#0 Alice's Addr	0.1000 BTC
		#1 Joe's Addr Tx Fee	0.0010 BTC 0.0005 BTC

“Spent”(previously unspent)

거래가 발생할 때마다 송금자는 수수료를 부담해야 하고 이 거래를 기록한 채굴자에게 보상(subsidy)으로 지급한다.

거래ID: 42			
INPUT	FROM	OUTPUT	TO
Alice	[25#0] 0.1000 BTC	#0 EDIYA's Addr	0.0150 BTC
		#1 Alice's Addr Tx Fee	0.0845 BTC 0.0005 BTC

“Unspent Output Transaction”(UTXO)

거래ID: 68			
INPUT	FROM	OUTPUT	TO
EDIYA	[42#0] 0.0150 BTC		

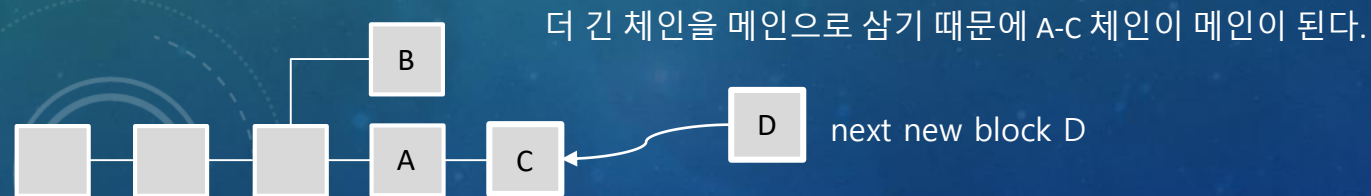
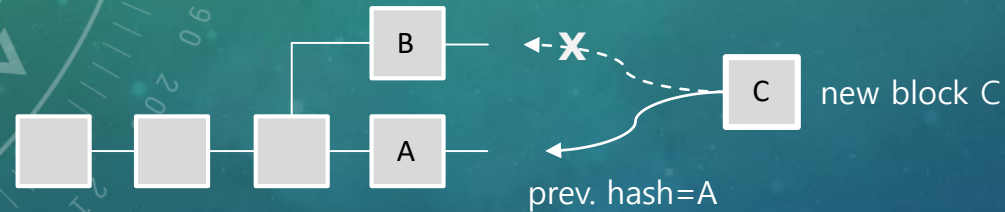
- Alice가 0.015 BTC를 EDIYA에게 지불하면 0.1 BTC는 “Spent”되고 지불금액과 잔액 두 개의 새로운 거래내역이 발생한다. Alice는 수수료를 제외한 0.0845 BTC 만큼의 “Unspent” BTC가, EDIYA는 0.015BTC “Unspent”가 생긴다.
- 종이 지폐를 자를 수 없듯이 UTXO는 나눌 수 없다. 작은 UTXO를 합쳐서 거래되는 값을 만들거나, 큰 UTXO를 내고 “잔돈”을 받아야 한다.
- 지갑(wallet) 소프트웨어는 블록체인으로 부터 해당 계정의 모든 UTXO를 찾아 “잔액”을 계산한다.

Alice’s UTXO(locking) → Alice’s digital signature(unlocking) → EDIYA’s UTXO(locking), Alice’s UTXO(locking)

비트코인의 동작

Fork

- 자연 발생적인 분기
채굴에 성공한 블록이 하나 이상이 발생했을 때 분기(splitting)될 수 있다. 그러나 더 긴 체인(다수 마이너들의 작업증명)이 메인이 되면서 해소된다. 분기되어 발생한 블록에 포함된 거래는 나중에(시간이 지난 후) 메인 체인에 포함된다.



비트코인의 동작

Fork

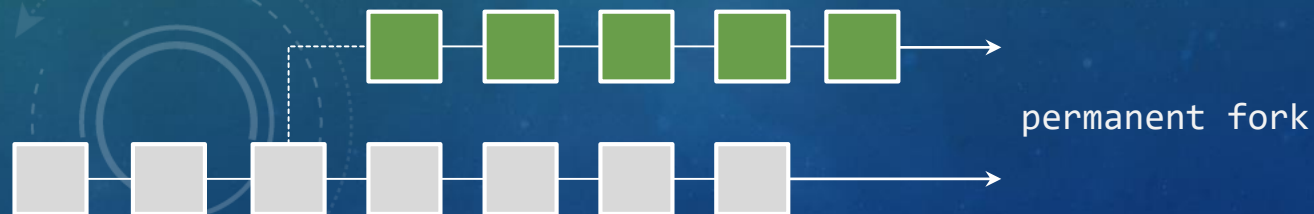
- Soft fork

분기 없이 이루어지는 소프트웨어의 부분적인 변경. 기존 노드와 업데이트된 노드들은 모두 새로운 규칙을 따르는 블록들을 유효한 블록으로 받아들인다. 그러나 업데이트된 노드들은 기존 규칙을 따르는 블록들을 거부한다. 다수의 노드들이 Soft fork에 찬성하는 경우 나머지 노드들도 따를 수 밖에 없다.



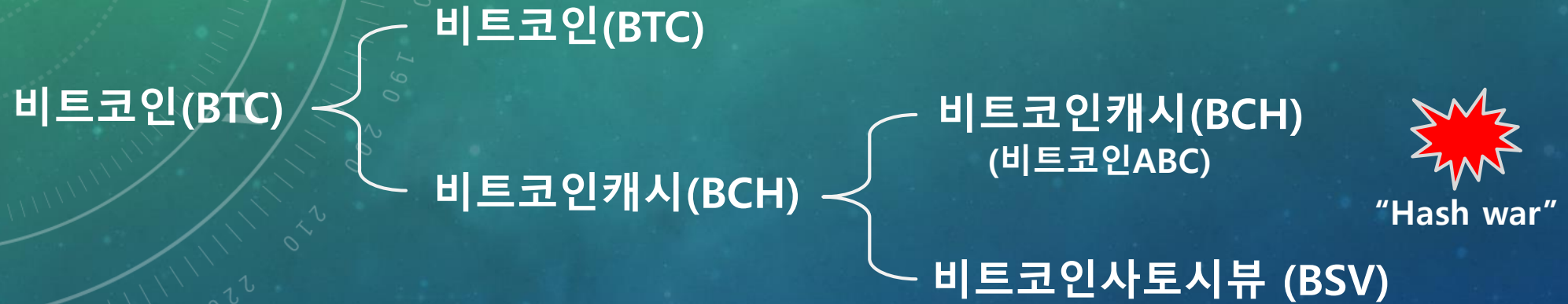
- Hard fork

분기가 발생하는 소프트웨어의 변경. 업데이트 노드에서 생성되는 블록들은 기존 노드에 의해 거부되고, 기존 노드에서 생성된 블록들은 업데이트된 노드에서 거부된다. 따라서 서로 독립적인 체인으로 진행된다. 결과적으로 다른 코인이 생기기도 한다(BTC vs BCH, ETH vs ETC).



비트코인의 동작

(예) 비트코인의 하드포크





비탈릭 부테린(Vitalik Buterin, 1994生)

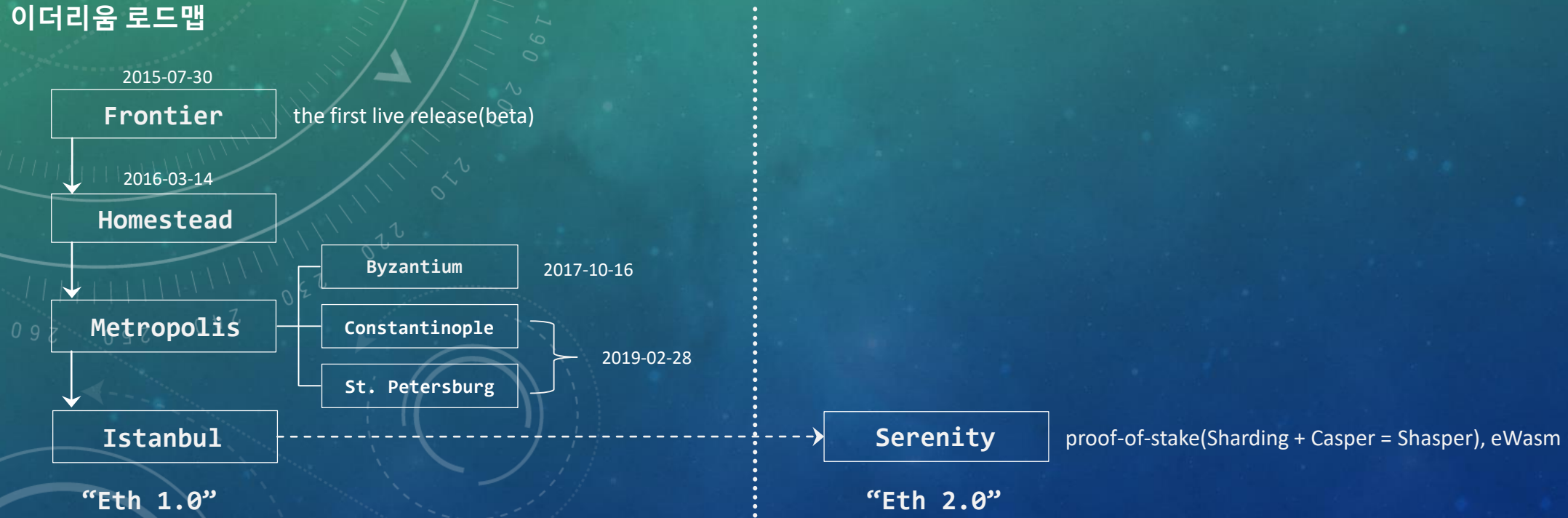
2011年 부터 비트코인 프로젝트에 합류했고 Bitcoin Magazine의 공동 창립자 중 한 사람이다.

애플리케이션 개발을 위한 스크립팅 언어를 도입하자는 그의 제안이 비트코인에서 받아들여지지 않자 몇몇 개발자들과 함께

2014年 부터 이더리움 프로젝트를 시작한다. crowdfunding으로 약 1,800만 달러(~30,000 BTC)를 모금하고(ICO: 1 BTC=2,000 ETH)

2015年 이더리움의 첫 번째 버전 “Frontier”를 발표한다.

이더리움 로드맵

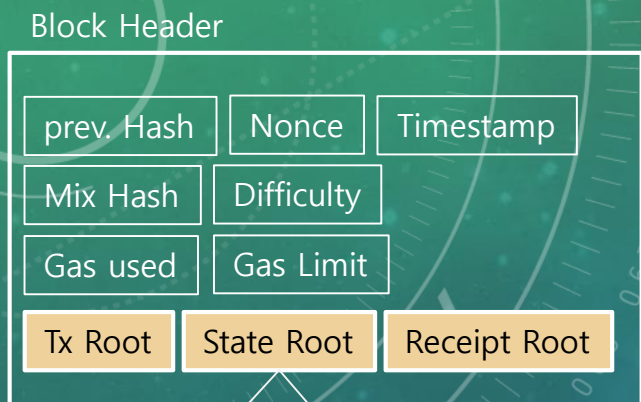


이더리움

비트코인 vs 이더리움

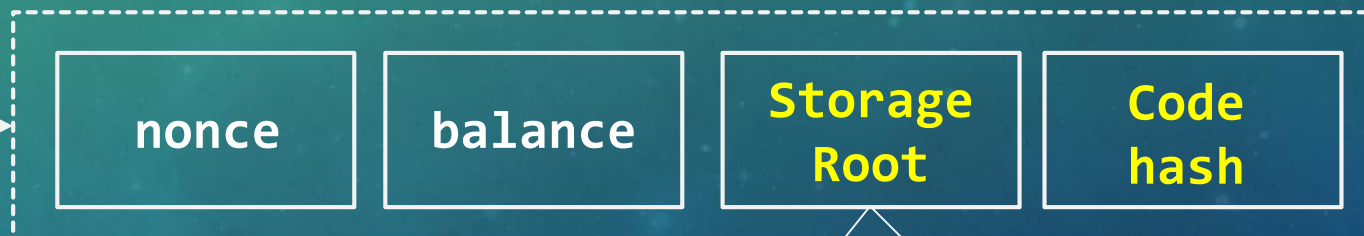
	비트코인	이더리움
블록생성	약 10분(5-7 TPS)	약 12-20초(10-15 TPS)
작업증명	double SHA256	Ethash, 장기적인 로드맵은 Proof of Stake(Casper)
인센티브	고정보상(12.5BTC)+수수료	고정보상(2ETH)+수수료+Uncle reference Uncle block에게도 보상, Gas fee
거래정보	Tx Root(Merkle tree)	Tx Root, State Root, Receipt Root(Merkle-Patricia tree)
자산	BTC	ETH
프로그래밍	Forth-like script (not Turing-complete)	Solidity, Serpent, Vyper(Turing-complete) DApp, Smart contract, Ethereum Virtual Machine
프로토콜	Bitcoin P2P	GHOST
계정	26-35 alphanumeric	20 bytes, External Owned Account, Contract Account
최대 발행	2,100만(현재 약 80%)	무제한

이더리움



State tree
(Merkle Patricia Tree)

Account



“UTXO-based” vs “Account-based”

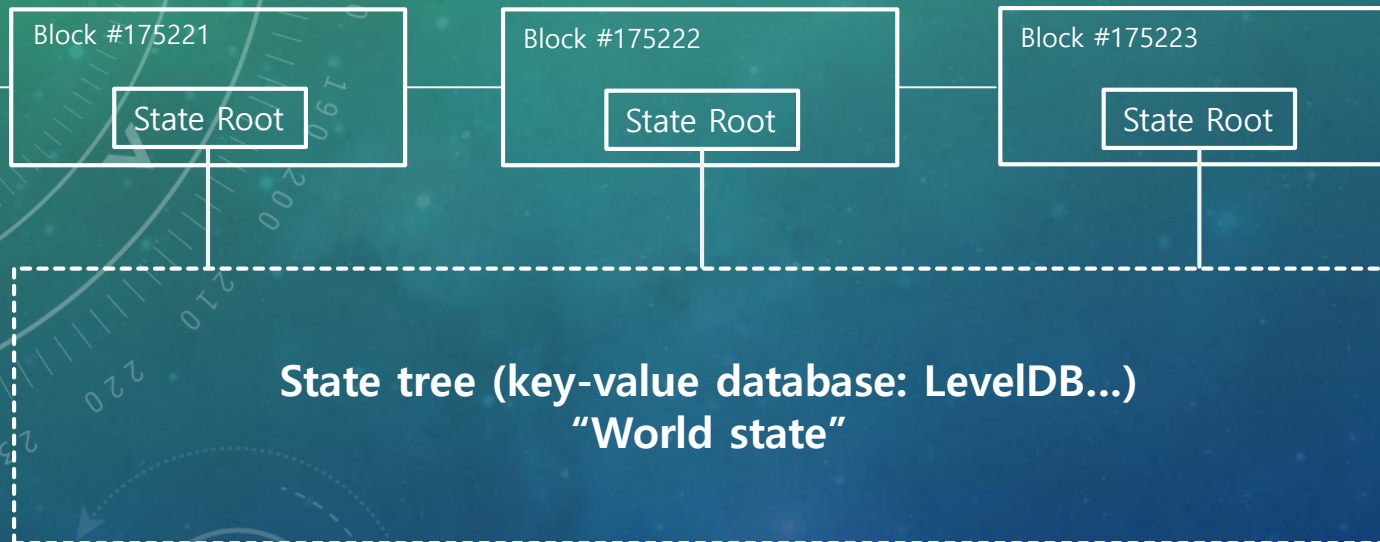
이더리움

nonce
balance
Storage Root
Code hash

```
[ <Buffer 01>,  
  <Buffer 06 d4 99 ec 6c 63 38 00 00>,  
  <Buffer 56 e8 1f 17 1b cc 55 a6 ff 83 45 e6 92 c0 f8 6e 5b 48 e0 1b 99 6c ad c0 01 62 2f b5 e3 63 b4 21>,  
  <Buffer c5 d2 46 01 86 f7 23 3c 92 7e 7d b2 dc c7 03 c0 e5 00 b6 53 ca 82 27 3b 7b fa d8 04 5d 85 a4 70> ]
```

`sha3('')` = "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470"

이더리움

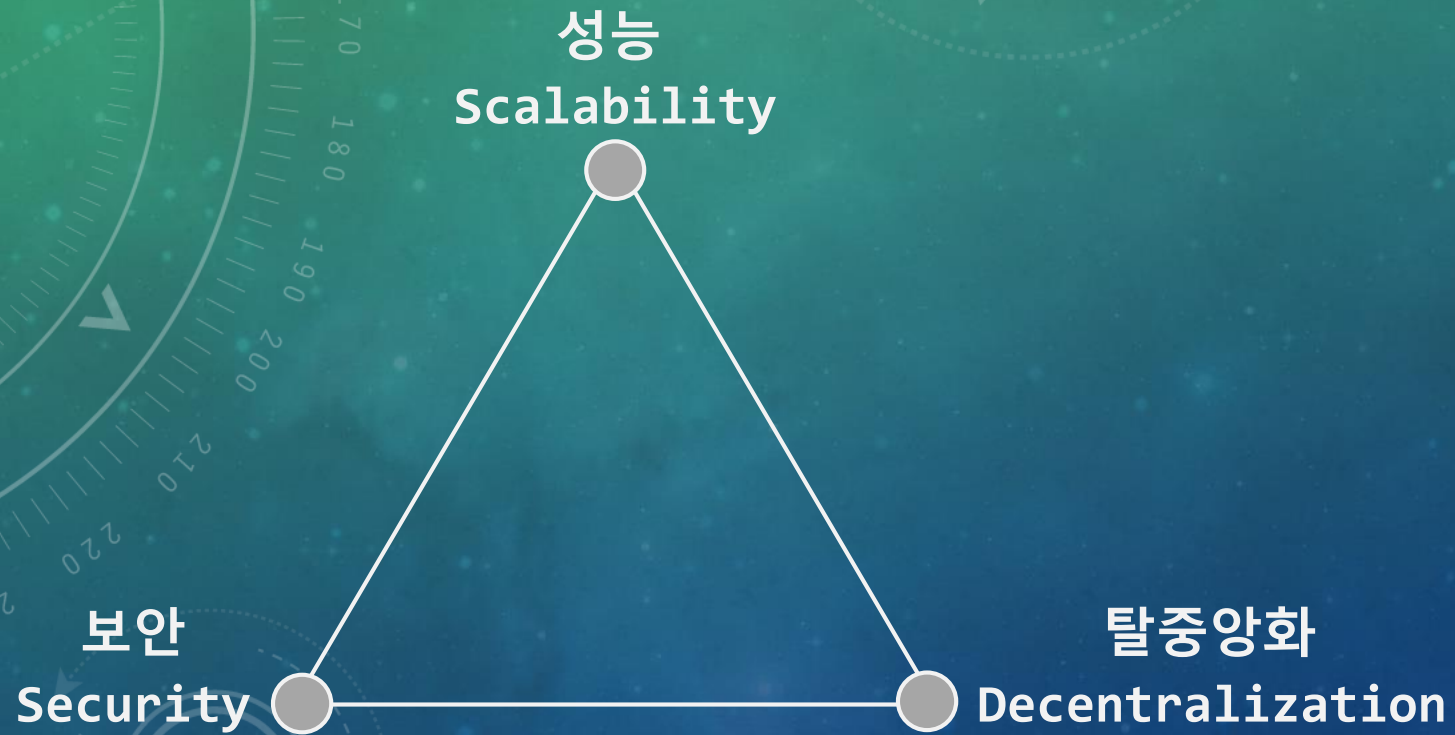


블록체인에 저장
변경되지 않음

로컬에 저장
트리를 새로 만드는 것이 가능
과거 데이터를 삭제할 수 있음
("tree pruning")

Trilemma

블록체인(public) 트릴레마



합의 방식(Consensus)

거래 내역들을 담은 블록을 네트워크를 구성하는 노드들이 검증하고 합의하는 과정
→ 원장에 영구 기록 여부(finality)를 결정하는 과정

과거 분산 시스템의 신뢰도를 보장(fault-tolerant)하려는 여러 가지 연구
→ 블록체인이 주목 받으면서 여러 형태의 합의 방식들이 제안됨

PoW(Proof of Work) – 비트코인, 이더리움

PoS(Proof of Stake) – 이더리움 2.0("Shasper")
DPoS(Delegated PoS) – EOS

PoA – 이더리움(Clique)
IBFT(Istanbul BFT) – Quorum
Raft – Quorum
PBFT(Practical BFT) – 코스모스(텐더민트)

합의 방식(Consensus)

- BFT(Byzantine Fault Tolerant)

BGP와 같은 상황에서 장애(오류, 실패, failure, fault)가 발생하더라도 정상 동작하도록 분산 시스템을 유지할 수 있으면 BFT 시스템(비잔틴 장애 허용 시스템)이라고 한다.

BGP = Byzantine Generals Problem

“A reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked – namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly as the Byzantine Generals Problem. We devote the major part of the paper to a discussion of this abstract problem and conclude by indicating how our solutions can be used in implementing a reliable computer system.”
(The Byzantine Generals Problem, 1982, LAMPORT, SHOSTAK, and PEASE)

- 분산 시스템의 장애를 BGP로 모델링하여 장애가 허용될 수 있는 상황을 제시
- 여러 전제와 가정 하에서 장애 허용이 가능한 알고리즘 제시

합의 방식(Consensus)

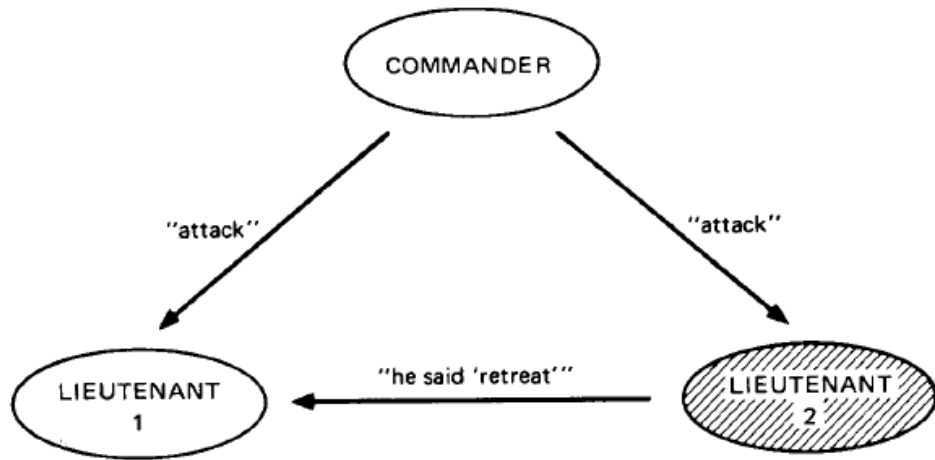


Fig. 1. Lieutenant 2 a traitor.

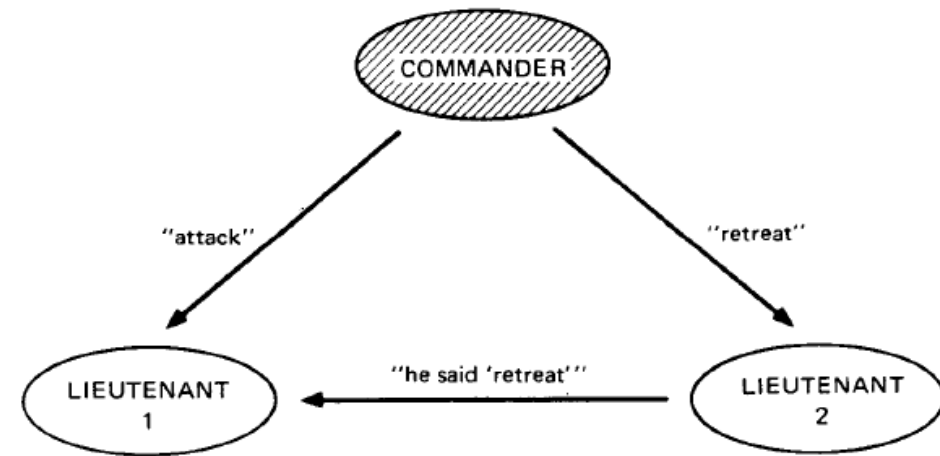


Fig. 2. The commander a traitor.

- 비잔틴 제국의 군대
COMMANDER = 총사령관
LIEUTENANT = 장군

- ① 총사령관의 명령은 메신저에 의해 전달되고 장군들끼리는 서로 직접 연락이 가능하다. 장군들은 총사령관의 공격/후퇴 명령을 받아 따라야 한다.
- ② 장군 중에는 배신자가 있을 수 있고 총사령관도 배신자일 수 있다.

이 경우에 다음 조건을 충족하는 합의에 이를 수 있는가?
(연합 작전을 성공하기 위한 조건)

- IC1. 배신자가 아닌 장군들은 모두 동일한 명령을 수행한다.
- IC2. 총사령관이 배신자가 아니라면 배신자가 아닌 모든 장군들은 명령에 복종한다.

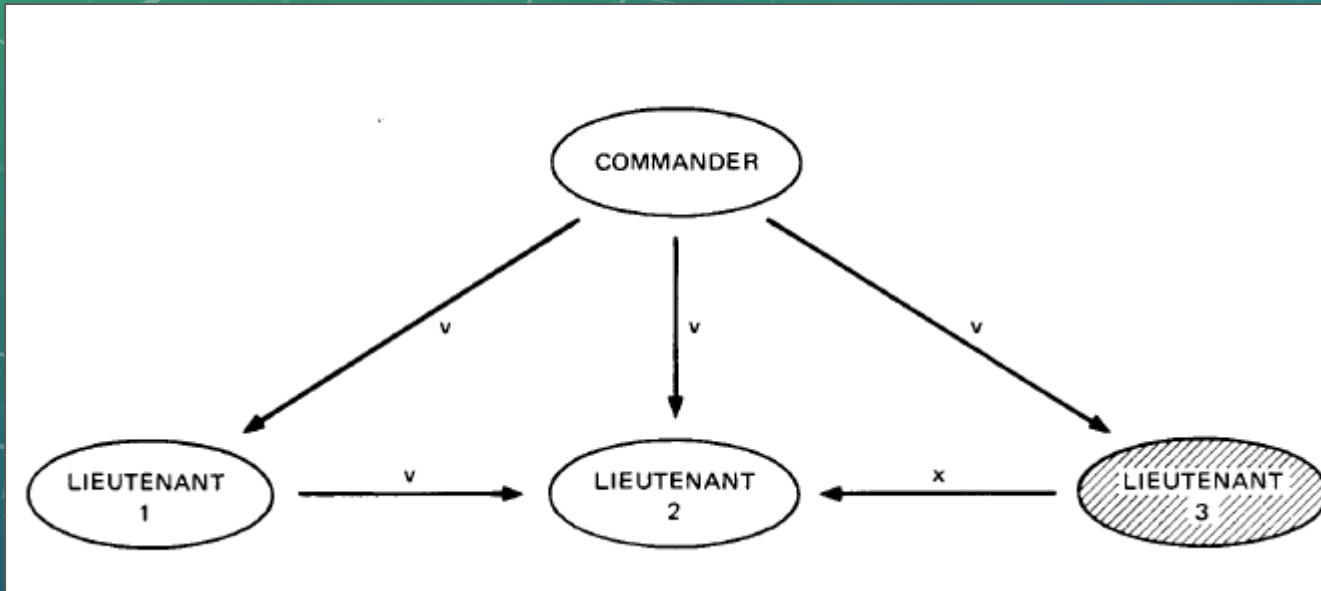
합의 방식(Consensus)

배신자가 m 명일 때 장군의 수가 $3m+1$ 미만이면 연합군은 작전에 성공할 수 없다.
→ m 개의 오류 노드가 발생해도 노드 수가 $3m+1$ 이상이면 장애 허용 가능

- (1) 전체 노드 수가 10개일 때 허용 가능한 비잔틴 노드의 최대 값은?
- (2) 전체 노드 수가 N 개일 때 허용 가능한 비잔틴 노드의 최대 값은?
- (3) 전체 노드 수가 $3m+1$ (m 은 비잔틴 노드) 일 때 정상(non-faulty) 노드의 수는?

합의 방식(Consensus)

전체 군대의 수가 4개이고 그 중에 하나가 비잔틴일 경우(장군3이 배신자)



$$[v, v, x] \rightarrow v$$

- A1. 모든 메시지는 조작되는 경우 없이 전달된다.
- A2. 수신자는 메시지 송신자를 알 수 있다.
- A3. 메시지가 오지 않았다는 것을 알 수 있다.

총사령관의 명령이 없으면 각 장군들은 후퇴 명령으로 판단한다.

알고리즘 OM(m), $m > 0$

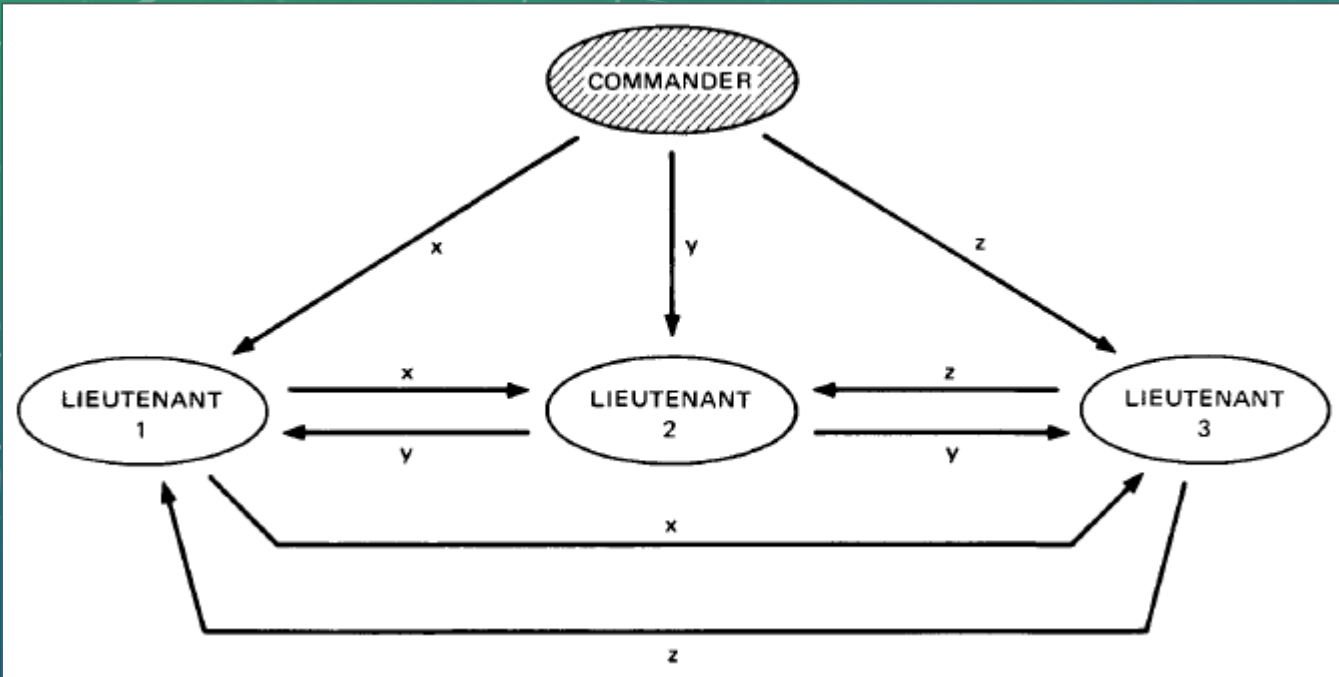
(1) 총사령관이 모든 장군에게 메시지를 보낸다.

(2) 장군은 받은 메시지에 따라 행동한다. 메시지를 받지 못한 경우 후퇴 명령으로 판단한다. 받은 메시지를 다른 장군들에게 보낸다.

(3) 각 장군들은 다른 장군들에게서 받은 메시지와 총사령관에게 받은 메시지를 종합하여 그 중에 다수(majority)를 차지하는 메시지에 선택하고 그에 따라 행동한다.

합의 방식(Consensus)

전체 군대의 수가 4개이고 그 중에 하나가 비잔틴일 경우(총사령관이 배신자)



(4) 다수가 없을 경우 중간 값을 선택한다.
(모두 동일한 순서대로 정렬할 수 있다)

결과적으로 모든 장군들이 동일한 합의에 도달한다.

장군1 $[x, y, z] \rightarrow y$

장군2 $[x, y, z] \rightarrow y$

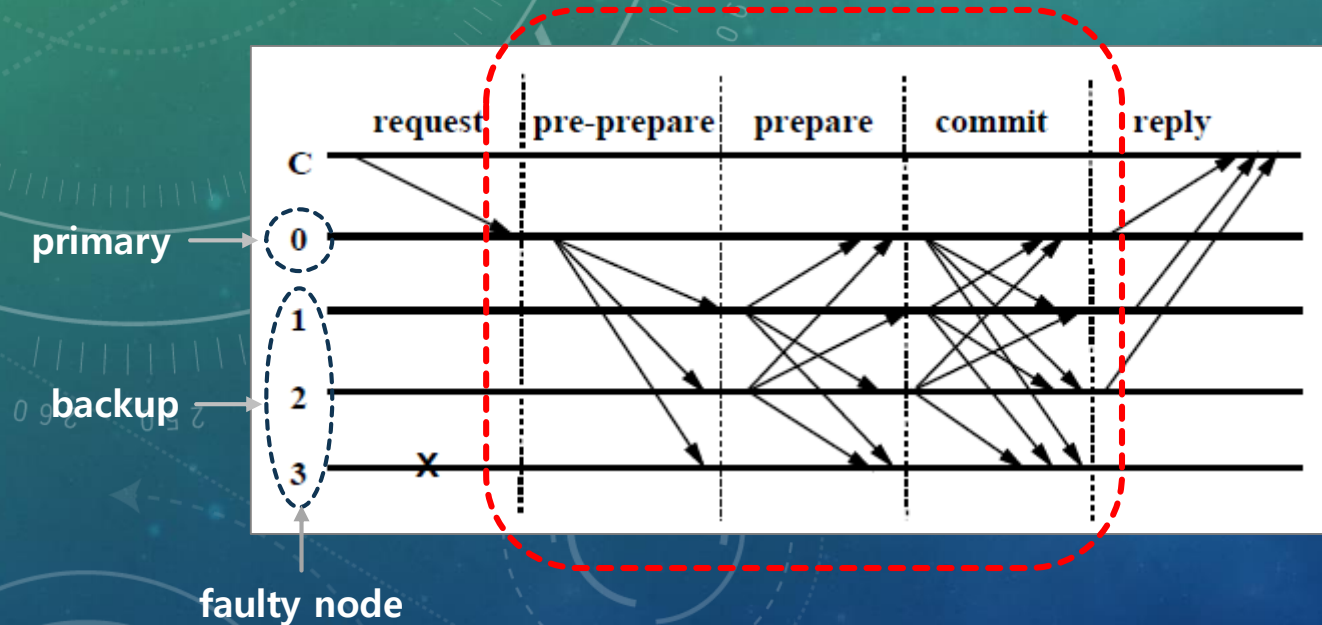
장군3 $[x, y, z] \rightarrow y$

합의 방식(Consensus)

- “Practical” BFT 현실에 좀 더 가까운 상황을 가정(비동기식)하여 BFT를 적용

BFT는 모든 노드들이 모든 메시지를 주고 받은 후(동기식)에 합의

PBFT는 4개의 단계(phase)를 거쳐서 합의(또는 3 phase, round), 오류 노드는 최대 $(N-1)/3$



- ① 클라이언트가 primary 노드에 요청 전송
- ② primary노드는 pre-prepare 메시지를 backup노드로 멀티캐스트
- ③ 각 backup노드는 pre-prepare 메시지 검증 후 prepare 메시지를 멀티캐스트
- ④ 각 노드는 자신이 받은 pre-prepare와 수신된 prepare 메시지를 매칭하여 검증("prepared")
→ 실행 순서 합의
- ⑤ commit메시지를 멀티캐스트
- ⑥ 각 노드는 자신이 받은 pre-prepare와 수신된 commit 메시지를 매칭하여 검증("committed-local")
→ 실행 합의
- ⑦ 각 노드는 클라이언트 요청 처리 후 클라이언트에게 결과 전송

합의 방식(Consensus)

- 불특정 다수가 참여하는 블록체인에서는 해시 파워로 경쟁하는 PoW 방식에서는 합의를 작업증명으로 대체하여 BGP를 해결(그러나 매우 큰 해시파워를 확보하는 경우 합의를 무효화할 수 있음 → 51% 공격)
- 소수의 노드가 합의를 이루는 구조에서는 소모적인 경쟁보다는 BFT 형태의 합의 방식이 적합할 수 있음
- 기업 블록체인에서는 BFT 스타일의 합의 방식을 선호

[참고] BaaS 를 통해 여러 합의방식 중에 선택할 수 있도록 제공하는 추세("pluggable" consensus)

Kaleido – PoA, Raft, IBFT

AWS – Hyperledger

Azure – Ethereum

루니버스 – LCA(PoA)



“open source blockchain for business”

하이퍼레저는 기업용 블록체인을 위한 여러 프로젝트, 개발도구들의 집합
금융, IoT, 물류, 제조, 기술 산업 등 여러 산업에 걸쳐 응용 가능한 블록체인 기술
오픈 소스 형태로 리눅스 재단에서 운영

<https://www.hyperledger.org/projects>



HYPERLEDGER
BURROW



HYPERLEDGER
FABRIC



HYPERLEDGER
GRID



HYPERLEDGER
INDY

하이퍼레저

- 하이퍼레저 패브릭

하이퍼레저 프로젝트들 중 핵심 프로젝트

블록체인 애플리케이션 플랫폼으로 plug-and-play 방식으로 모듈화

→ “modular and configurable architecture”

일반적인 프로그래밍 언어(Go, Java, Javascript)로 스마트 컨트랙트 작성이 가능

스마트 컨트랙트 = 체인코드

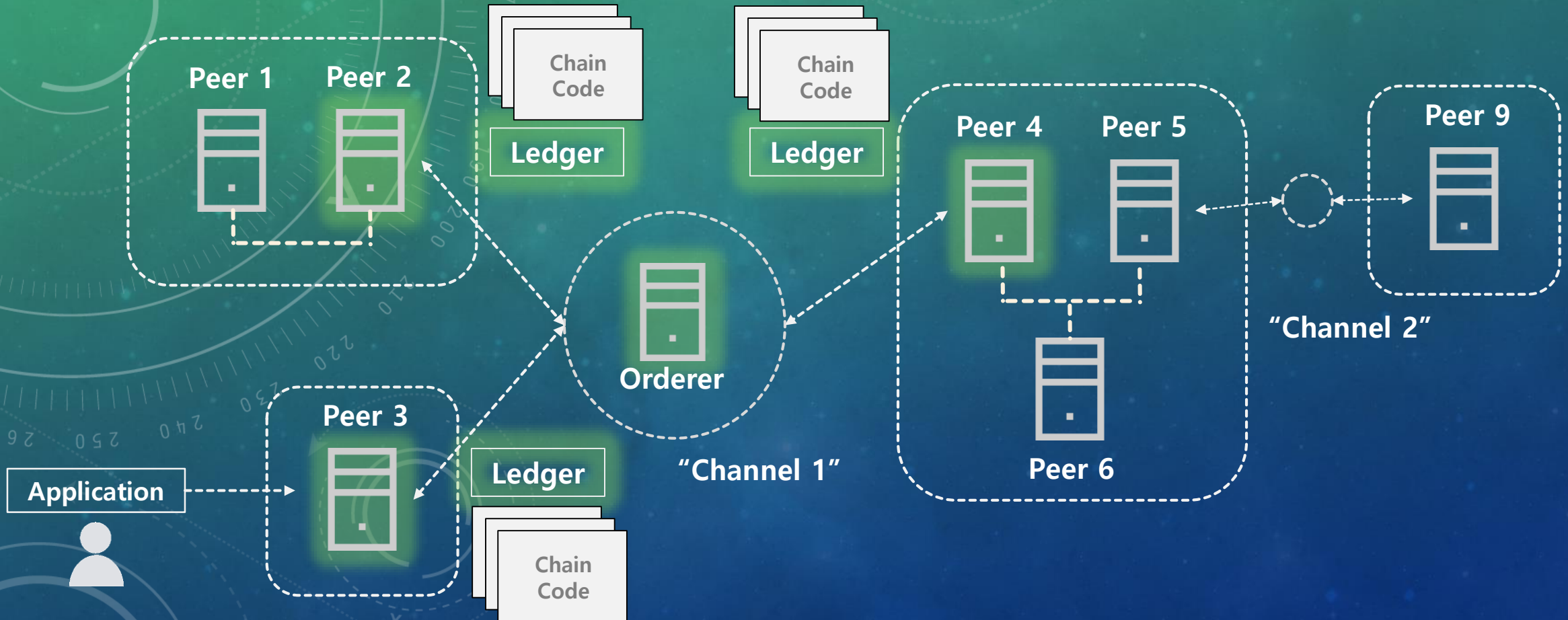
암호화폐 제거 : 비효율적인 채굴이나 트랜잭션 수수료 없음

→ “do not require a native cryptocurrency to incent costly mining or to fuel smart contract execution.”

인증된 클라이언트와 노드만이 참여 – Membership Service Provider (MSP)

하이퍼레저

- 하이퍼레저 패브릭



하이퍼레저

- 하이퍼레저 패브릭

트랜잭션 사이클

Execute → Order → Validate → Ledger Update



트랜잭션 순서(order)에 대한 합의 – “Kafka”, 분산시스템을 위한 고성능 메시지 처리 시스템

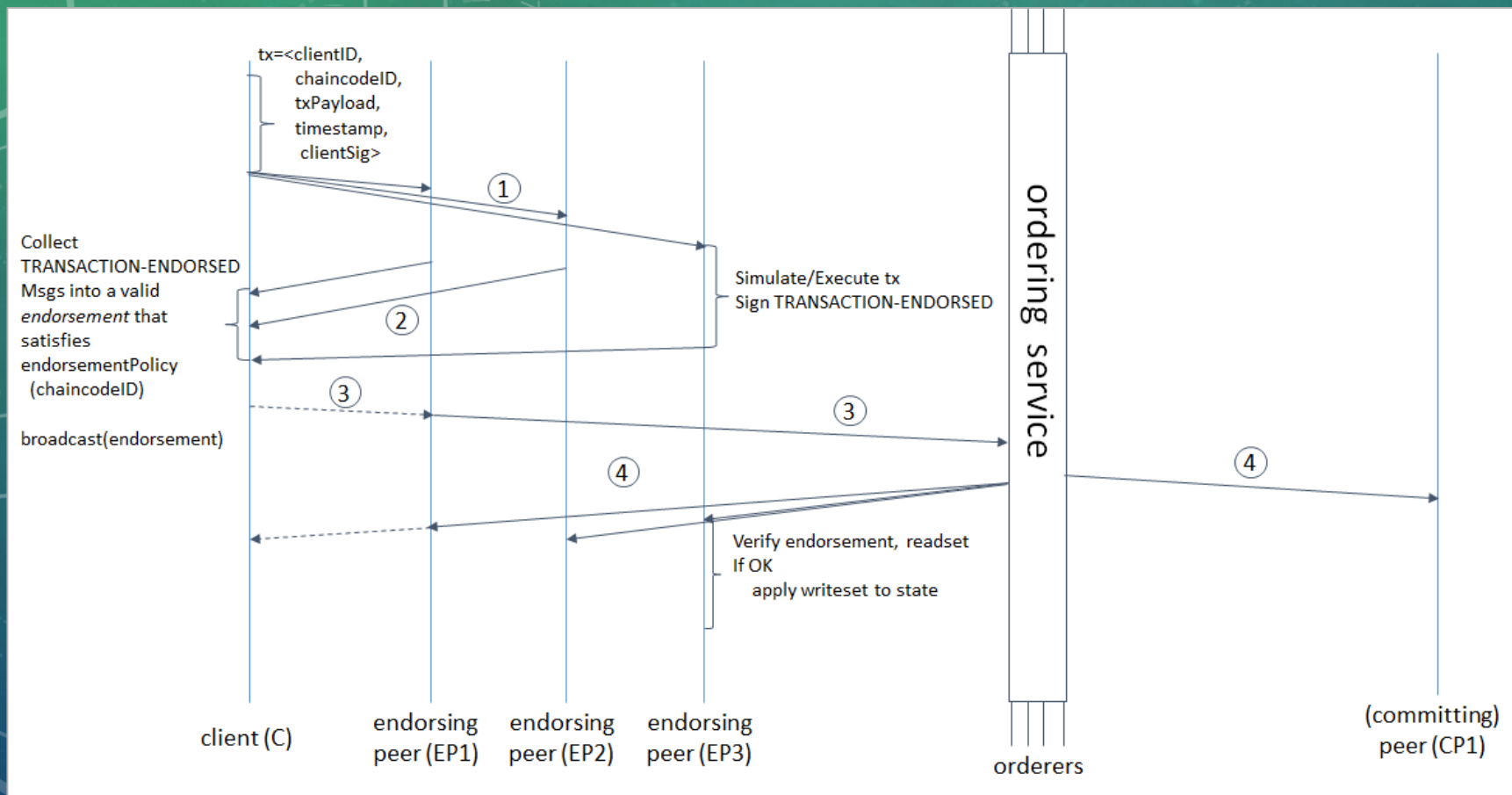
Apache Kafka® is a distributed streaming platform

하이퍼레저 패브릭에서 말하는 합의(Consensus)는 어떤 알고리즘을 의미하기 보다는 트랜잭션의 시작부터 확정되기 까지의 모든 과정(from proposal and endorsement, to ordering, validation and commitment)을 포괄한다.

“In a nutshell, consensus is defined as the full-circle verification of the correctness of a set of transactions comprising a block.”

하이퍼레저

• 하이퍼레저 패브릭



스마트 컨트랙트

- 블록체인 2.0
- Smart Contract의 개념은 암호학자 Nick Szabo가 1996년 Extropy(제 16호)라는 잡지에 “Smart Contracts: Building Blocks for Digital Markets” 라는 글을 기고하면서 소개되었다.
- Nick Szabo는 법에 의해 보호되고 그 이행을 법으로 강제하는 현실 세계의 계약을 사이버 시대, 온라인에서의 계약관계에 어떻게 적용할 수 있는가라는 문제로 부터 Smart contract의 개념을 도입했다. 그는 종이에 씌어진 계약보다 더 다양한 기능을 가지고 있다는 의미로 “smart”라고 명명했고, 계약이 모든 곳(hardware, software)에 내장될 수 있다고 보았다(embedded in the world). 예를 들어 자판기를 Smart contract의 원시적인 형태라고 생각했다.
- 기존의 계약을 사람들이 상호 합의하는 약속(a set of promises agreed to in a “meeting of the minds”)이라고 한다면 Smart contract는 그 약속들을 이행하는 당사자들이 지켜야 하는 프로토콜을 포함하는, 디지털 형태의 약속(a set of promises, specified in digital form, including protocols within which the parties perform on these promises)이라고 정의하고 있다.
- 그는 Smart contract가 계약을 이행(또는 위반)하면 프로토콜에 따라 자동으로 확인되며(verifiability), 계약의 이행은 스스로 이루어질 수 있고(enforceability), 그 내용이 안전하게 보호되며 제 3자의 개입을 차단하고(privacy), 계약의 이행 과정을 당사자들이 감독할 수 있게 되는(observability), 계약의 4가지 기본 요소를 충족할 수 있다고 생각했다.
- 블록체인은 Smart contract를 실현하기 위한 플랫폼을 제공한다.

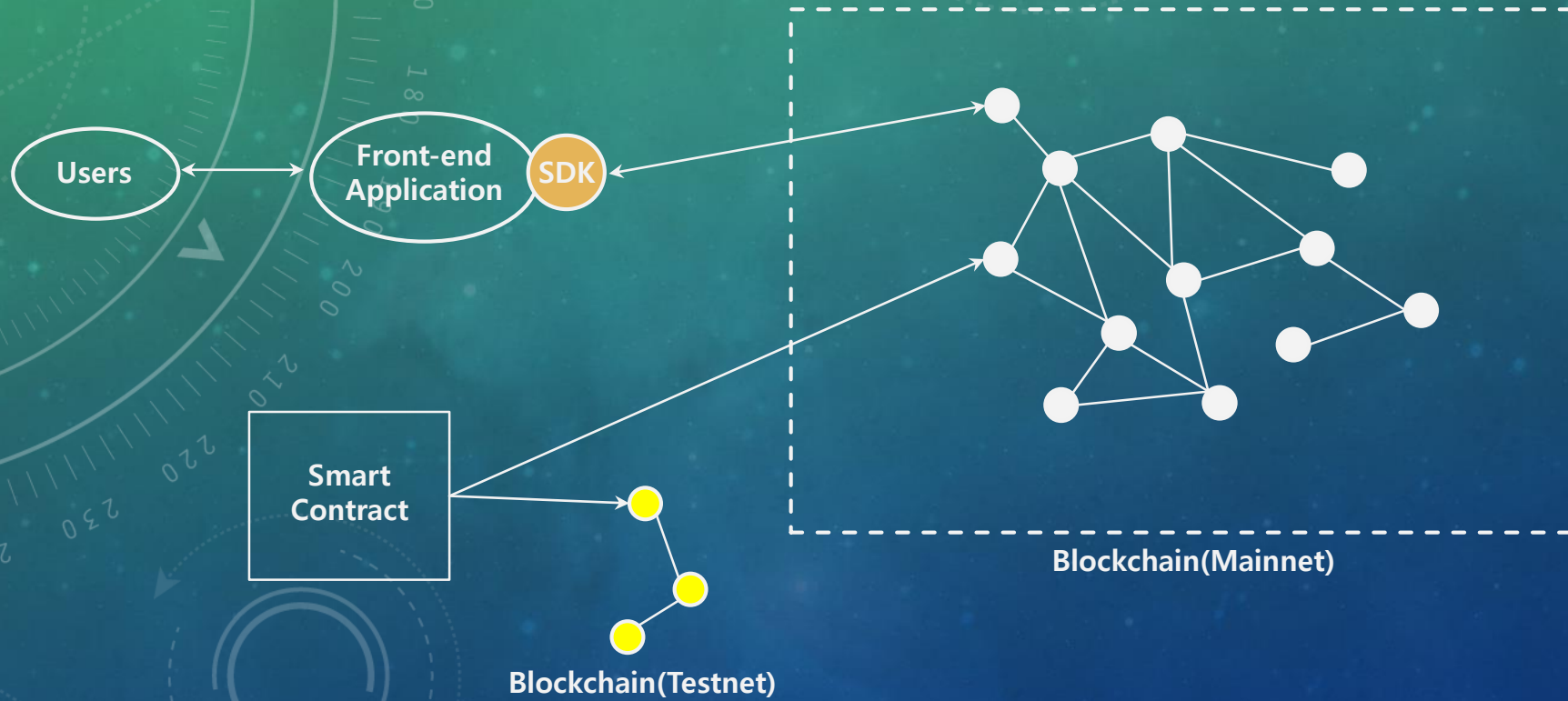
스마트 컨트랙트

Smart contract를 작성할 수 있는 블록체인 플랫폼과 개발 언어

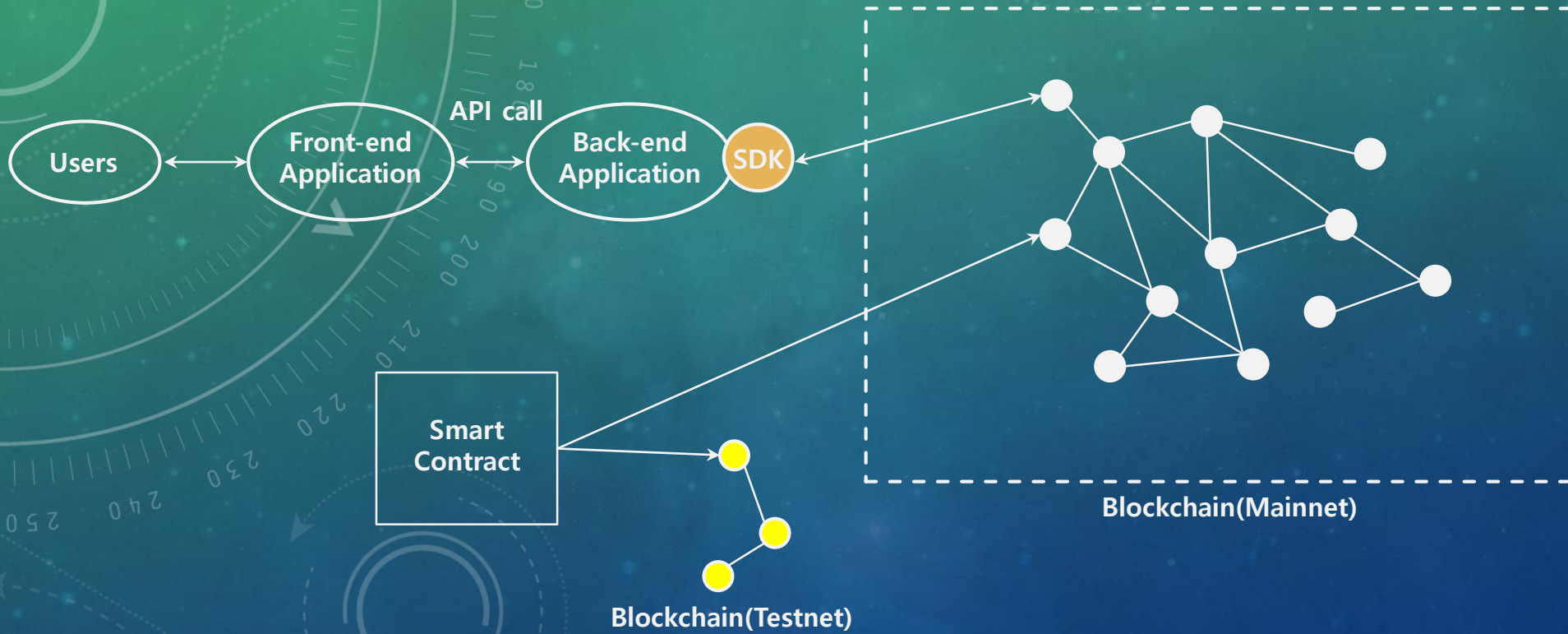
플랫폼	언어
Ethereum	Solidity, LLL, Serpent, Vyper
Bitcoin	Ivy-lang, Balzac
Cardano	Plutus (Haskell inspired)
Qtum	Solidity
EOS	C/C++ (compiles to WASM)
Hyperledger Fabric	Go, Javascript
Neo	C#, VB.Net, F#, Java, Kotlin, Python

Solidity is not a general-purpose language!

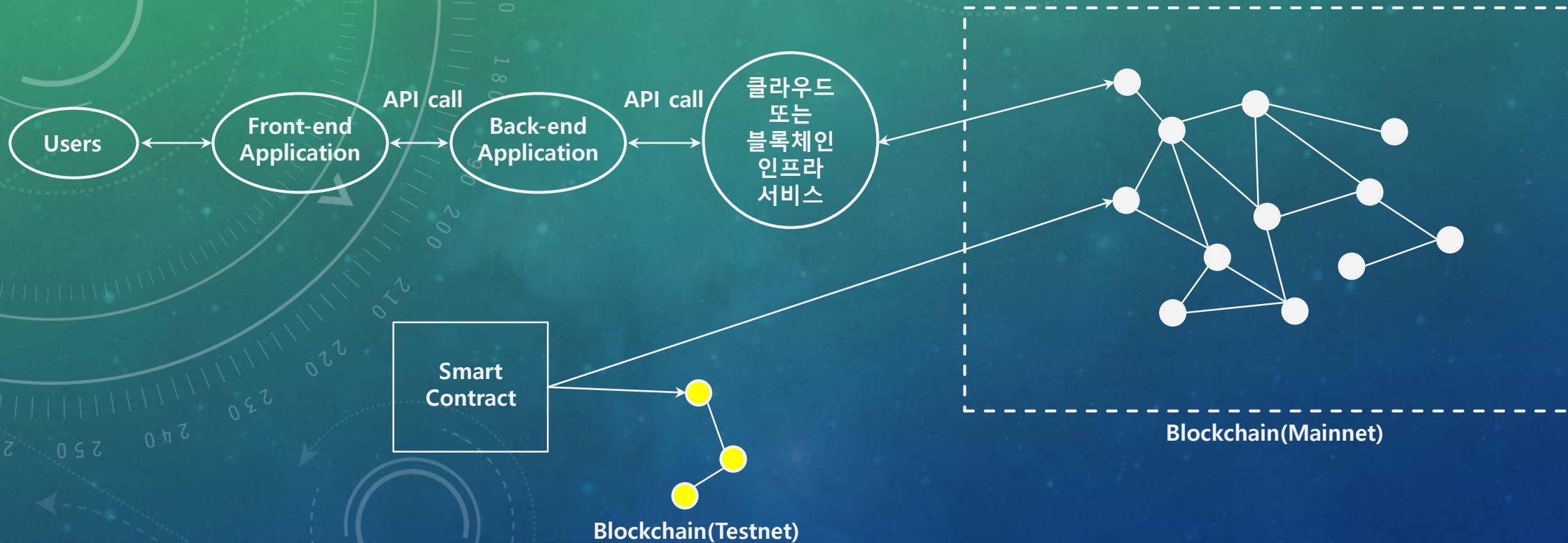
블록체인 애플리케이션



블록체인 애플리케이션



블록체인 애플리케이션

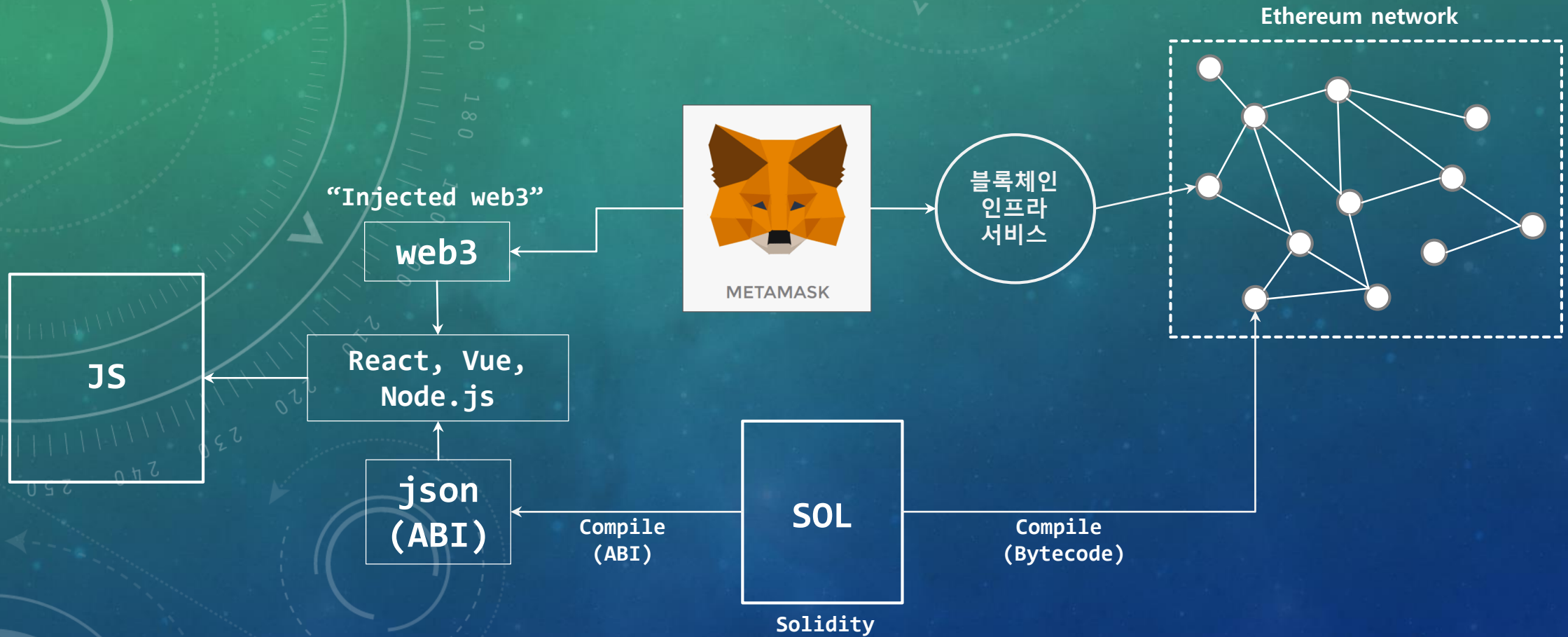


이더리움 Dapp

	자바 웹 개발	이더리움 컨트랙트 개발
개발환경	IDE(Eclipse) + plug-in... Java(JDK) Framework(Spring MVC) Tomcat	IDE + plug-in... Solidity, ... Truffle(solc-js) 컴파일, 테스트, 배포 자동화 도구 Ganache
운영환경	WAS : Weblogic, Jboss, Jeus, Websphere ... Database : RDBMS(Oracle, MS-SQL, MySQL) ... "Centralized" 기업이 관리하는 전산(데이터)센터	이더리움 : 분산공유 네트워크 "Decentralized" 노드들(누구나 가능)

이더리움 Dapp

(예) Javascript Frontend application



하이퍼레저 – Chain code using dev mode

(1) 도커 이미지 설치

```
docker pull hyperledger/fabric-orderer@1.4.1
docker pull hyperledger/fabric-peer@1.4.1
docker pull hyperledger/fabric-ccenv@1.4.1
docker pull hyperledger/fabric-tools@1.4.1
```

(2) fabric-samples 설치

```
git clone -b release-1.4 https://gerrit.hyperledger.org/r/fabric-samples
```

(3) peer, orderer, channel 실행

```
\fabric-samples\chaincode-docker-devmode> docker-compose -f docker-compose-simple.yaml up
```

(4) chain code 컴파일, 실행

```
docker exec -it chaincode bash
CORE_PEER_ADDRESS=peer:7052 CORE_CHAINCODE_ID_NAME=mycc:0 ./chaincode_example02
```

(5) chain code 설치, 인스턴스 생성, 호출(invoke)

```
peer chaincode install -p chaincodedev/chaincode/chaincode_example02/go -n mycc -v 0
peer chaincode instantiate -n mycc -v 0 -c '{"Args":["init","a","100","b","200"]}' -C myc
peer chaincode query -n mycc -c '{"Args":["query","a"]}' -C myc
```

<https://github.com/hyperledger/fabric-samples/tree/release-1.4/chaincode-docker-devmode>

Q&A

[참고]

<https://github.com/ConsenSys/ethereum-developer-tools-list>

The background features a gradient from light green at the top to dark blue at the bottom. On the left side, there are several overlapping circular elements, including a large semi-circular scale with numerical markings from 140 to 260 in increments of 10, and various concentric circles and arcs. The text 'Thank You' is centered in a large, white, sans-serif font.

Thank You