비트코인 이해

P2P (Peer-to-Peer) 버전의 전자 현금은 금융 기관을 거치지 않고 한 쪽에서 다른쪽으로 직접 온라인 지불을 보낼 수있게합니다.

- Satoshi paper

P2P 기반 가상화폐 시스템의 문제점

1. 악의적인 참가자에 의한 변조.부정 (the Byzantine Generals' Problem)
2. 정보 전달 지연으로 인한 불일치 - 이중 지불
3. P2P 네트웍의 유지 및 운영 시스템 부재

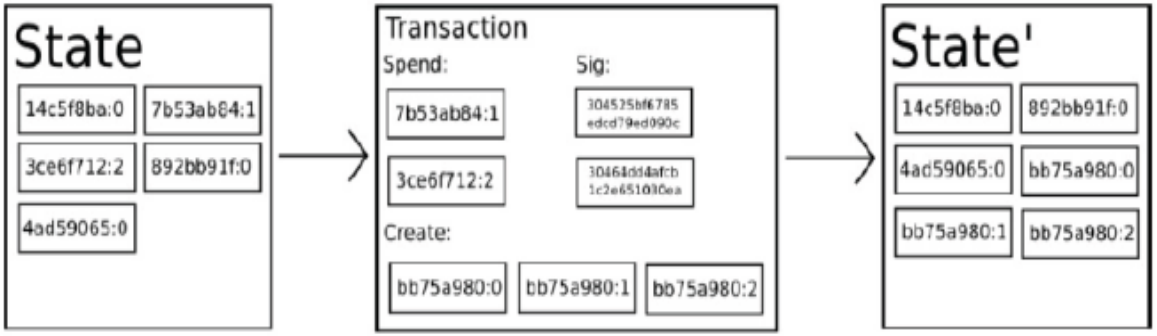
1. 블록체인 - "financial institution 을 BlockChain이라는 Shared ledger 로 대체!!
2. 전자서명 – 이전 소유자와 본인 보증과 부인 방지
3. PoW(Proof of Work) – 블록을 임의대로 만들어 조작하는 것을 방지
4. 인센티브에 의한 네트웍 운영(마이닝)

Bitcoin = State Transition System

상태(State)와 상태 변환 함수(State transition function)로 구성된 시스템

- 상태: UTXO (Unspent Transaction Outputs)의 집합
  UTXO: 코인 금액과 소유자의 비트코인 주소(Encrypted Public Key)

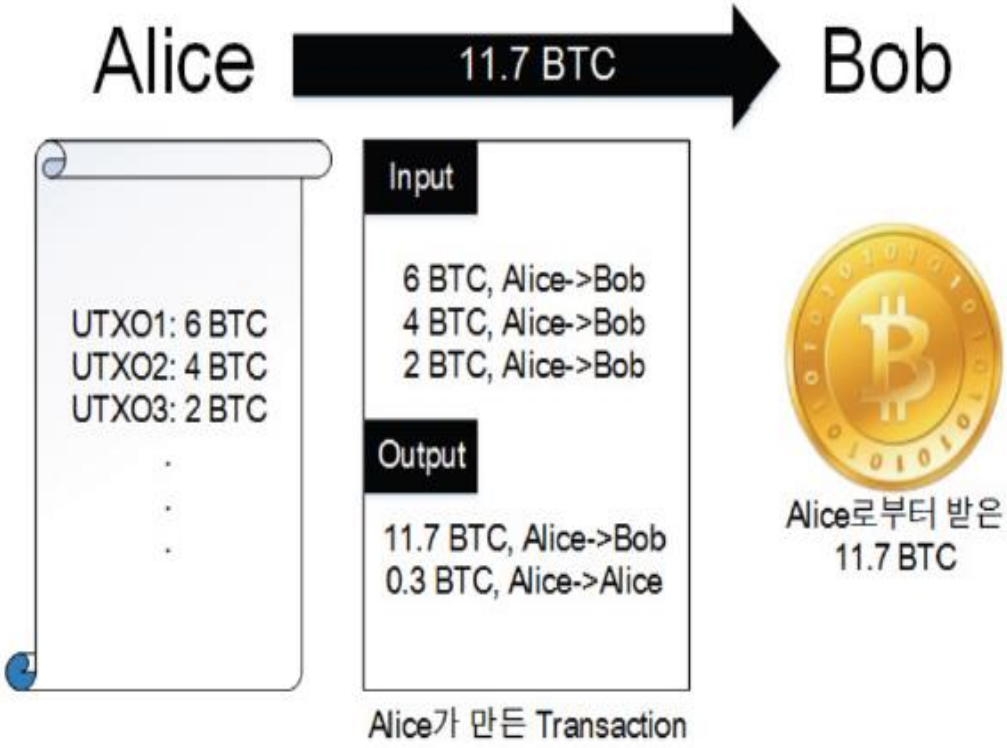
- 상태 변환 함수: APPLY(S, TX) -> S' or ERROR
  TX (트랜잭션): 입력 + 출력
  입력 : 발신자의 UTXO 참조 정보, 발신자의 서명
  출력 : 새로운 상태에 추가될 UTXO 정보



Bitcoin = State Transition System

"Alice [11.7BTC] -> Bob"

"Alice [0.3 BTC] -> Alice"는 거스름돈을 표시하는 Output



BlockChain

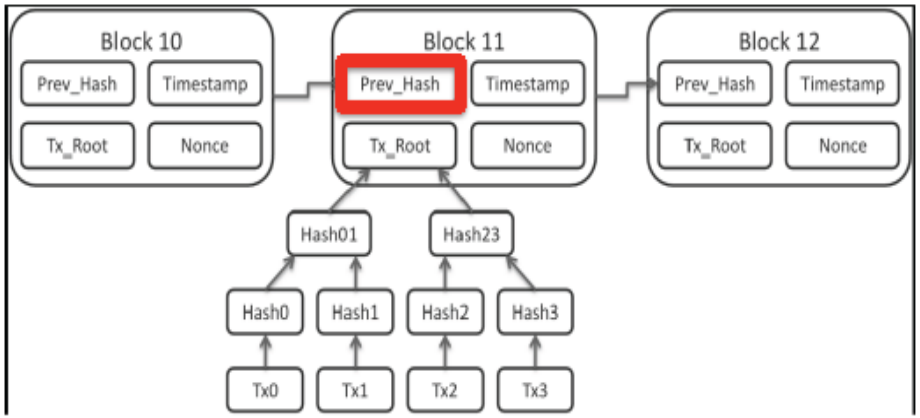
- 블록체인은 시간순으로 정열됨

  (앞에 블록의 내용이 변경되면 뒤에 것도 모두 변경해야 함, 조작이 어려운 불가역 데이터)

- 블록체인은 분산된 공유 원장.

  모든 사용자가 모든 거래를 기록한 원장을 공유하기 때문에 새로운 블록이 체인에 추가되려면

  거래의 타당성을 검증받아야 한다.(부당 거래 방지)



- ❖ Version: The block version number.
  17년 기준 현재는 2, 4Bytes
- ❖ Timestamp : the current timestamp. 4Bytes
- ❖ Hash of the previous block,32Bytes(256Bits)
- ❖ Nonce, 4Bytes
  (보통 0x00000000 부터 0xFFFFFFFF)
- ❖ The current **difficult target**. 4bytes
- ❖ Hash of the Merkle Root,32Bytes(256Bits)

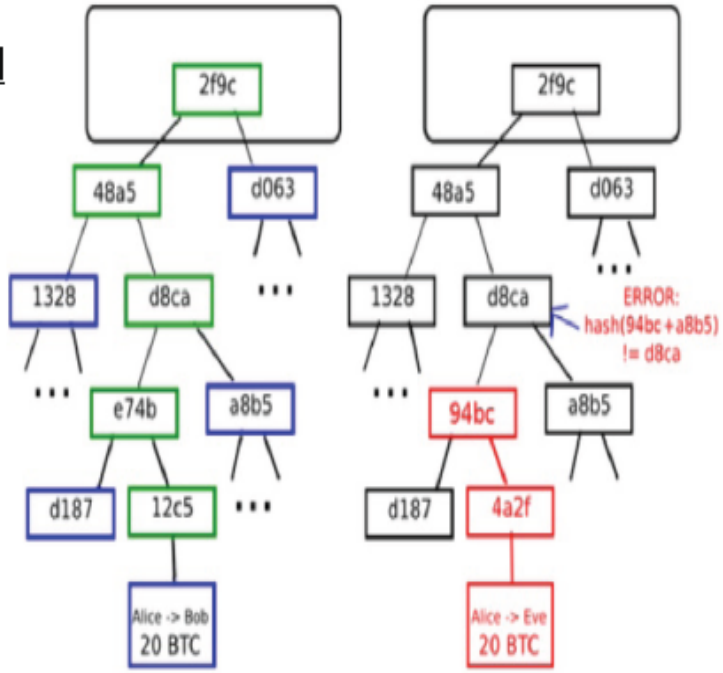
Merkle Tree(Hash Tree)

블록헤더의 Merkle root는 트랜젝션들의 거래 내용이 변조되었는지를 증명해준다.

이더리움 SPV (Simplified Payment Verification) Protocol

- **Full node**: 비트코인 네트워크에서 각 블록의 모든 정보를 저장하고 처리하는 노드
  - : 전체 블록체인을 저장 , 많은 디스크 공간을 필요로 함
  - : 트랜잭션의 유효성을 검증 (Confirmation)
- **Light node** (Thin client): 원하는 블록의 블록헤더를 다운로드하여 작업 증명 여부를 검증하는 노드
  - : 전체 블록체인을 저장하지 않고, 블록의 헤더만을 저장하여 Merkle root 를 통한 특정 트랜잭션의 유효성 확인

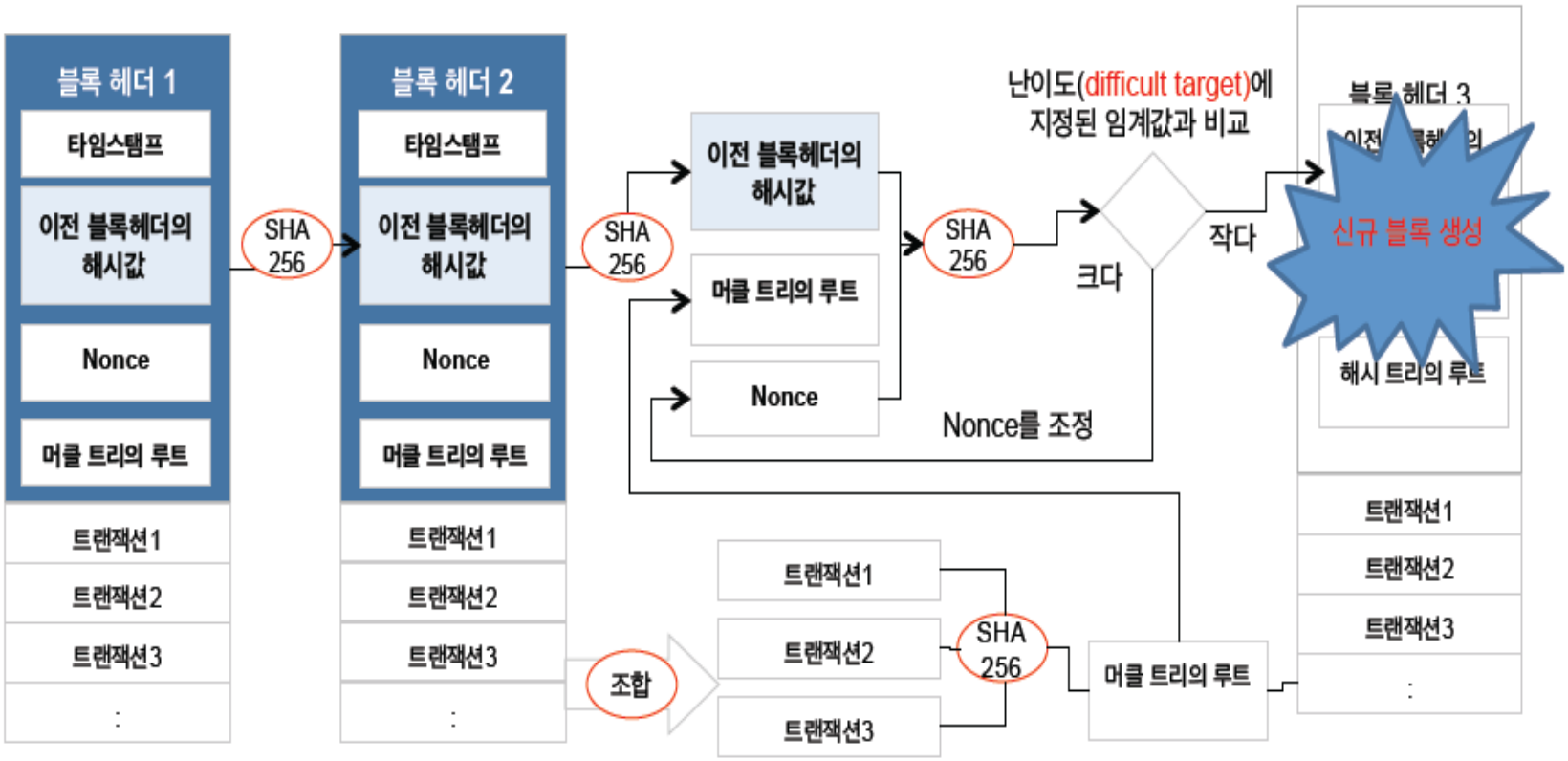
→ 강한 안전성을 보장하면서도, 임의의 트랜잭션에 대한 상태와 잔고를 알 수 있음



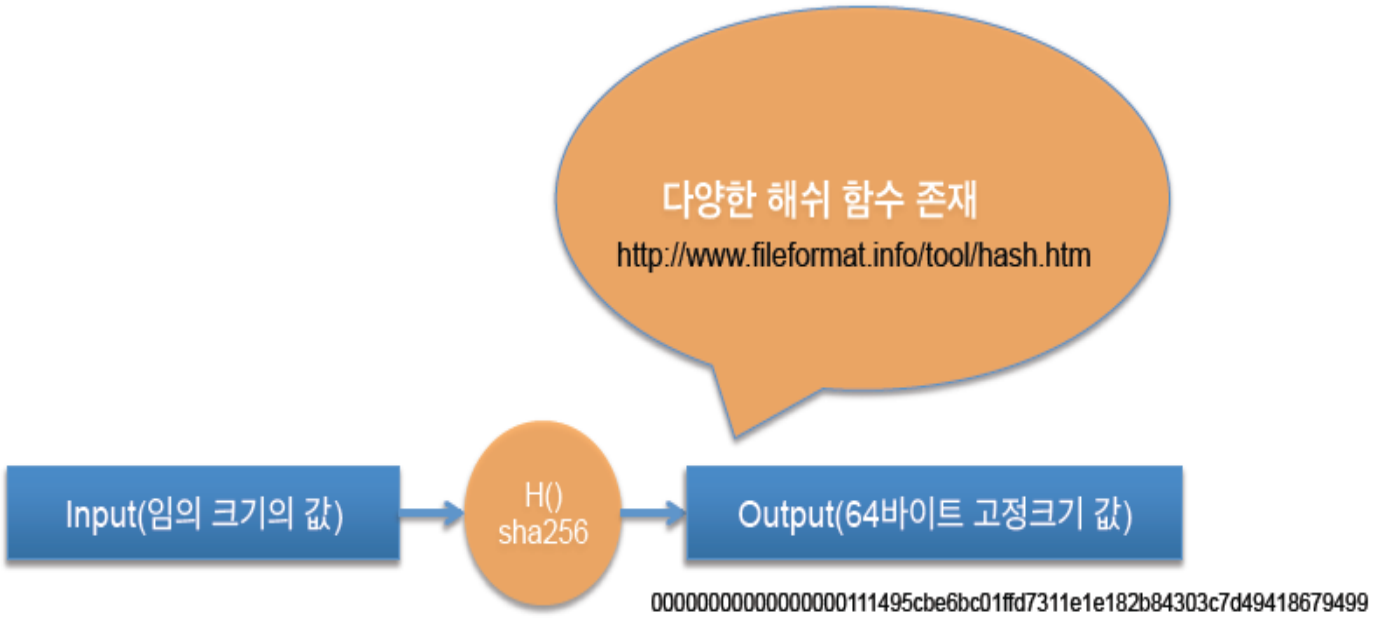
비트코인의 블록 처리 프로세스

1. 사용자 A가 송금을 의뢰 트랜젝션을 발생시킨다.
   트랜잭션 발행시 본인 확인을 위해 **전자서명**을 첨부한다.
2. 해당 트렌잭션은 P2P 네트웍을 통해 모든 노드에게 **브로드캐스팅**된다.
3. 트랜잭션 수신자중 마이너들은 **전자서명을 확인**하고 블록을 생성하기 위해
   SHA256으로 해쉬값을 계산한다. 현재 비트코인의 블록생성 시간은 10분이다.
4. 처음 조건을 만족하는 해답을 발견한 마이너는 새로 생성한 해당 블록을 모든 노드에
   브로드캐스트한다.
5. 해당 블록을 받은 다른 노드들은 이전 블록 정보와 찾은 해답을 매개변수로 해쉬값을 구해
   조건을 만족하는 지 검증한 후, 정당한 블록이면 로컬 디스크에 있는 블록체인에 추가한다.
6. 최초 해답을 발견한 **마이너**는 블록 생성에 대한 보상을 받는다.

비트코인의 블록 처리 프로세스



해쉬함수


다양한 해쉬 함수 존재
http://www.fileformat.info/tool/hash.htm
Input(임의 크기의 값)
H()
sha256
Output(64바이트 고정크기 값)
000000000000000000111495cbe6bc01ffd7311e1e182b84303c7d49418679499

합의 알고리즘, Proof of Work

P2P 네트웍은 정보 도달에 시간차가 있어 지연과 미도달 사태가 발생할 수 있다. 이로 인해 이중 송신에 따른 중복 처리나 오작동이 가능한다 . 이를 해결하기 위해서는 해당 정보가 정확한 지 확인하기위한 방법이 필요하다. 이 방법이 바로 합의 알고리즘이다.

계산량에 따른 증명, Proof of Work을 합의 (비트코인, 이더리움)

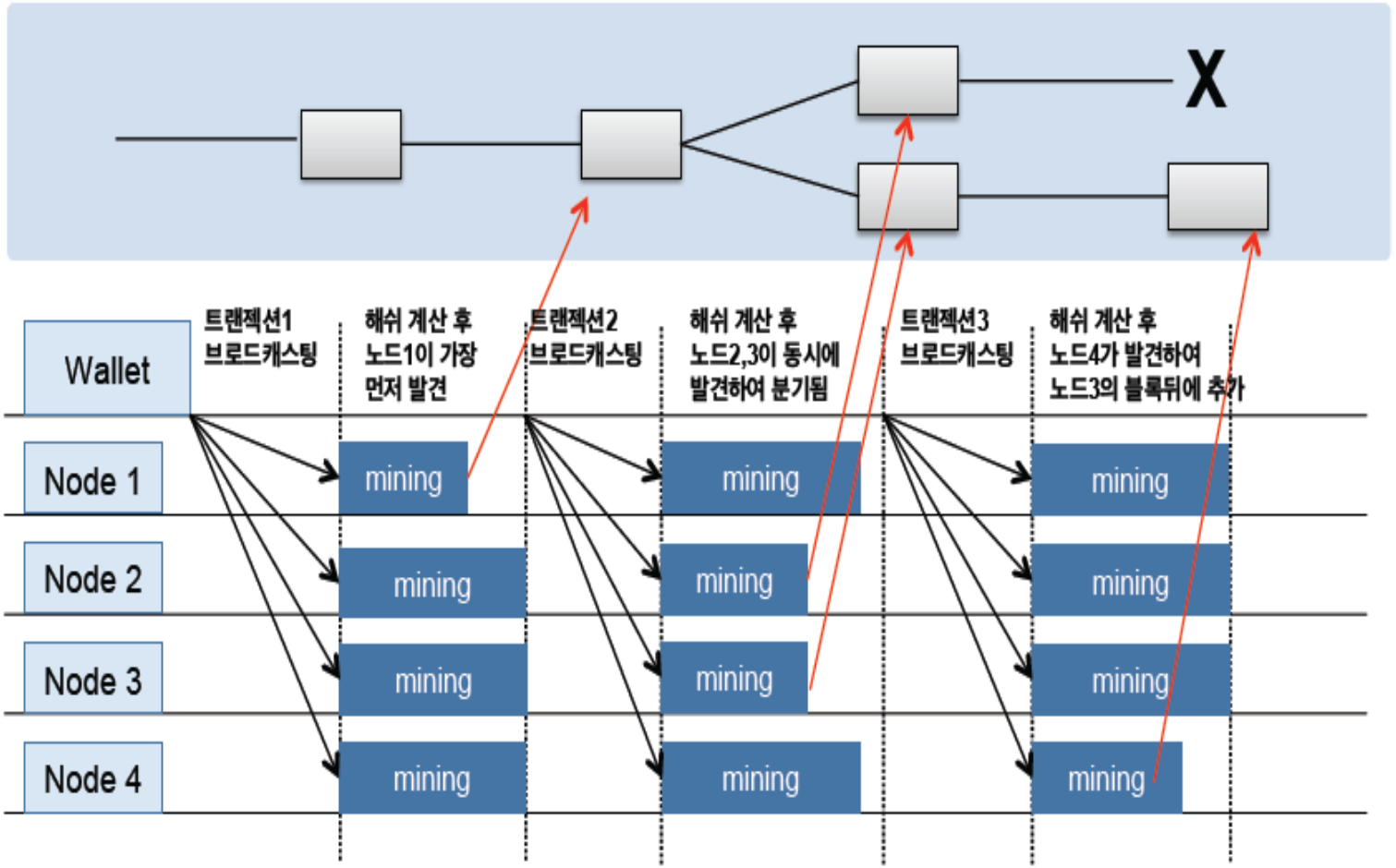
해당이 어려운 문제를 가장 빨리 해결한 사람에게 블록을 만들 수 있도록 하고 가상 화폐로 보상.
보상으로 받기 위해 자발적으로 참여하기 때문에 시스템이 자율적으로 운영됨.
거래를 승인하고 동시에 블록 생성을 계산적으로 어렵게 만들어서 공격자들이 마음대로 블록체인을 조작하는 것을 방지

- 생성한 블록에 포함된 트랜잭션 중 출력 금액보다 입력 금액이 큰 트랜잭션이 있다면 그 차액을 '거래 수수료'로 획득
- 수수료는 사용자가 자유롭게 정할 수 있으나, 수수료를 0으로 설정 하면 마이너가 블록에 포함시키기를 선호하지 않아 확정되는데 보 통 1-7일 이 걸림 (권장 수수료: 0.0005 BTC, 약 500원)
- "파이얼리티 불확실성 문제" 해결을 위해 6-confirmation(거래가 확정되더라도 6째 블록 생성시까지 대기 후 처리)
  예. 20000번째 블록에서 fork를 일으켜도 20005번째 블록이 생성되었을 때 거래를 확정하면 이중 지불 공격 방어가능

문제점

- 공격자 노드가 51% Attack이 가능할 정도의 hash rate를 가졌다면 6-confirmation도 보안책이 될 수 없음
- 파이널리티 문제 , PoW는 블록체인 분기시 긴 체인을 채택함. 짧은 체인을 사용하던 노드에 계좌 잔액이 변경되거나 이중 지불 또는 거래 자 체가 없었던 일이 될 수 있음

합의 알고리즘, PoW 의 구조

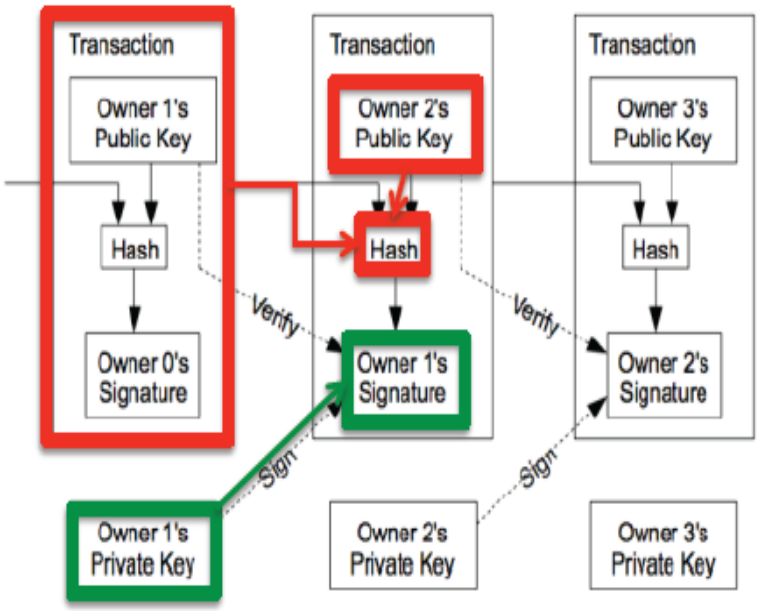


전자서명

비대칭 알고리즘 – 공개 키와 개인 키 사용 (VS 대칭키 알고리즘은 비밀키만 사용)

전자서명 – 블록체인 트랜젝션

Owner 1이 Owner 2에게 비트코인을 전달하는 과정



Owner 1은, 비트코인 거래 내역과 Owner 2의 공개 키를 가져와서 해싱(hashing)을 한다. 그리고 자신(Owner1)의 개인키를 사용해서 서명(Sign, ECDSA)을 한다.

서명된 결과를 Owner 2에게 보낸다. Owner 2는, 자신의 공개키를 활용해 암호화된 문서가 도착했으므로, 자신의 개인키를 사용해서 암호를 풀 수 있다. 암호가 제대로 풀리면(Verify) 비트코인의 거래가 완료된다.

이더리움 이해

"이더리움은 정확히 프로그래밍 한대로 동작하는

스마트 컨트랙트를 동작시키는 분산된 플랫폼이다."

https://ethereum.org/

비트코인 대비 차이점

1♪ 튜링완전성 (Turing-Completeness)

- 비트코인: Script를 사용하고 거래 증며시 무한 루프에 빠지는 것을 방지하기 위해 의도적으로 반복문을 제거하고 if 문만 지원.
- 이더리움: 튜링완전언어를 지원하여 if 문 뿐만 아니라 반복/ 조건 명령문도 지원

  (이더리움이 지원 언어 - Serpent, Solidity, LLL, Mutan)

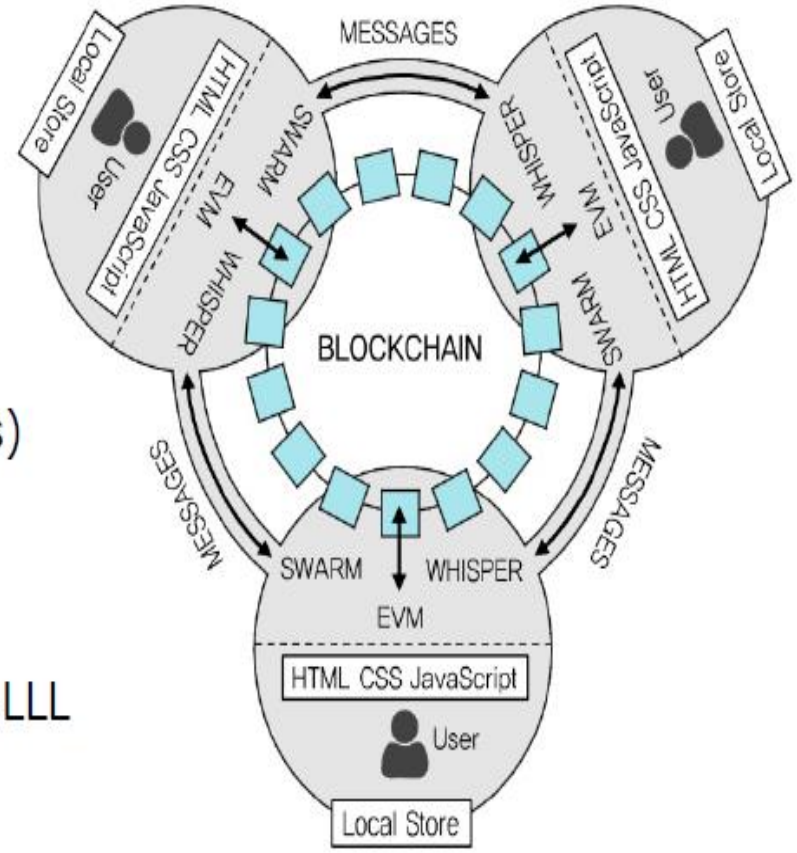
2♪ 개발 플랫폼을 이용한 응용성

- 비트코인: 전자 화폐 서비스
- 이더리움: 다양한 서비스를 개발할 수 있는 플랫폼
- 스마트 컨트랙트 : 블록체인에 실행가능한 코드 형태 업로드

  특정 조건이 달성 되면 자동적으로 코드가 실행되어 계약이 이행됨

이더리움 환경

- Mist Browser/Wallet
- Decentralized Applications (DApps)
- Swarm,Whisper
- EVM (Ethereum Virtual Machine)
- Smart Contract : Solidity,Serpent,LLL



Account

- 이더리움의 상태(State)는 어카운트(Account)라고 하는 객체(Object)들로 구성됨
- 각 계정은 20바이트의 주소와 상태 변환(state transition)을 가지고 있다.
- 어카운트의 4개 필드 구성

필드	설명
NONCE	각 거래가 오직 한번만 처리되게 하는 일종의 카운터
ETHER BALANCE	어카운트의 이더 잔고
CONTRACT CODE	어카운트의 계약 코드 (존재할 경우)
STORAGE	어카운트의 저장 공간 (초기설정 상에는 비어있음)

Account

• 어카운트의 종류 : EOA & CA

외부 소유 어카운트 (EOA, Externally Owned Account)

일반적인 사용자 계정을 말하며 , 아무런 실행 코드를 가지고 있지 않다.

EOA에 메세지를 보내기 위해서는 새로운 트랜 잭션(거래)을 만들고 서명해야 한다.

비트코인의 지갑과 같은 개념이다.

컨트랙트 어카운트(CA , Contract Accounts)

메시지를 받을 때마다 내부의 코드를 활성화시키고 , 이에 따라 컨트랙을 생성하거나 다른메세지를 읽거나 보낸다. 또한 내부 저장 공간에 데이타를 저장한다.

cf) contract account를 보통 contract라고 하며 EOA에 의해 작동된다.

Account

외부 소유 어카운트 (EOA, Externally Owned Account)

```
// Jay Account 생성
> personal.newAccount("Jay")   // 또는 $ geth account new
INFO [08-01|08:46:29] New wallet appeared  url=keystore:///Users/jaehyunpark-a… status=Locked
0xb2cf02bea7e2538a90b634fcbe2cbf1dd9ee6662
```

컨트랙트 어카운트(CA , Contract Accounts)

0xec5bee2dbb67da8757091ad3d9526ba3ed2e2137

Transaction(거래)

상대방에게 보낼 메시지를 가지고 있는 서명된 데이터

- 컨트랙트는 트랜잭션을 구성하는 필드의 데이터를 읽어서 어플리케이션에 맞는 코드를 실행
- 트랜잭션의 필드 구성

필드	설명
수신처 주소	거래를 받을 수신처
발신자 서명	발신자를 확인할 수 있는 서명
이더 (value)	발신처가 수신처로 보내는 이더의 양
데이터(data)	옵션으로 임의의 메세지, 함수호출, 계약 생성 코드
STARTGAS	거래가 실행시 수행되도록 허용된 최대 트랜젝션 수행 횟수
GASPRICE	계산 단계마다 발신처가 지불하는 수수료

Message

- (외부 액터가 아니라)컨트랙에 의해서만 생성되며 다른 컨트랙에게 전달된다(funciton call).
  : 메세지 생성을 지시하는 CALL/DELEGATECALL opcode 을 호출하여 생성됨.
- 메세지는 트랜잭션과 다르게 가상 객체로 별도 저장할 필요가 없으며 EVM실행환경내에서만 존재
- 메시지의 필드 구성

필드	설명
발신자 주소(implicit)	메시지를 전송한 발신처
수신처 주소	메시지를 수신할 수신처
이더(Value)	메시지와 함께 전달되는 이더 (wei)
데이터(optional)	데이터 필드
STARTGAS	거래가 실행시 수행되도록 허용된 최대 트랜젝션 수행 횟수

Ether

이더리움에서 사용하는 화폐 단위.
EVM 연산 대가를 지불하는 데 사용.

변동성이 있기 때문에
트랜젝션 비용으로
직접 사용하지 않고 GAS를 통해 사용

wei:	1
kwei:	1.000
ada:	1.000
femtoether:	1.000
mwei:	1.000.000
babbage:	1.000.000
picoether:	1.000.000
gwei:	1.000.000.000 (1e-9)
shannon:	1.000.000.000
nanoether:	1.000.000.000
nano:	1.000.000.000
szabo:	1.000.000.000.000
microether:	1.000.000.000.000
micro:	1.000.000.000.000
finney:	1.000.000.000.000.000
milliether:	1.000.000.000.000.000
milli:	1.000.000.000.000.000
ether:	1.000.000.000.000.000.000(1e-18)
kether:	1.000.000.000.000.000.000.000
grand:	1.000.000.000.000.000.000.000
einstein:	1.000.000.000.000.000.000.000
mether:	1.000.000.000.000.000.000.000.000
gether:	1.000.000.000.000.000.000.000.000.000
tether:	1.000.000.000.000.000.000.000.000.000.000

https://converter.murkin.me/

Ether

```
➢ eth.sendTransaction(
{from : '0xb2cf02bea7e2538a90b634fcbe2cbf1dd9ee6662', to : '0x87c4ef09c4e94
249ed94b74d6d573c3dc902f15d' , value:web3.toWei(1,"ether")})

INFO [08-01|17:03:49] Submitted transaction                    fullhash=0x265514bd91
1fbb8dab9cb54a7e5be57c35ce445cb07d1762cd0f0f61b78a5843 recipient=0x87c
4ef09c4e94249ed94b74d6d573c3dc902f15d
"0x265514bd911fbb8dab9cb54a7e5be57c35ce445cb07d1762cd0f0f61b78a5843"

➢ eth.pendingTransactions
➢ miner.start()
```

GAS

- 가스는 거래를 보낸 사람이 이더 리움 블록 체인에서 이루어지는 모든 작업에 대해 지불해야하는 실행 수수료의 이름

- Total cost = gasUsed * gasPrice
  gasUsed : 해당 트렌젝션 수행에 소비되는 총 가스양
  gasPrice : 1가스 단위 가격

Gas는 계산 비용과 연결되어 있는 반면,
Ether는 시장 변동성과 연결되어 있다.

EVM이 GAS를 다 쓰면 실행중인
트랜젝션이 멈추고 이전 상태로 rollback된다.
가스는 그냥 낭비됨.

Operation Name	Gas Cost	Remark
step	1	default amount per execution cycle
stop	0	free
suicide	0	free
sha3	20	
sload	20	get from permanent storage
sstore	100	put into permanent storage
balance	20	
create	100	contract creation
call	20	initiating a read-only call
memory	1	every additional word when expanding memory
txdata	5	every byte of data or code for a transaction
transaction	500	base fee transaction
contract creation	53000	changed in homestead from 21000

GAS

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead transition*.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.

https://ethereum.github.io/yellowpaper/paper.pdf

default gasPrice of 0.05e12 wei.

(예제)

2개의 숫자만 추가하는 계약을 봅시다.
위의 스프레드 시트에서 ADD는 3 개의 가스를 소비합니다.

기본 가스 가격을 사용한 대략적인 비용은 다음과 같습니다.

3(gasUsed)*0.05e12(gasPrice)=1.5e11 Wei

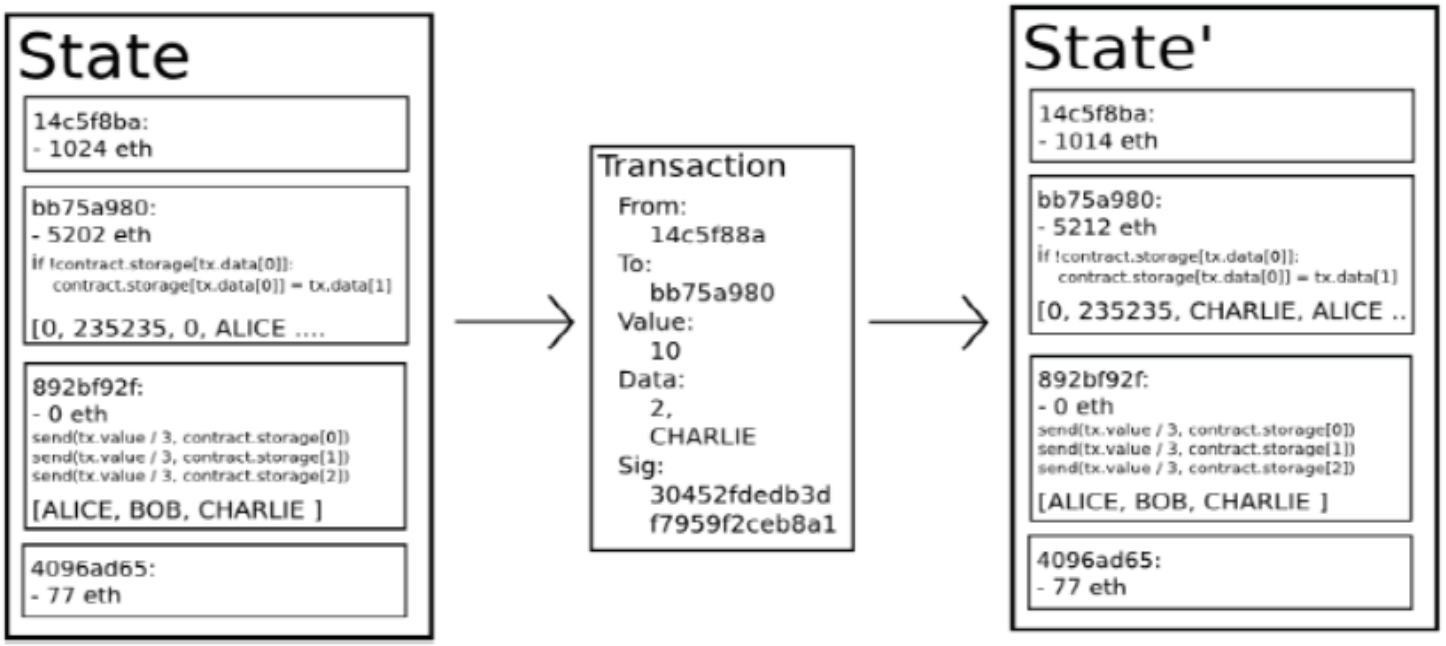
1 Ether은 1e18 wei이므로 총 비용은 0.00000015 Ether입니다.

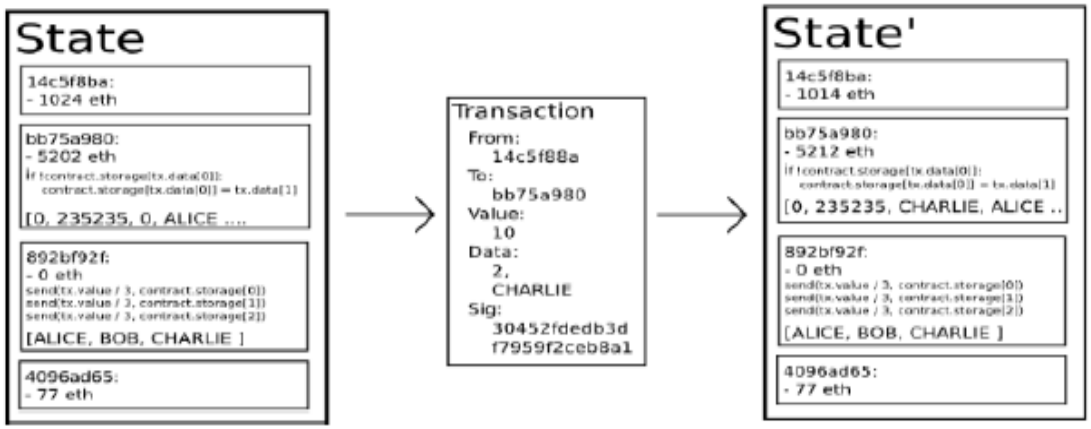
이는 두 숫자를 계약에 전달하는 비용과 같은 일부 비용을 추가하기 전에 무시하기 때문에 단순화 된 것입니다.

State Transition System

APPLY(S)=S'

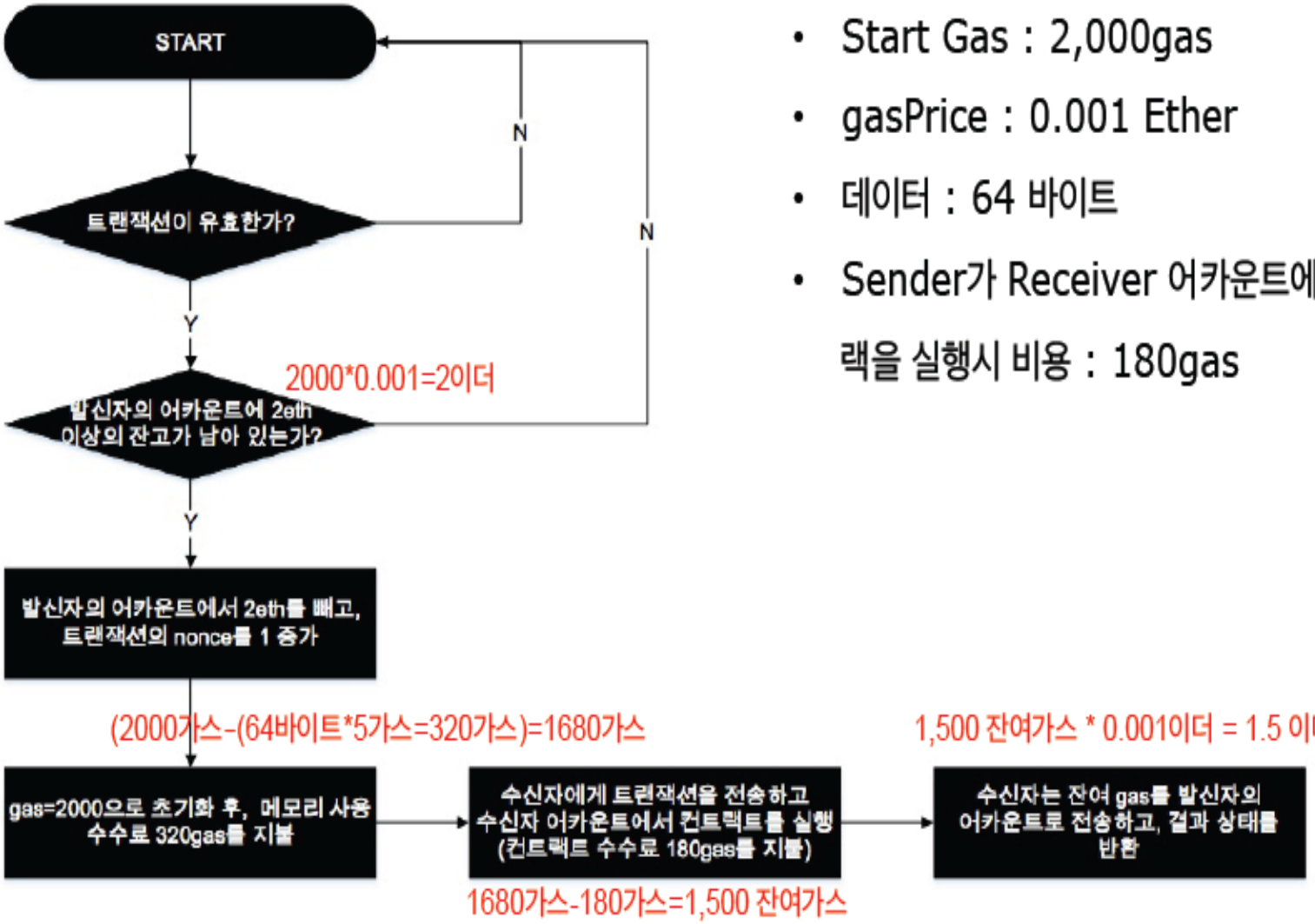
(S:현재 상태 → APPLY: 상태변이 함수 → S': 변이된 상태 또는 에러)





1. 해당 트랜젝션이 문법에 맞게 구성되어 있는지, 서명은 유효한지, Sender의 계정에 있는 NONCE와 맞는지 점검.
2. 거래 비용을 STARTGAS * GASPRICE로 계산하고 서명으로 부터 전송 주소를 확인한다. Sender의 계정 잔액에서 요금을 빼고 NONCE를 1 올린다. 잔액이 충분하지 않을 경우 에러를 반환한다.
3. GAS=STARTGAS 로 초기화 한다. 트랜젝션에서 사용한 메모리 당 수수료를 1바이트당 5Gas 만큼 가져간다.
4. 거래 값을 Sender 계정에서 Receiver 계정으로 전송한다. 만약 Receiver 계정이 존재하지 않는다면 계정을 생성한다. Receiver 계정이 Contract account일 경우, Contract 코드를 동작시킨다. 이 코드 동작은 계약이 완료되거나 가스가 다 소비될 때까지 계속된다.
5. 만약 Sender가 충분한 돈이 없거나, 가스가 소비되어 코드 실행을 할 수 없을 경우 등의 이유로 전송하는 것이 실패한다면, 거래 요금에 대한 지불을 제외한 모든 상태를 원복하고 거래 요금을 채굴자의 계정으로 추가한다.
6. 거래가 성공할 경우 남아 있는 모든 가스를 이더로 환산한 후 Sender에게 반환하고, 거래 요금을 Miner에게 전송한다.

State Transition System



- Start Gas : 2,000gas
- gasPrice : 0.001 Ether
- 데이터 : 64 바이트
- Sender가 Receiver 어카운트에서 컨트랙을 실행시 비용 : 180gas

Smart Contract & EVM

Nick Szabo

스마트계약 개념 창시자

신뢰할 수 없는 컴퓨터 인터넷 환경에서 고도로 발달된 계약을 준수하도록 하는 프로토콜 창시(1994년)

기존 비트코인에는 Script 라고 불리는 단순한 실행 프로그램만 지원.

채굴자(Miner)들의 거래 검증 알고리즘으로 사용

Vitalik Butterin

이더리움의 창시자

스마트 계약을 단순한 검증과 연산 수준이 아닌 '상태'와 '함수'로 정의하고, 상태에 따른 '분기'와

데이터 '저장'이 가능한 '튜링완전언어(Turing Complete Code)'로 발전시킴.

스마트 계약은 이더리움의 실행환경 안에 살아있는 일종의 자율 에이전트(autonomous agents)로서,

분산N/W 환경하에서도 중앙화된 컴퓨팅 환경과 동일한 통제와 서비스가 가능함

Smart Contract & EVM

- 컨트랙은 실행가능한 스택 기반의 코드로 EVM에서 실행됨

  코드 실행은 0부터 시작하는 프로그램 카운터를 증가시키며 반복적으로 연산(loop)을 수행한다.

- 코드는 종료 조건이 만족했을 때 실행을 멈춘다 : 오류, STOP 명령어, RETURN 명령어, 실행 완료

- 연산 수행을 위해서는 세가지 타입의 공간에 접근이 가능해야 한다.

  · 스택 : LIFO(Last-In-First-Out) 컨테이너에 value를 push 하거나 pop함.

  · 메모리 : 무한대로 확장 가능한 바이트 배열

  · 저장소 : Key/value 저장소(simpleDB). 계산이 끝나면 리셋되는 스택이나 메모리와는 달리 저장소는 영속적으로 유지됨

- 코드는 블록 헤더 데이터 뿐만 아니라 특정 값이나, 발신자 및 수신되는 메시지의 데이터에 접근할 수 있고, 결과값으로 데이터의 바이트 배열을 반환할 수도 있다.

Smart Contract & EVM

- 스마트 컨트랙 작성

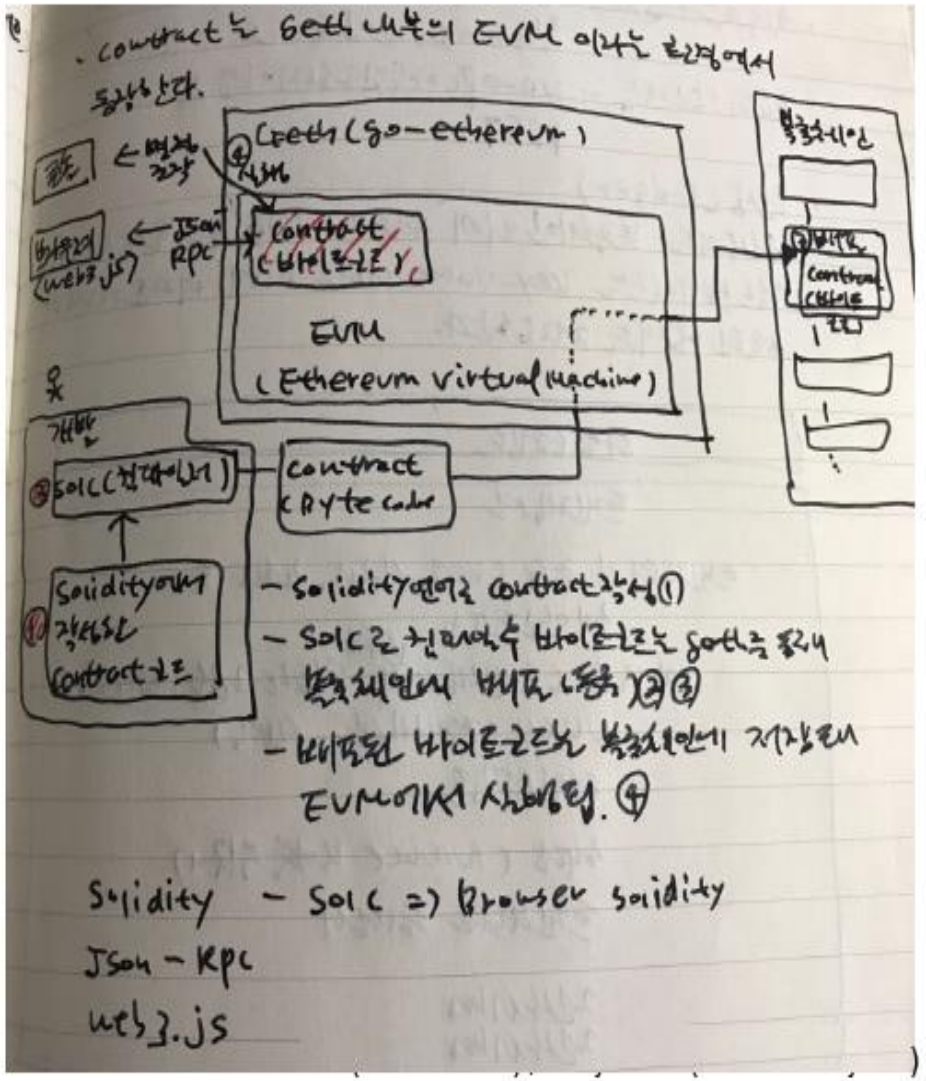
- **Solidity**, Serpent, LLL, Mutan 로 작성
- 컴파일 후 생성된 바이트 코드를 블록체인에 Deploy

- 스마트 컨트랙트가 실행되는 시점 ?

- EOA에 의해 발생한 블록의 트랜젝션에 의해 실행
- 다른 컨트랙트에 의해 실행

- 3가지 instruction 유형

- 새로운 스마트 컨트랙 생성
- 특정 스마트 컨트랙의 함수 실행
- 이더 전송



Smart Contract & EVM - OPCODES

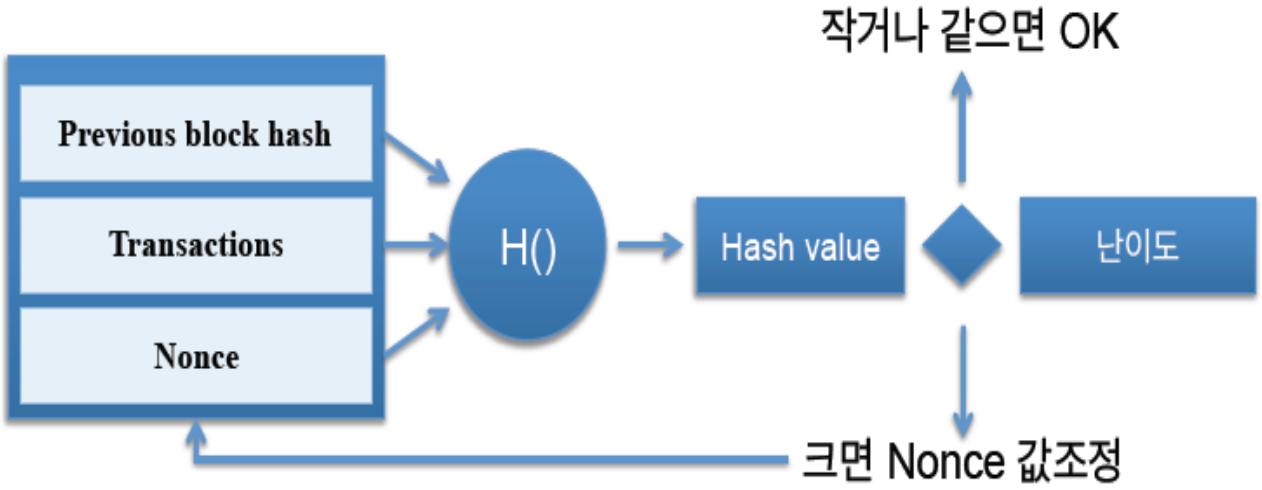
0s: Stop and Arithmetic Operations

0x00	STOP	Halts execution
0x01	ADD	Addition operation
0x02	MUL	Multiplication operation
0x03	SUB	Subtraction operation
0x04	DIV	Integer division operation
0x05	SDIV	Signed integer
0x06	MOD	Modulo
0x07	SMOD	Signed modulo
0x08	ADDMOD	Modulo
0x09	MULMOD	Modulo
0x0a	EXP	Exponential operation
0x0b	SIGNEXTEND	Extend length of two's complement signed integer

https://github.com/ethereum/yellowpaper
https://ethereum.stackexchange.com/questions/119/what-opcodes-are-available-for-the-ethereum-evm

합의 알고리즘, 마이닝(Ethash)

블록헤더의 Hash값이 난이도 목표(nBits)에 제시된 값보다 작은값이
나오게 하는 Nonce값을 찾아 나서는 과정



합의 알고리즘, 마이닝(Ethash)

- 해쉬 값의 앞에 0이 많은 이유는?

- Brute force 식으로 Nonce에 있는 값을 1비트씩 바꿔가면서 계산하기 때문.

- 앞자리 0의 갯수를 조정하여 난의도(nBits) 조정

The target is the threshold below which a block header hash must be in order for the block to valid, and nBits is the encoded form of the target threshold as it appears in the block header.

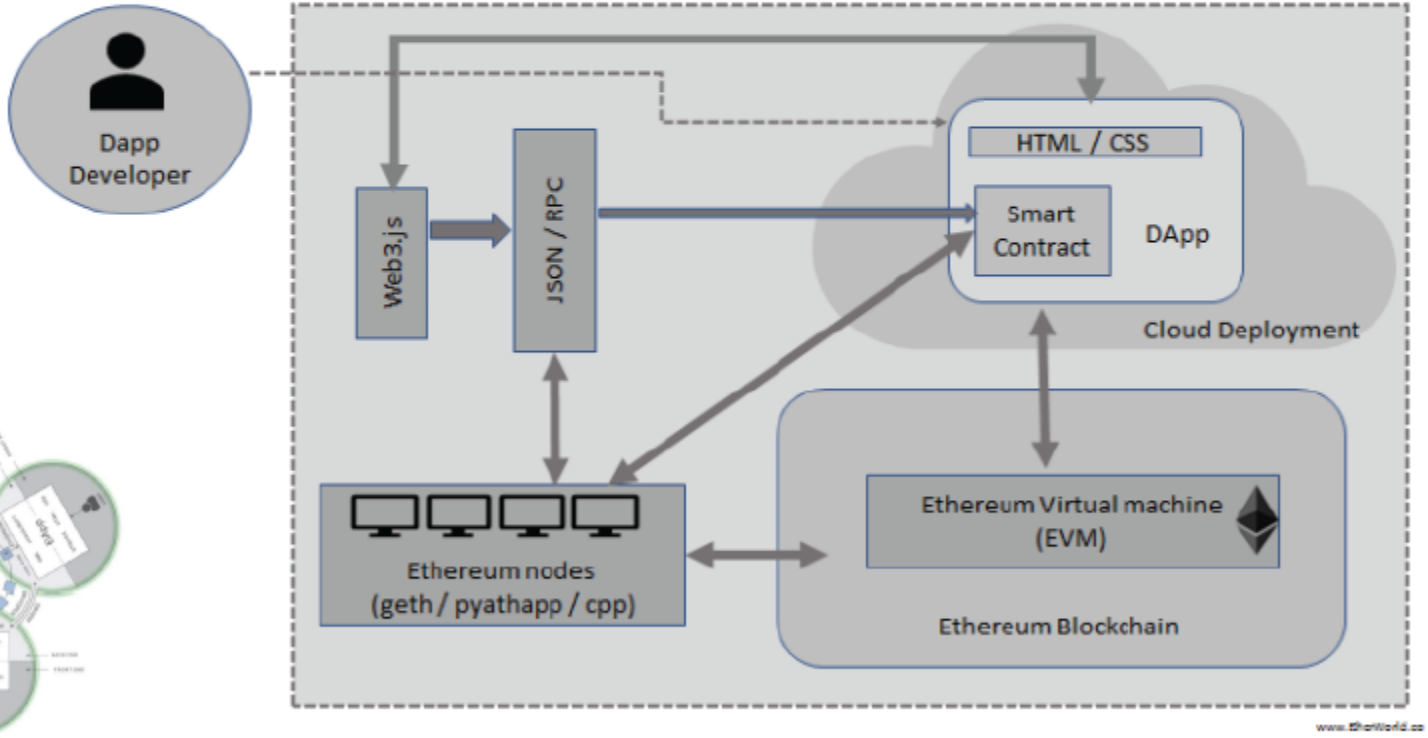
Mining

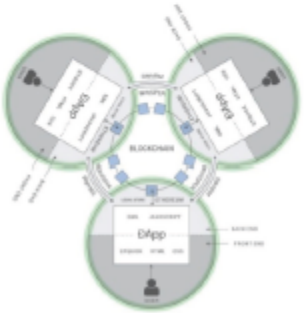
- Ethash: 수정된 작업증명 (PoW) 방법
  : 약12초에 하나의 블록이 생성될 수 있도록 알고리즘이 설계 되어 있음
  : 2차원 배열 데이터인 DAG(Directed acyclic graph) 파일이 사용됨

- 고속으로 생성되는 블록으로 인해 생기는 엉클 블록이 이더리움 보안성을 저하시키는 문제 발생
  : 엉클 블록: 블록생성에 성공하였고 검증에 오류가 없어서 네트워크를 통해 전파되었으나,
  더 빨리 전파된 다른 채굴자들 에 의해 순위가 밀려 주체인에 들어가지 못한 블록
  (비트코인에서는 Stale block으로 표현함)
  : 수정된 GHOST 프로토콜을 도입
  - 7세대까지의 엉클 블록을 주 체인에 포함
  - 엉클 블록 중 유효한 블록에 수수료를 지급 (기본 보상의 93.75%)

DApp

- Dapp은 이더리움 Contract 기반의 Frontend Client(App = Frontend Client + Server)





DApp

- 금융 DApps: 자산을 블록체인 위에 올리고 스마트 컨트랙트의 대상으로 사용
  : 채권, 주식, 파생상품, 보험, 복권, 도박 등
- 준/비금융 DApps: 토큰, 쿠폰, 네임코인, 투표 등
- 탈중앙화 조직/회사: 회사나 조직을 블록체인 상에 올려서 운영
  : 월급지급, 금전거래, 회계장부기록, 지분표시 등
- 탈중앙화 자율조직/회사 (DAO/DAC): 블록체인 상의 알고리즘으로 의사를 결정하여 영업, 회계, 구매, 판매 및 수익 분배 등을 실현

* Dapps 리스트 :https://docs.google.com/spreadsheets/d/1VdRMFENPzjL2V-vZhcc_aa5-ysf243t5vXIxC2b054g/htmlview?pli=1

BlockChain Issues

1. P2P 네트웍의 안정성,신뢰성,성능
2. Security
3. 합의 알고리즘
4. 블록체인
   - 확장성 : 블록체인의 크기 증가 (비트코인 70GB/이더리움16GB)
   - Gas 비용
   - 마이닝 파워 집중화
   - 트랜젝션 처리 능역
     : Block interval: 하나의 블록이 전파되는데 걸리는 시간
     : Block size: 블록의 크기

- Case: Bitcoin
  : Block interval: 10 min (average)
  : Block size: 1MB (maximum)
  : 3.5 tps (TPS: Transactions per second)
  : block interval이 적고, block size가 클수록 확장성은 커짐

비트코인/이더리움 비교 (17년 8월 17일)

	비트코인	이더리움
블럭생성	평균 10 분 소요	평균 18.42초 소요
Tx/S	7개 처리	15개 처리
노드수 (2017/08/17)	9103 https://bitnodes.21.co	24088 https://www.ethernodes.org/network/1
블럭증명	PoW	PoW → PoS
최초블록생성	2009/01/03	2014/07
최대발행한도	2150년 21,000,000	2128년 이후 1억개 유지
블록보상	12.5BTC	5ETH
적용 범위	가상화폐	가상화폐 포함 전 산업 분야(플랫폼)

* 블록체인에서 TPS는 10분 후 생성되여 연결되는 블록내에 담을 수 있는 트렌젝션 수