Delivering Trusted Connectivity
㈜아이오트러스트

D'CENT
Delivering Trusted Connectivity

블록체인 지갑 기술 세미나

유민호 (Chief Strategy Officer)

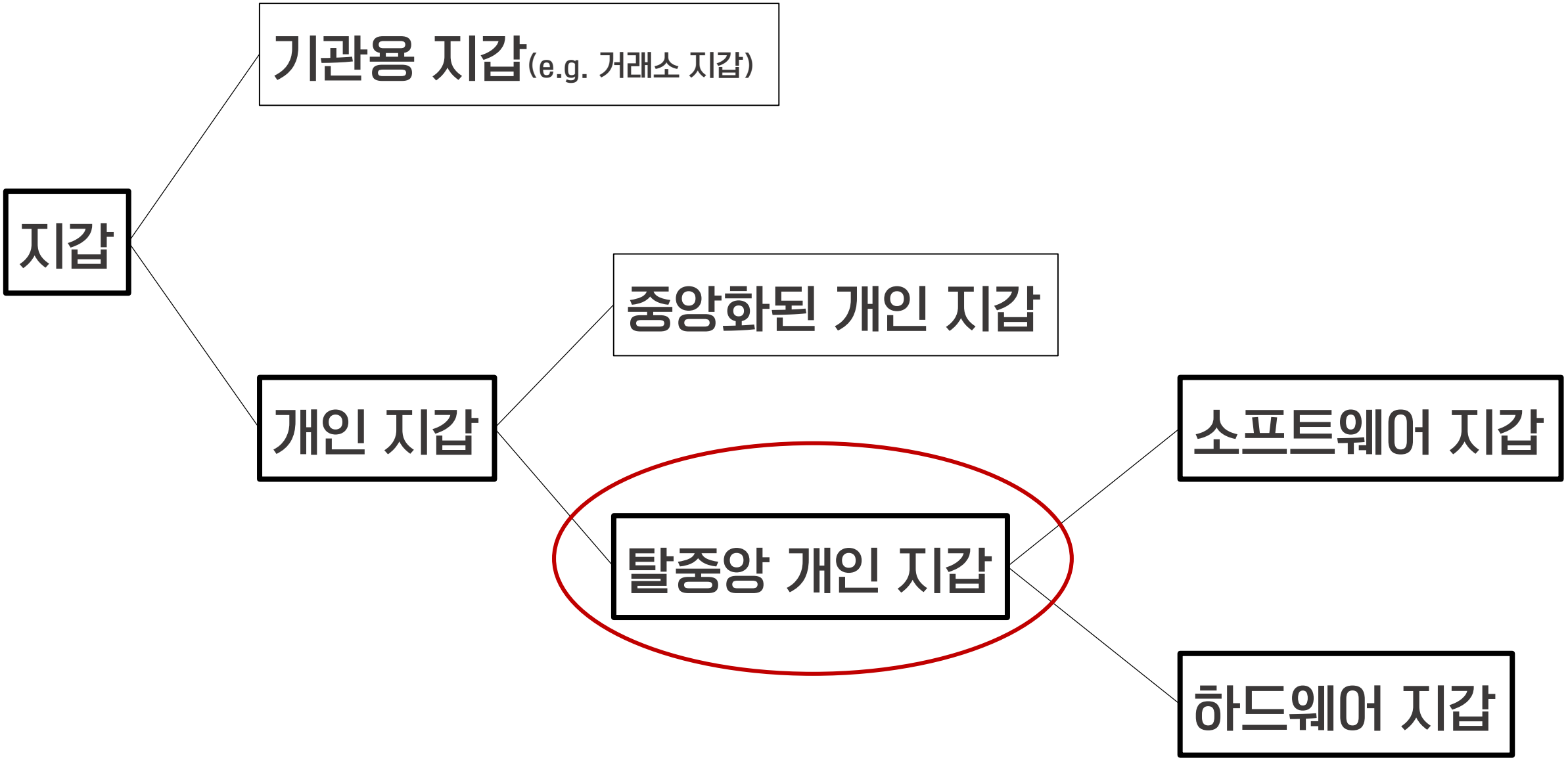
Delivering Trusted Connectivity
IOTRUST

AUG 2021

오늘 이야기 할 것

1. 블록체인 트랜잭션 (비트코인, 이더리움 중심)
2. 개인키 생성 및 관리 방식 (BIP 표준)
3. 하드웨어 지갑의 보안

지갑의 종류



블록체인과 지갑, 하드웨어 지갑의 역할

블록체인 : 공통의 프로토콜로
이루어진 노드들의 집합

ethereum

bitcoin

ripple

RSK

BINANCE

블록체인 네트워크

- Address & Account Scheme
- Transactions
- Data
- Virtual Machine

[블록체인 지갑]

- 블록체인 정보 조회
- 트랜잭션 전송
- GUI
- 주소 관리
- Unsigned TX 생성

모바일 or PC

하드웨어 지갑
인터페이스

- 주소 생성
- Unsigned TX 분석
- 트랜잭션 서명

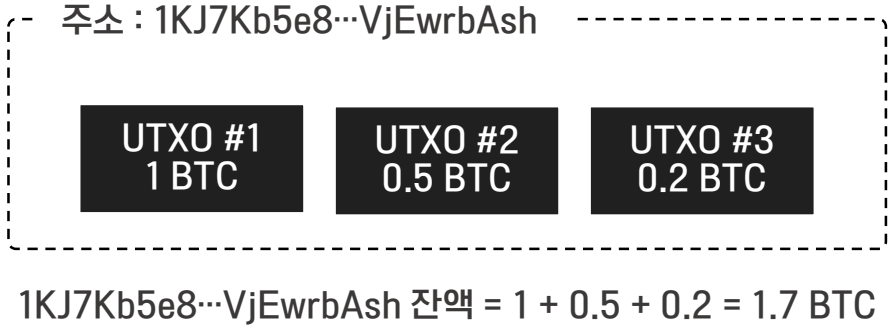
하드웨어 지갑

블록체인 트랜잭션 구조

트랜잭션 처리 방식에 따른 분류

UTXO 방식

UTXO : Unspent Transaction Output
별도의 계정 정보 없이, 사용되지 않은 트랜잭션 결과물 기반으로 관리



1KJ7Kb5e8…VjEwrbAsh 잔액 = 1 + 0.5 + 0.2 = 1.7 BTC

UTXO 방식의 블록체인 예시:
비트코인, 라이트코인, 비트코인캐시, 지캐시 등

어카운트 방식

계정 상태 정보를 기반으로 관리

어카운트 방식의 블록체인 예시:
이더리움, 바이낸스체인, XRPL, 클레이튼 등

UTXO 기반 트랜잭션 처리 방식 (비트코인 예시)

UTXO #1

TX #1: 주소 A —1 BTC→ 주소 C

UTXO #2

TX #2: 주소 B —3 BTC→ 주소 C

주소 C : 4 BTC

UTXO #1
1 BTC

UTXO #2
3 BTC

주소 C에서 주소 D로 3.5 BTC 전송

TX #3:

Input

주소 C

UTXO #1 (1 BTC)

UTXO #2 (3 BTC)

3.5 BTC → 주소 D (UTXO #3)

0.5 BTC → 주소 C (UTXO #4)

주소 C : 0.5 BTC

UTXO #4
0.5 BTC

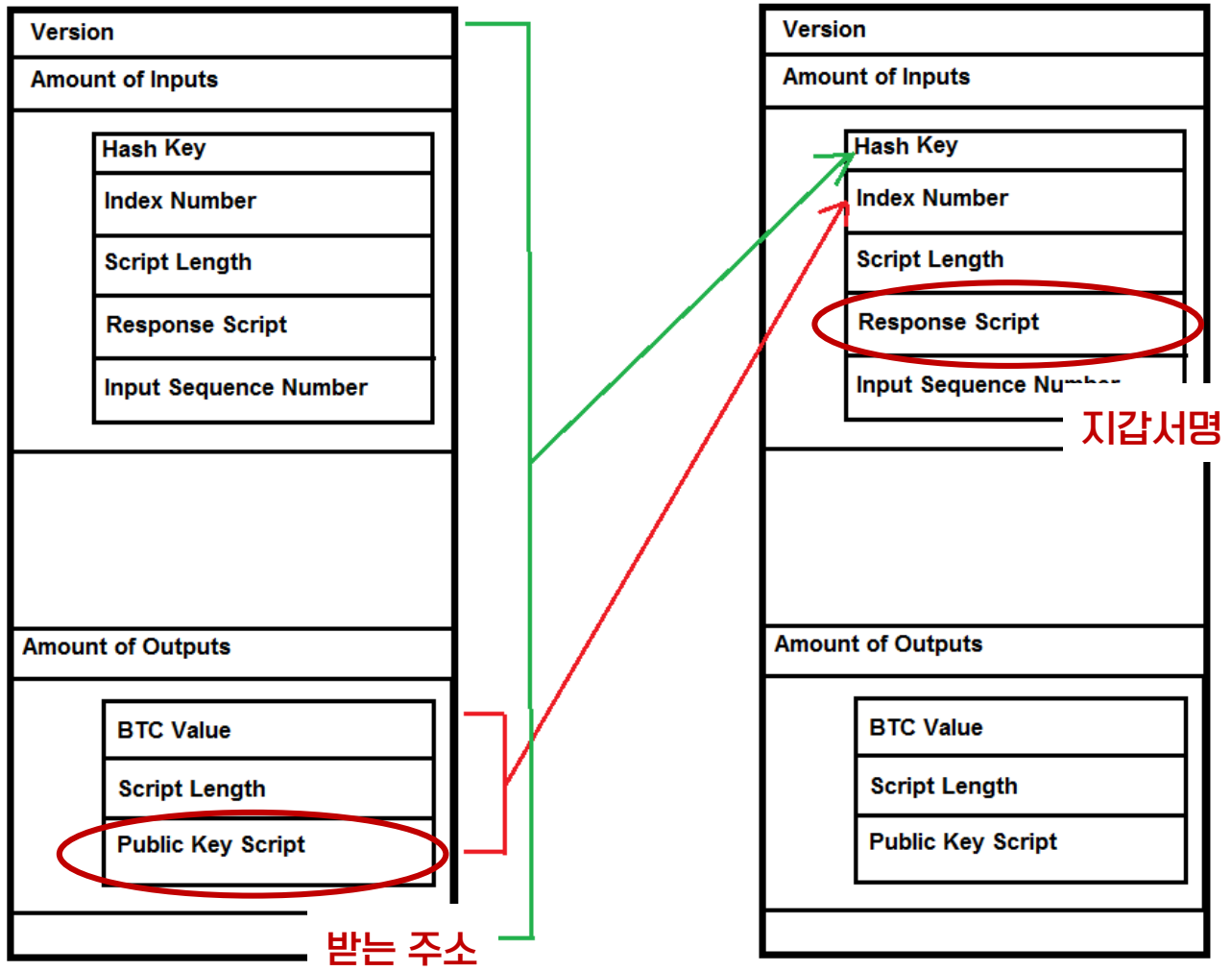
주소 D : 3.5 BTC

UTXO #3
3.5 BTC

전송시 어떤 UTXO를 Input으로 사용할지 판단이 복잡함

비트코인 트랜잭션의 구조

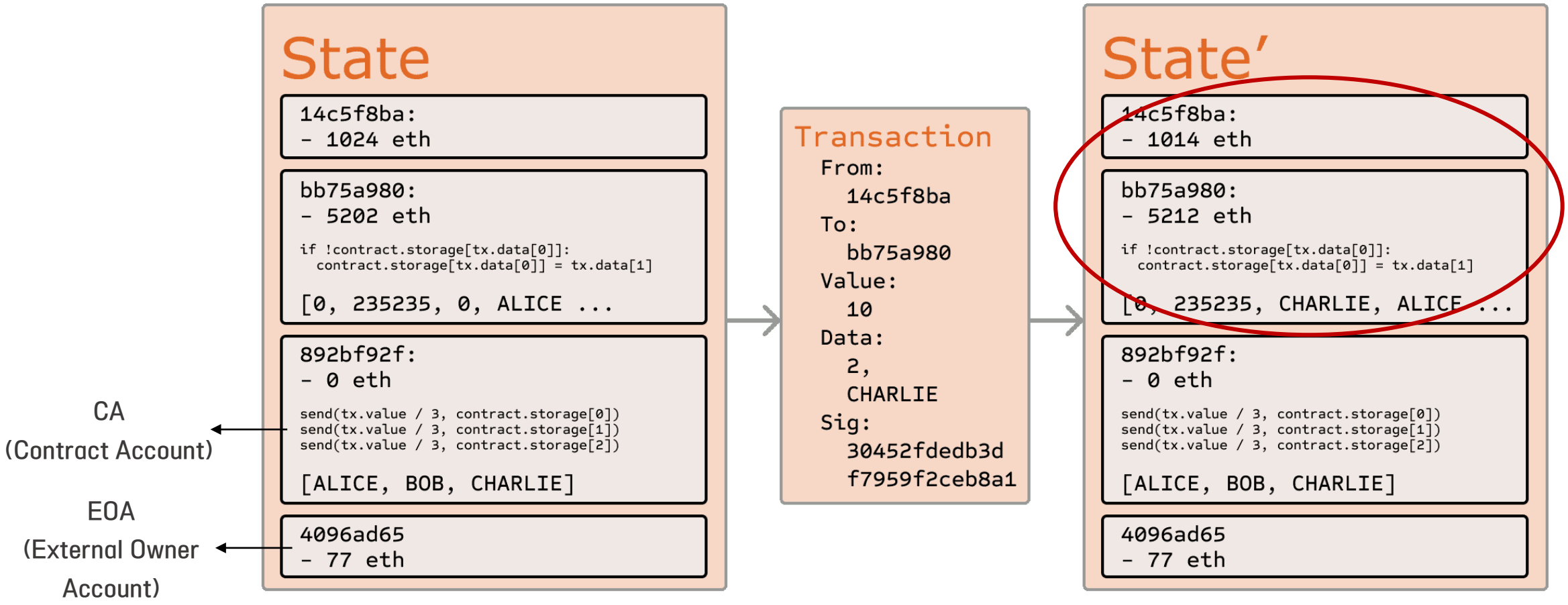
Field	Description
Version no	currently 1
Flag	If present, always 0001, and indicates the presence of witness data
In-counter	positive integer VI = VarInt
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)
Out-counter	positive integer VI = VarInt
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block
Witnesses	A list of witnesses, 1 for each input, omitted if flag above is missing
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final



스크립트 기반으로 서명 정보를 포함하기 때문에 멀티시그와 같은 기능 적용이 가능함

어카운트 기반 블록체인 (이더리움 예시)

계정마다 상태 정보 보유 → 트랜잭션을 통해 계정 상태가 변경됨



이더리움 트랜잭션의 구조

- **Ethereum Transaction**
  https://medium.com/@codetractio/inside-an-ethereum-transaction-fa94ffca912f

```
transaction = {
    nonce: web3.toHex(0),
    gasPrice: web3.toHex(20000000000),
    gasLimit: web3.toHex(100000),
    to: '0x687422eEA2cB73B5d3e242bA5456b782919AFc85',
    value: web3.toHex(1000),
    data: '0xc0de'
}
```

트랜잭션 일련번호

네트워크 수수료

계정 주소

ETH 수량

트랜잭션 데이터

rlp + hash

```
0x6a74f15f29c3227c5d1d2e27894da58d417a484ef53bc7aa57ee323b42ded656
```

sign with privateKey

```
v: '0x1c'
r: '0x668ed6500efd75df7cb9c9b9d8152292a75453ec2d11030b0eec42f6a7ace602'
s: '0x3efcbbf4d53e0dfa4fde5c6d9a73221418652abc66dff7fddd78b81cc28b9fbf'
```

signature

지갑에서 처리

```
signedTransaction = {
    nonce: web3.toHex(0),
    gasPrice: web3.toHex(20000000000),
    gasLimit: web3.toHex(100000),
    to: '0x687422eEA2cB73B5d3e242bA5456b782919AFc85',
    value: web3.toHex(1000),
    data: '0xc0de',
    v: '0x1c',
    r: '0x668ed6500efd75df7cb9c9b9d8152292a75453ec2d11030b0eec42f6a7ace602',
    s: '0x3efcbbf4d53e0dfa4fde5c6d9a73221418652abc66dff7fddd78b81cc28b9fbf'
};
```

rlp + hash

```
0x8b69a0ca303305a92d8d028704d65e4942b7ccc9a99917c8c9e940c9d57a9662
```

transaction id

개인키 생성 및 관리

개인키와 주소의 관계

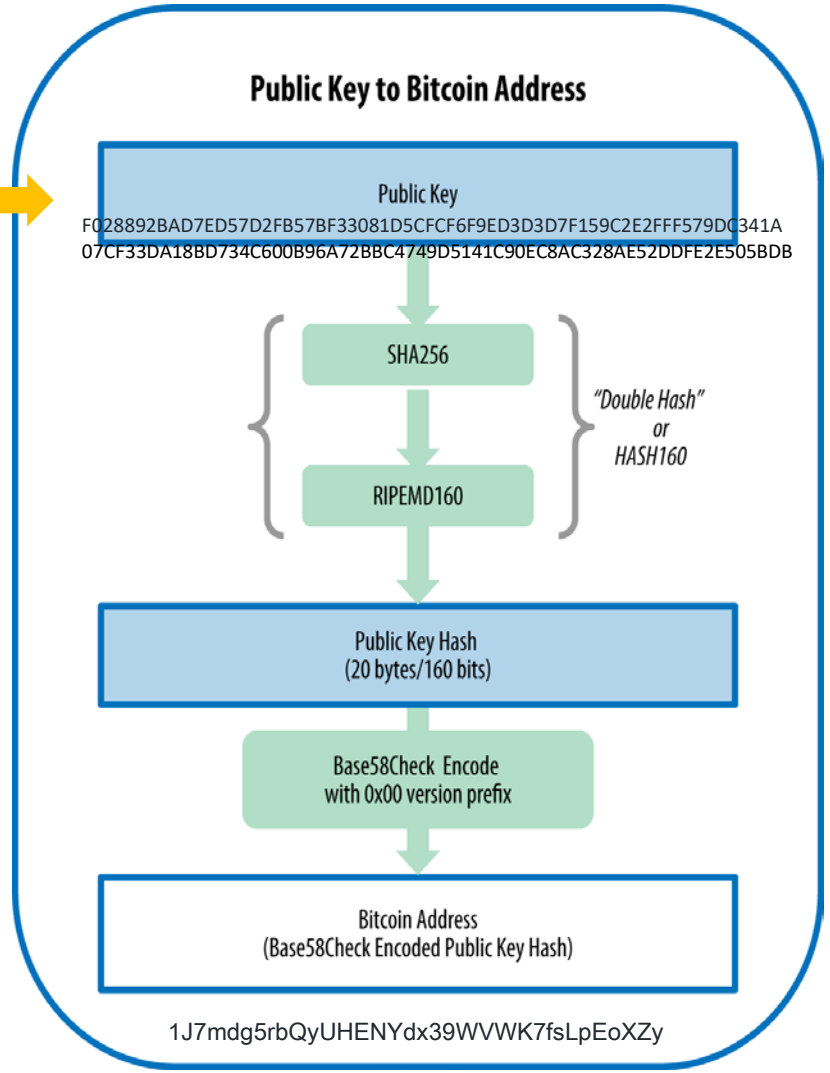
비밀 정보

공개 정보

개인키

1E99423A4ED27608A15A2616A2B0E9E5
2CED330AC530EDCC32C8FFC6A526AEDD

주소의 트랜잭션 서명 권한



- 하드웨어 지갑 내에서 개인키를 기반으로 주소 생성 후 주소 추출
- 주소 생성 방식은 블록체인 별로 달라짐
- 개인키는 하드웨어 지갑 밖으로 나오지 않음

추가 고려사항

- 지갑 분실/파손을 대비한 키의 백업과 복구는?
- 각 코인별, 계정별로 다른 키를 사용할 수 있을까?

개인키 백업과 복구 : 니모닉 코드 (BIP-39)

숫자로 이루어진 값을 사람이 읽을 수 있는 단어로 치환

→ 잘못 기록하는 상황을 방지

1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD + CS

2진수

11110100110 / 01010000100 / 01110100100 / 11101101001 / 00111011000 / 001000 …

11 bit

인덱스

1958

https://github.com/bitcoin/bips/blob/master/bip-0039/bip-0039-wordlists.md

1	abandon	…	2037	write
2	ability		2038	wrong
3	able		2039	yard
4	about		2040	year
5	above		2041	yellow
6	absent		2042	you
7	absorb		2043	young
8	abstract		2044	youth
9	absurd		2045	zebra
10	abuse		2046	zero
11	access		2047	zone
12	accident		2048	zoo

```
CS = ENT / 32
MS = (ENT + CS) / 11
```

ENT	CS	ENT+CS	MS
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

CS : Checksum Length
Checksum : First CS bits of SHA256(ENT)
ENT : Entropy Length
MS : Mnemonic Sentence

니모닉 코드

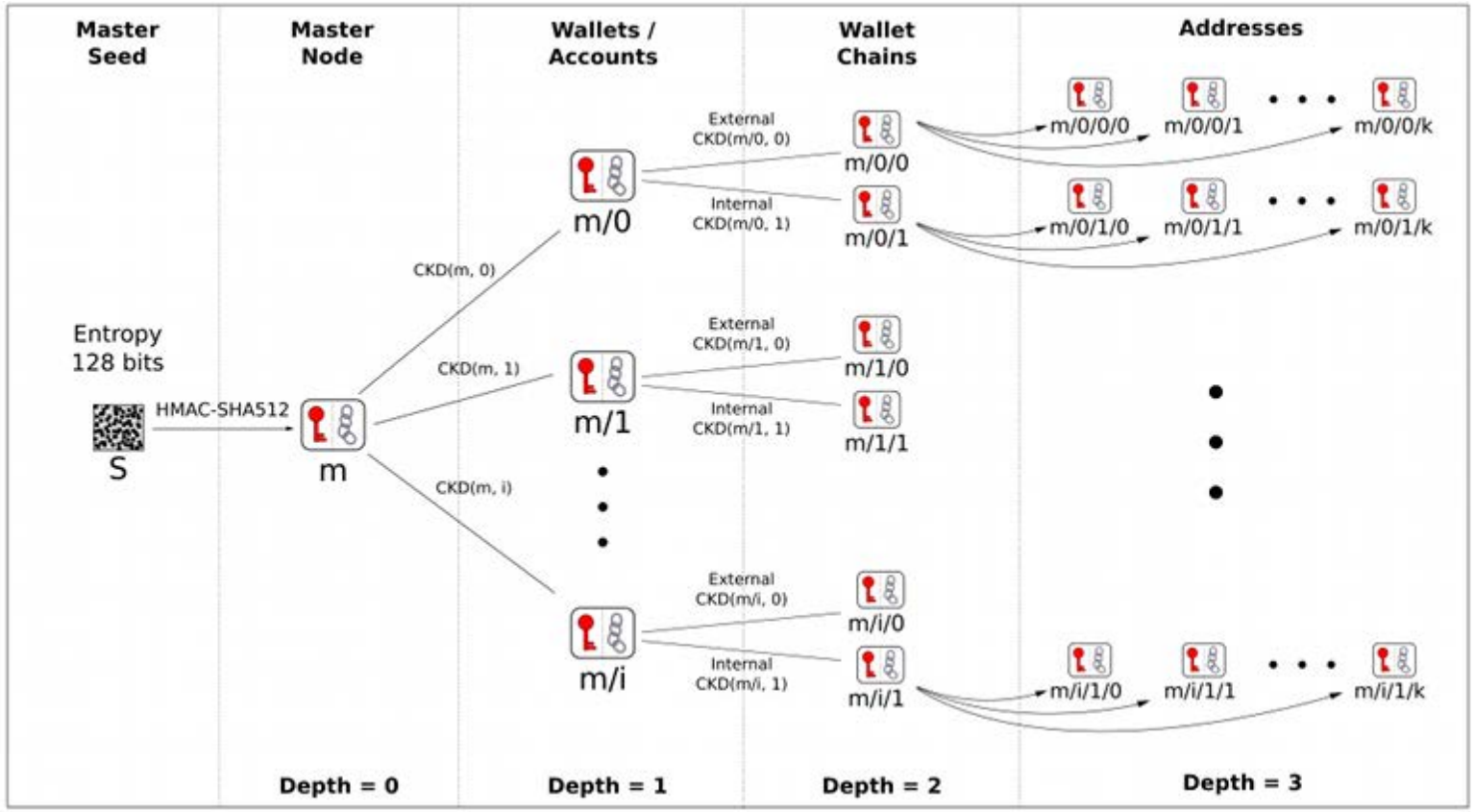
tragic industry what device tide return select west mass cute only opinion search omit maze labor cupboard sun slender radar clinic protect blade bomb

키의 계층 구조 : HD 지갑 (BIP32)

각 자산 종류별, 계정 별로 키를 분리해서 관리할 필요성

→ 하나의 키로부터 하위 키들을 유도해서 사용

→ 마스터키만 복원할 수 있다면 모든 자산에 대한 개인키 복원 가능



- **BIP44 (Multi-Account HD Wallet)**

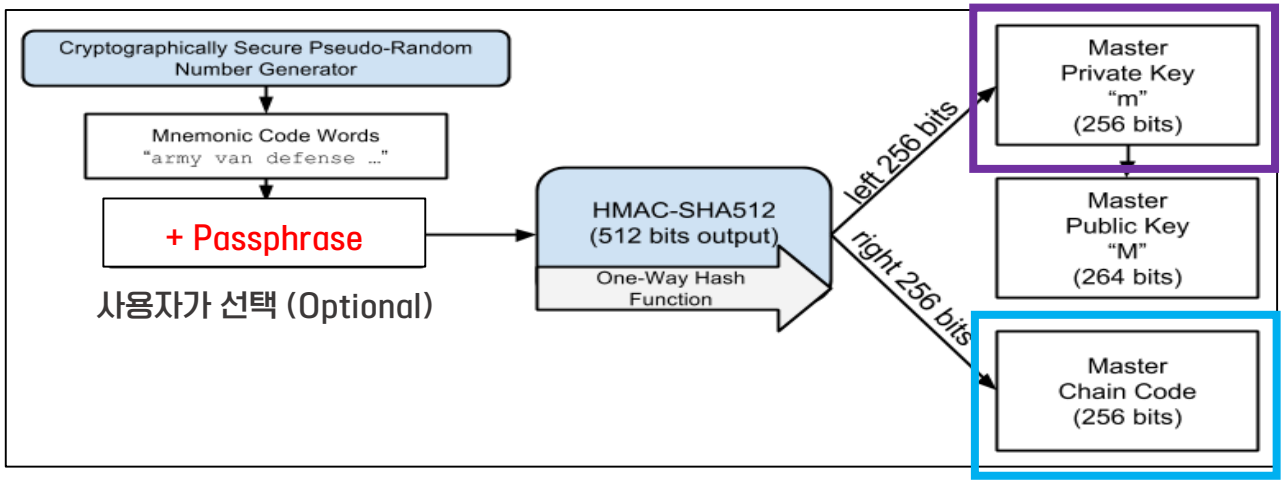
```
m / purpose' / coin_type' / account' / change / address_index
```

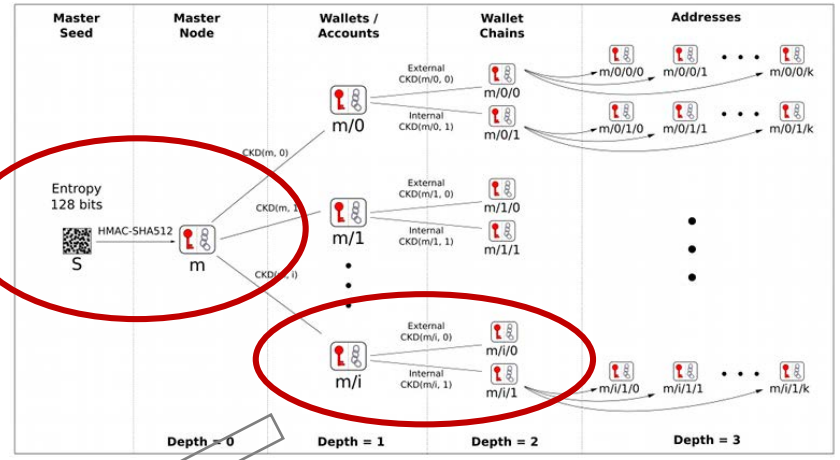
- m/44’/0’/0’/0/0
- m/44’/0’/1’/0/0
- m/44’/60’/0’/0/0

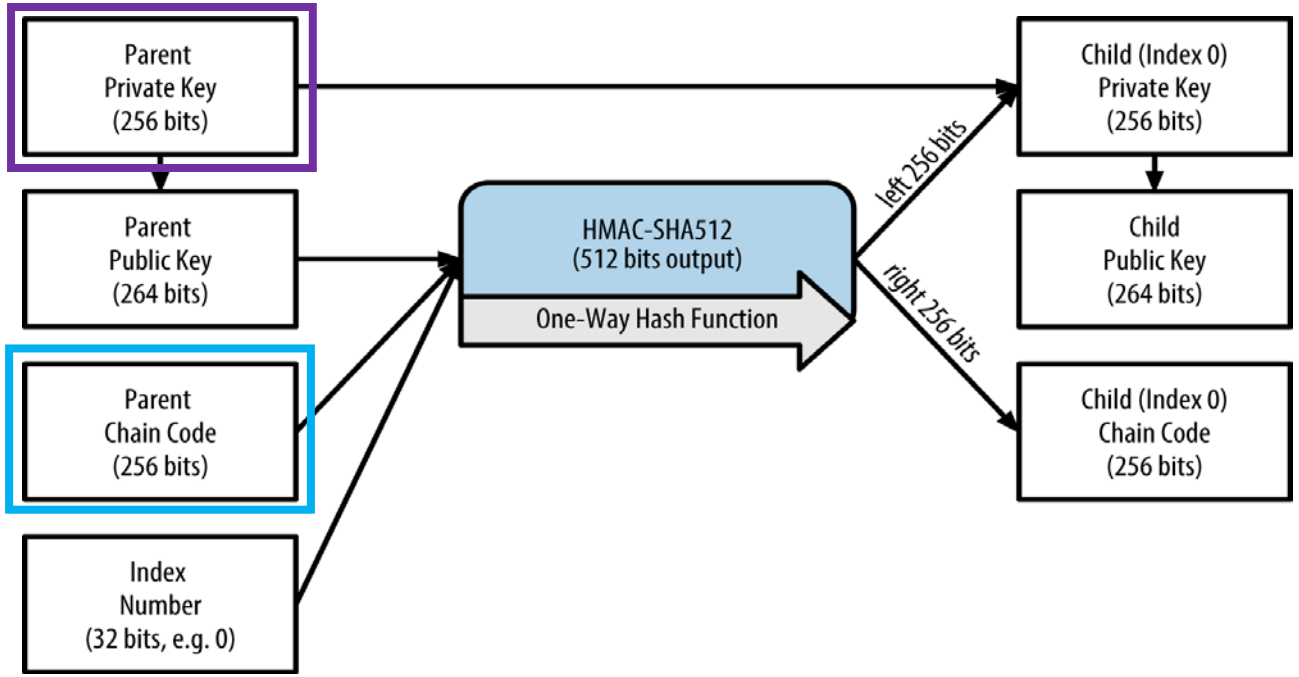
BIP44 경로의 운영 방식은 지갑마다 다름.

따라서, BIP44 경로를 알고 있다면 지갑간 호환 가능

BIP32 키 유도 방식







마스터 키만 있으면 모든 하위 키를 동적으로 계산할 수 있음.

→ 모든 종류의 개인키를 저장하고 있을 필요가 없음

하드웨어 지갑 및 보안 기술

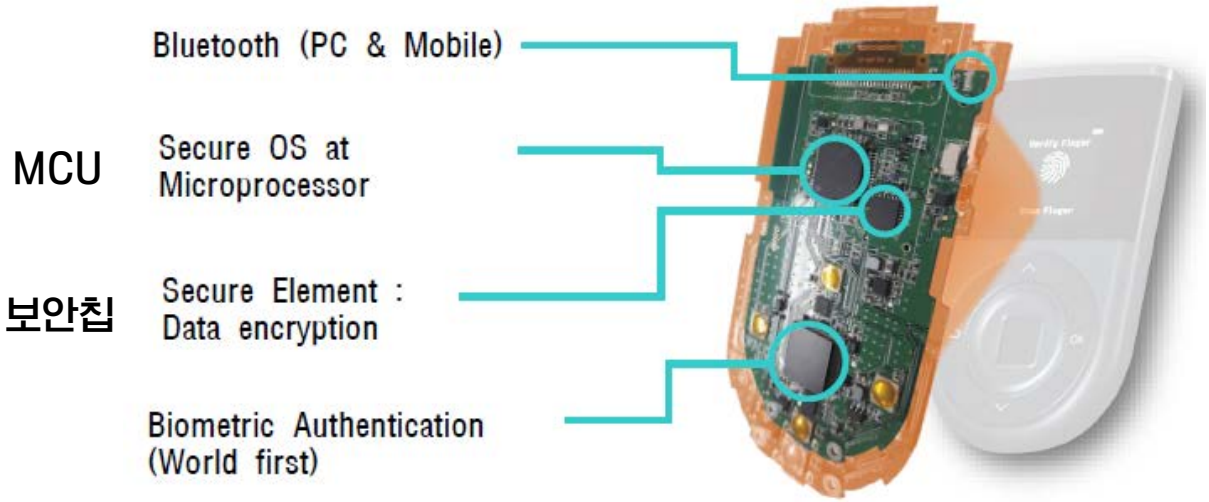
하드웨어 지갑 패키지

정상 패키지

비정상 패키지

공급망(Supply Chain) 공격 방지

디센트 지문 인증형 하드웨어 지갑



기본 보안 특징

- 신용카드, 유심칩에 사용되는 보안칩 탑재
- 개인키는 보안칩 안에서 생성 및 서명
- 사용자 PIN 및 지문 정보는 보안칩에서 처리 (지문 알고리즘 in SE)
- 시큐어 코딩 적용
- MCU 펌웨어 보호
- 트랜잭션 raw 데이터 처리 (중간자 공격 방지)

시큐어 코딩 예시

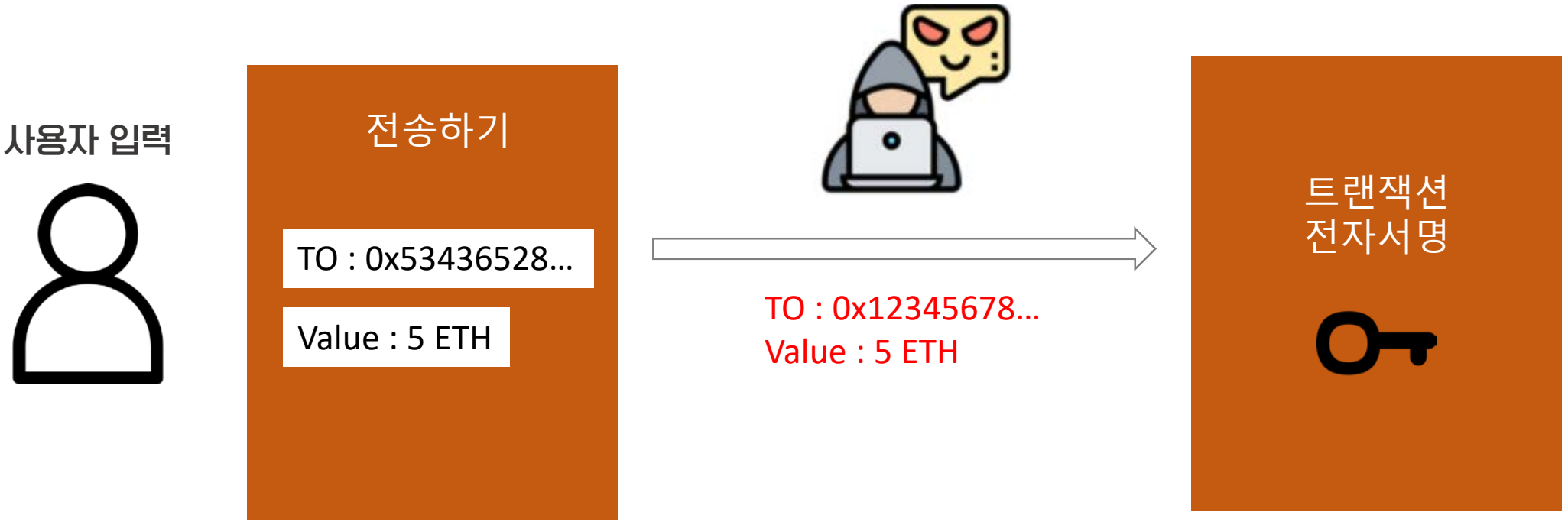
```
if (userVerified) {
   // Do something
} else {
   // Error procssing
}
```

```
#define TRUE 0x5A5A
#define FALSE 0xA5A5

if (userVerified == TRUE) {
   // Do something
} else if (userVerified == FALSE) {
   // Error processing
} else {
   // security assert
}
```

중간자 공격 (Man In The Middle Attack)

시나리오 #1: 보낼 때

주소 A에서 주소 B로 전송하려는 사용자에게
화면으로는 주소 B로 표시
내부 트랜잭션은 해커의 주소 C로 생성하여 서명 요청



중간자 공격 방지

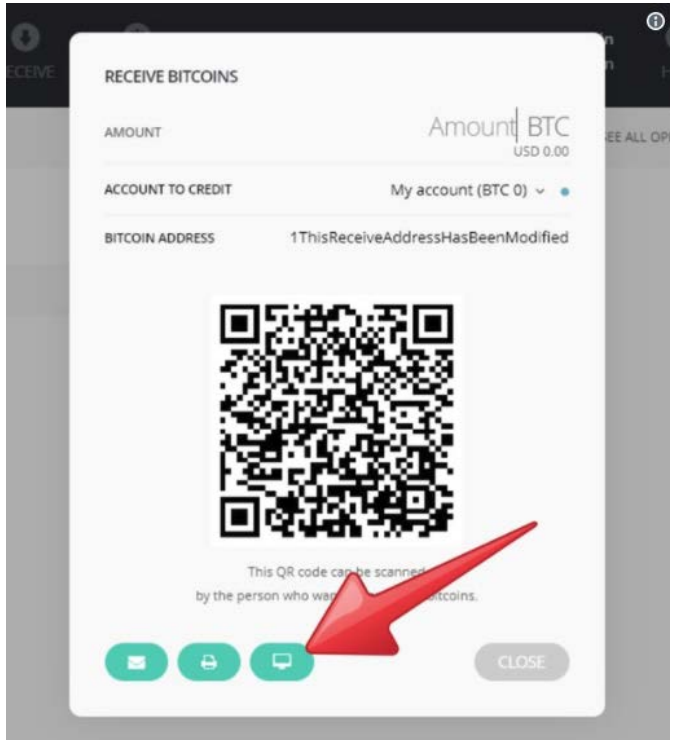
서명하려는 전체 트랜잭션 데이터를 하드웨어 지갑에서 처리
(하드웨어 지갑 펌웨어에 각 블록체인 트랜잭션 처리 모듈 구현)
하드웨어 지갑 디스플레이에는 실제 서명하려는 정보 표시


```
message transaction_begin_req_parameter_t {
        required uint32 version = 1;
        required uint32 tx_blk_size = 2;
        repeated transaction_input_t input = 5 [(nanopb).max_count = 50];
        repeated transaction_output_t output = 7 [(nanopb).max_count = 10];
        required uint32 locktime = 8;
}

message transaction_input_t {
        required uint32 prev_tx_size = 1;
        required uint32 utxo_idx = 2;
        required transaction_type_t type = 3;
        required string key_path = 4 [(nanopb).max_size = 51];
        optional uint32 sequence = 5;
}

message transaction_output_t {
        required transaction_type_t type = 1;
        required uint64 value = 2;
        repeated string to_address = 3 [(nanopb).max_size = 120, (nanopb).max_count = 3];
}
```

다른 중간자 공격 시나리오

시나리오 #2: 받을 때

받는 지갑 주소를 공격자의 주소로 변환



하드웨어 지갑에서 받는 주소 표시

타이핑 에러 방지를 위한 QR코드 생성

사용자 인증

간단한 구현 방식

- 보안칩(SE)에는 지문 템플릿만 저장
- MCU에서 지문 알고리즘 수행
- 지문 매칭 수행시 SE에서 지문 정보를 MCU로 읽어와서 매칭 확인

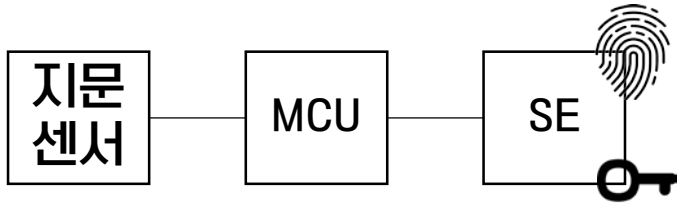
장점:

- 구현이 간단함

단점:

- 개인키 접근 권한 우회 가능

디센트 구현 방식



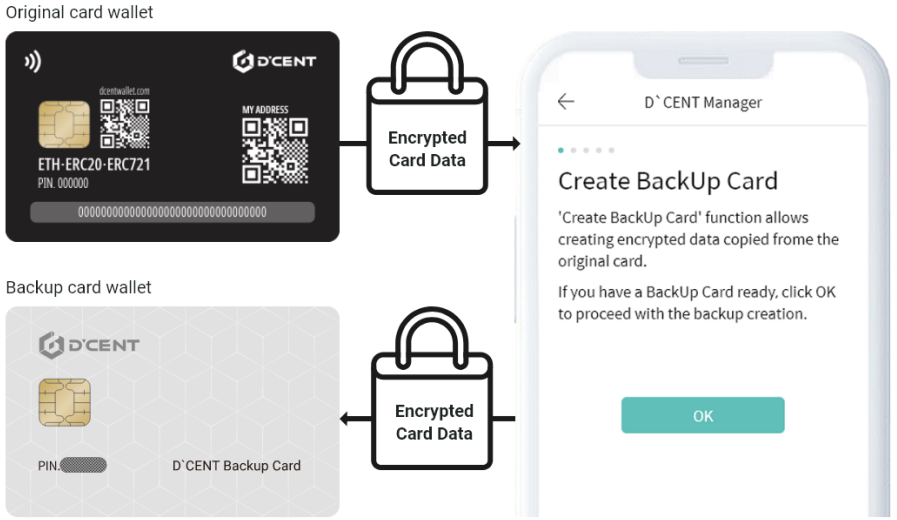
- 보안칩(SE)에 지문 템플릿 저장
- 보안칩에서 지문 매칭 알고리즘 수행
- 지문 매칭이 된 경우에만 개인키 접근 가능

사용자 인증 정책

- 사용자 인증에 반복적으로 실패할 경우,
  → 보안칩의 개인키 삭제 및 기기 초기화

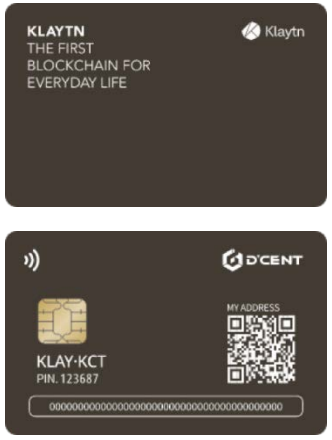
디센트 카드타입 하드웨어 지갑

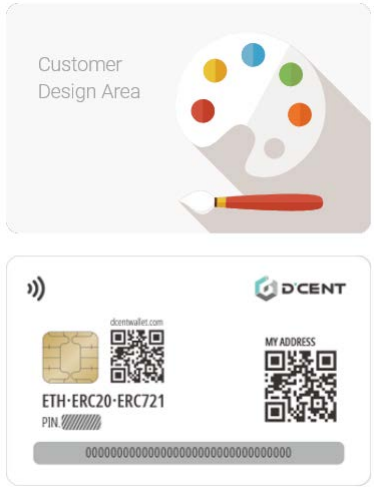
No Mnemonic Code

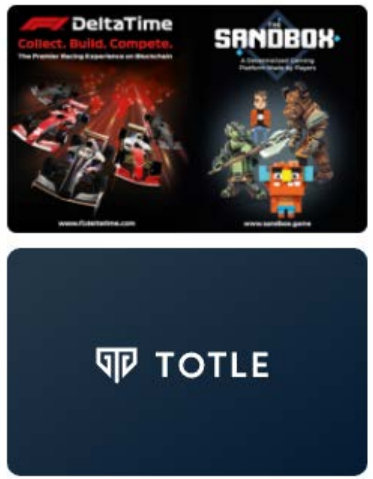


Ethereum Card

Klaytn Card



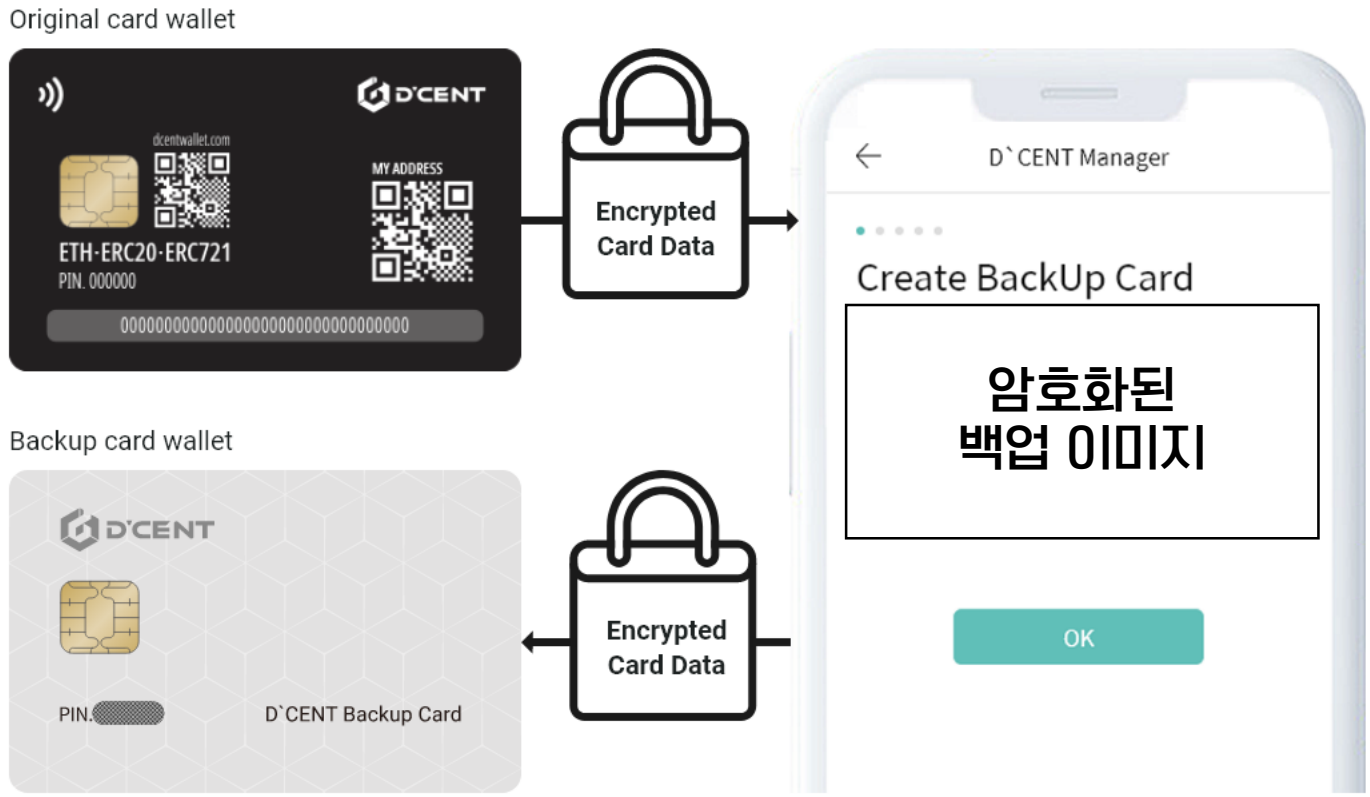




카드타입 지갑 백업 및 복구

사용자의 비밀번호를 기반으로 개인키 및 지갑 발급 정보에 대한 백업 이미지 생성

백업 이미지를 별도의 복구 카드에 복원



Thank You!

페이스북에서 “블파스“를 검색하세요!

https://www.facebook.com/groups/blpas