Let’s play with Bitcoin Cash

Try Swift New York 2019

September 10th, 2019

Jean-Baptiste Dominguez @jbdtky

Jean-Baptiste DOMINGUEZ

- Led wallet team @Bitcoin.com
- Swift lover
- Blockchain enthusiast
- Twitter / Github @jbdtky

Introduction

Bitcoin Cash

- Decentralized public ledger
- Blockchain technology
- Proof of work

 **No middleman / Public database**

 **Immutable**

 **Fork of Bitcoin**

 itcoin Cash

- 0 conf
- Low fee

 **Instant payment**

 **less than a penny**

Actors

- Full nodes
- Miners
- “Light nodes”

Interact with the actors

App light or heavy

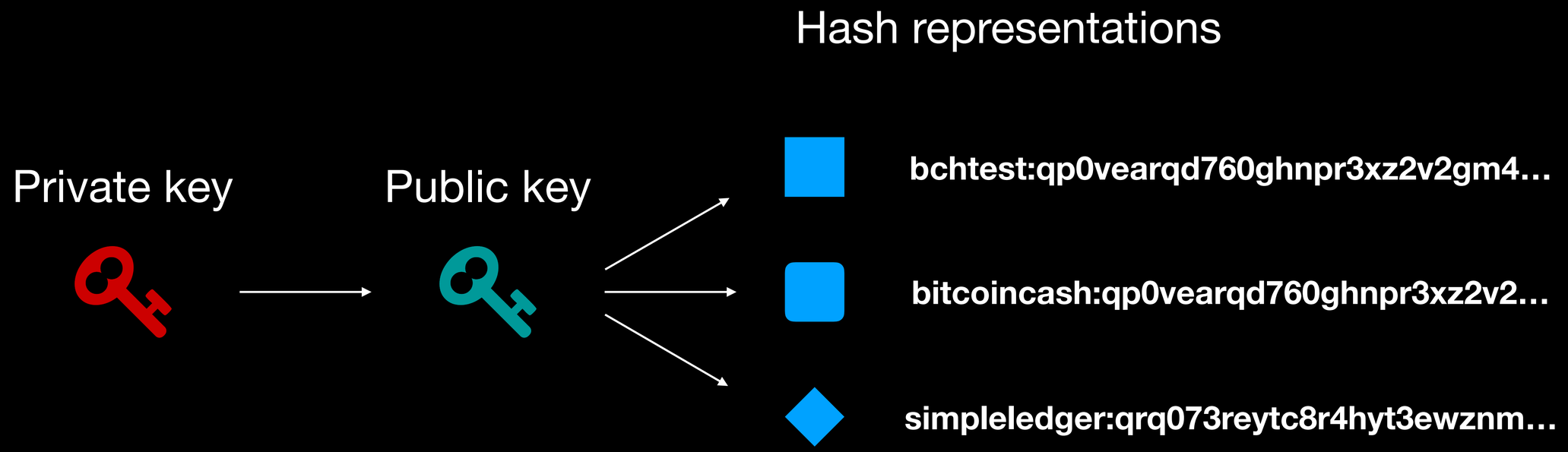
Katsumi Kishikawa

Yenom team

What is a Wallet?

Private key

What is a Wallet?



What is a transaction?

fee* = SUM(inputs.value) - SUM(outputs.value)

UTXO = Unspent Transaction Output

Best practices

- BIPs = Bitcoin Improvement Proposals
- BIP39 “Mnemonic code for generating deterministic keys”
- BIP32 “Hierarchical Deterministic Wallets”
- BIP43 “Purpose Field for Deterministic Wallets”
- BIP44 “Multi-Account Hierarchy for Deterministic Wallets”

BIP39

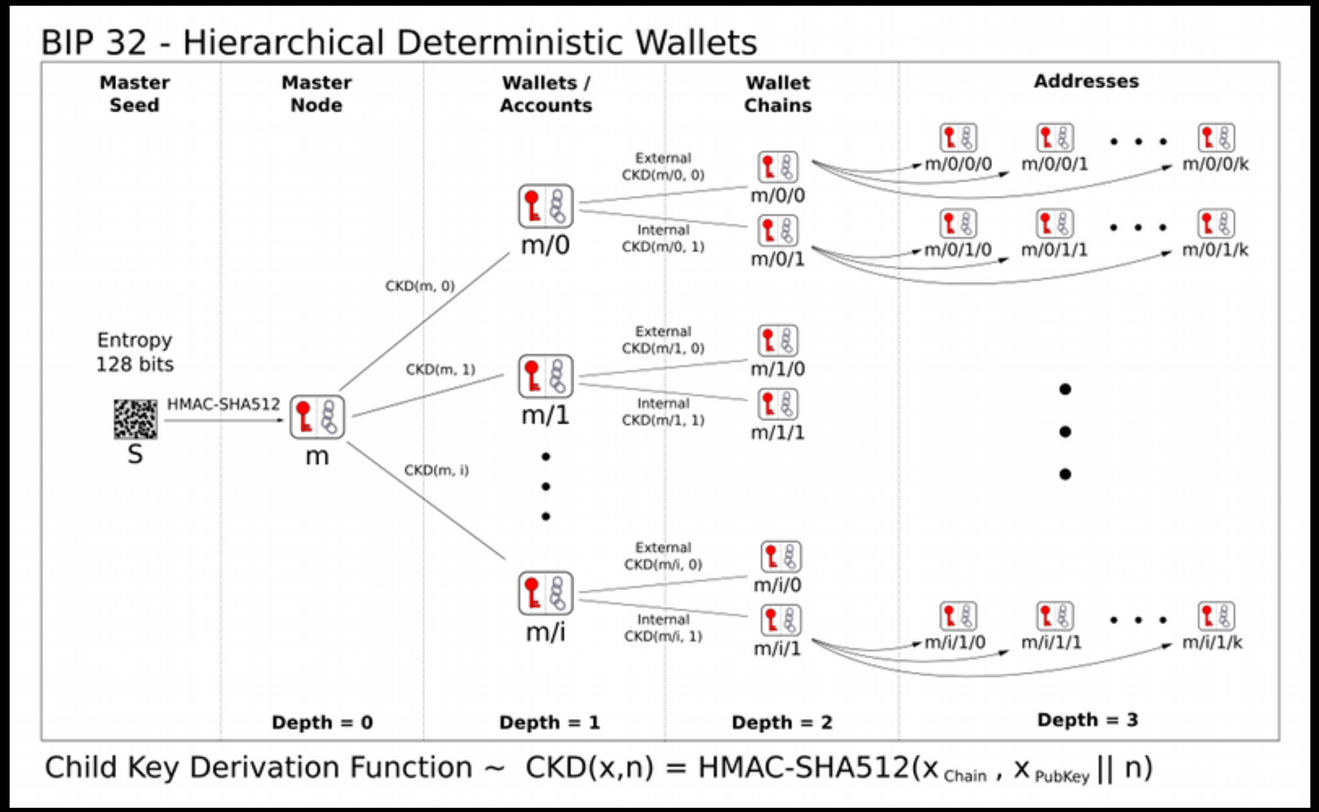
Mnemonic is a list of words

The following table describes the relation between the initial entropy length (ENT), the checksum length (CS) and the length of the generated mnemonic sentence (MS) in words.

```
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  |
```

BIP32


BIP 32 - Hierarchical Deterministic Wallets
Master Seed
Master Node
Wallets / Accounts
Wallet Chains
Addresses
Entropy 128 bits
S
HMAC-SHA512
m
CKD(m, 0)
CKD(m, 1)
CKD(m, i)
m/0
m/1
m/i
External CKD(m/0, 0)
Internal CKD(m/0, 1)
External CKD(m/1, 0)
Internal CKD(m/1, 1)
External CKD(m/i, 0)
Internal CKD(m/i, 1)
m/0/0
m/0/1
m/1/0
m/1/1
m/i/0
m/i/1
m/0/0/0
m/0/0/1
m/0/0/k
m/0/1/0
m/0/1/1
m/0/1/k
m/i/1/0
m/i/1/1
m/i/1/k
Depth = 0
Depth = 1
Depth = 2
Depth = 3
Child Key Derivation Function ~ CKD(x,n) = HMAC-SHA512(x Chain , x PubKey || n)

BIP32 + 43 + 44

Derivation path

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

m / 44' / 145' / 0' / 0 / 0
m / 44' / 145' / 0' / 0 / 1
m / 44' / 145' / 0' / 0 / 2
m / 44' / 145' / 0' / 0 / 3
...

Create a master key

```
let mnemonic = try Mnemonic.generate()
let seed = Mnemonic.seed(mnemonic: mnemonic)
let masterPrivKey = HDPrivateKey(seed: seed, network: .mainnet)
```

Derived child keys

```
// m/purpose'/coin'/account'
// m/44'/145'/0'
let masterBchPrivKey = try masterPrivKey
    .derived(at: 44, hardened: true)
    .derived(at: 145, hardened: true)
    .derived(at: 0, hardened: true)

// m/44'/245'/0'
let masterSlpPrivKey = try masterPrivKey
    .derived(at: 44, hardened: true)
    .derived(at: 245, hardened: true)
    .derived(at: 0, hardened: true)
```

Derived addresses

```
// m/change/address
// m/0/0
let privKey = try masterBchPrivKey
    .derived(at: 0)
    .derived(at: 0)
    .privateKey()

let cashAddress = privKey
    .publicKey()
    .toCashaddr()
```

Locking script

- P2PKH = Pay to Public Key Hash
- P2SH = Pay to Script Hash
- Null Data

OP_CODE

```
45  /** Script opcodes */
46  enum opcodetype {
47      // push value
48      OP_0 = 0x00,
49      OP_FALSE = OP_0,
50      OP_PUSHDATA1 = 0x4c,
51      OP_PUSHDATA2 = 0x4d,
52      OP_PUSHDATA4 = 0x4e,
53      OP_1NEGATE = 0x4f,
54      OP_RESERVED = 0x50,
55      OP_1 = 0x51,
56      OP_TRUE = OP_1,
57      OP_2 = 0x52,
58      OP_3 = 0x53,
59      OP_4 = 0x54,
60      OP_5 = 0x55,
61      OP_6 = 0x56,
62      OP_7 = 0x57,
63      OP_8 = 0x58,
64      OP_9 = 0x59,
65      OP_10 = 0x5a,
66      OP_11 = 0x5b,
67      OP_12 = 0x5c,
68      OP_13 = 0x5d,
69      OP_14 = 0x5e,
70      OP_15 = 0x5f,
71      OP_16 = 0x60,
72
```

https://github.com/Bitcoin-ABC/bitcoin-abc/blob/master/src/script/script.h

Make a P2PKH script

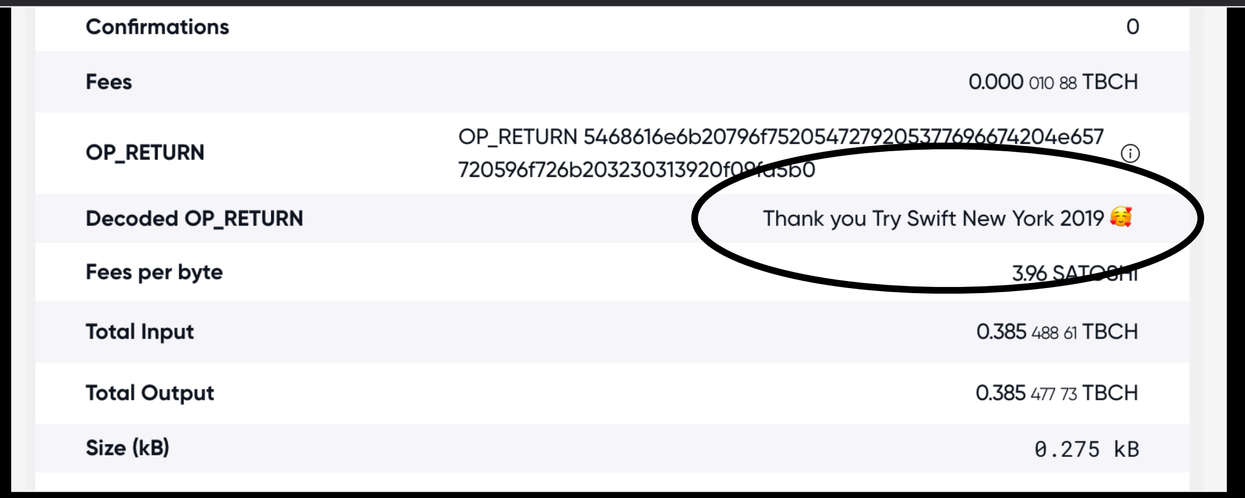
```
let toCashAddress = try
    AddressFactory.create("bchtest:qryyn74spvqgg75hk8rzsum4j6nuzkddkvqszzuwem")

let scriptTo = Script()
try scriptTo.append(.OP_DUP)
    .append(.OP_HASH160)
    .appendData(toCashAddress.data)
    .append(.OP_EQUALVERIFY)
    .append(.OP_CHECKSIG)

let scriptToEasy = Script(address: toCashAddress)
```

Make a null data script

```
let scriptMessage: Script?
if let messageData = "Thank you Try Swift New York 2019 🥰".data(using: .utf8) {
    scriptMessage = try Script()
        .append(.OP_RETURN)
        .appendData(messageData)
}
```



Simple Ledger Protocol

- Token protocol
- Leverage OP_RETURN
- Low cost

https://simpleledger.cash/

Conclusion

- Powerful technology
- Great opportunity for businesses
- Simple Ledger Protocol

References

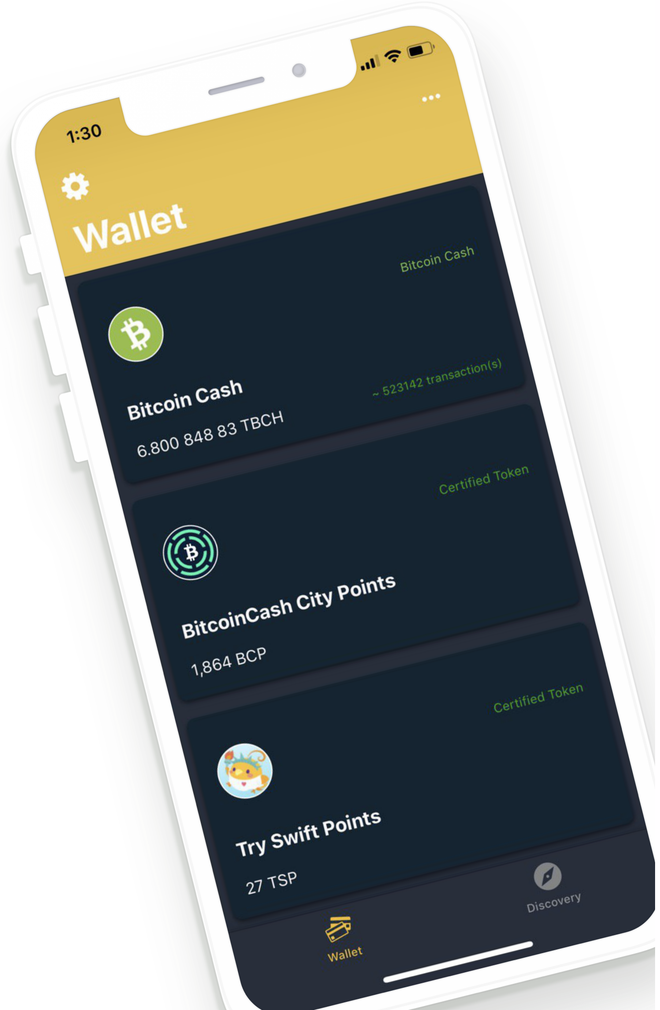
- https://www.bitcoincash.org/specs/
- https://github.com/bitcoin/bips
- https://en.bitcoin.it/wiki/Main_Page
- https://simpleledger.cash/
- https://github.com/jbdtky/BitcoinKit
- https://github.com/yenom/BitcoinKit
- https://learnmeabitcoin.com
- https://github.com/Bitcoin-ABC/bitcoin-abc

Thank you 😀✌🏻

- Jean-Baptiste Dominguez
- Twitter @jbdtky
- Github @jbdtky
- Telegram @jbdtky

Honey Points

Get crypto rewards from your favorite places.



Jean-Baptiste Dominguez @jbdtky