

Secure wallet application for cryptocurrency and blockchain transactions

by Viet-Sang Nguyen

Monday, September 6th, 2021



CryptoExperts

- ◆ Founded in 2009, based in Paris
- ◆ Research team & service company
- ◆ Strong focus on cryptography & security of embedded systems
- ◆ Services of custom crypto design, implementation, evaluation
- ◆ Software & technologies
 - Secure embedded crypto libraries
 - White-box cryptography
 - Fully Homomorphic Encryption
- ◆ Website: www.cryptoexperts.com

Outlines

1. Introduction

- ◆ Context
- ◆ Goal

2. Keys and addresses for cryptocurrencies

- ◆ Address derivation
- ◆ Privacy problem
- ◆ Tree-like structure of keys (HD Wallet)

3. Transactions

- ◆ Transaction components (Bitcoin)
- ◆ How to create a transaction?

4. Secure wallet architecture

- ◆ Account on wallet
- ◆ Token generator and usage

5. Summary

Outlines

1. Introduction

- ◆ Context

- ◆ Goal

2. Keys and addresses for cryptocurrencies

3. Transactions

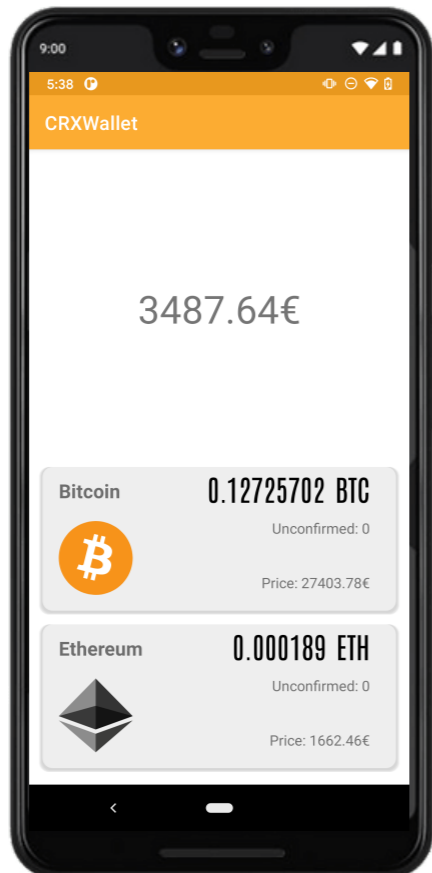
4. Secure wallet architecture

5. Summary

Context

- ◆ A valid transaction is signed by ECDSA
- ◆ One has control over coins if she has the private key

Responsibilities of a wallet



Manage Keys
Create transactions
Sign transactions
Broadcast transactions

→ Key protection is extremely important

Context

- ◆ Many wallets “help” users to manage keys
- ◆ Risk: keys are stored on smartphone (open environment)



HELEN PARTZ MAR 31, 2021

iPhone user blames Apple for \$600K Bitcoin theft via fake app

Apple removed the fake Trezor app several times, but it kept appearing on the App Store days later.

16874 Total views 377 Total shares Listen to article 2:05

An illustration showing a hand holding a smartphone with a cracked screen. The screen displays a Bitcoin symbol. The background is dark with several red Bitcoin symbols floating around. The word 'COINTELEGRAPH' is visible in the top left corner of the illustration, and a 'NEWS' tag is in the bottom right corner.

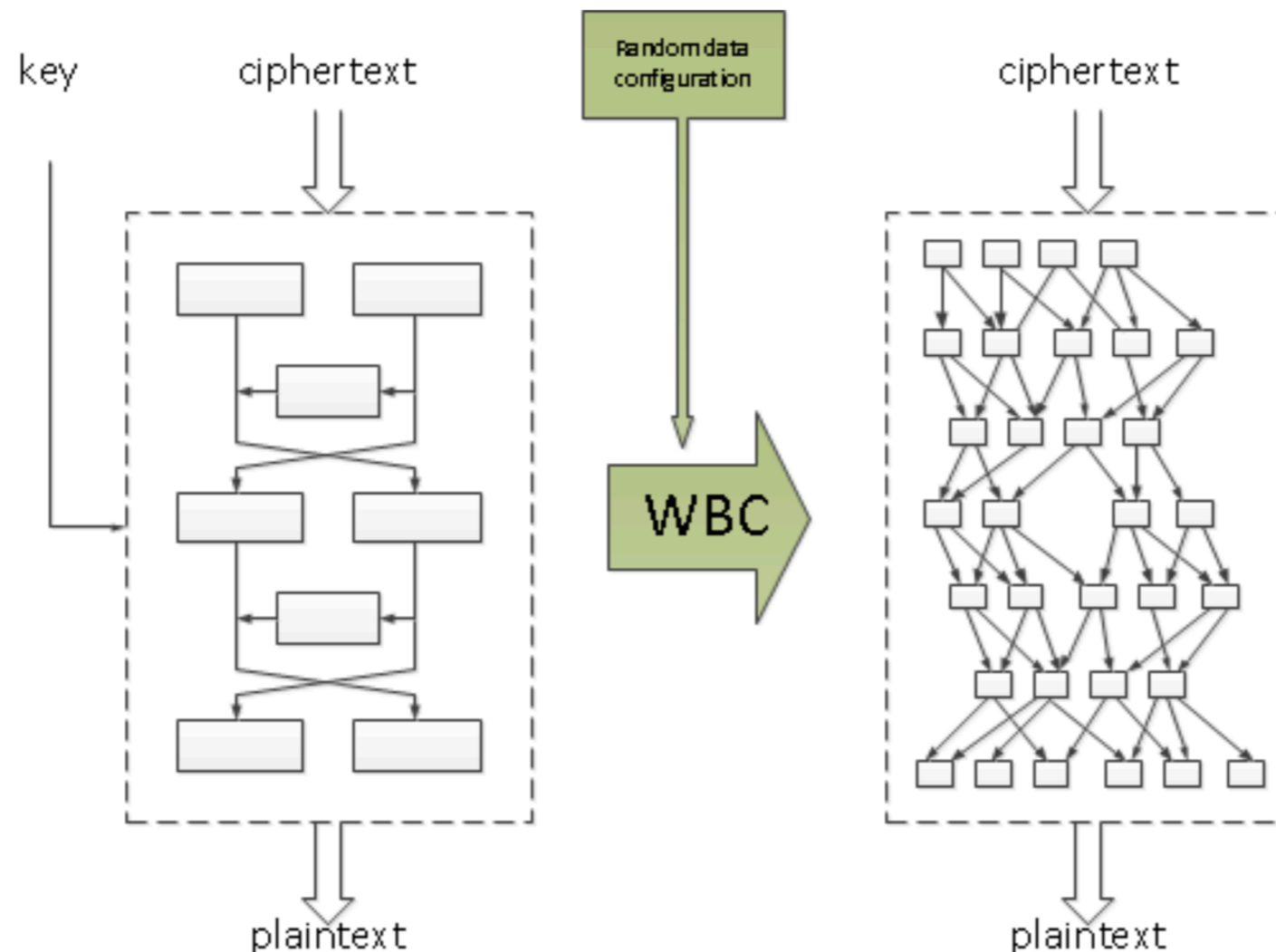
f A scam cryptocurrency app on Apple's app distribution service App Store has reportedly stolen \$600,000 Bitcoin (BTC) from one iOS user.

t Cryptocurrency holder Phillippe Christodoulou fell victim to a scam app on the App Store, losing nearly all his life savings to a fake crypto wallet application, The

→ Need a secure wallet app

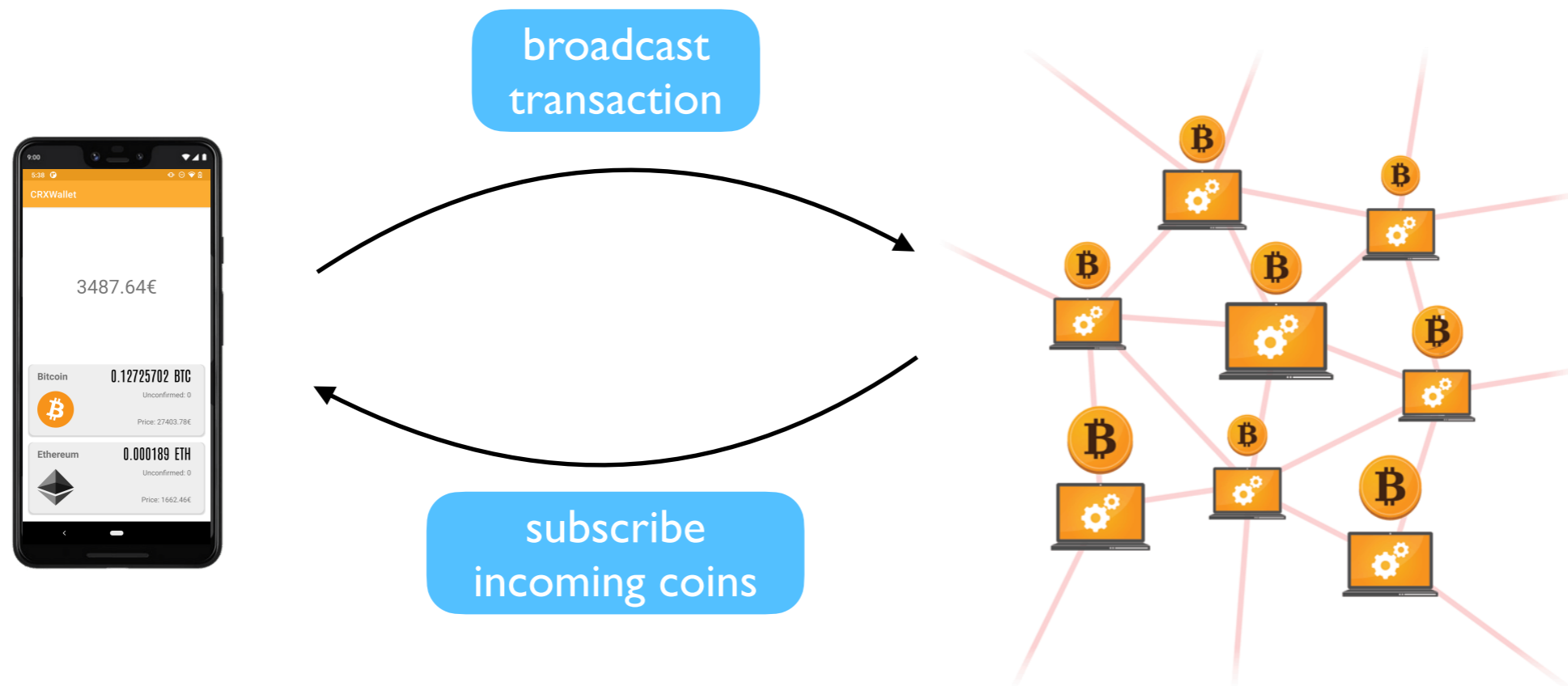
White-Box Cryptography

- ◆ Hide the secret key in an obfuscated cryptographic implementation
- ◆ An attacker is assumed to have
 - full access to the software
 - control of the execution environment
- ◆ Our main goal is to make the key extraction difficult



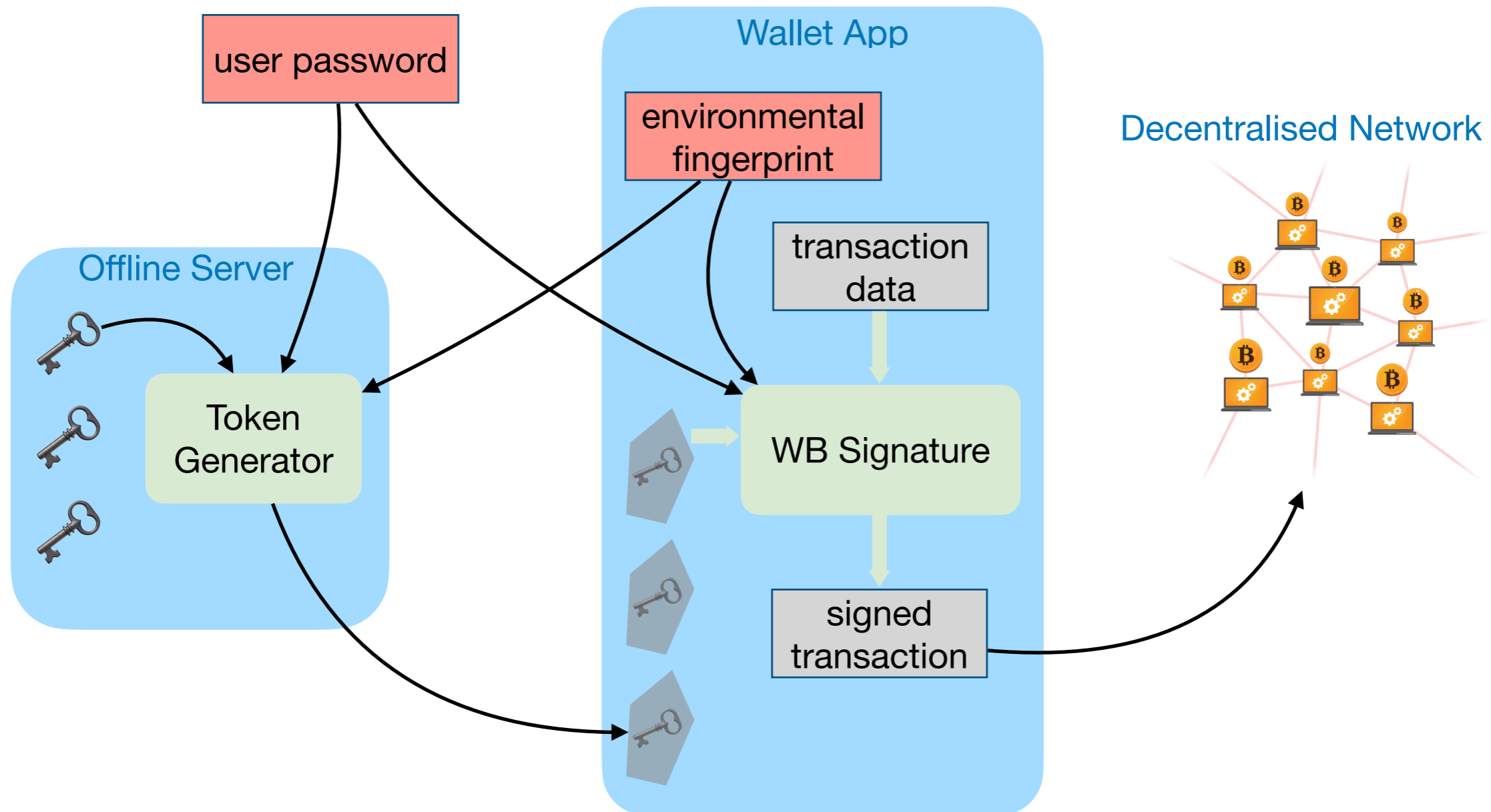
Goal

- ◆ To build a cryptocurrency wallet application
 - It is capable of sending/receiving coins
 - Transactions are *signed by White-box ECDSA*
- ◆ This app supports Bitcoin and Ethereum transactions



Overview of Architecture

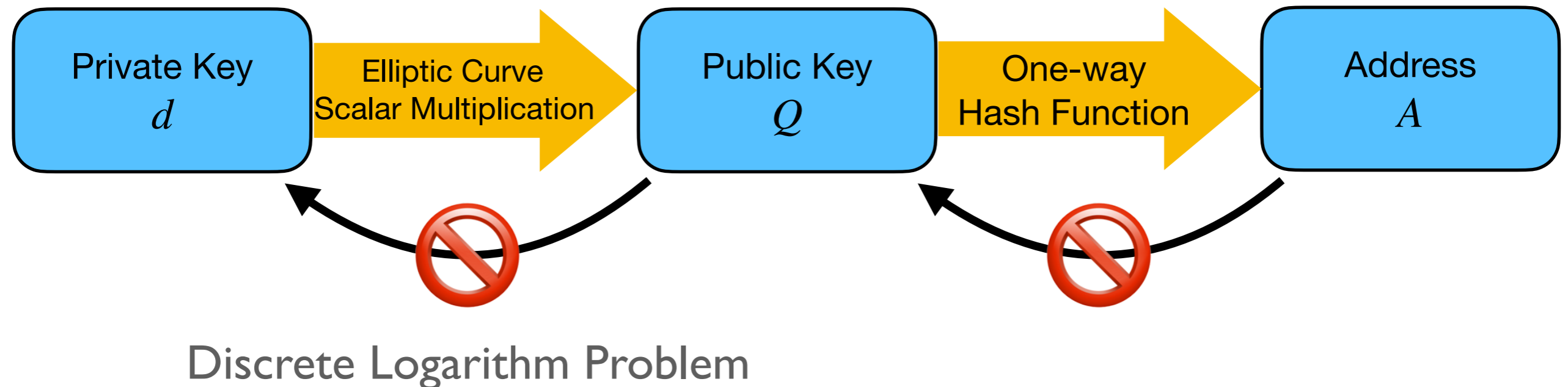
- ◆ A token is a secure container for a key
 - generated by a trusted server
 - operated by a white-box signature
- ◆ Server is deployed on a trusted and isolated environment



Outlines

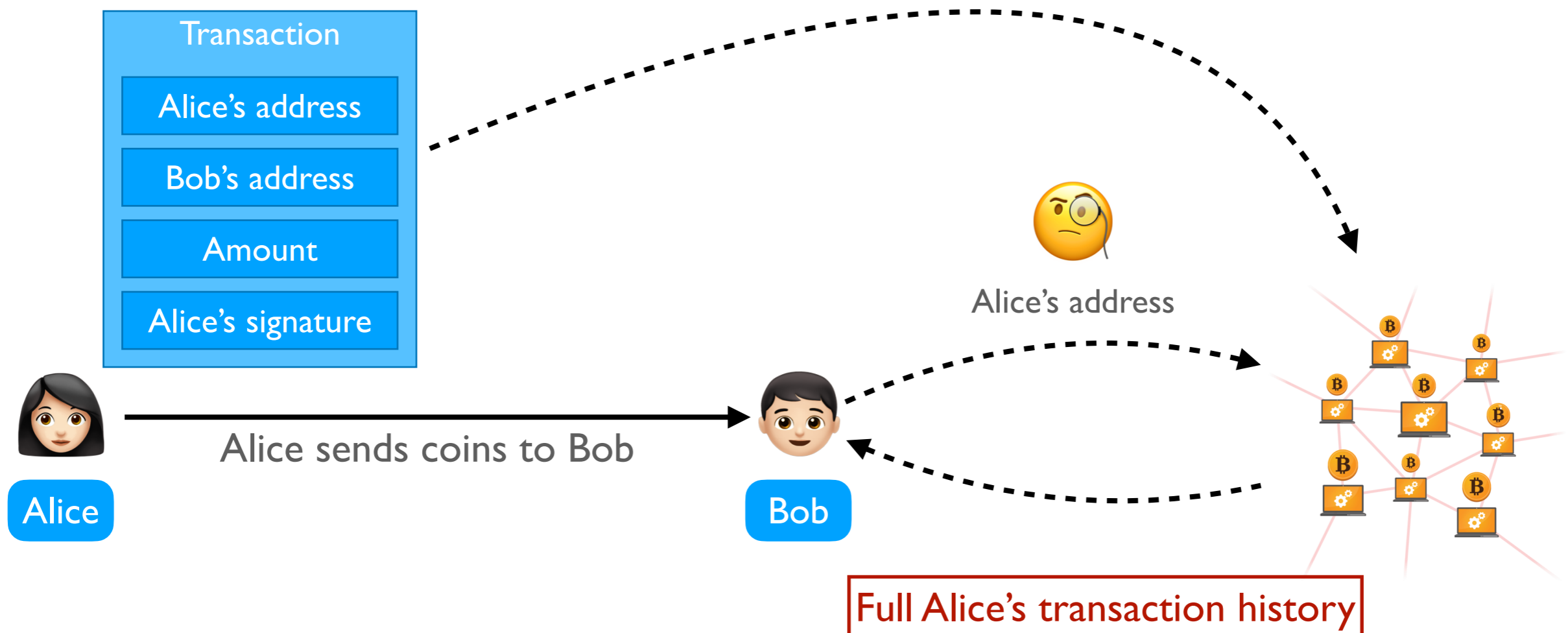
1. Introduction
2. Keys and addresses for cryptocurrencies
 - ◆ Address derivation
 - ◆ Privacy problem
 - ◆ Tree-like structure of keys (HD Wallet)
3. Transactions
4. Secure wallet architecture
5. Summary

Key and Address in Cryptocurrencies



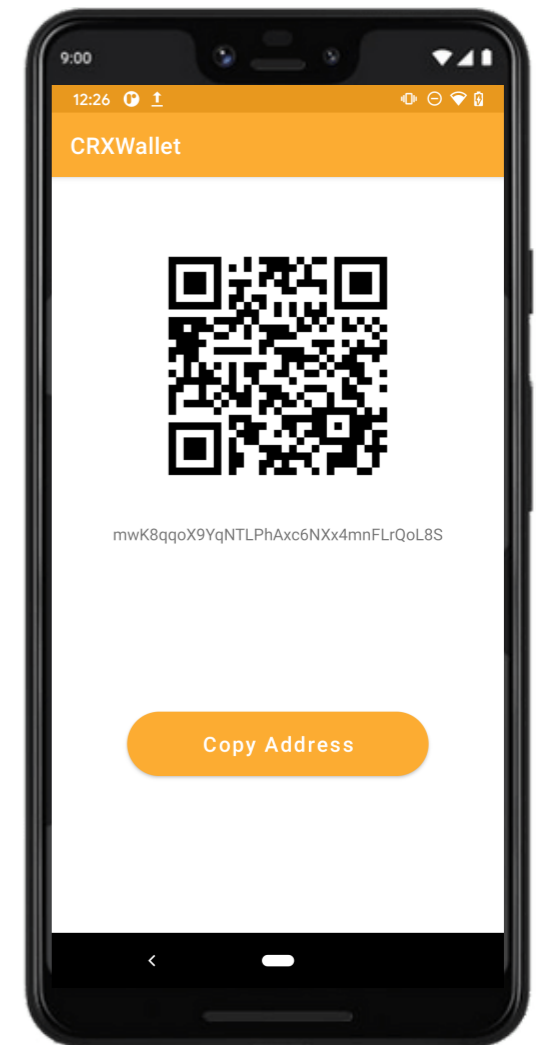
Privacy problem

- ◆ If Alice uses only one address for many transactions...
It is fine. BUT...



Privacy problem: solution

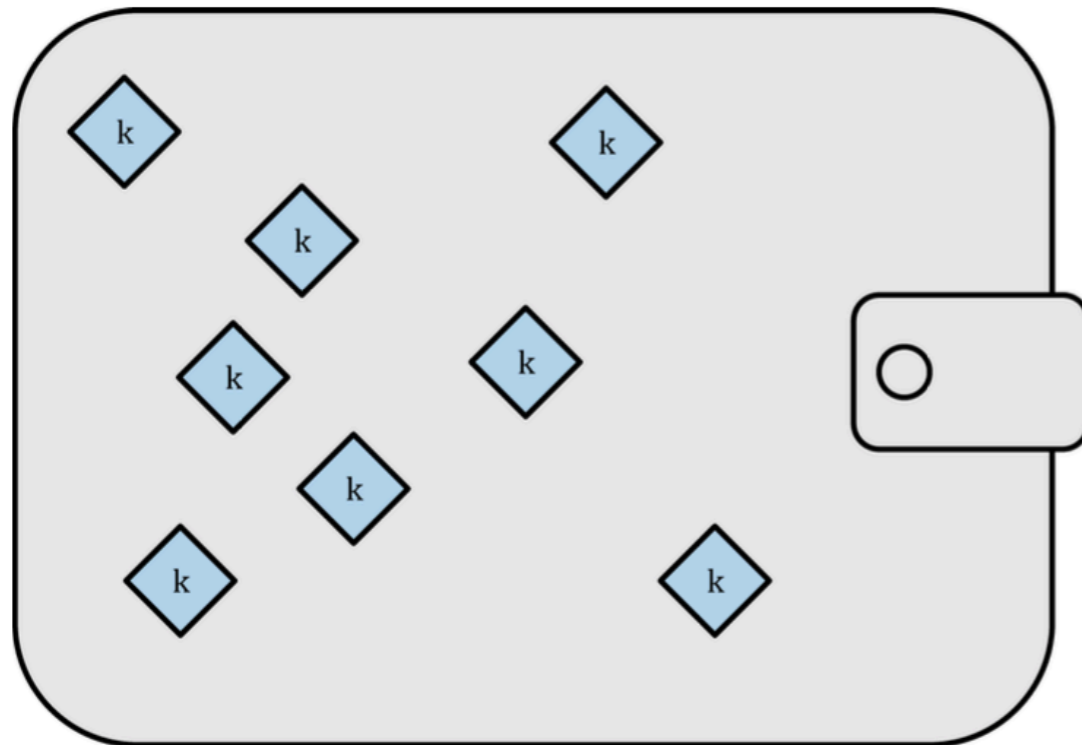
- ◆ Should avoid reusing addresses
- ◆ One address involves in only two transactions
 - Receive coins from another address
 - Send coins to another address
- ◆ Change receiver's address right after receiving coins from someone
- ◆ → Split total balance into small amounts contained by different addresses



How to manage many addresses and keys?

Wallet Types

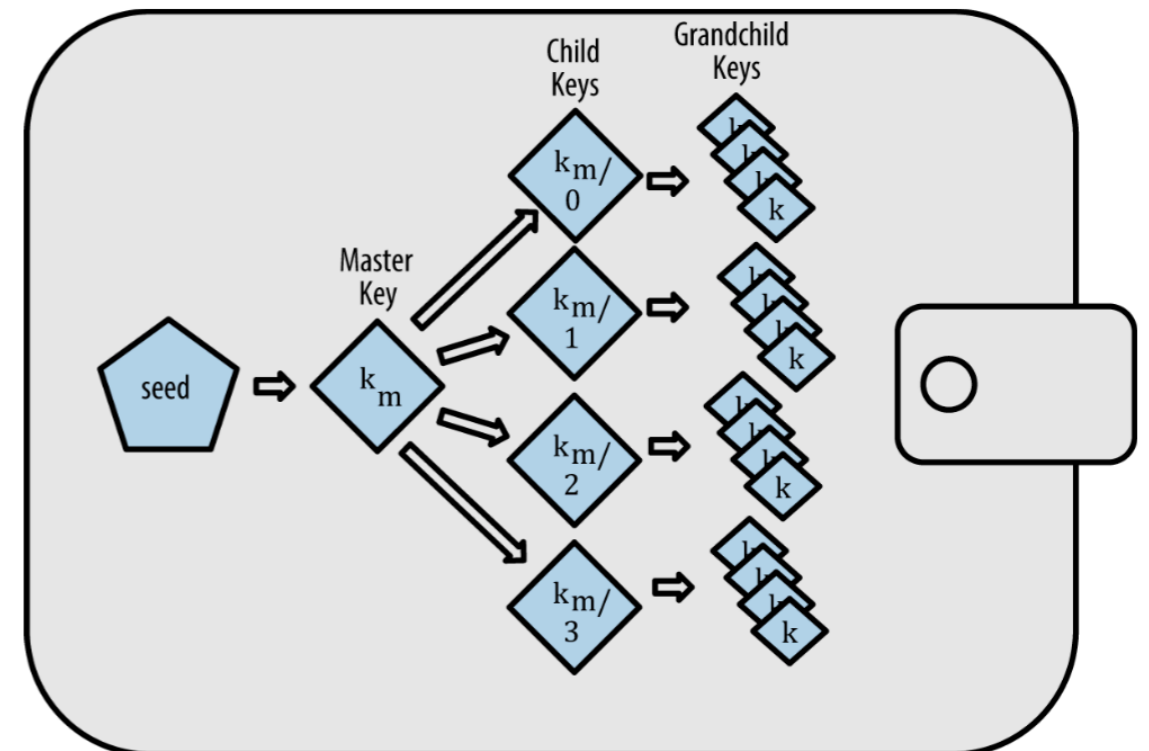
Non-deterministic



- ◆ Independent generation
- ◆ No relation

→ Bad choice

Deterministic

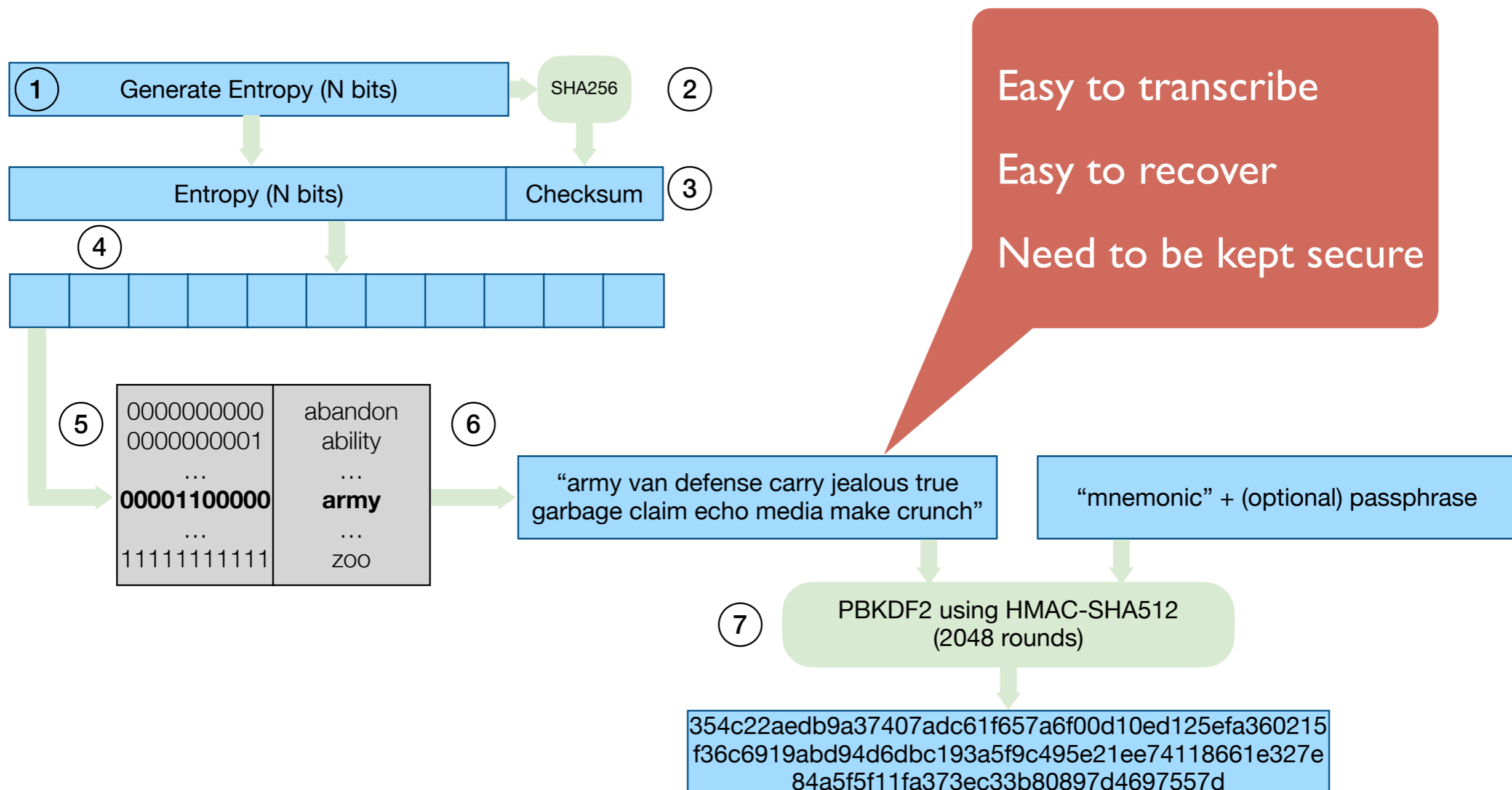


- ◆ Tree-like structure
- ◆ Keep secret only the seed

→ Good choice

Mnemonic Code Words

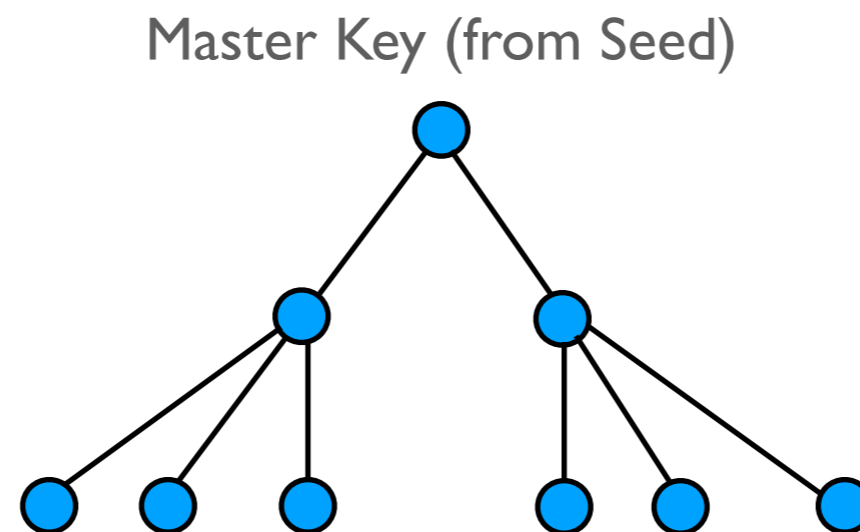
- ◆ BIP-39: Mnemonic code for generating deterministic keys*



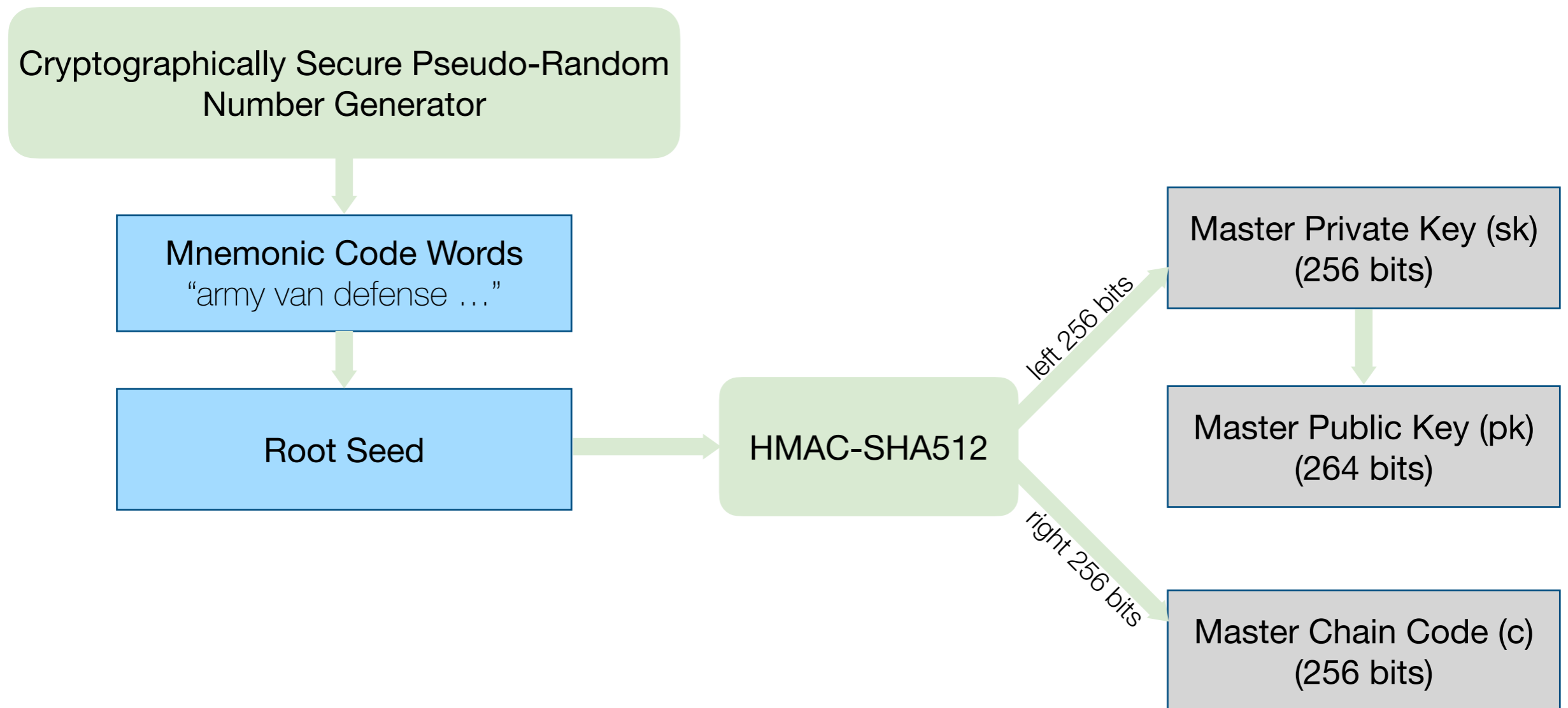
(* Source: <https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki>

HD wallet from the Seed

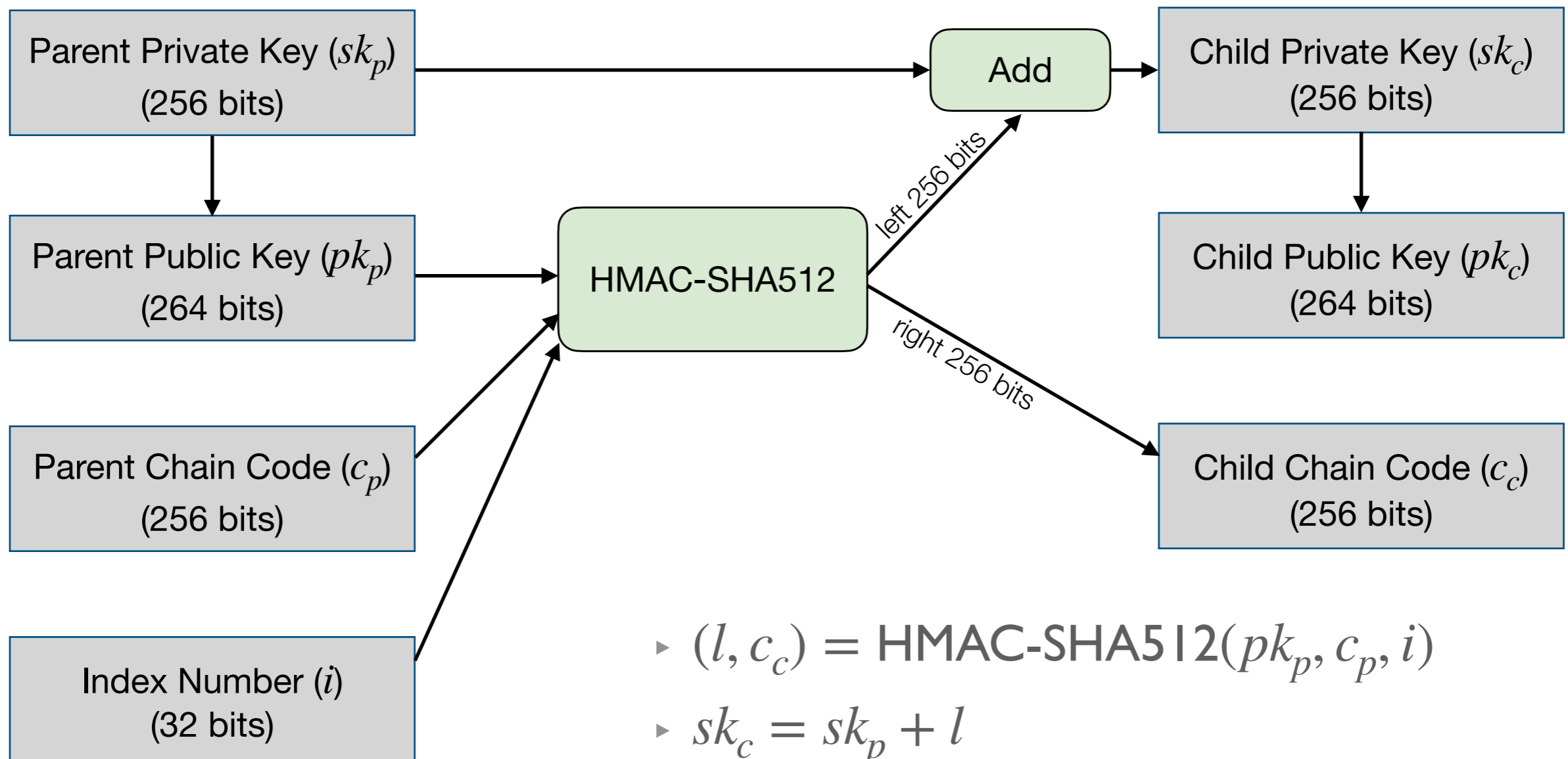
- ◆ BIP-32: Hierarchical Deterministic Wallets*
- ◆ A **tree-like structure** of keys:
 - Generate Master Key from Seed
 - Generate a child private key from a parent private key
 - Generate **a child public key from a parent public key** (without the need of the private key)



Generate Master Key from Seed

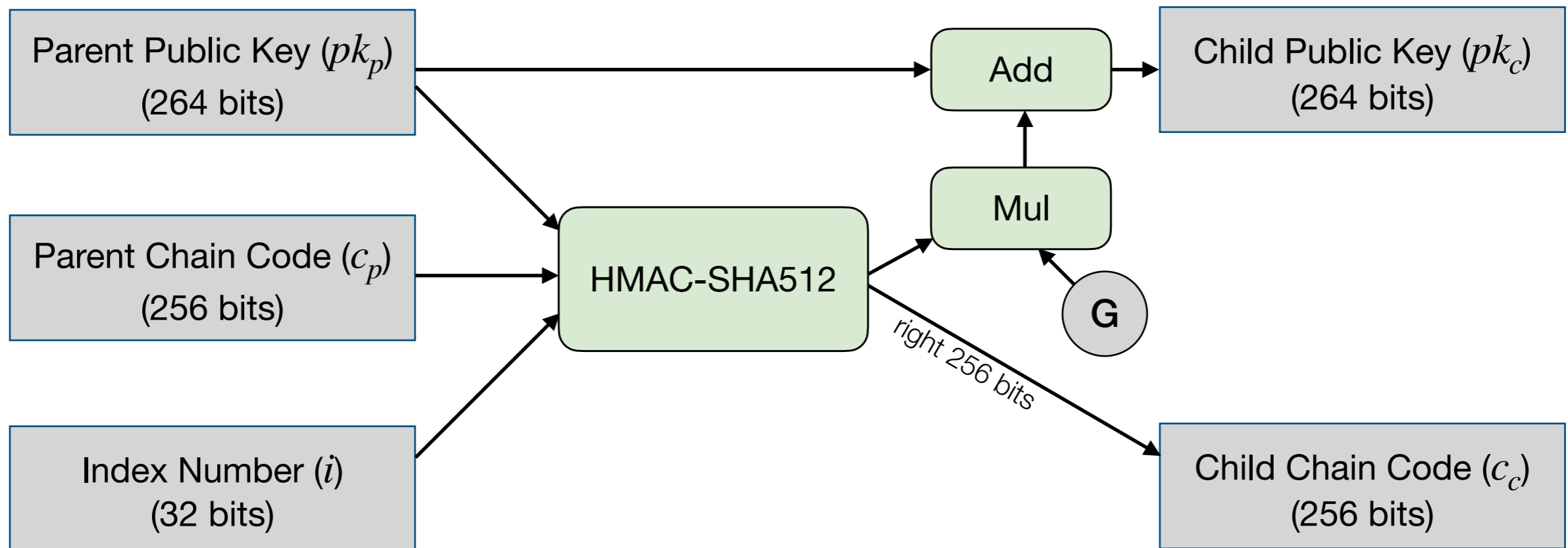


Child *private* key from parent *private* key



- ▶ $(l, c_c) = \text{HMAC-SHA512}(pk_p, c_p, i)$
- ▶ $sk_c = sk_p + l$
- ▶ $pk_c = sk_c \times G = (sk_p + l) \times G$
- ▶ $xprv = (sk \parallel c)$: enough to generate

Child *public* key from parent *public* key

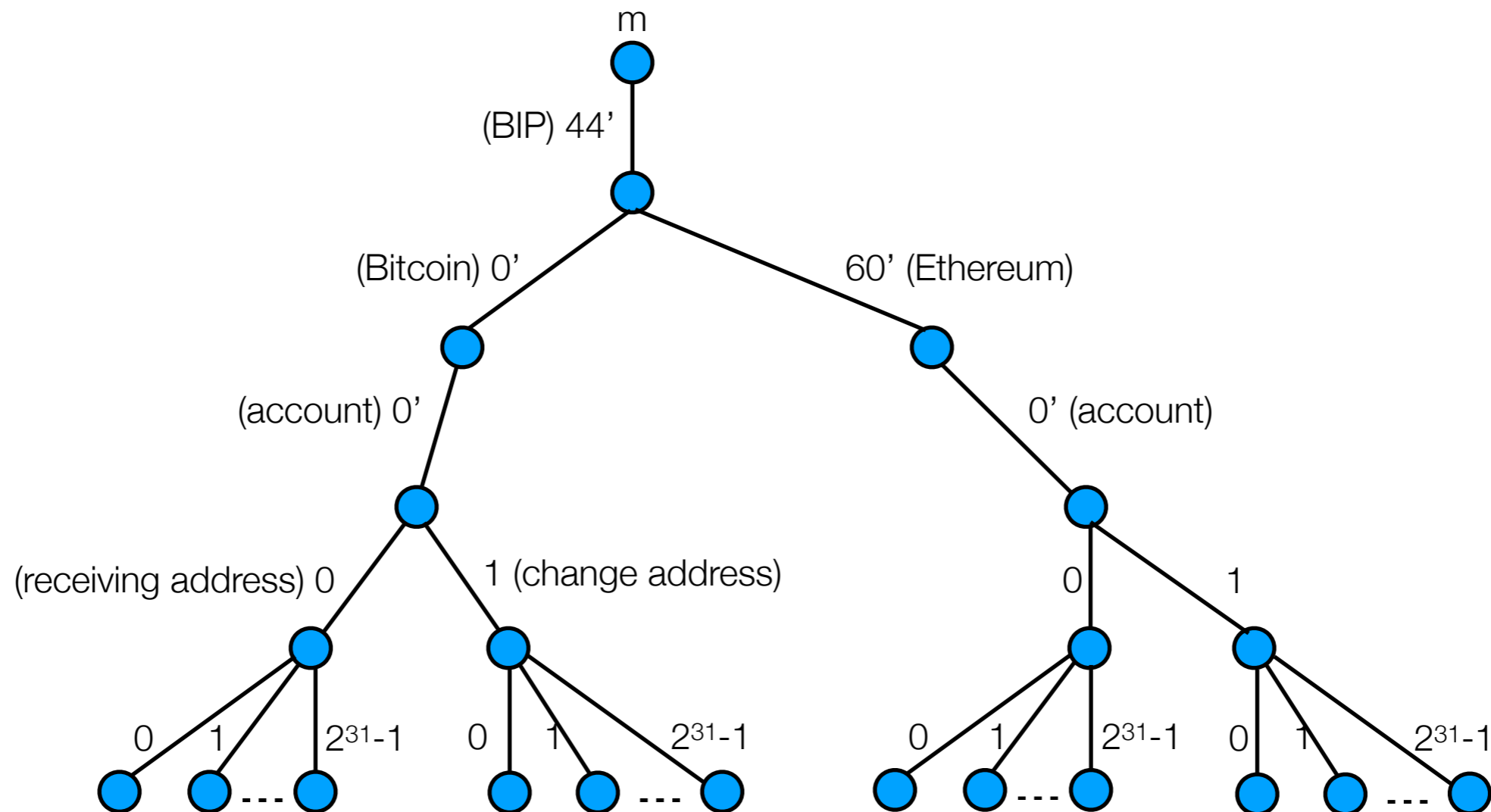


- ▶ $(l, c_c) = \text{HMAC-SHA512}(pk_p, c_p, i)$
- ▶ Previous calculation: $pk_c = sk_c \times G = (sk_p + l) \times G$
- ▶ Now: $pk_c = pk_p + l \times G = sk_p \times G + l \times G = (sk_p + l) \times G$
- ▶ $x_{pub} = (pk \parallel c)$: enough to generate

Public key is generated without the need of private key

Key Structure Specification

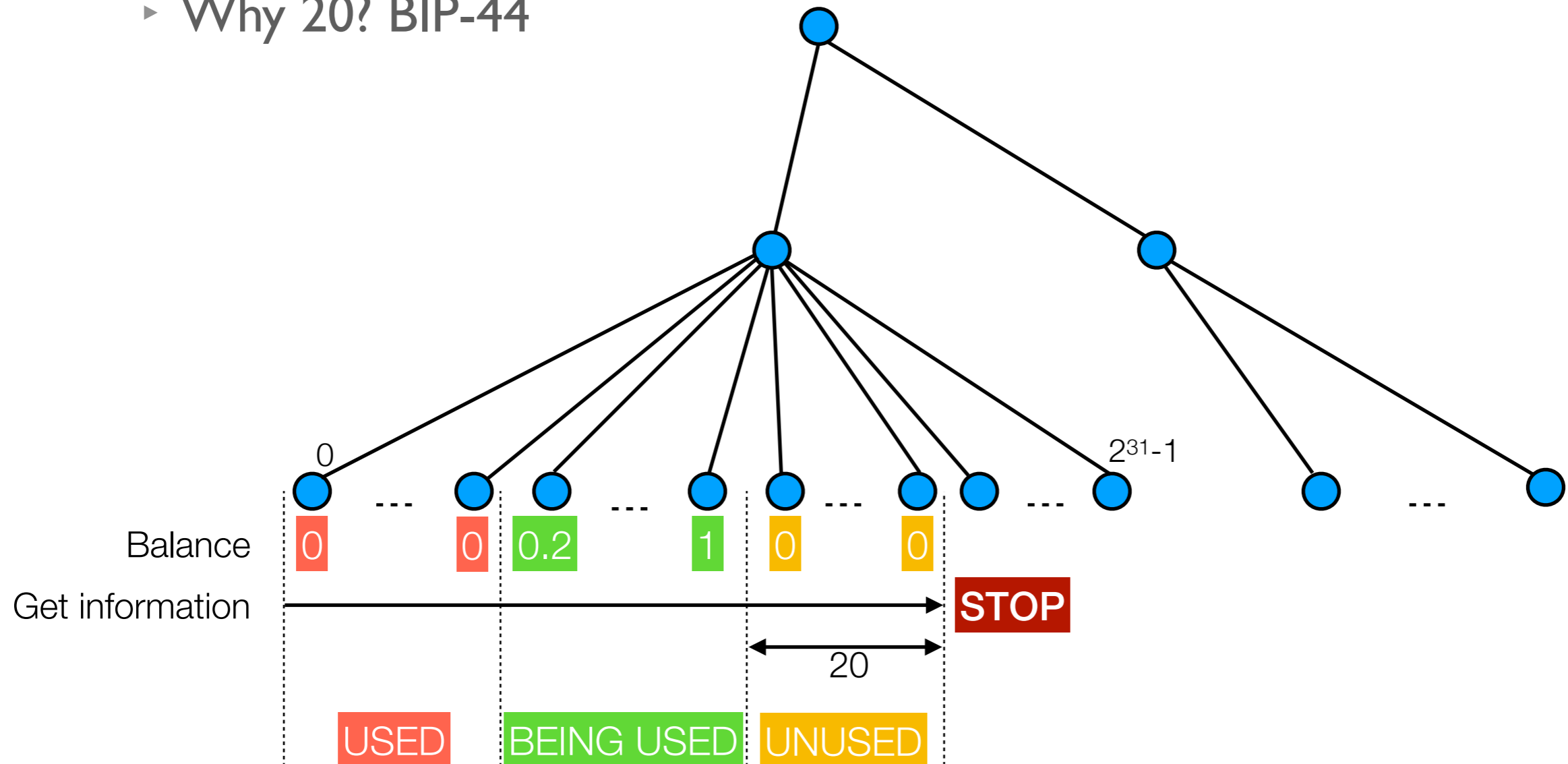
- ◆ BIP-44: Multi-Account Hierarchy for Deterministic Wallets*
 - `m / purpose' / coin_type' / account' / change / address_index`
 - Example: `m / 44' / 0' / 0' / 0 / 1`



(*) Source: <https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki>

How to get balance?

- ◆ Get information on Bitcoin network for each address
- ◆ When to stop?
 - 20 consecutive *fresh addresses* (no transaction)
 - Why 20? BIP-44

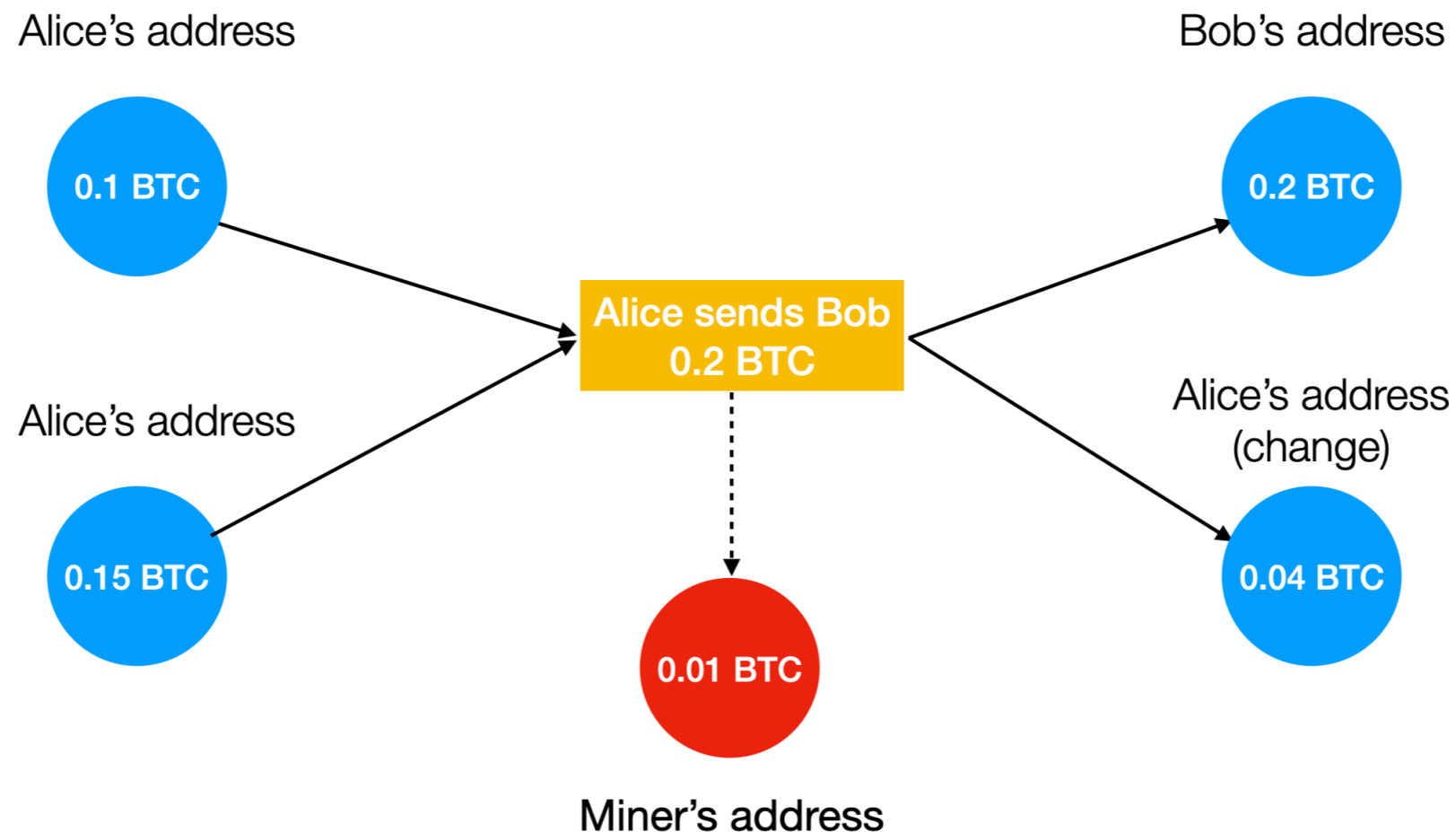


Outlines

1. Introduction
2. Keys and addresses for cryptocurrencies
3. Transactions
 - ◆ Transaction components (Bitcoin)
 - ◆ How to create a transaction?
4. Secure wallet architecture
5. Summary

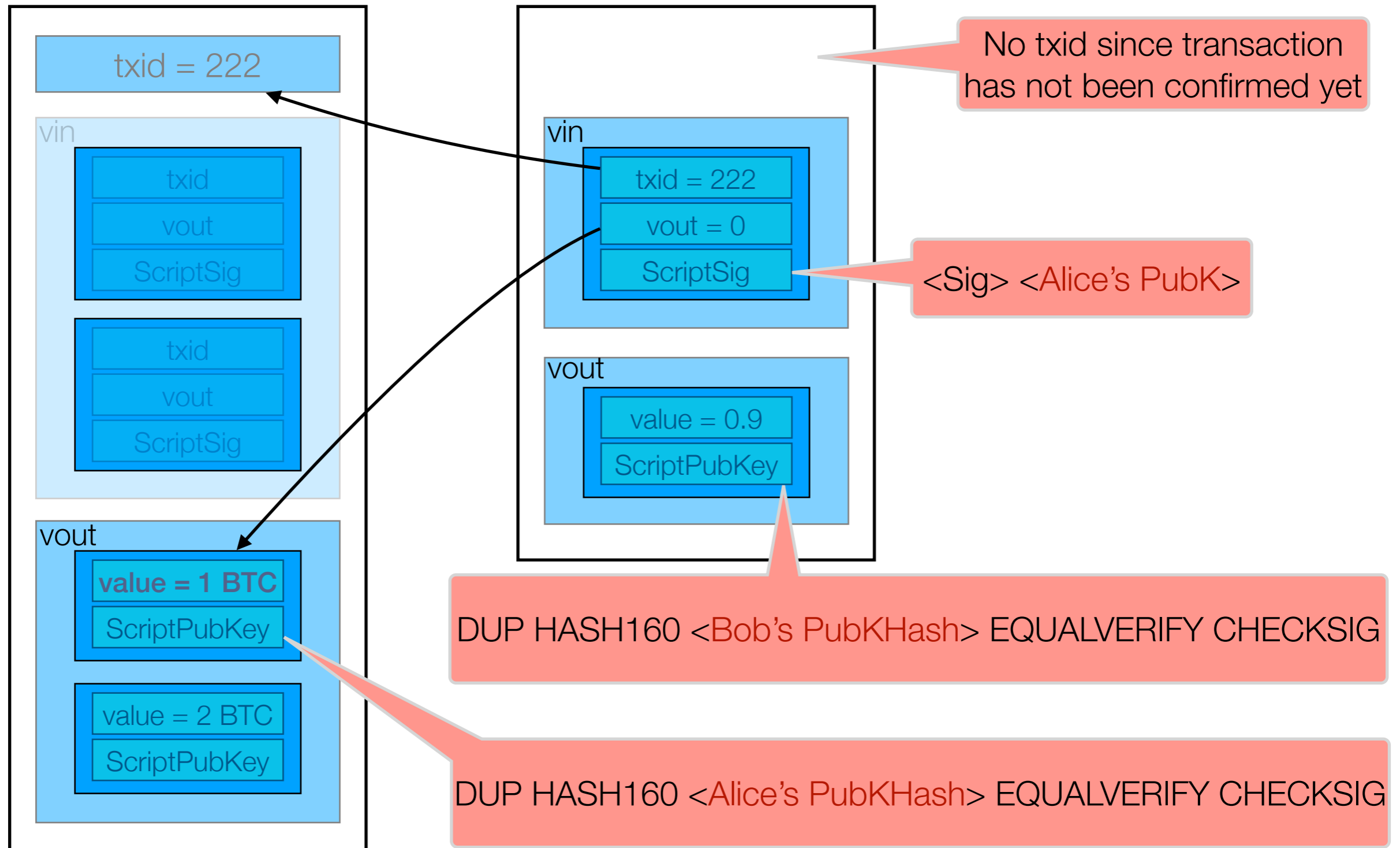
Transaction Components

- ◆ A transaction can include
 - One or many addresses as inputs
 - One or many addresses as outputs
- ◆ The change is not automatically sent back to the sender
- ◆ Transaction fee
 - $\text{Fee} = \text{Sum}(\text{inputs}) - \text{Sum}(\text{outputs})$



Create a new transaction

- ◆ Example: Alice has 1 BTC and wants to send Bob 0.9 BTC

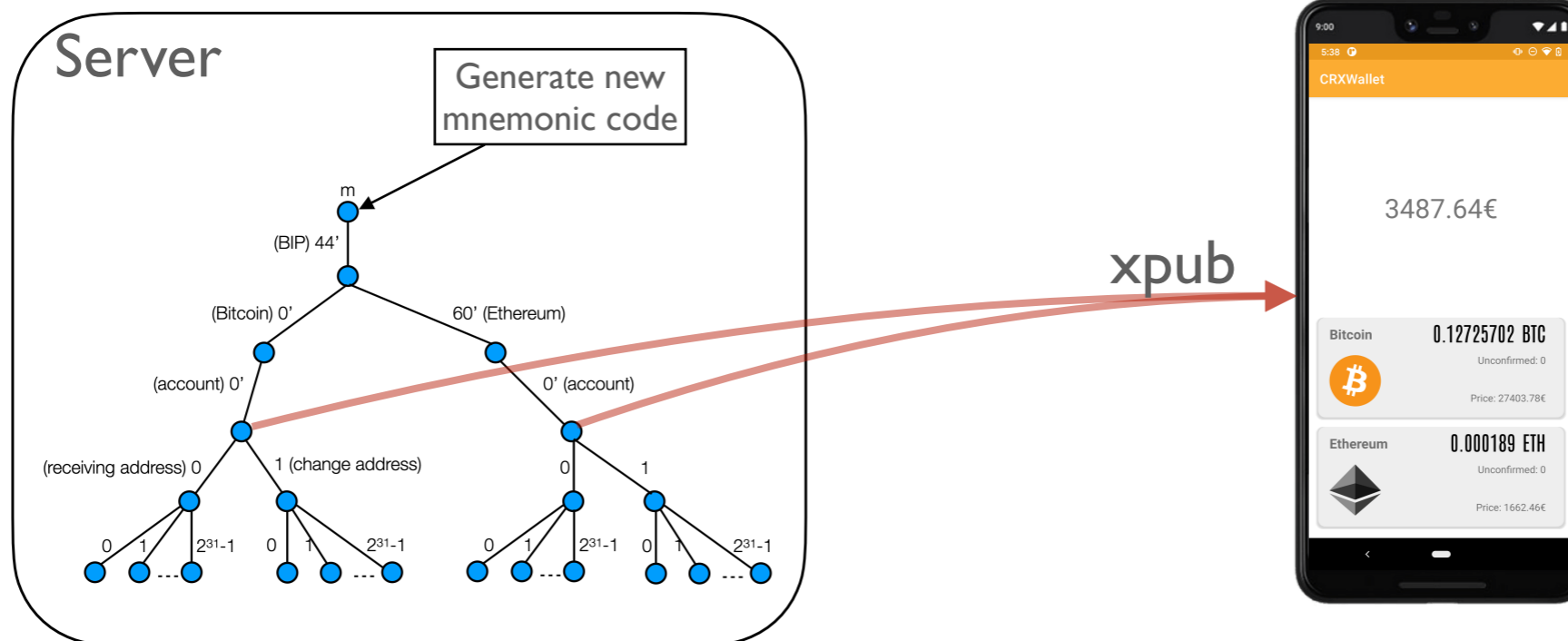


Outlines

1. Introduction
2. Keys and addresses for cryptocurrencies
3. Transactions
4. Secure wallet architecture
 - ◆ Account on wallet
 - ◆ Token generator and usage
5. Summary

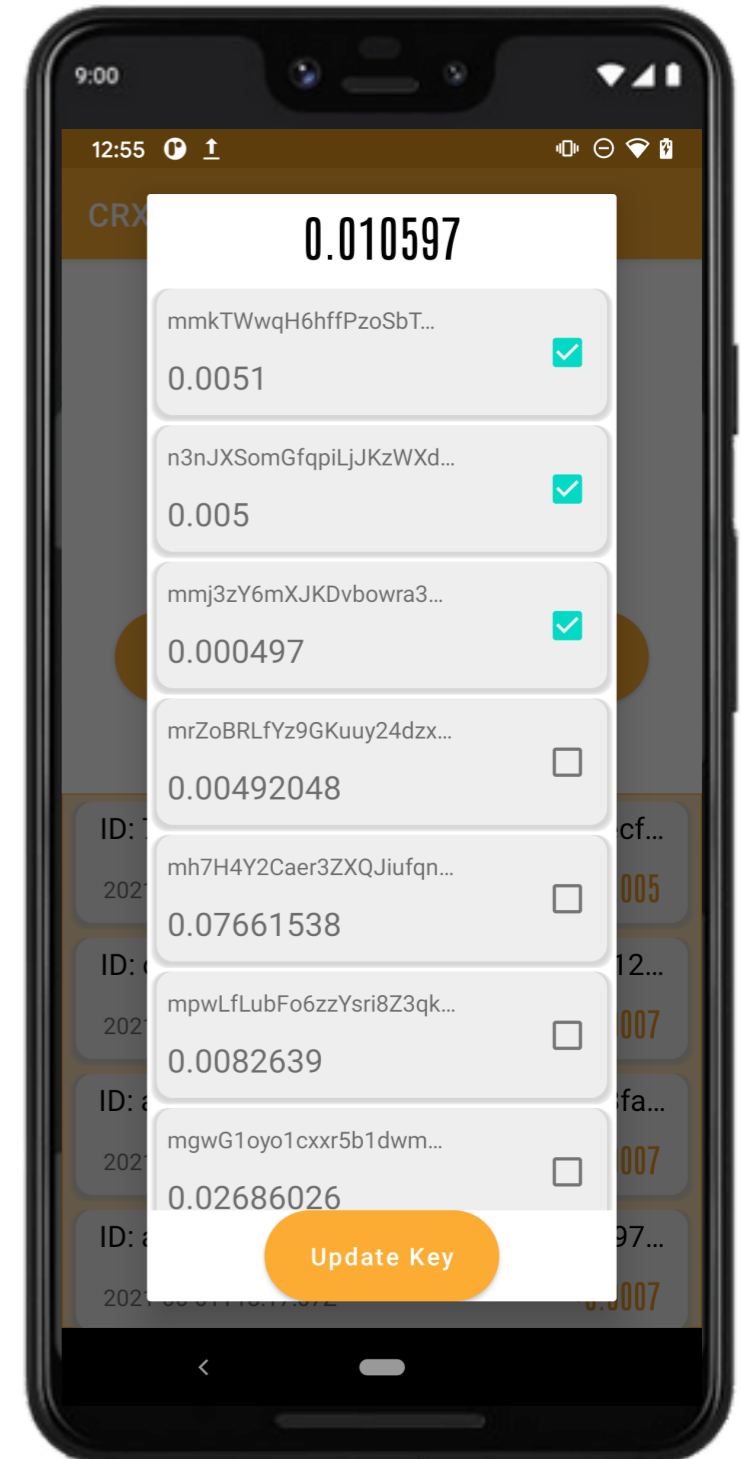
Create account on wallet

- ◆ Wallet app only stores xpub (of account node)
- ◆ From xpub, it can generate addresses and public keys



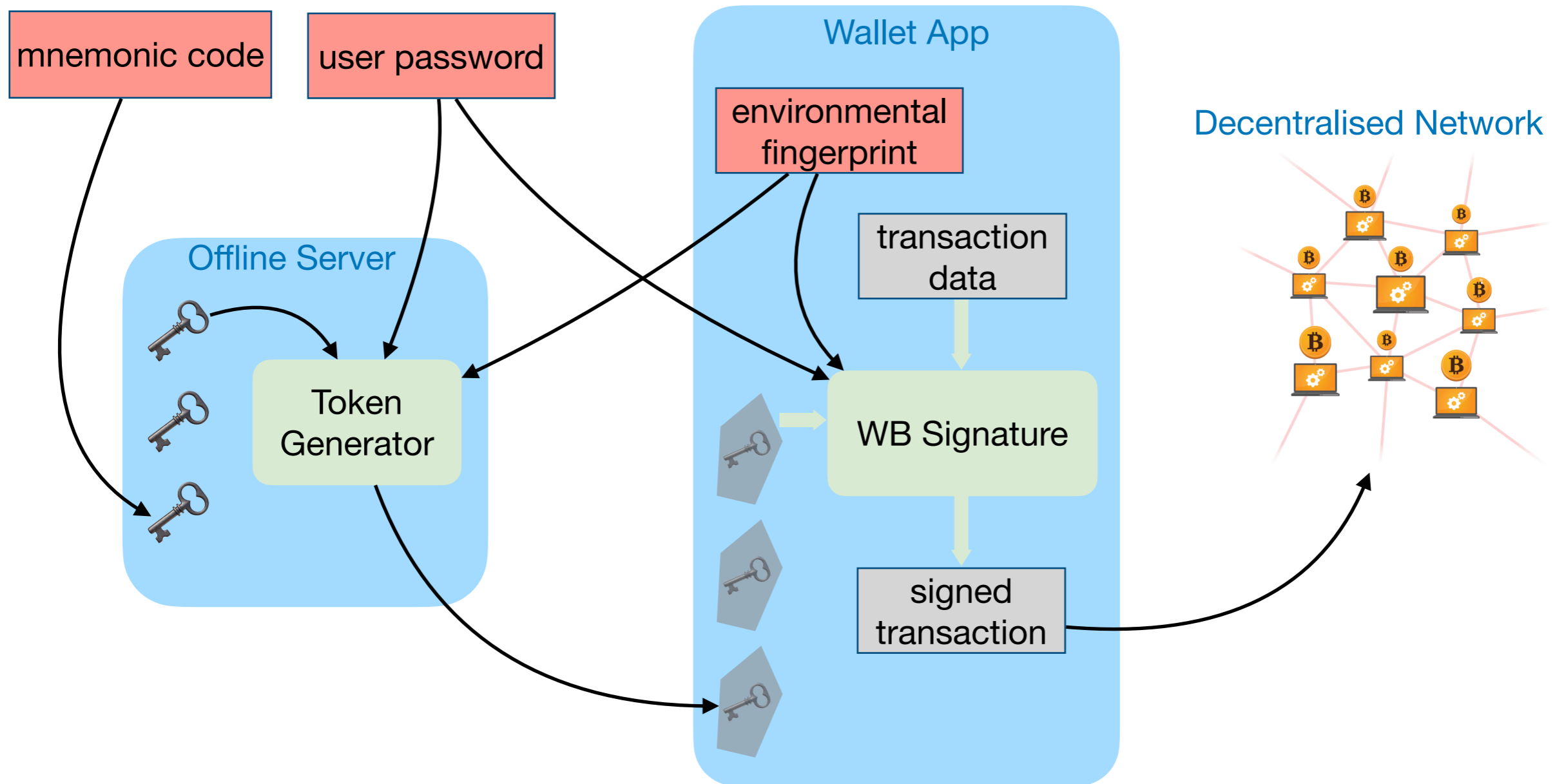
Spendable Amount

- ◆ Private keys are not stored in the app
- ◆ What is spendable amount?
 - Sum of positive balance of addresses
 - Tokens (private keys) are available in the app
- ◆ Increase spendable amount
 - Connect to server (by cable)
 - Update tokens



Overview of Architecture

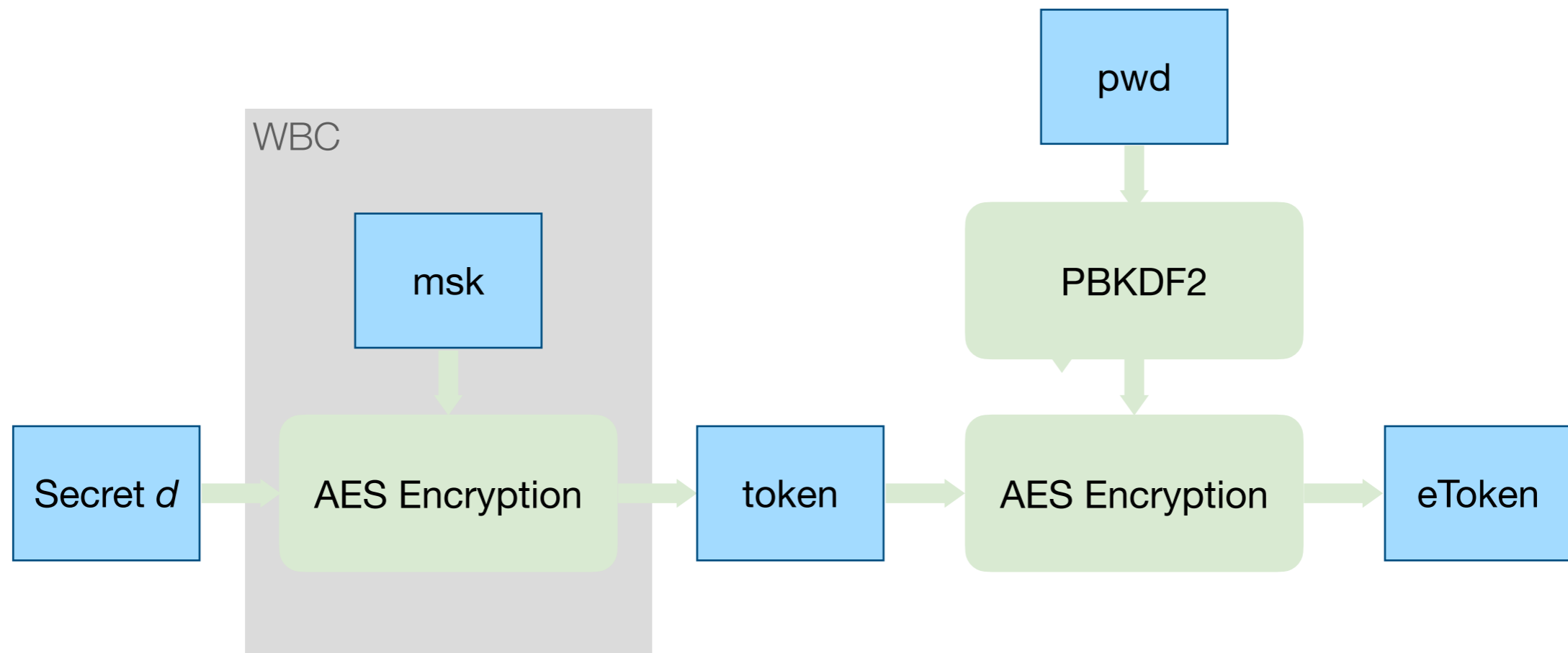
- ◆ A token is a secure container for a key
 - generated by a trusted server
 - operated by a white-box signature generator



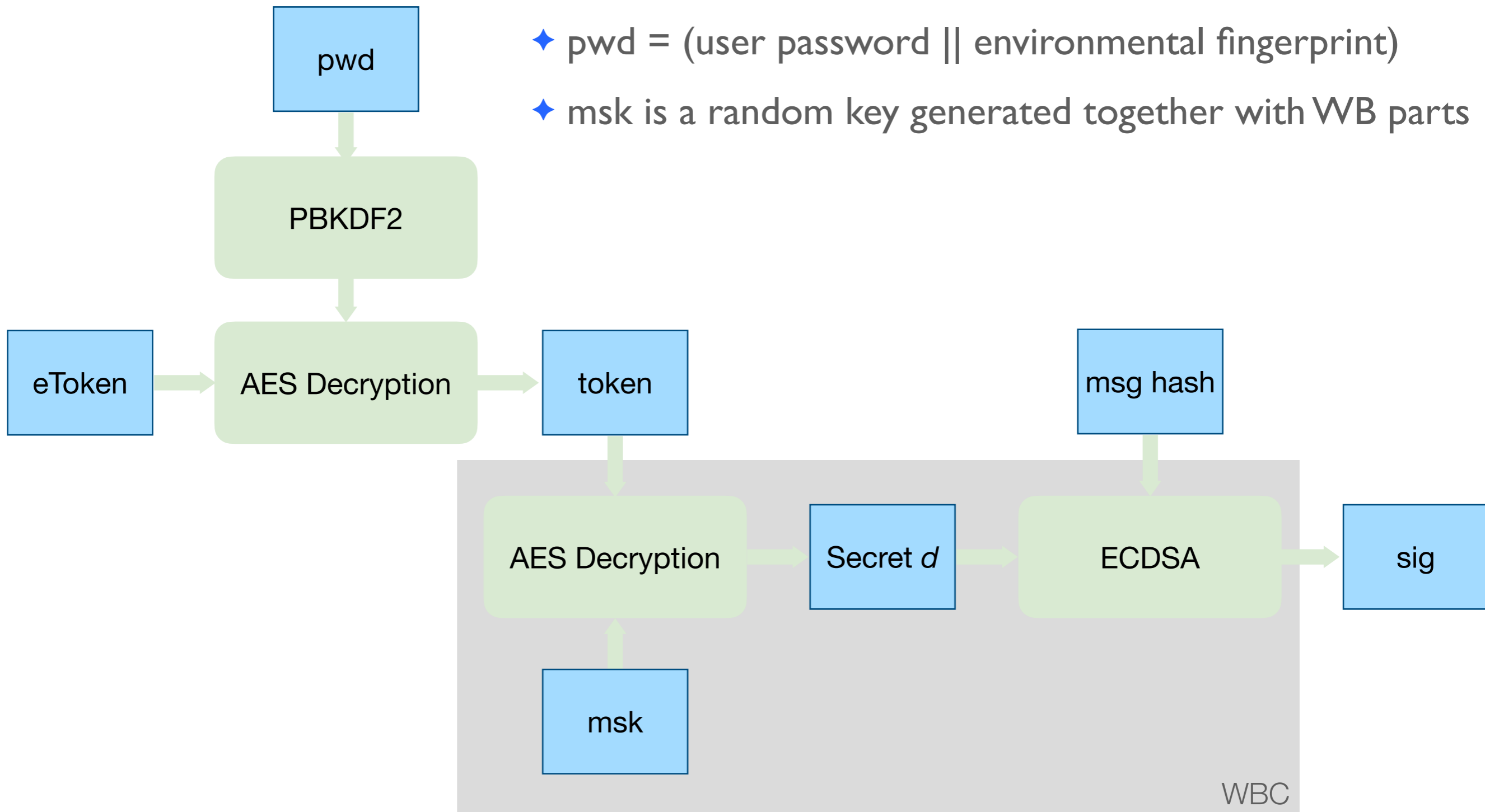
→ Why secure?

Server: token generator

- ◆ $\text{pwd} = (\text{user password} \parallel \text{environmental fingerprint})$
- ◆ msk is a random key generated together with WB parts



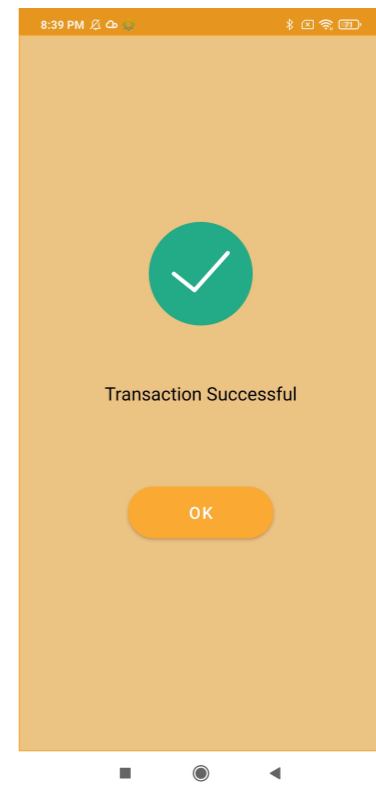
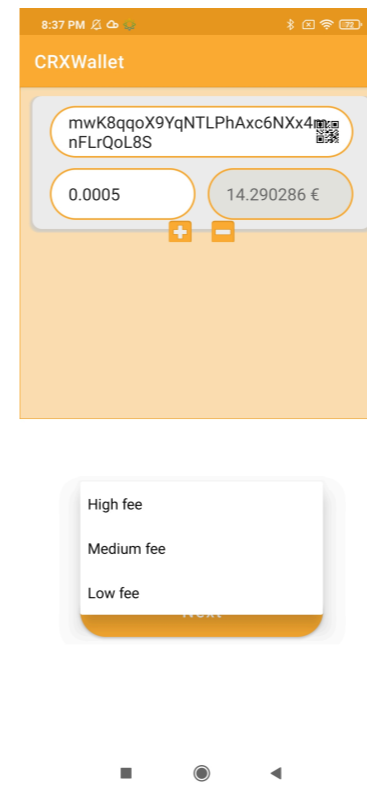
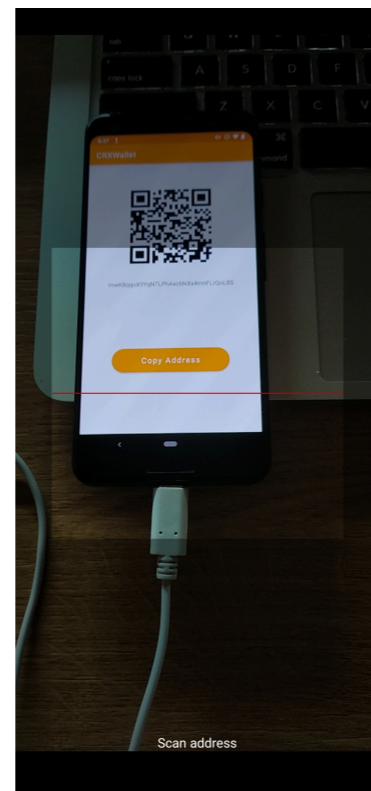
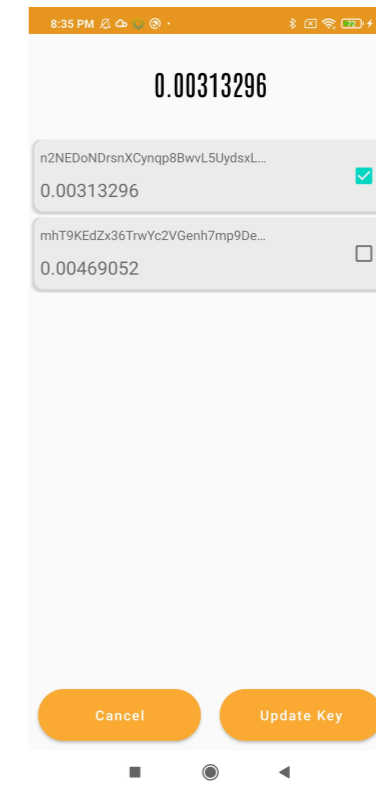
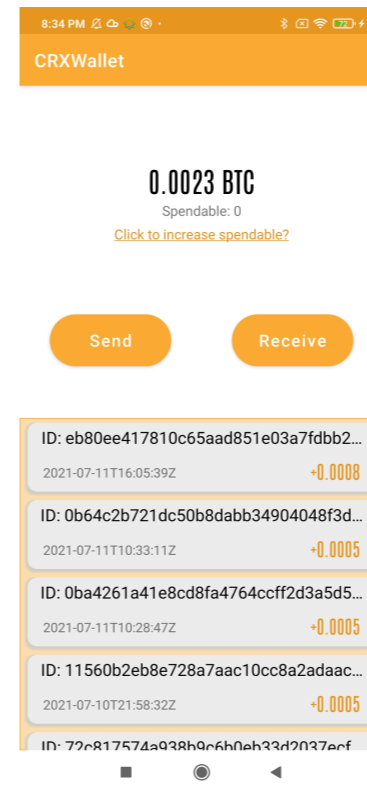
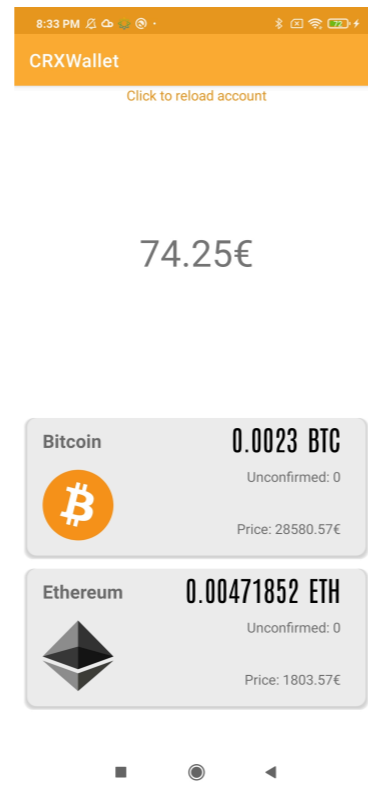
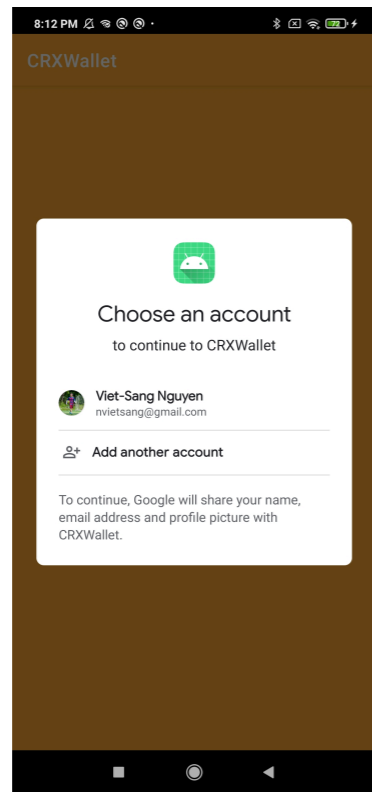
Wallet app: signature generator



Outlines

1. Introduction
2. Keys and addresses for cryptocurrencies
3. Transactions
4. Secure wallet architecture
5. Summary

Screenshots



Demo available at: https://youtu.be/Y9EIZL_G5A8

Summary

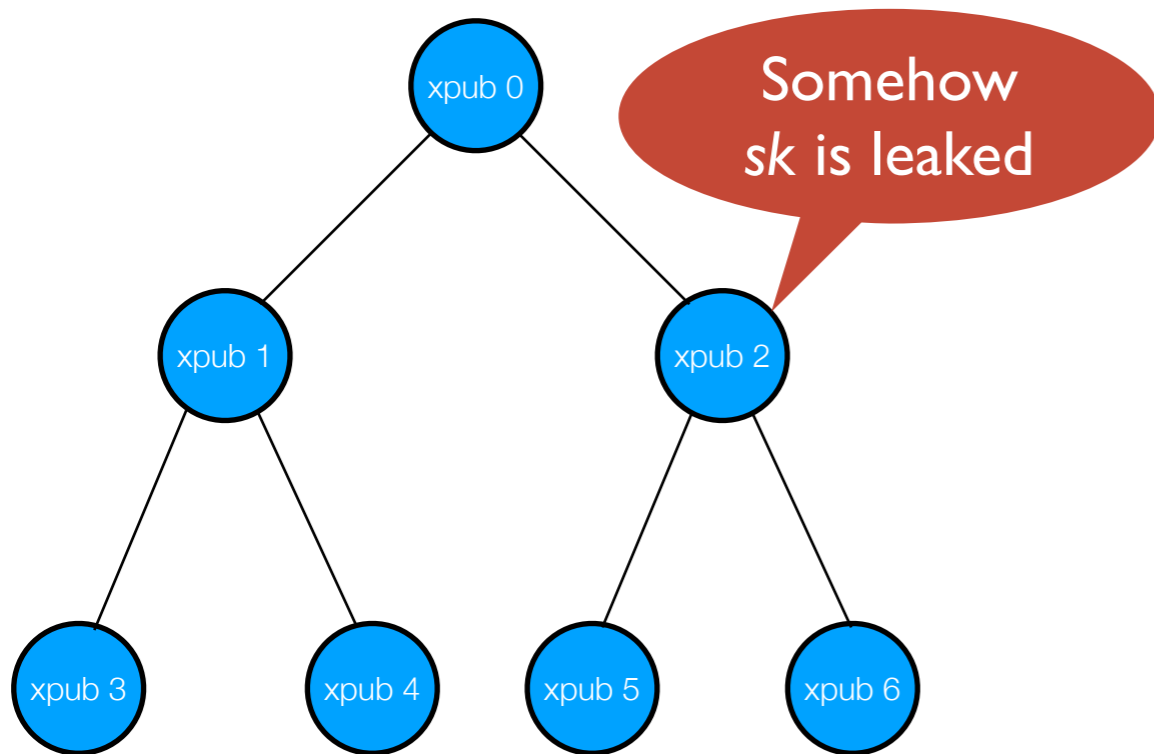
- ◆ Generation and management of keys in a wallet
 - Mnemonic code
 - Tree structure of keys
- ◆ Creation of a new Bitcoin (and Ethereum) transaction
- ◆ Architecture of a secure wallet application
 - Token generation
 - Token usage with white-box cryptography
- ◆ Survey attacks and countermeasures on ECDSA (not presented here)
- ◆ White-box ECDSA is still a challenge

Thank you

Any question?

Appendix

A possible risk

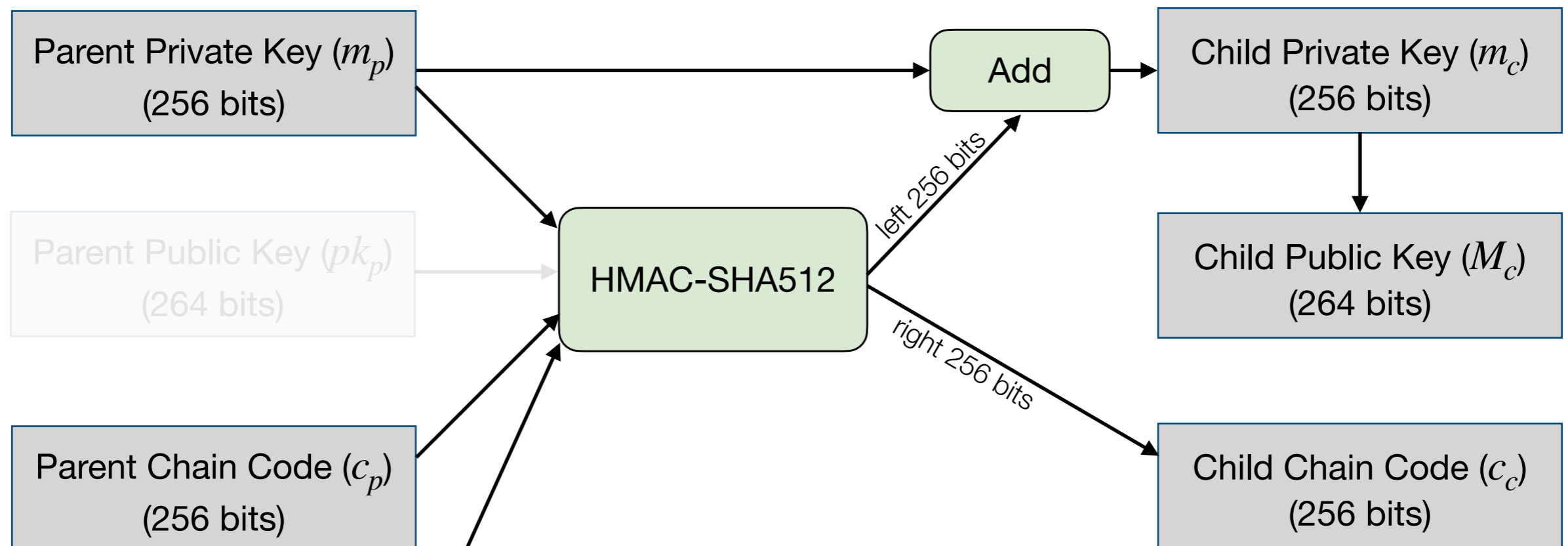


$xprv = (sk \parallel c)$
 $xpub = (pk \parallel c)$
Same chain code in xpub and xprv

- ◆ Private keys of its children are revealed (xpub 5, 6)
- ◆ Private key of xpub 0 can be deduced
 - $(l, c_2) = \text{HMAC-SHA512}(xpub_0, i)$
 - $sk_0 = sk_2 - l$
- ◆ → Harden child key derivation

Harden child key derivation

- ◆ Break the relationship between parent public key and child chain code
- ◆ Use parent private key to derive child chain code, instead of the parent public key
- ◆ Cannot generate child public key without the need of private key anymore

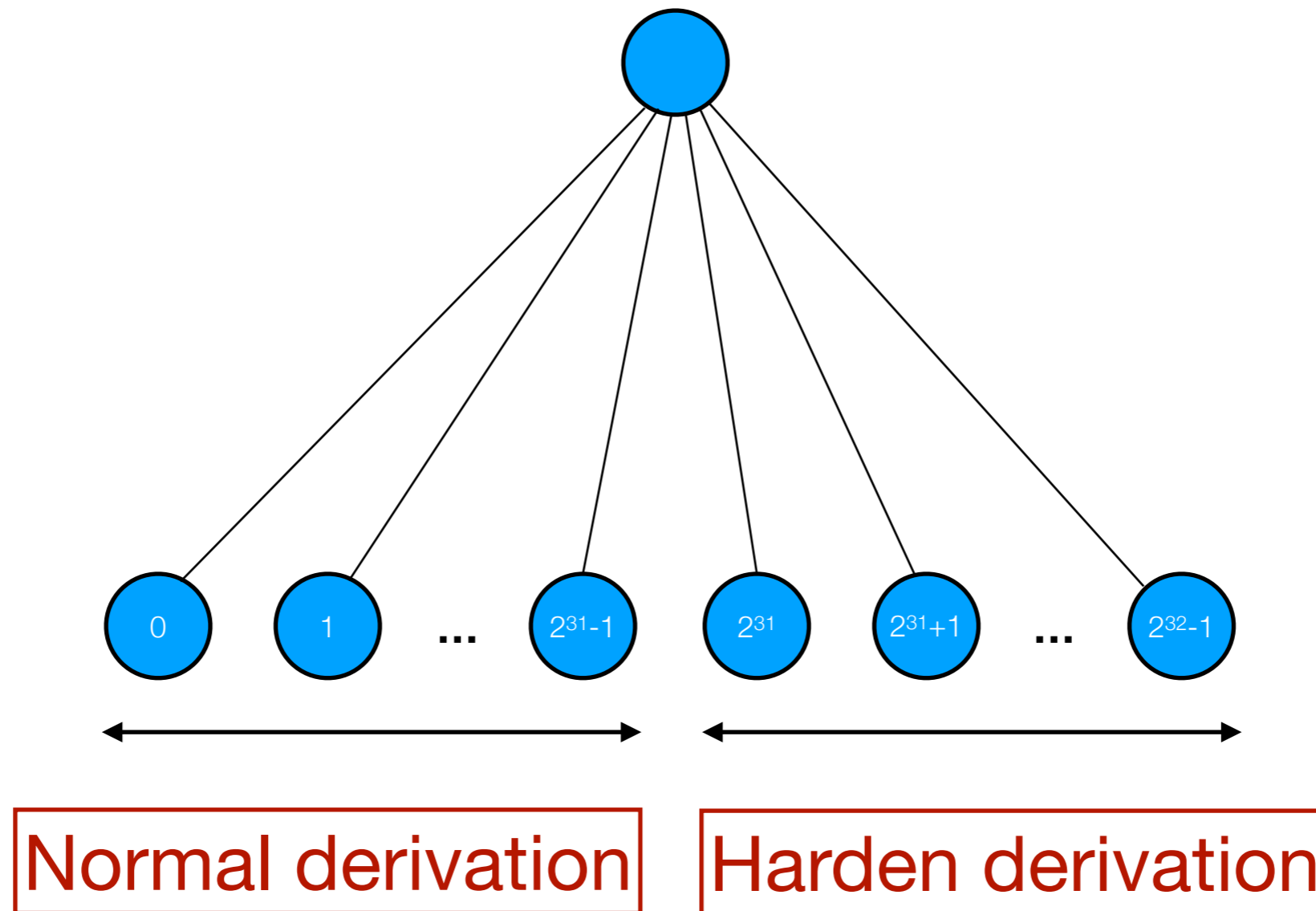


$$\triangleright (l, c_c) = \text{HMAC-SHA512}(sk_p, c_p, i)$$

$$\triangleright sk_c = sk_p + l$$

$$\triangleright pk_c = sk_c \times G = (sk_p + l) \times G$$

Index Number



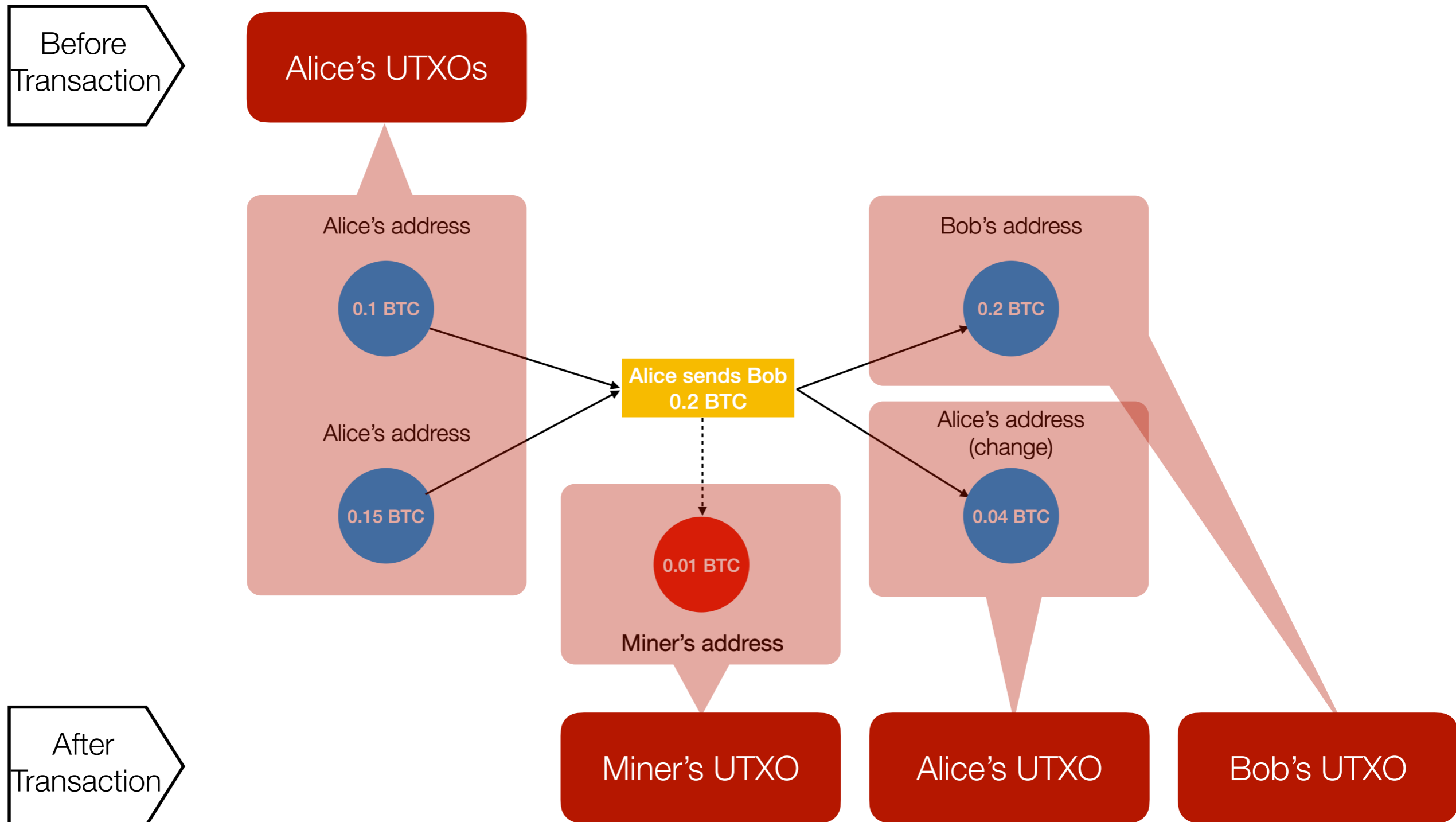
- ◆ Use prime symbol to denote index for a harden child
 - $i' = 2^{31} + i$
 - Example: $2' = 2^{31} + 2$

Transaction Fee

- ◆ $\text{Fee} = \text{Sum}(\text{inputs}) - \text{Sum}(\text{outputs})$
- ◆ Calculated based on the size of transaction
 - A block has a limited size (1 MB)
 - Miners want to include many transactions in a block
 - Large-size transaction (may) contains many inputs, which needs more efforts to refer to
- ◆ Use API to know suitable fee (satoshi/byte)

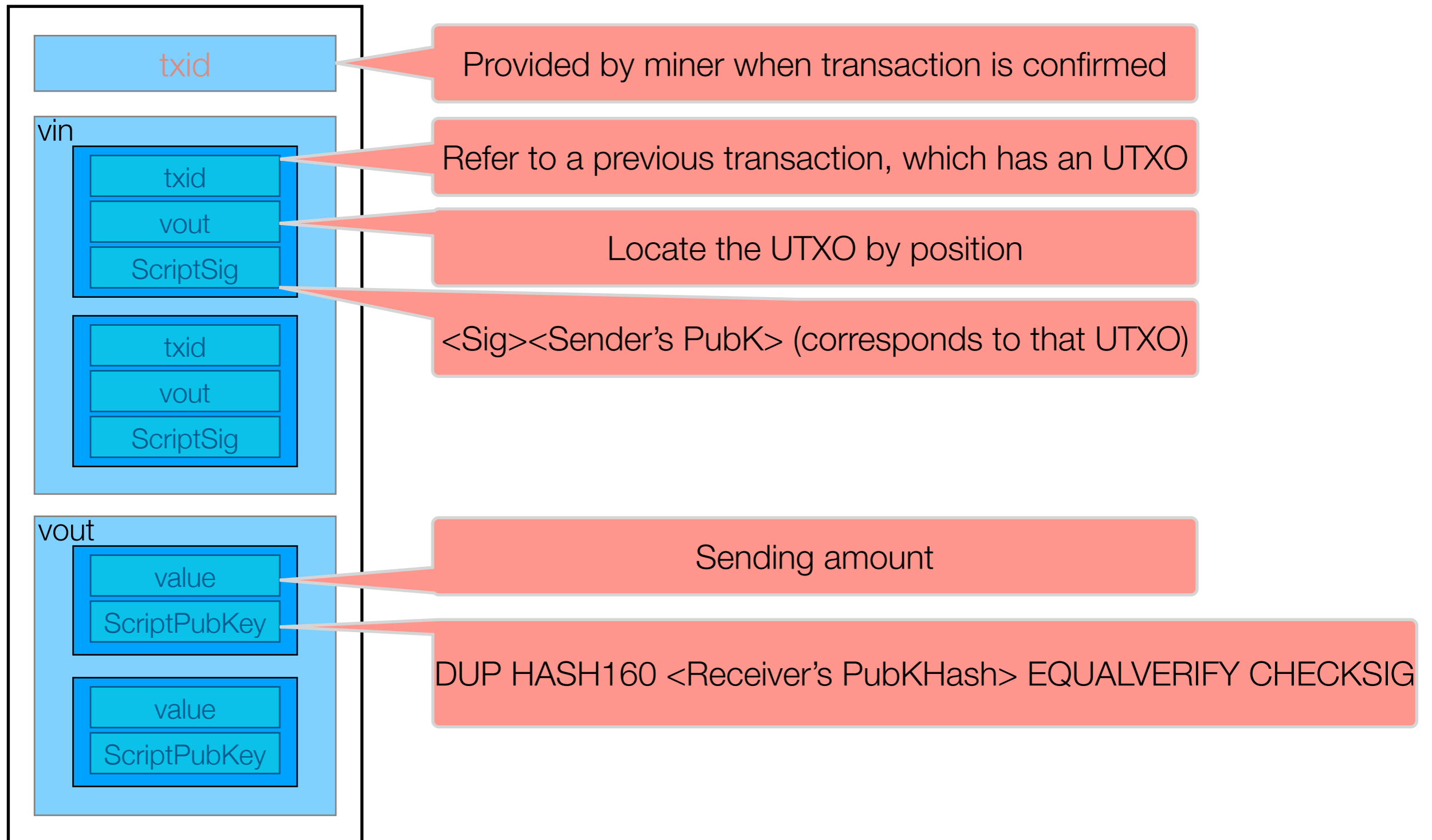
Unspent Transaction Output (UTXO)

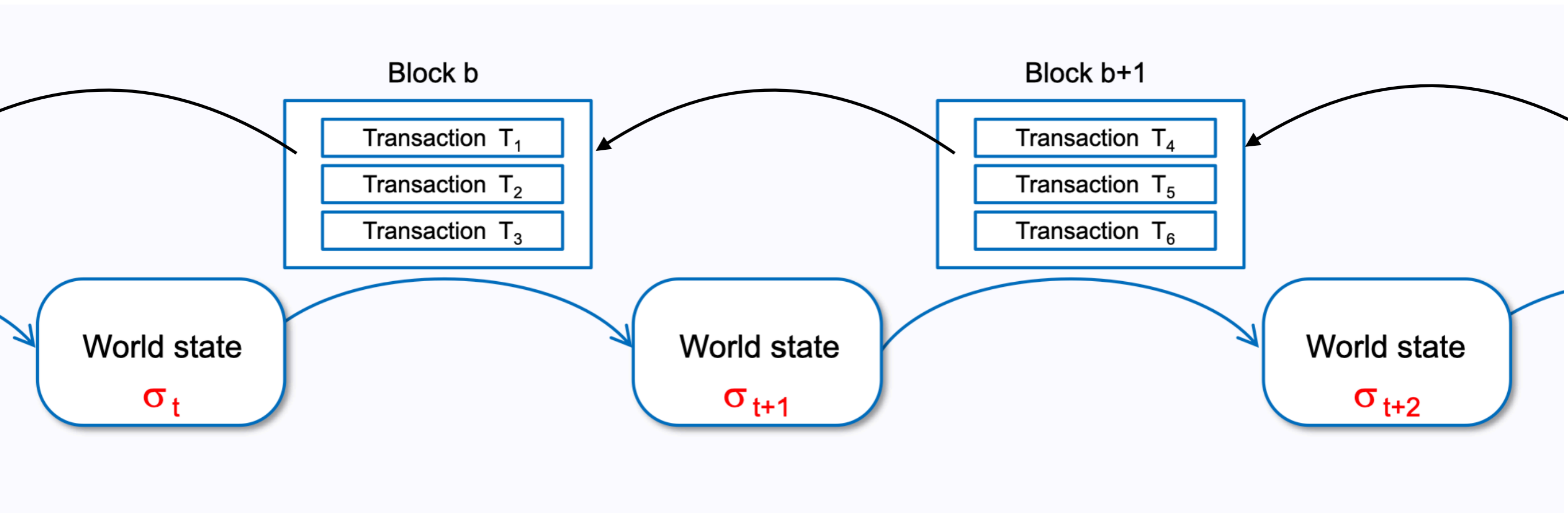
- ♦ UTXO refers to a transaction output that can be used as input in a new transaction

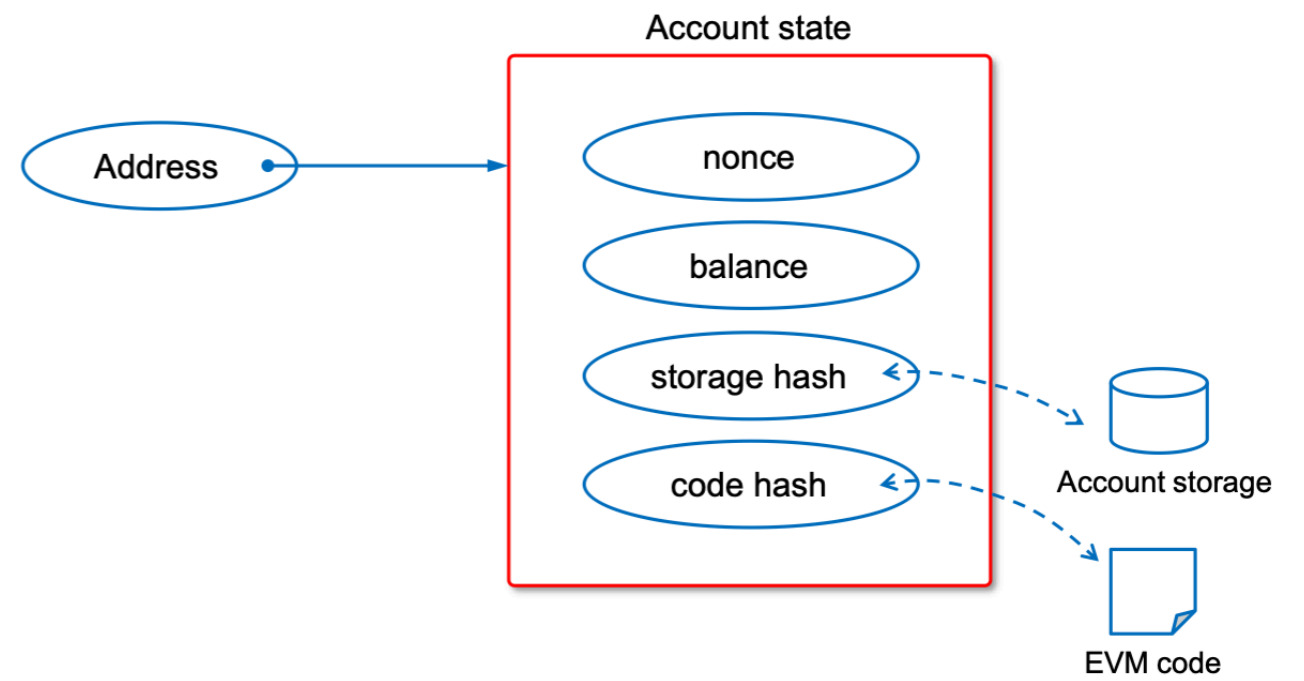
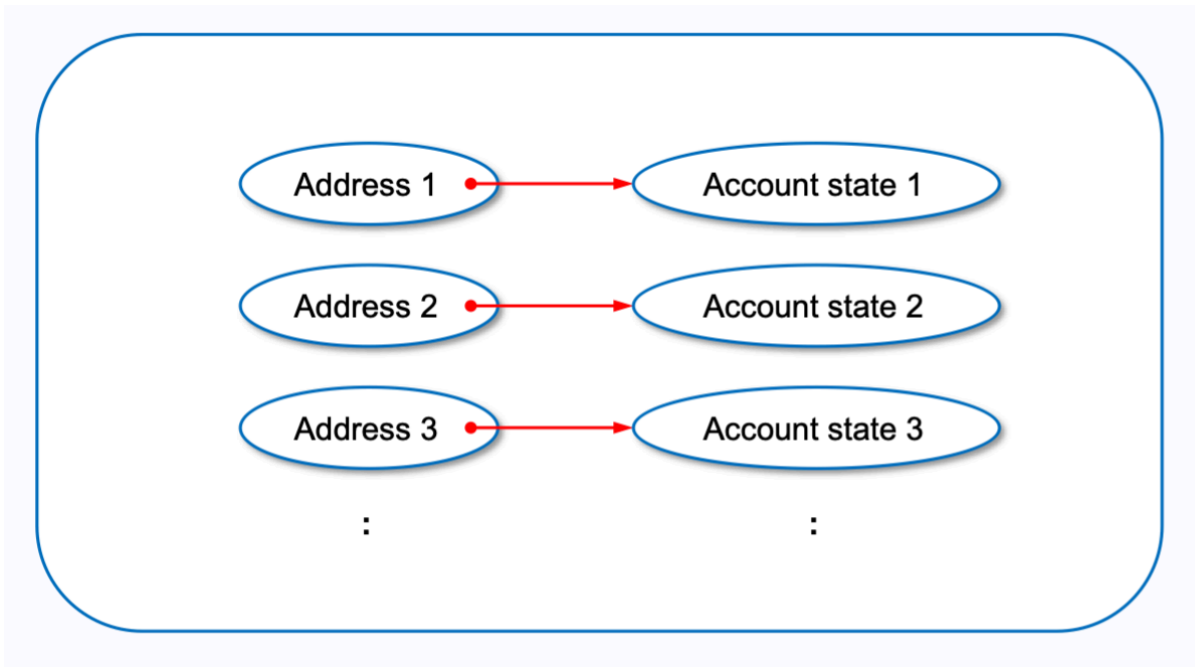


Transaction in detail

- ◆ Example: a transaction with 2 inputs and 2 outputs

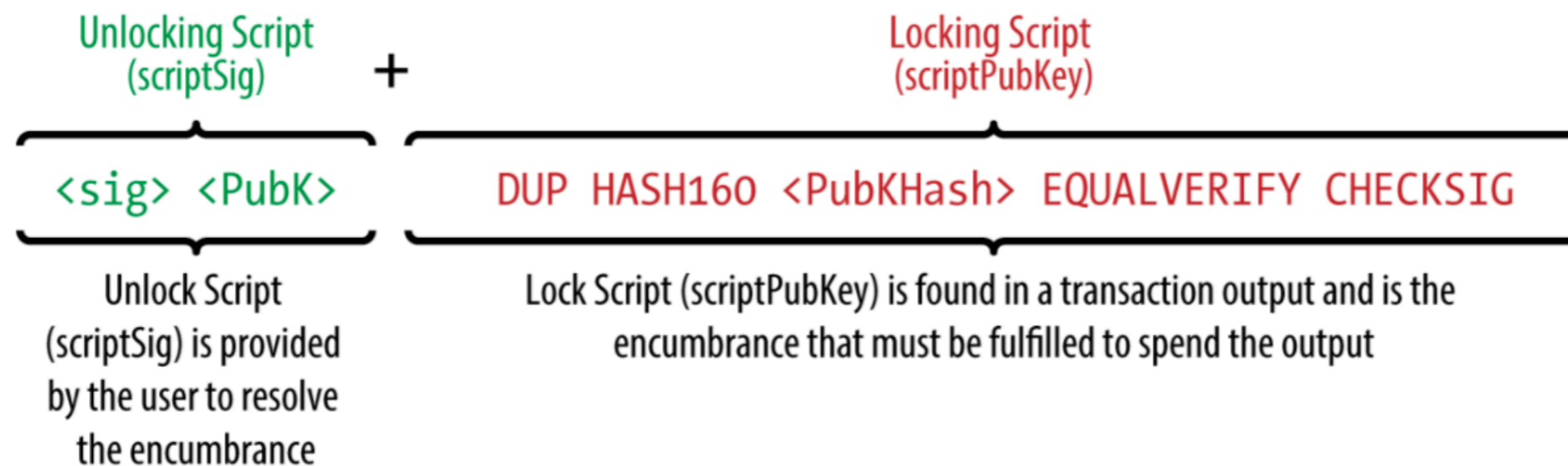






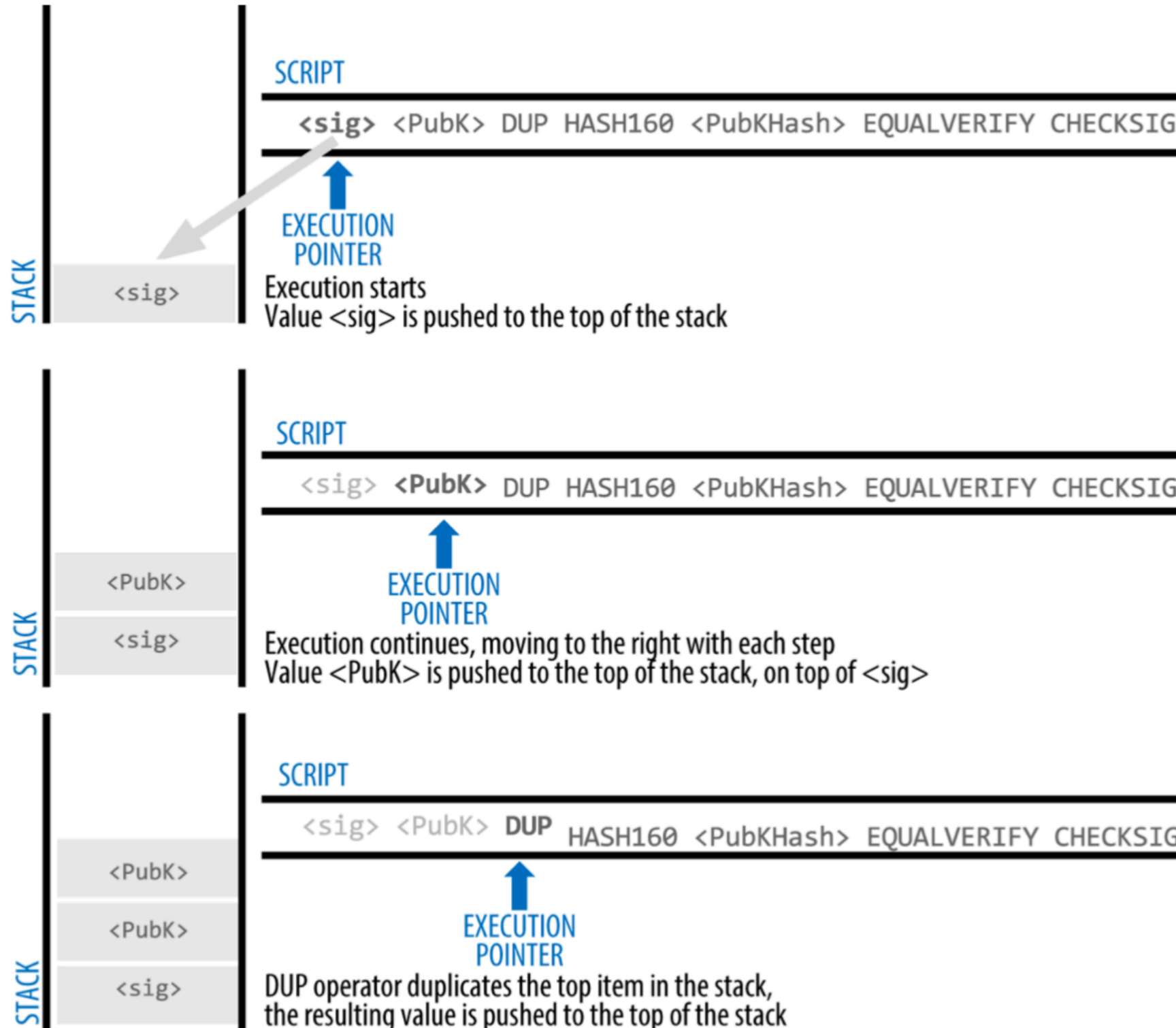
How to validate this transaction?

- ◆ Concatenate ScriptSig and ScriptPubKey

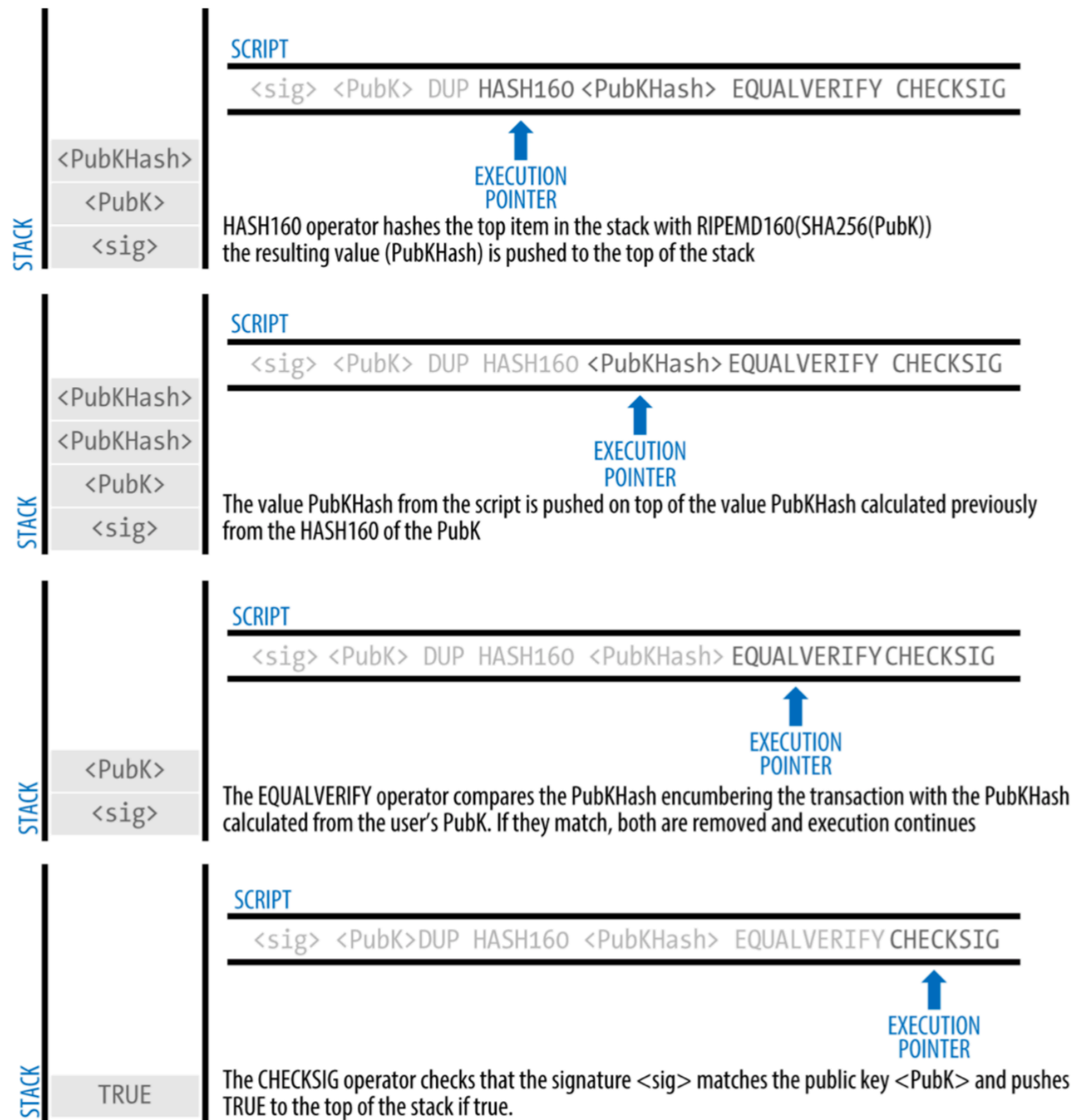


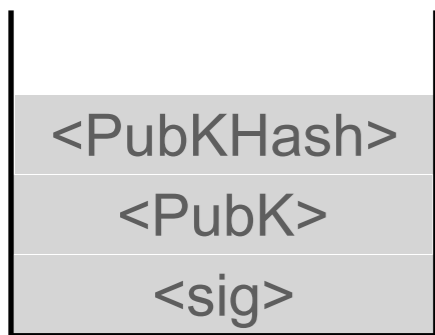
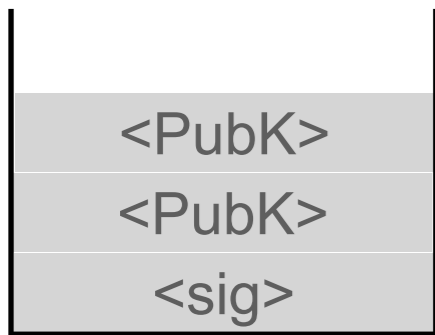
- ◆ Execute a program by a stack
- ◆ If it returns true, the transaction is valid

Validate transaction by Stack



Validate transaction by Stack





<PubKHash>
<PubKHash>
<PubK>
<sig>



<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
scriptSig scriptPubKey

<PubK>
<sig>



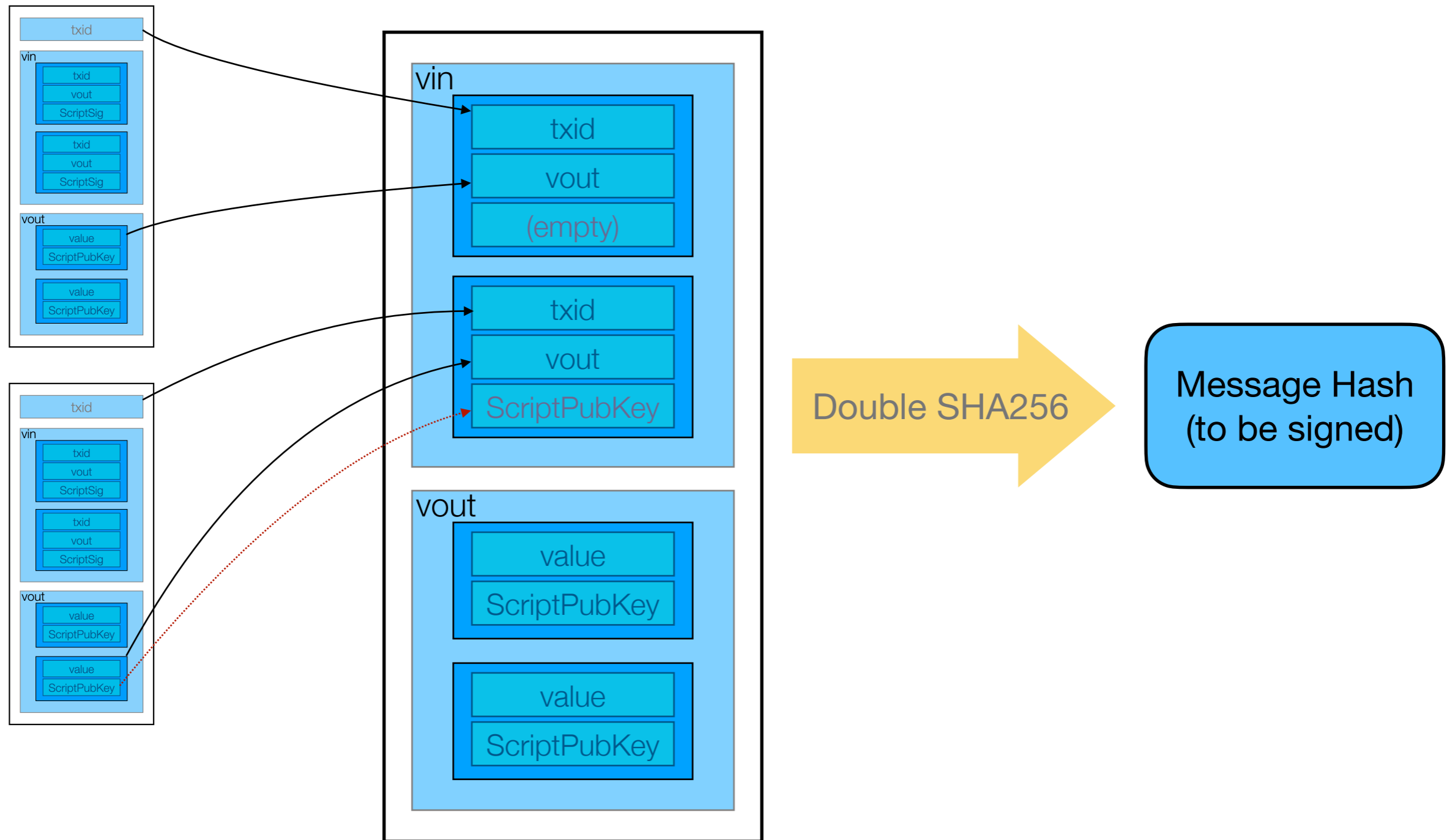
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
scriptSig scriptPubKey

TRUE



<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
scriptSig scriptPubKey

Transaction in detail



WhibOx Contest 2021

The screenshot shows a web browser window at `contest2021.whibox.io`. The page title is "WhibOx Contest 2021 - CHES 2021 Challenge". A navigation menu at the top includes links for "crypto", "English", "Good blogs", "News", "Coding", "Twitter", "Popular", "Medium", "Hacker Noon", "TED", "root-me", "Hackathon", "Live Messi", "GG Calendar", and "GG Security Blog". A user profile "Yoptimus" is logged in.

The dashboard features four key statistics:

- REGISTERED USERS: 81
- CHALLENGES: 97
- BROKEN! / STANDING!: 97/0 (100%)
- BREAKS: 890

The "Challenges" section is titled "Challenges ranking by strawberry scores". It includes a "Show 25 entries" dropdown and a search box. Below is a table with the following data:

| Rank ↑↓ | id ↑↓ | Name ↑↓ | 🍓 Peak ↑↓ | User ↑↓ | Status ↑↓ | Performance ↑↓ | Public Key | Proof of knowledge |
|---------|-------|--------------|-----------|---------|-----------|----------------|----------------------|----------------------|
| 1 | 227 | keen_ptolemy | 20.39 🍓 | zerokey | Broken! | 10.68 | view | view |

WhibOx Contest 2021

contest2021.whibox.io

crypto English Good blogs News Coding Twitter Popular Medium Hacker Noon TED root-me Hackathon Live Messi GG Calendar GG Security Blog

Banana Scores

Your challenge breaks

| Date ↕ | User ↕ | Strawberries ↕ | Challenge Name ↕ |
|----------------------|--------|----------------|---------------------------|
| 2021-08-20 10:11 UTC | You! | 0.00 🍓 | practical_cori (139) |
| 2021-08-20 10:06 UTC | You! | 0.00 🍓 | amazing_aryabhata (185) |
| 2021-08-20 10:06 UTC | You! | 0.00 🍓 | optimistic_jennings (174) |
| 2021-08-20 10:05 UTC | You! | 0.00 🍓 | wonderful_roentgen (187) |
| 2021-08-20 10:03 UTC | You! | 0.00 🍓 | nifty_lamport (235) |
| 2021-08- | | | |

↑