Bitcoin Wallets

Leonardo Comandini
leonardocomandini@gmail.com

March, 2024

Contents

- Recap
- Custodial Wallets
- Wallet Essentials
- Bitcoin Script
- Public Keys and Addresses
- Singlesig, Multisig and More
- Non-Deterministic Wallets
- BIP32: HD wallets
- BIP39: Mnemonics
- Output descriptors
- Modern Wallet Structure
- Hot vs Cold Wallets
- Hardware Wallets
- Blockchain Data Sources

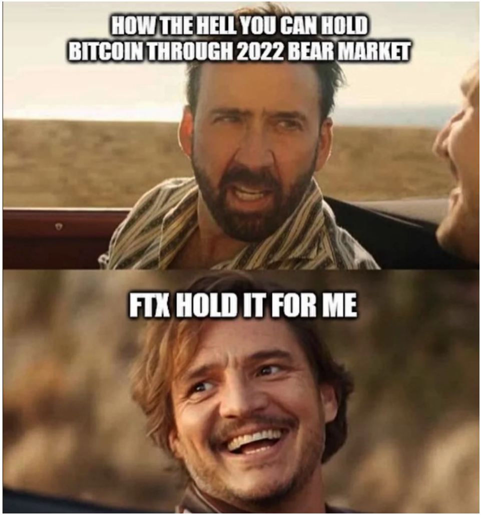
About Last Time

- Hash Functions
- Merkle Trees
- Symmetric Cryptography
- Elliptic Curve Cryptography
- Elliptic Curve Discrete Logarithm Problem
- Signing Algorithm

Key concept: knowledge of private keys $\approx$ ownership of the coins

Custodial Wallets Are Not Actual Wallets

Custodial Wallets Are Not Actual Wallets!



Wallet Essentials

- **Receive**: share your public key and receive some coins to it
- **Show balance** (and history): get data from the blockchain (and mempool)
- **Send**: create, sign and broadcast transactions

Bitcoin Script

Each UTXO (coin) has some spending conditions associated to it, in the form of a script.
To spend a coin it is necessary to provide another script that satisfies those conditions.

- ScriptPubkey
  e.g. `<pubkey> OP_CHECKSIG`
- SigScript
  e.g. `<sig>`

Each node *verifies* that each SigScript satisfies the corresponding ScriptPubkey conditions before accepting a new block.

Public Keys and Addresses

Usually you don't share a public key, but rather an address

private key		public key		scriptpubkey		address
x	$\rightarrow$	P	$\rightarrow^*$	$spk(P)$	$\leftrightarrow$	$addr(P)$

The address maps the scriptpubkey which commits to the spending conditions chosen by the receiver.

Singlesig, Multisig and More

With Bitcoin you choose your spending conditions before receiving.

- Single sig
  - P2PK
  - P2PKH
  - P2WPKH
  - P2SH-P2PWKH
  - P2TR
- Multi sig
  - $multi(2, A, B)$
  - $multi(2, A, B, C)$
  - $and(A, B, C)$
- More
  - $or(A, and(B, C))$
  - $or(A, and(B, older(1000)))$

Non-Deterministic Wallets

- Generate a random private key x_1
- Share the corresponding private key P_1 (or $addr(P_1)$)
- ...
- Generate a random private key x_n
- Share the corresponding private key P_n (or $addr(P_n)$)

Each time you generate a new private key, you must back it up.

BIP32: Deterministic Wallets

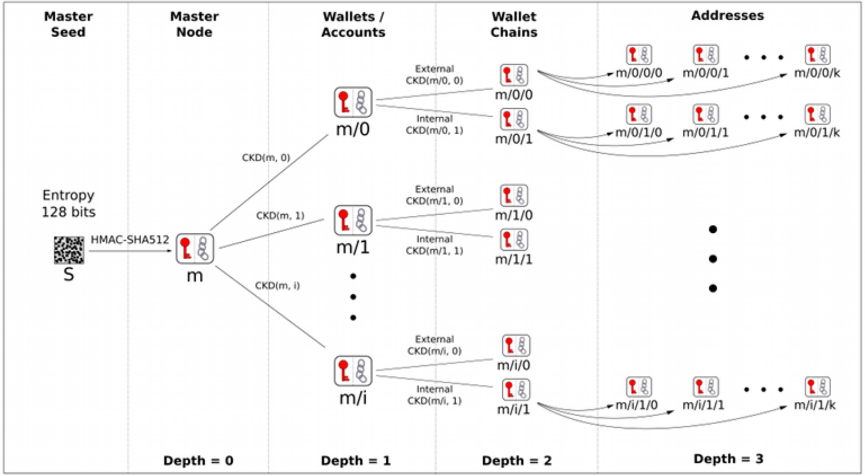
BIP32 (2012) defines a standard for **deterministic** wallets to avoid the drawbacks of incremental backups.

BIP32 seed (16 to 64 bytes) → master private key → private keys

You **just need to backup the seed** and you have backed up an ”unlimited” number of private keys.

BIP32: Hierarchical Wallets

BIP 32 - Hierarchical Deterministic Wallets



Child Key Derivation Function ~ $\mathrm{CKD}(x,n) = \text{HMAC-SHA512}(x_{\text{Chain}}, x_{\text{PubKey}} \,||\, n)$

BIP32: Public Derivations

BIP32 introduces **extended keys** (the nodes in the tree):

prefix	key	chain code	to child i	to child i'
xprv	x	c	yes	yes
xpub	P	c	yes	**no**

i is in $0..2^{31}$ (**normal derivation**).
i' stands for $i + 2^{31}$ (**hardened derivation**).

Key concept: *from an xpub you can derive an "unlimited" set of public keys*, without (!) knowing (or learning) the xprv (or any of the private keys)

BIP32: Popular Extensions

- BIP44: P2PKH, e.g. $m/44'/0'/0'/0/*$
- BIP49: P2SH-P2WPKH, e.g. $m/49'/1'/0'/0/*$
- BIP84: P2WPKH, e.g. $m/84'/0'/2'/0/*$
- BIP86: P2TR, e.g. $m/86'/0'/0'/1/*$

$$m/purpose'/coin_type'/account'/is\_change/*$$

- SLIP132: *xpub*, *zpub*, *ypub*, ...

Secret Backups

- Single keys (WIF)
- Extended private keys (xprv)
- BIP32 seeds

Not very user friendly

BIP39: mnemonics

BIP39 defines a standard for generating a set of words from which a BIP32 seed can be derived.

BIP39 mnemonic → BIP32 seed → xprv → xpub

A BIP39 mnemonic usually consists in 12 or 24 words from a dictionary of 2048 words.

"cat swing flag economy stadium alone churn speed unique patch report train"

This can be derived to a BIP32 seed from which derive a BIP32 HD wallet.

"deb5f45...87b46541f5", "xprv9s2...MDLXdk95DQ"

BIP39 is *unanimously discouraged for implementation* but it's the most common standard for mnemonic backups.

Proper Backups: Output Descriptors

In order to correctly derive addresses, you need to remember the spending conditions. These can be expressed with the **output descriptors**.

Few examples of public output descriptors:

- pk(0279..98)
- sh(multi(2,022f..01,03ac...be))
- pkh([d34db33f/44'/0'/0']xpub6E...EL/1/*)
- wsh(multi(1,xpub66...uB/1/0/*,xpub69...PH/0/0/*))

Modern Wallets Structure

Modern wallets are usually split in 2 parts:

- **Watch-only**: public descriptors, addresses, xpubs, public keys
- **Signers**: mnemonics, BIP32 seeds, xprvs, private keys

This allows to threat the signers security with additional care.

Hot vs Cold Wallets

Hot wallets: private keys are on the device

Cold wallets: private keys are somewhere else

Hardware Wallets

You can move your private keys to a specialized hardware, commonly referred to as a **Hardware Wallet** (HWW).
A better name would be **signing device**.
HWWs works with a host application (**companion app**).

Tasks that a HWW must perform:

- initialize and show backup
- get xpub
- register descriptor
- confirm address
- sign transaction

Some HWW vendors

- Ledger
- Trezor
- Coldcard
- BitBox
- Seed Signer
- Blockstream's Jade
- TwentyTwoHW's Portal

Blockchain data sources

- Bitcoin Core
- Electrum Servers
- More specialized servers (e.g. esplora)

Different trade-offs, in terms of performance, bandwidth, storage requirements and trust.

Exercise

Describe the process of setting up a 2of2 wallet with software and hardware signer.

Describe the backup for each party in details.

Thank you