Bitcoin and Cryptocurrency Technologies
Lecture 8: Bitcoin Wallets

Yuri Zhykin

Mar 27, 2025

UTXO Set

- All bitcoin in circulation is represented by a **UTXO set** - a set of all unspent transaction outputs.
- Every "coin" (**UTXO**) consists of the amount of **satoshis** and the corresponding lock script.
- In order to verify the received transaction, a Bitcoin user has to check that transaction is correctly constructed and if the outputs used by the transaction are included in the **UTXO set**.
- Whole Bitcoin protocol works to ensure the **consistency** of of the **UTXO set**.

Bitcoin Ownership

- **Owning bitcoin** means that some entity can provide the correct **unlock script** to the **lock script** of some of the outputs in **UTXO set**.
- Lock scripts are visible publicly, so ideally every UTXO should have a different lock script.
- Otherwise, it is immediately visible how much bitcoin a certain entity owns.
- **Bitcoin wallet** is usually a piece of software for storing data needed to construct unlock scripts for the associated UTXOs.

Standard Lock Scripts

- **P2PK** - Pay to Public Key

```
<pubKey> OP_CHECKSIG;
```

- **P2MS** - Pay to Multi-Signature

```
<M> <pk1> ... <pkN> <N> OP_CHECKMULTISIG;
```

- **P2PKH** - Pay to Public Key Hash

```
OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG;
```

- **P2SH** - Pay to Script Hash

```
OP_HASH160 <scriptHash> OP_EQUAL;
```

- **P2WPKH** - Pay to **Witness** Public Key Hash

```
OP_0 <20-byte-witness-data>;
```

- **P2WSH** - Pay to **Witness** Script Hash

```
OP_0 <32-byte-witness-data>;
```

Segregated Witness

- Softfork in the network, activated on 24 August 2017.
- Proposed in a series of **Bitcoin Improvement Proposals** - BIP-0141, BIP-0143, BIP-0144 and BIP-0148.
- Main idea is to move the large unlock scripts out of the transaction data that is included in the blocks.

Bitcoin Addresses 1/4

- Bitcoin uses several human-oriented encodings to encode addresses and keys:
  - **Base58Check**

    $$Base58Check(t,data) = Base58(t + data + HASH256(t + data)[0{:}4])$$

    where *Base58*

    $$123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz$$
  - **Bech32**

    $$Bech32(t,data) = t + \text{"1"} + Base32'(data + Bech32Checksum(t,data))$$

    where *Base32'*

    $$qpzry9x8gf2tvdw0s3jn54khce6mua7l$$

Bitcoin Addresses 2/4

- **P2PK** and **P2MS** do not have defined address formats.
- **P2PKH** address format

```
OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG;
```

$A_{p2pkh} = Base58Check(0x00 + pubKeyHash)$

17VZNX1SN5NtKa8UQFxwQbFeFc3iqRYhem

$A'_{p2pkh} = Base58Check(0x6F + pubKeyHash)$

mipcBbFg9gMiCh81Kj8tqqdgoZub1ZJRfn

Bitcoin Addresses 3/4

- **P2SH** address format

```
OP_HASH160 <scriptHash> OP_EQUAL;
```

$A_{p2sh} = Base58Check(0x05 + scriptHash)$

3EktnHQD7RiAE6uzMj2ZifT9YgRrkSgzQX

$A'_{p2sh} = Base58Check(0xC4 + scriptHash)$

2MzQwSSnBHWHqSAqtTVQ6v47XtaisrJa1Vc

Bitcoin Addresses 4/4

- **P2WPKH/P2WSH** address format

```
OP_0 <20-or-32-byte witnessData>;
```

$$A_{p2wpkh/p2wsh} = Bech32(\text{"}bc\text{"} + witnessVersion + witnessData)$$

bc1qw508d6qejxtdg4y5r3zarvary0c5xw7k**v8f3t4**

$$A'_{p2wpkh/p2wsh} = Bech32(\text{"}tb\text{"} + witnessVersion + witnessData)$$

tb1qw508d6qejxtdg4y5r3zarvary0c5xw7k**xpjzsx**

Cryptographic Key Storage 1/2

- All standard lock/unlock scripts are based on providing a signature matching the public key, whose hash is included in the script (either lock script, or unlock script in case of P2SH, P2WPKH and P2WSH).
- Since the form of the script is standardized, *the only component that differs* is **the hash of the public key**.
- The only piece of data needed to construct a standard unlock script for the standard lock script is the *corresponding private key*.

Cryptographic Key Storage 2/2

- All bitcoin wallets currently in use are simply **cryptographic key stores** with additional functionality:
  - securely store private keys for owned "coins",
  - generate new private keys/public keys/addresses,
  - for every new block or transaction, verify if its lock script corresponds to a standard lock/unlock script that matches any of the stored keys (optional),
  - construct new transactions by combining a subset of known UTXOs and a set of new UTXOs and providing corresponding unlock scripts (optional).

Simple Key Pool Wallets

- Simplest Bitcoin wallet is a **single private key**.
- The address is included in the public chain data, so if **addresses are reused**, it is easy to calculate the amount of bitcoin owned by the same entity.
- Since *reusing addresses* is bad, the solution is to simply generate a new key for every new incoming transaction.
- Wallet is a file containing a list of keys for all "coins" owned.
- Backup is needed after every received transaction.
- The size of the key storage continues to grow and its unsafe to remove old keys as their addresses might still get used in the future.

Hierarchical Deterministic Wallets 1/2

- **Hierarchical deterministic wallets** (**HD wallets**) were first introduced in 2011 and standardized by BIP-0032 in 2012.
- The core idea behind HD wallets is to generate a **master private key** and derive all future private keys from it.
- **Any private key in the hierarchy can be used to generate any child private keys**.

  $CKD_{priv}(k_{par}, c_{par}, i) = HMACSHA512(c_{par}, k_{par} G || i) = I$

  $k_i = I[0{:}32] + k_{par} \pmod{n}$

  $c_i = I[32{:}64]$

- **Any public key in the hierarchy can be used to generate any child public keys but not their private keys**.

  $CKD_{pub}(K_{par}, c_{par}, i) = HMACSHA512(c_{par}, K_{par} || i) = I$

  $K_i = (I[0{:}32])G + K_{par} = (I[0{:}32] + k_{par})G = k_i G$

  $c_i = I[32{:}64]$

Hierarchical Deterministic Wallets 2/2

- BIP-0039 defines a way to encode the master key as a sequence of words.
- Most modern wallets show BIP-0039 **seed** on initialization.
- 2048 words in the dictionary.
- 12-word sequence contains 128 bits of security.
- Example:

```
fortune flush   weekend current
key     hero    snake   leopard
brisk   climb   timber  appear
```

Security: Mobile Wallets

- Dozens of wallet applications exist for mobile devices (both iOS and Android):
  - **BlueWallet** (iOS, Android, centralized server),
  - **Green** (iOS, Android, centralized server),
  - **Bitcoin Wallet** (Android, SPV node).
- Main disadvantage of mobile wallets is that the keys are stored on a network-connected device, which is inherently insecure.
- Any security breach that allows attackers to access the data on the device may result in keys being stolen.
- OK to use for day-to-day transactions involving small amounts of bitcoin.

Security: Hardware Wallets

- Hardware wallets are dedicated **air-gapped** devices that specialized at generating and storing cryptographic keys:
  - **Trezor devices**
  - **Ledger devices**
  - **Coinkite's Coldcard**
  - **Blockstream's Jade**
  - **Shift Crypto's BitBox02**

Security: Cold Storage

- **Cold storage** is any method of storage that does not involve software or devices at all.
- HD seed can be written on a piece of paper and stored in a secure place, or even simply memorized.
- In order to use the bitcoin from cold storage, one needs to install the key on one a signing-capable device and only then create a transaction.

Security: General Approach

- **Resonably secure approach** to storing bitcoin:
  - generate a new key with a dedicated hardware device (e.g. hardware wallet),
  - create a cold backup of the **master private key** or seed,
  - import the **master public key** to a device that will be used to track the storage,
  - **delete the private key from the dedicated device**.

The End

Thank you!