HIERARCHICAL DETERMINISTIC WALLETS

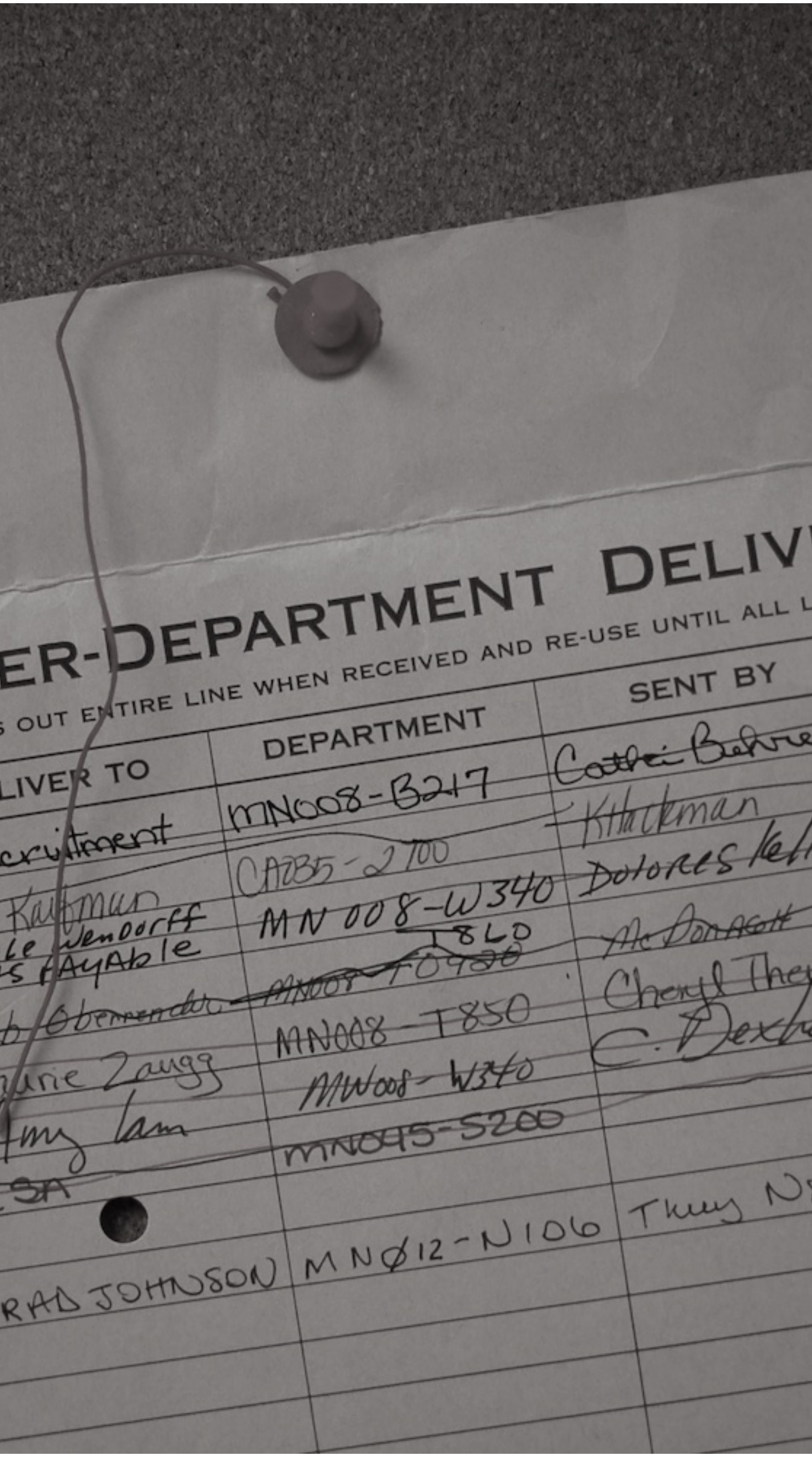
JOHN NEWBERY

@jfnewbery

github.com/jnewbery

HIERARCHICAL DETERMINISTIC WALLETS

- The problems of address reuse
- BIP 32 - HD Wallets
- BIP 39 - Mnemonics for HD wallet seeds
- BIPs 43 and 44 - Multi-Account Hierarchy for HD Wallets



THE PROBLEMS OF ADDRESS REUSE

THE PROBLEMS OF ADDRESS REUSE

- Privacy
- ECDSA Security
- Quantum Security

PRIVACY

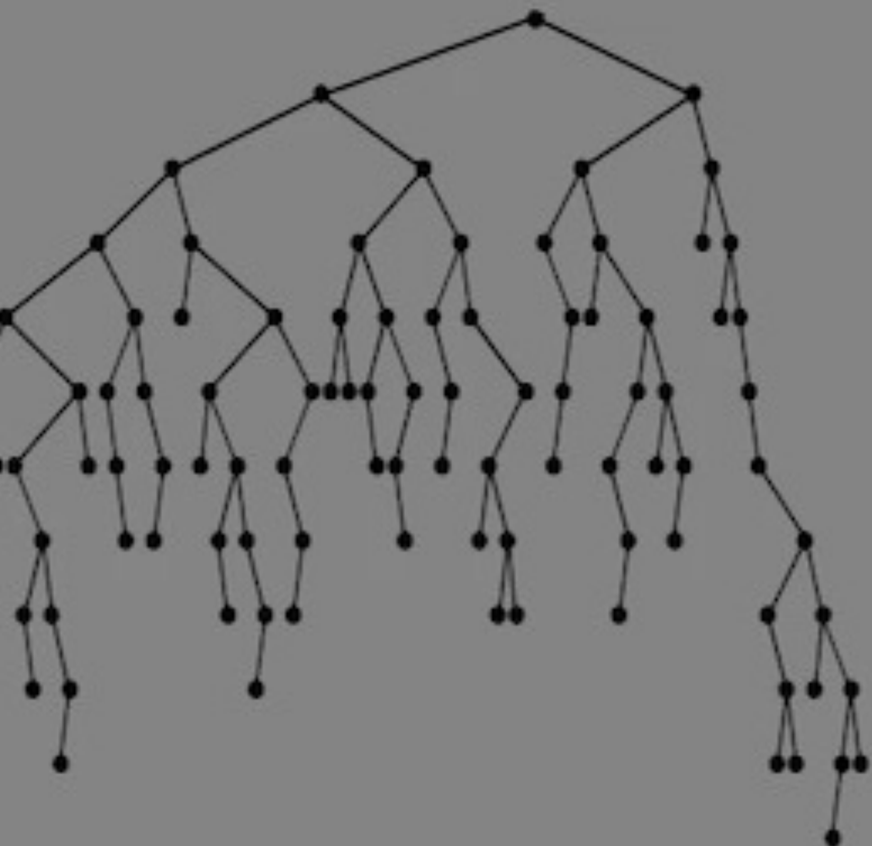
- Bitcoin transactions are public
- *Chain analysis* can uncover patterns
- Re-using addresses can reveal information

SECURITY

- ECDSA requires a (cryptographically secure random) ephemeral key
- Signing with the same ephemeral key reveals the private key
- If your PRNG is broken, then reusing an address can reveal the private key

QUANTUM SECURITY

- Sending *to* an address (using P2PKH) does not reveal the public key
- Spending *from* an address reveals the public key
- ECDSA is not quantum secure
- SHA256 is more quantum resistant



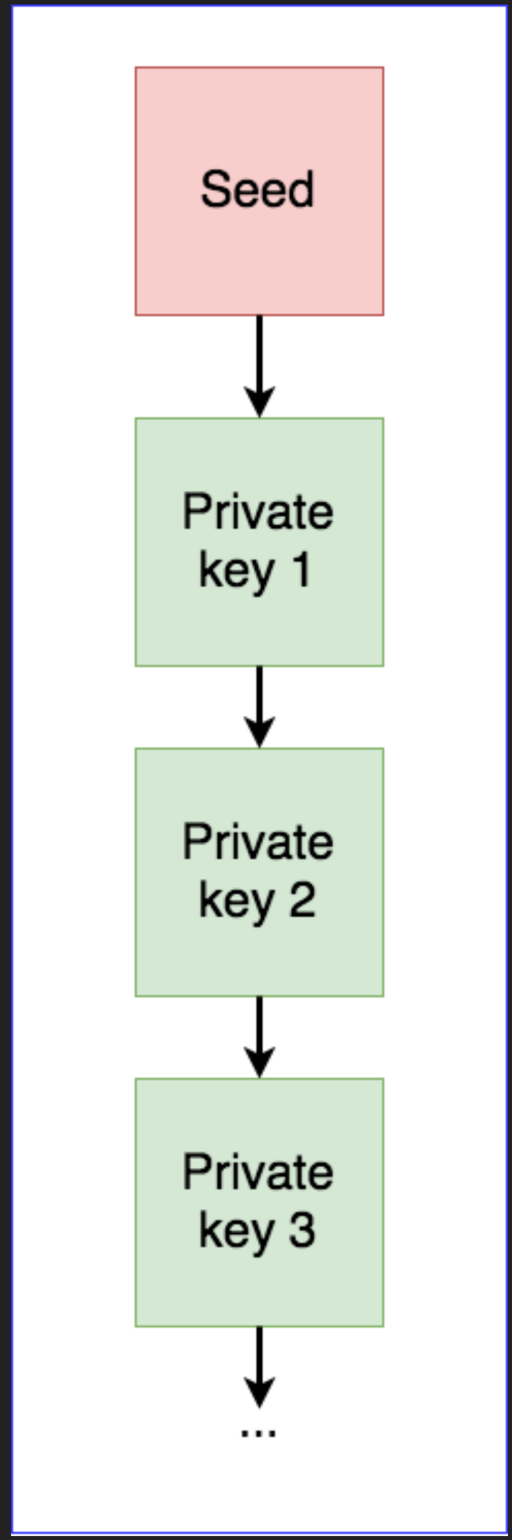
BIP 32
HD WALLETS

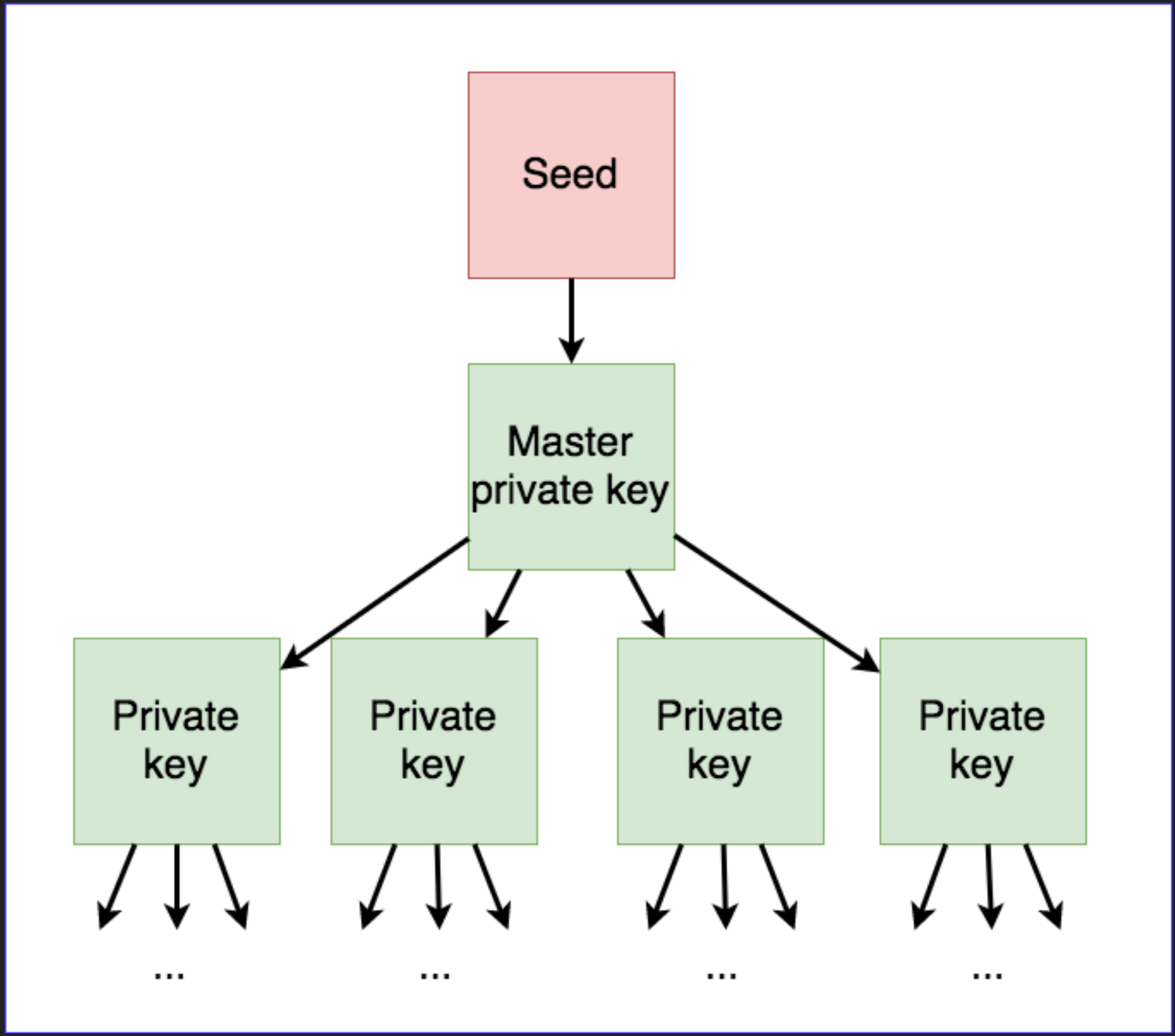
HD WALLETS – USE CASES

- Full wallet sharing
- Per-office or per-department balances
- Recurrent transactions
- Unsecure money receiver

SINGLE-USE ADDRESSES

- Best practice is to only use addresses once
- Having many unlinked private keys is difficult to backup and share
- Better to have a *seed* and a way to deterministically derive new private keys
- Sharing a hash chain is all or nothing
- A tree allows sub-branches to be shared individually


Seed
Private key 1
Private key 2
Private key 3
...



Seed
Master private key
Private key
Private key
Private key
Private key
...
...
...
...

BIP 32 OVERVIEW

- Generate a random 128-512 bit seed ***S***
- Use HMACs (Hash Message Authentication Codes) to derive child nodes.
- For the *master key* **m**:
  - I = HMAC-SHA512(Key = “Bitcoin seed”, data = S)
  - I_L (the left 256 bits) is the *master private key*.
  - I_R (the right 256 bits) is the *master chain code*.

CHILD KEY DERIVATION (PRIVATE)

- Derive child key from parent key using HMACs
- $I = \text{HMAC-SHA512}(key = C_{par}, data = K_{par} \,||\, i)$
  non-hardened, $i < 2^{31}$
- $I = \text{HMAC-SHA512}(key = C_{par}, data = 0x00 \,||\, k_{par} \,||\, i)$
  hardened, $i >= 2^{31}$ notation: $i_H == i' == i + 2^{31}$
- $I_L + k_{par}$ is the *child private key*
- I_R is the *child chain code*
- This function is called CKDpriv

CHILD KEY DERIVATION (PUBLIC)

- Only possible for non-hardened child keys
- $I = \text{HMAC-SHA512}(\text{key} = C_{par}, \text{data} = K_{par} \| i)$
  (non-hardened)
- $I_L + K_{par}$ is the *child public key*
- I_R is the *child chain code*
- This function is called CKDpub

SERIALIZATION FORMAT

- BIP 32 defines a 78 byte *extended key format*:
  - 4 bytes: “version” (eg 0x0488b21e - “xpub”) for Bitcoin main net
  - 1 byte: depth in key derivation tree
  - 4 bytes: fingerprint of the parent’s key
  - 4 bytes: child index
  - 32 bytes: chain code
  - 33 bytes: pub key compressed or 0x00 || priv key

KEY IDENTIFIER AND FINGERPRINT

- *Identifier* of extended key is HASH160(Public Key)
- This is the same data used in the Bitcoin address
- *fingerprint* of key is first 32 bits of identifier

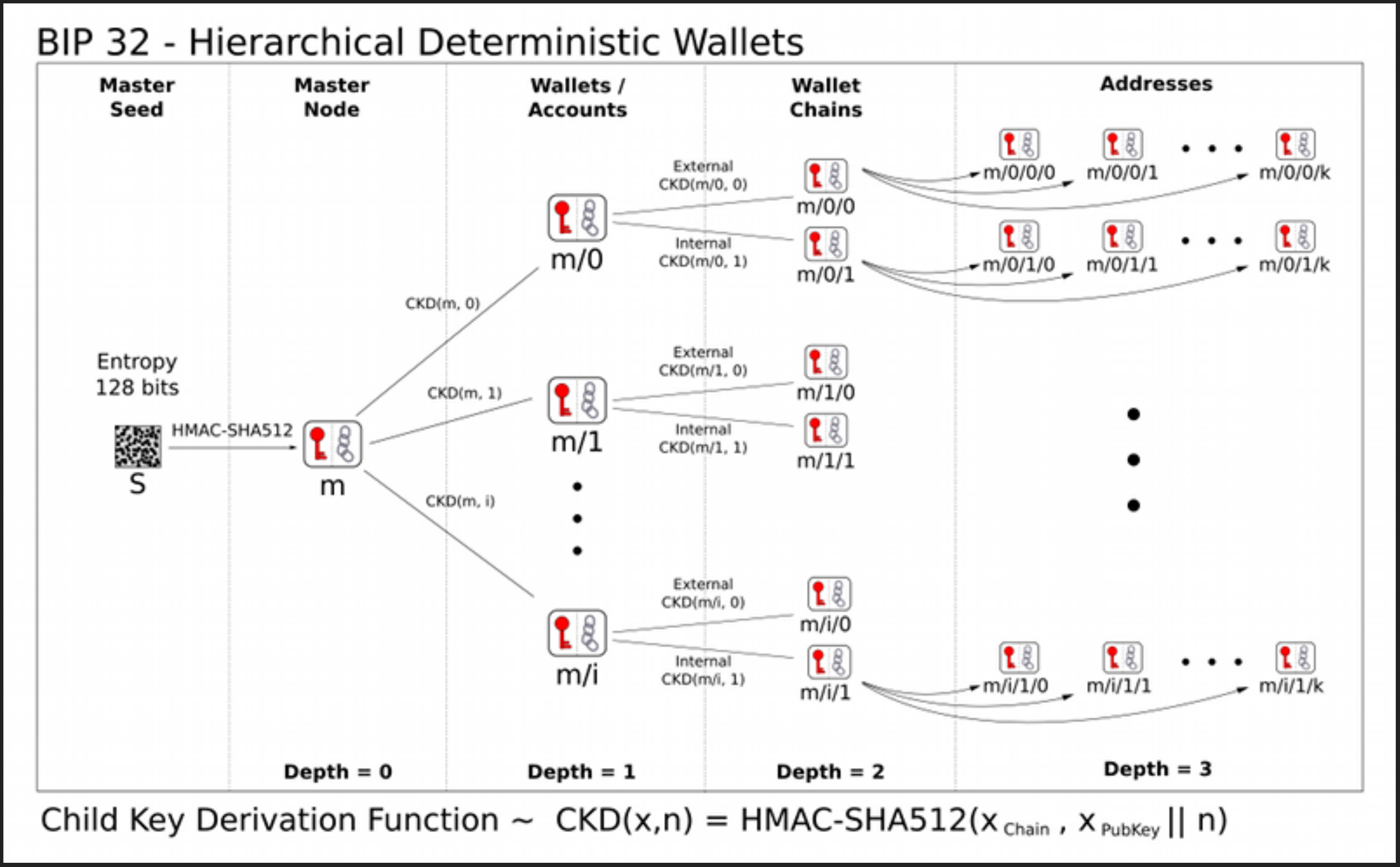
KEY TREE

- Construct a tree of keys by repeatedly applying CKDpriv
- Notation: index of each child key, separated by slashes
- eg: m/3_H/2/5 or m/3'/2/5

DEFAULT WALLET LAYOUT (1)

- Wallet is organized as several ‘accounts’, indexed by i
- Each account has two keypair chains:
  - internal: used for giving out addresses.
    Key notation: $\mathbf{m/i_H/1/k}$
  - external: used for change addresses, etc.
    Key notation $\mathbf{m/i_H/0/k}$

DEFAULT WALLET LAYOUT (2)



SECURITY OF HD WALLETS

- Given a child extended private key (k_i, c_i) and i, attacker cannot derive parent private key
- given any number of extended private keys (k_{i_j}, c_{i_j}) and i_j, attacker cannot determine if they are from a common parent
- HOWEVER!
- given a parent extended public key (K_{par}, c_{par}) and a non-hardened child private key, it is possible to derive a parent extended private key
  - a compromised extended private key compromises all private keys up to the first hardened parent

BIP 39 MNEMONICS

BIP 39

- A way to generate a BIP 32 seed using a mnemonic
- Submitted by Slush (Satoshi Labs) and used in Trezor
- Used in the Trezor hardware wallet
- There are some criticisms of this method

GENERATING THE MNEMONIC SENTENCE

- Generate 128-256 bits of entropy. Call these bits **ENT**
- Append the first len(ENT)/32 bits of SHA256(ENT)
- Split the concatenated bits into 11 bit chunks
- Each 11 bit chunk corresponds to an entry in a 2048 word list
- Example:
  SCHEME SPOT PHOTO CARD BABY MOUNTAIN DEVICE KICK CRADLE PACT JOIN BORROW

LENGTH OF MNEMONIC SENTENCE

len(ENT)	len(CS)	len(ENT + CS)	Number of words
128	4	132	12
192	6	198	18
256	8	264	24

FROM MNEMONIC SENTENCE TO SEED

- Generate 128-256 bits of entropy. Call these bits **ENT**
- Append the first len(ENT)/32 bits of SHA256(ENT)
- Split the concatenated bits into 11 bit chunks
- Each 11 bit chunk corresponds to an entry in a 2048 word list

GENERATING THE SEED FROM THE MNEMONIC SENTENCE

- Use PBKDF2 (Password-based Key Derivation Function 2)
  - 2048 rounds of HMAC-SHA256
- Password: the mnemonic sentence
- Salt: “mnemonic” + optional passphrase

CRITICISMS

- A fixed wordlist is required (because of the way the checksum is computed)
- Does not have ‘versioning’ - the seed does not indicate how the tree should be derived
- Relies on the security of the CSPRNG. Not clear whether using a random input to the PBKDF is any better than using a user-supplied password

BIPS 43 & 44 – MULTI-ACCOUNT HIERARCHIES

BIPS 43 AND 44

- Another two BIPs from Slush (Satoshi Labs)
- Imposes structure on the key tree
- Intended for portability between wallet implementations

BIP 43

- First level of tree hierarchy should be ‘purpose’
  m / purpose’ / *
- For example, BIP 44 hierarchy starts:
  m / 44’ / *

BIP 44

- Defines entire structure for trees
- **m / purpose' / coin_type' / account' / change / address_index**
- purpose: 44'
- coin_type: defined in Satoshi Labs SLIP-0044. Bitcoin main net is 0'
- account: used for wallet user organization
- change: 0 for external chain, 1 for internal chain (same as BIP 32 default layout)
- address_index: set of addresses for use by the wallet

ACCOUNT DISCOVERY

- Used to restore wallet from backup seed
- Account field starts from 0
- Scans external chain until there’s a gap of 20 unused addresses
- If account **i** has transactions, also try scanning account **i + 1**