Bitcoin:
Programming the Future of Money

Topics in Computer Science - ITCS 4010/5010, Spring 2025

Dr. Christian Kümmerle

Lecture 16

Wallets

Some figures are taken from:

- “Mastering Bitcoin: Programming the Open Blockchain”, (Andreas Antonopoulos, David Harding), 3rd Edition, O’Reilly, 2023.

What is a “wallet”?

WHAT IS A WALLET IN THE CONTEXT OF BITCOIN?

A **wallet** is an application that interacts with the Bitcoin blockchain and stores (private and/or) public key information.
Types of wallets:

- **Paper “wallets”**
- **Desktop wallets**
- **Mobile wallets**
- **Web “wallets”**
- **Hardware wallet / hardware signing device**

PAPER WALLET

- Contains information about a private key / associated information
- Often uses “Wallet Import Format” (WIF) for private key information
- **Not best practice to “store” larger amount of bitcoin!**

Is not a “wallet” in strict sense.

WALLET IMPORT FORMAT

- Start with byte 0x80 (main net) and 0xef (test net)
- Append: Private key integer in 32-byte big-endian.
- If compressed SEC is used for public key: Append 0x01, otherwise append nothing
- Apply hash256 to bit string above, get first 4 bytes (checksum),
  append this to above.
- Encode in Base58.

- Purpose: Have a format for private key that is “easily” human readable.

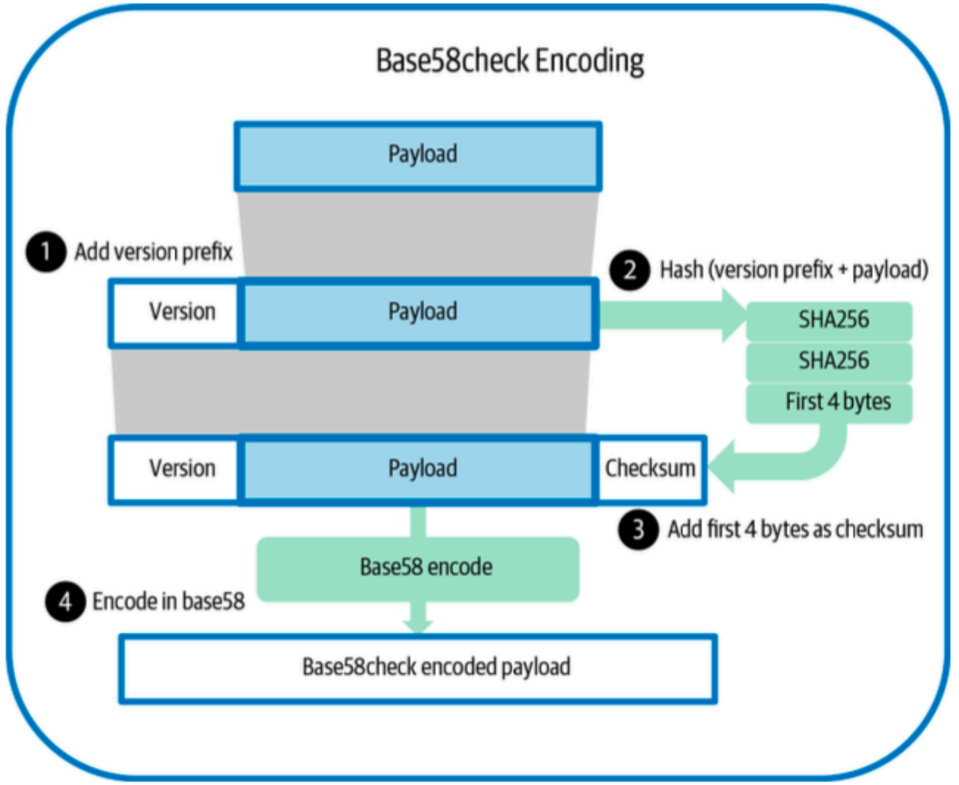


Table 4-1. Base58check version prefix and encoded result examples

Type	Version prefix (hex)	Base58 result prefix
Address for pay to public key hash (P2PKH)	0x00	1
Address for pay to script hash (P2SH)	0x05	3
Testnet Address for P2PKH	0x6F	m or n
Testnet Address for P2SH	0xC4	2
Private Key WIF	0x80	5, K, or L
BIP32 Extended Public Key	0x0488B21E	xpub

DESKTOP WALLETS

- Software for a desktop computer.
- May also serve as a “bitcoin node” that keeps a copy of entire blockchain (currently: more than 600 GB) and that is able to validate blocks and transactions.
- Example for desktop wallet w/ bitcoin node:
  Bitcoin Core (https://bitcoincore.org/)
  Derived from reference implementation developed by Satoshi Nakamoto

Other desktop wallets:

- Electrum (https://github.com/spesmilo/electrum)
- Specter Desktop (https://github.com/cryptoadvance/specter-desktop)
- Sparrow (https://sparrowwallet.com/)

DESKTOP WALLETS

- Desktop wallets can, but **don’t have to store your private keys**.
- Best practice: **Don’t store your private keys on desktop wallets**!

Downsides of desktop wallets (if used to store keys):

- **Security:**
  Prone to cybersecurity issues / hacks. May lose your funds!

DEGREES OF AUTONOMY OF DIFFERENT WALLET TYPES

Categorization applicable for desktop/mobile wallets:

- **Integration with full Bitcoin node**
  Validates entire history of transactions, can share copies of blocks =>
  Best security & privacy, large storage requirement (>600 GB)
- **Pruned Bitcoin node**
  Validates entire history of transactions, but discards older blocks
  except for header & keeps valid UTXO set
  => **Security as full node, lower storage requreiment,** but cannot
  share blocks to other network participants
- **Lightweight Client (Simplified Payment Verification)**
  Only downloads block headers, relies on trusted nodes for verification.
  Minimal store requirements
- **Third-party API client**
  Interacts through third-party system with Bitcoin network
  Full trust into third party required, limited security/privacy

Downsides of lightweight wallets/ third-party API wallets:

- **Security:**
  Only partially validate (if at all) if new blocks follow the rules of the Bitcoin protocol
- **Privacy:**
  When checking your balance, send Bitcoin addresses to a “trusted” service / “trusted” bitcoin node to receive balance and transaction history.
  -> Third party can spy on user

Comparison Pruned Node vs. Lightweight Client

Aspect	Pruned Node	Lightweight (SPV) Client
Verification	Fully Independent	Trust-based (partial verification)
Security	High (same as full node)	Lower (depends on trust)
Storage & Bandwidth	Moderate (several GB)	Very Low (MB)
Privacy	Good	Limited
Network Participation	Active validation	Minimal (consumer)

MOBILE WALLET

- Software for mobile phones
- Similar to desktop wallets
- Sometimes **lightweight clients**, mostly **third-party API client**

WEB WALLET

- Often offered by Bitcoin exchanges (private companies that allow you to buy / sell cryptocurrencies for “fiat” money)
- Usually, keys are **fully controlled by external service**

Basic principle:

Not your keys, not your coins!

HARDWARE WALLET

Better name: “Hardware signing device”

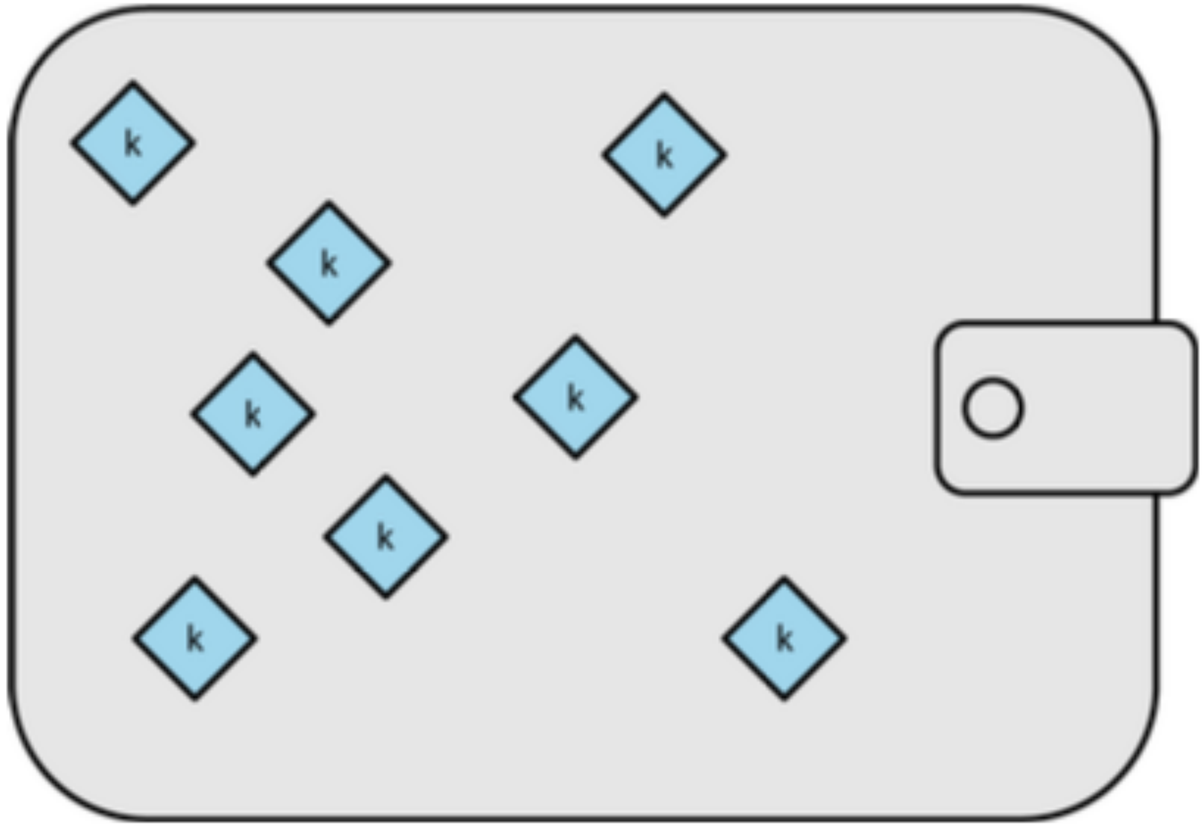
Better name: “Hardware signing device”

- Specialized hardware to securely **store private key information** and **sign transactions**
- Interact with desktop/mobile/web wallet via USB, NFC, or QR-codes.
- Commonly thought of as method **of choice to secure keys controlling larger amount of funds**

- Best practice: Use “analog" backup to secure against hardware failure / have redundancy in case of theft / loss.

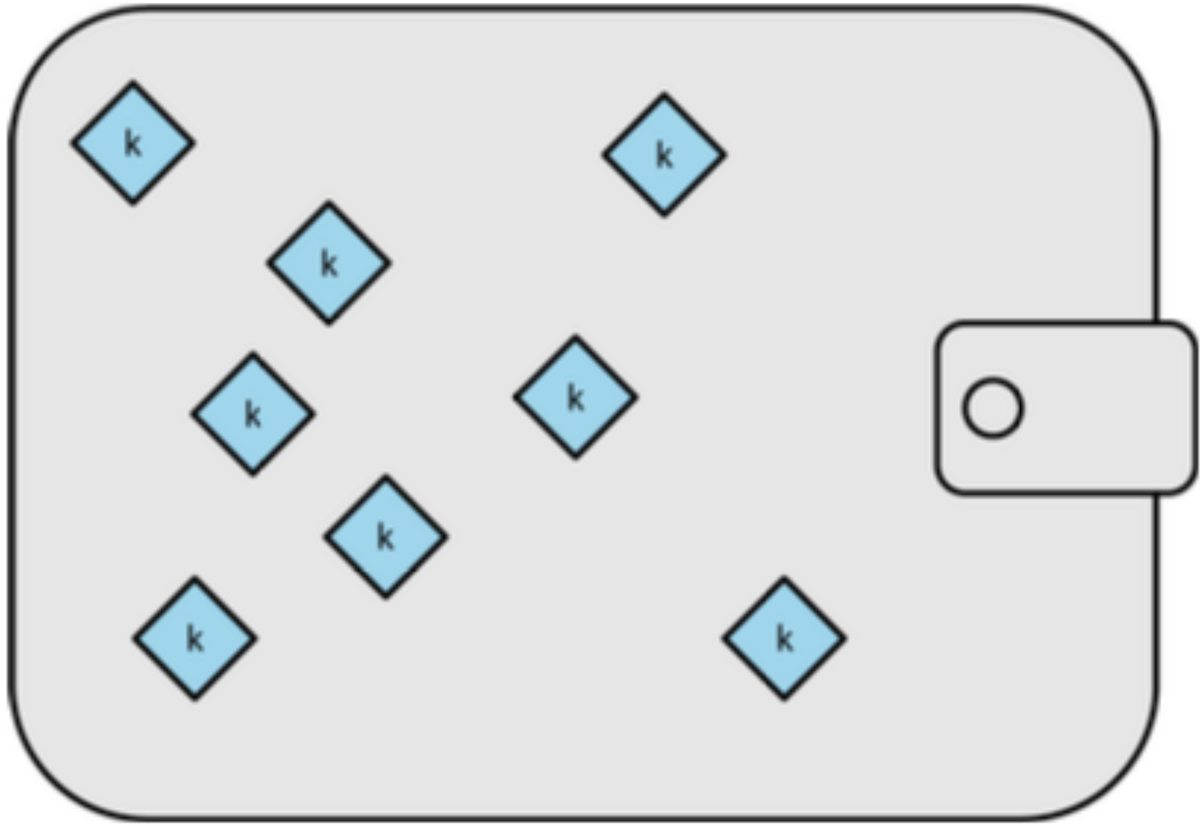
Relationship of Wallets & Addresses

NON-DETERMINISTIC KEY GENERATION



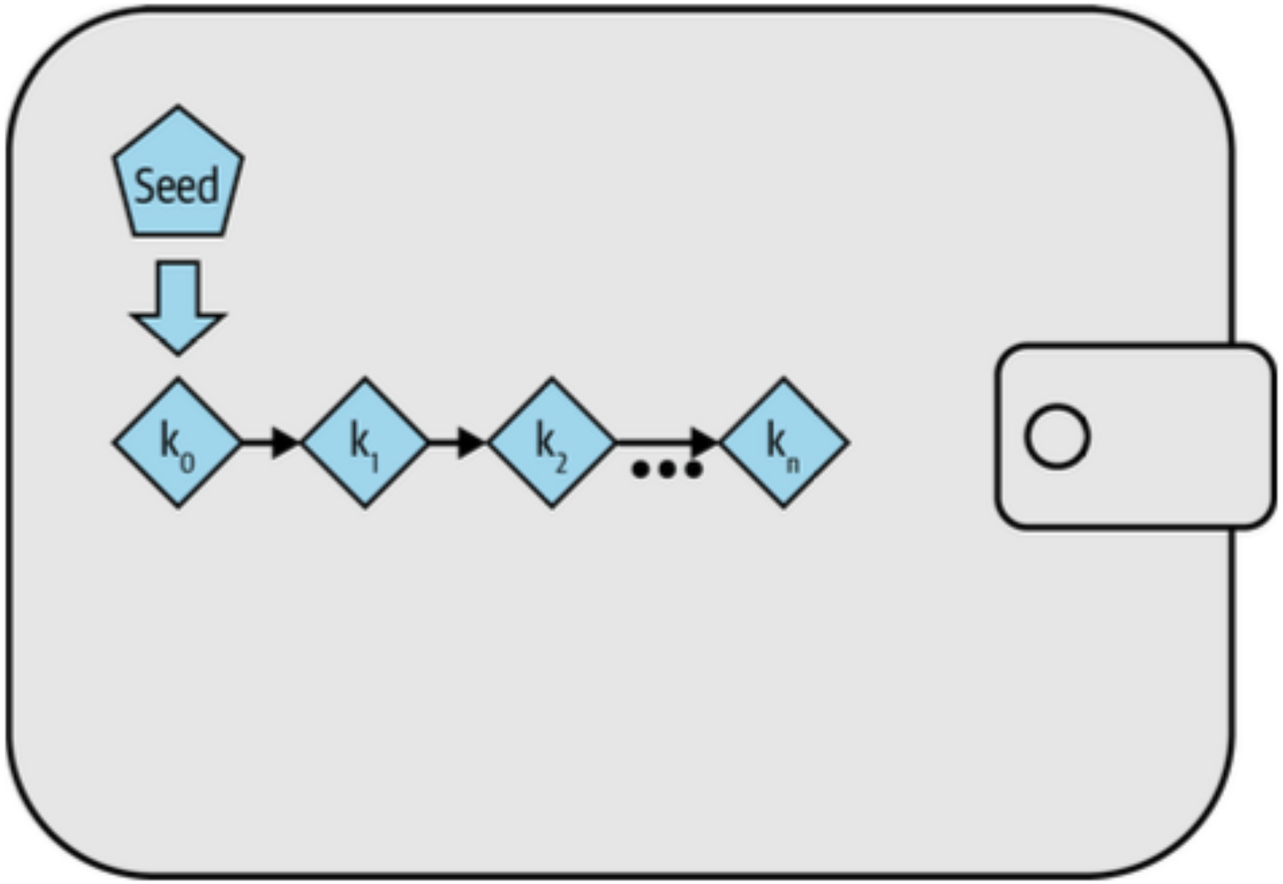
Idea: For each address, generate new, random seed (new private key)

NON-DETERMINISTIC KEY GENERATION



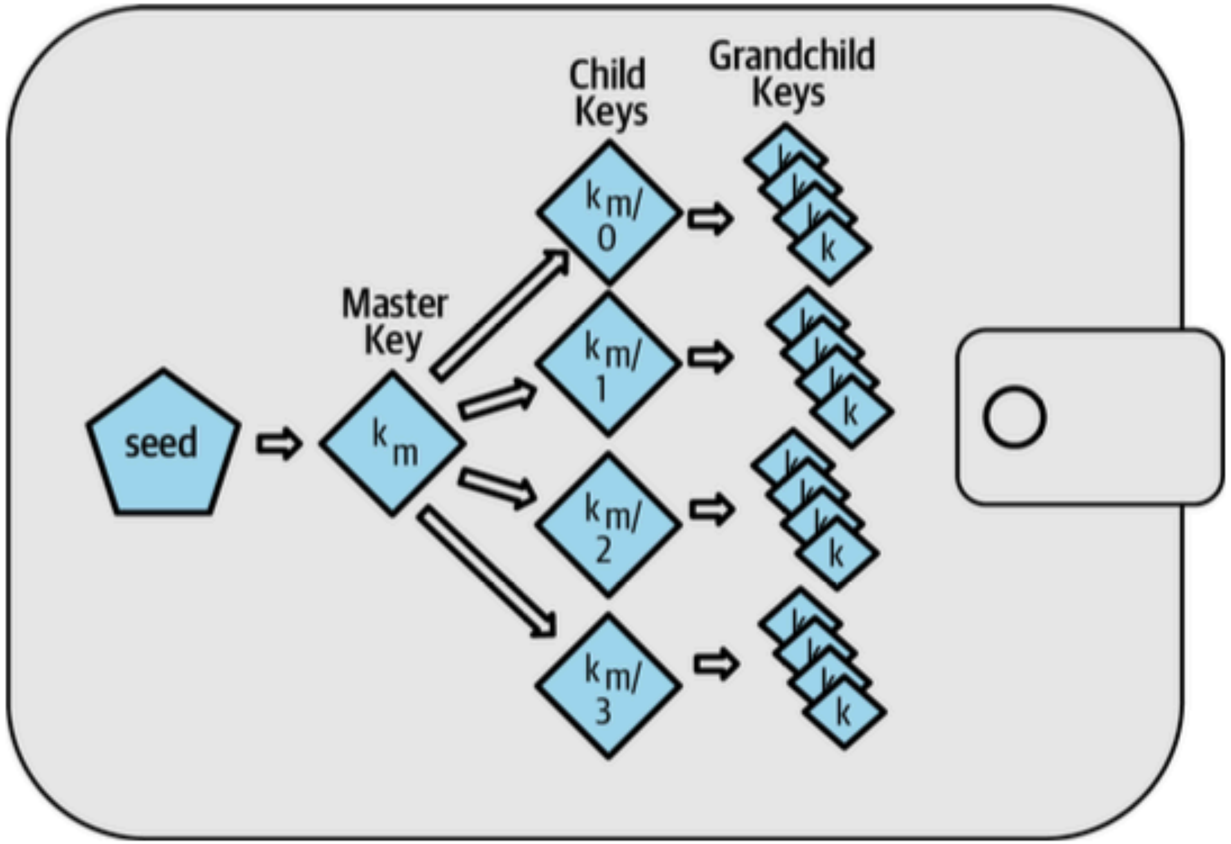
Problem: Need for backup of one private key per transaction (if privacy is desired)

DETERMINISTIC KEY GENERATION



Idea: Have **one** random seed, derive **different private keys** from each other through hashing.

HIERARCHIC DETERMINISTIC KEY GENERATION

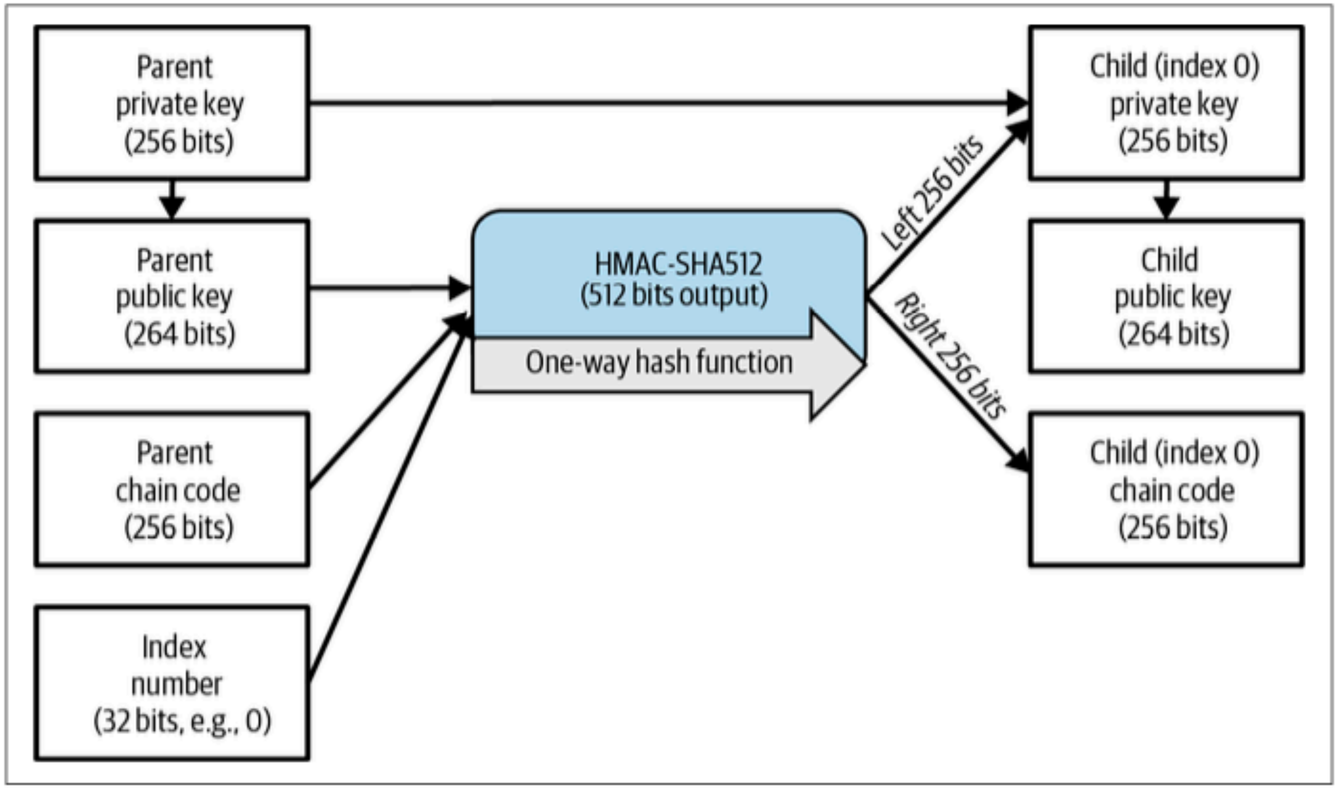


Generalization: Different addresses for different purposes (accounts, spending/change address, etc.)

BIP 32: HIERARCHICAL DETERMINISTIC WALLETS

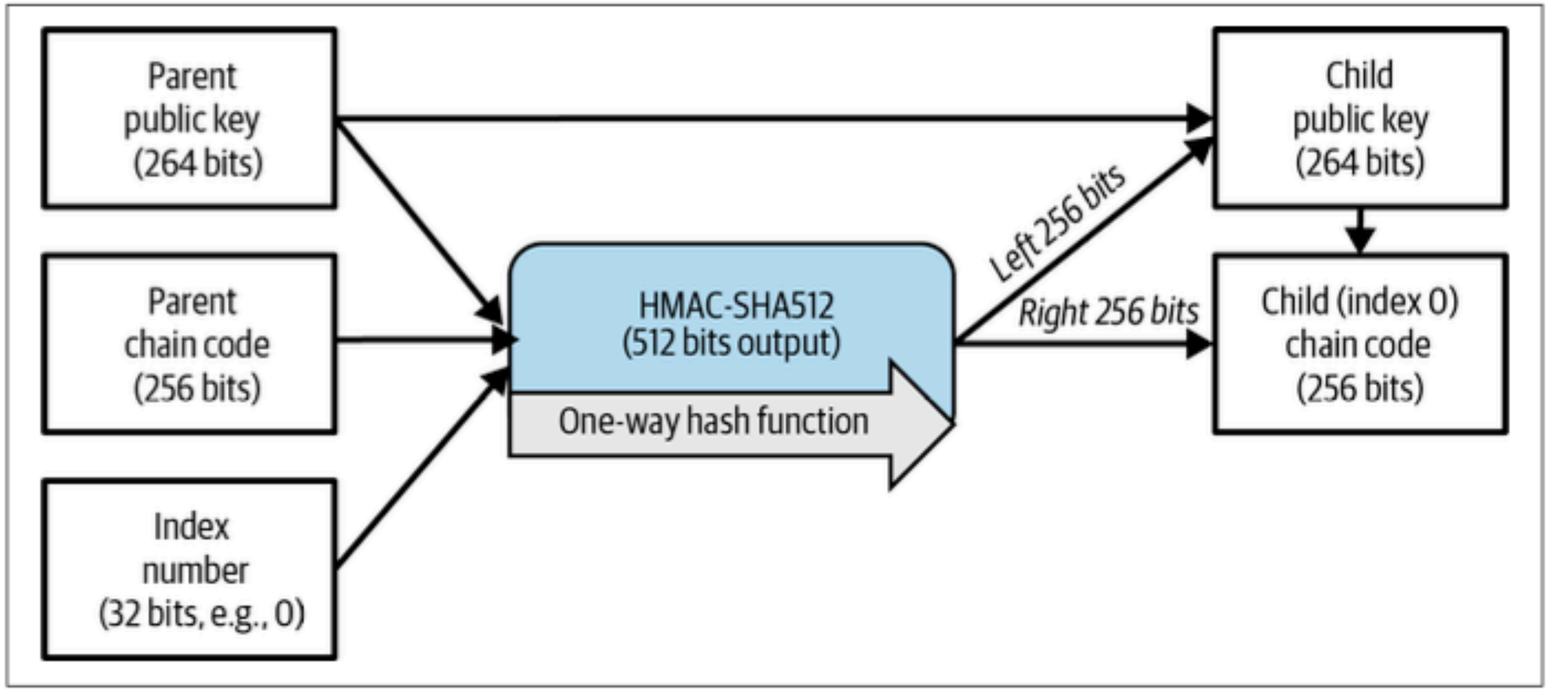
- Given a **master private key "m"** (256 bits), and
- a **master chain code "c"** (256 bits),

Derive child private key (256 bits) and child chain codes (256 bits) which can themselves serve as parent private key/ parent chain codes.

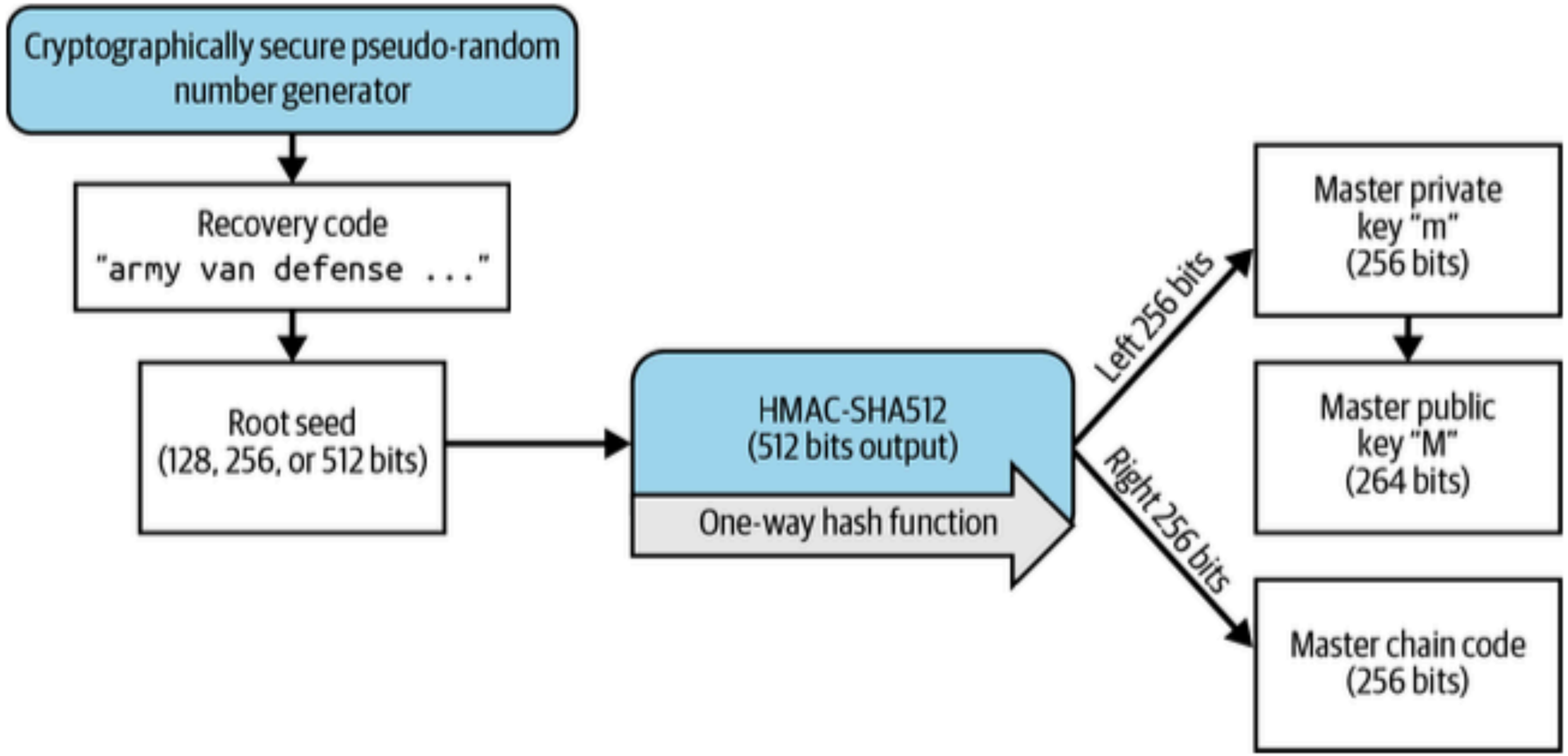


BIP 32: HIERARCHICAL DETERMINISTIC WALLETS

Same ability for parent public keys and parent chain code pairs.



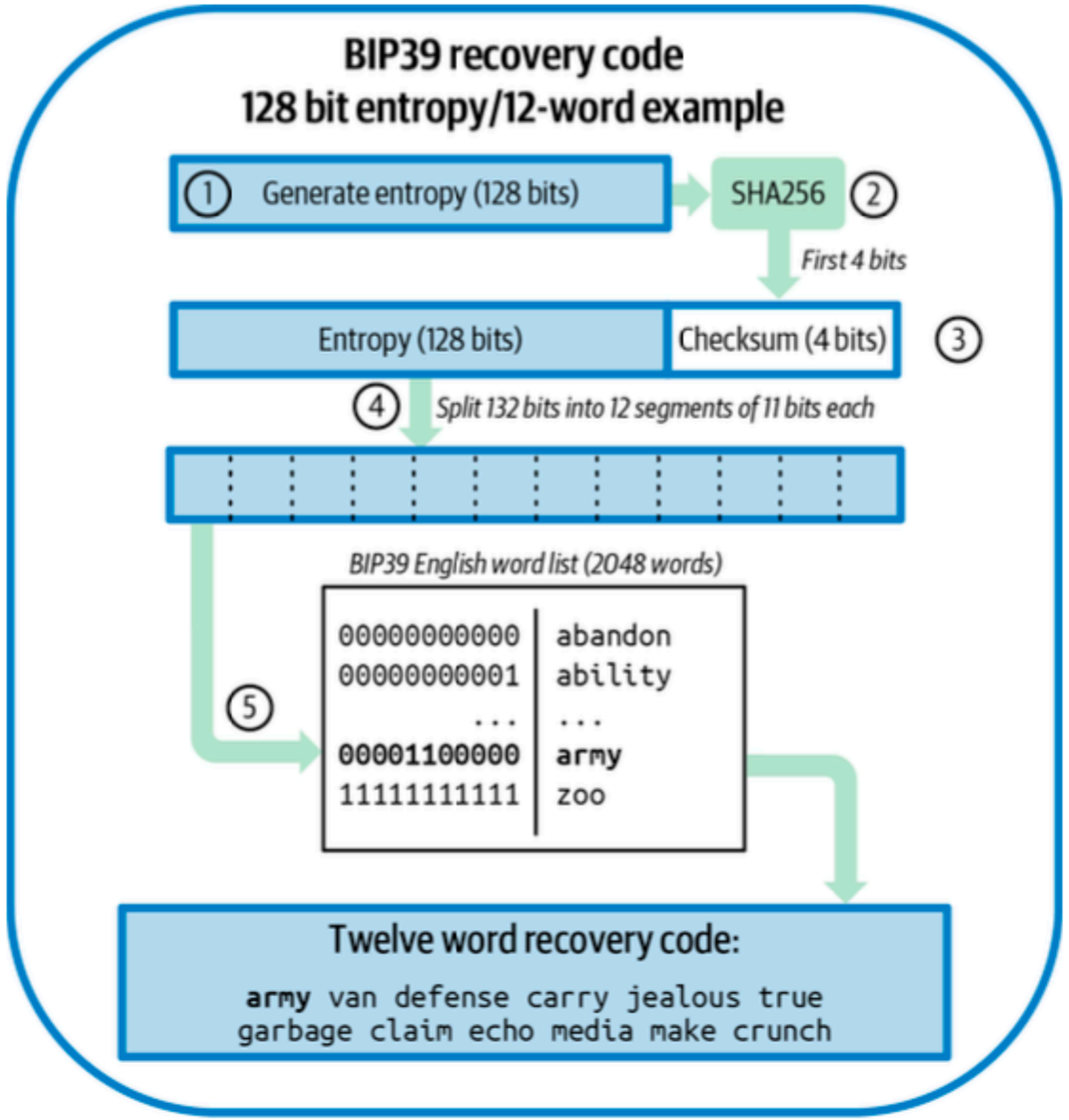
HIERARCHIC DETERMINISTIC KEY GENERATION AS IN BIP32



HMAC-SHA512

- Hash-based message authentication code
- “Keyed” variant of a cryptographic hash function (here: SHA-512)

BIP 39: MNEMONIC CODE FOR GENERATING DETERMINISTIC KEYS



BIP 39: MNEMONIC CODE FOR GENERATING DETERMINISTIC KEYS

Table 5-4. BIP39: entropy and word length

Entropy (bits)	Checksum (bits)	Entropy + checksum (bits)	Recovery code words
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24