Blockchain Programing

Revision: 2025

Slides: Henri LIEUTAUD, reworked & updated by Nathan HERVIER

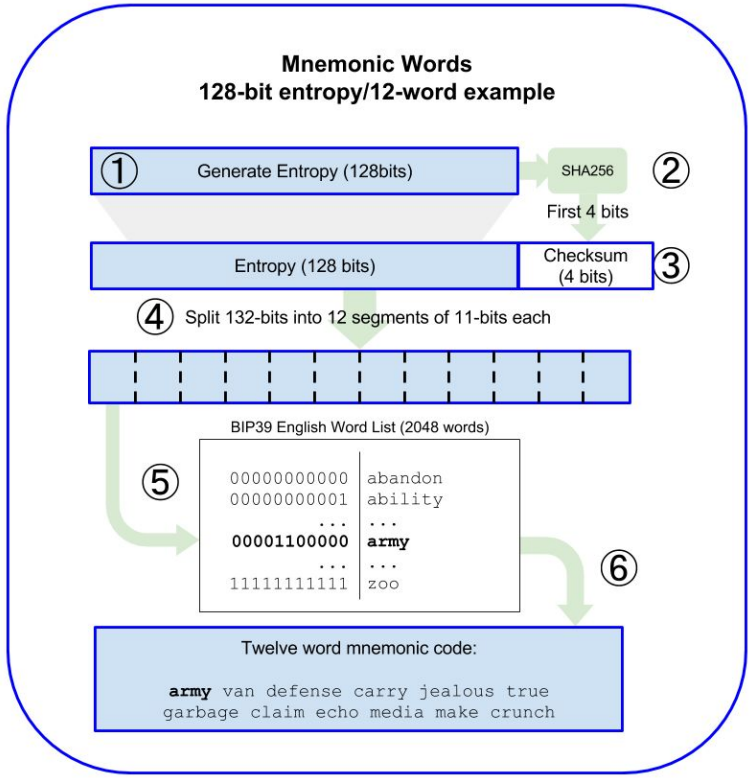
Today's agenda+

- Wallet management
- BIP 39
- BIP 32

BIP 39

Seeds & Mnemonics BIP 39

- The seed of a wallet is a random number that allows the generation of the wallet's keys.
- To facilitate the generation and storage of this seed, BIP39 specifies a mechanism to represent the seed with a set of specific words.
- Different standards and dictionaries are used by different cryptocurrencies and wallets.



Seeds & Mnemonics BIP 39

- https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki
- https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch05.asciidoc

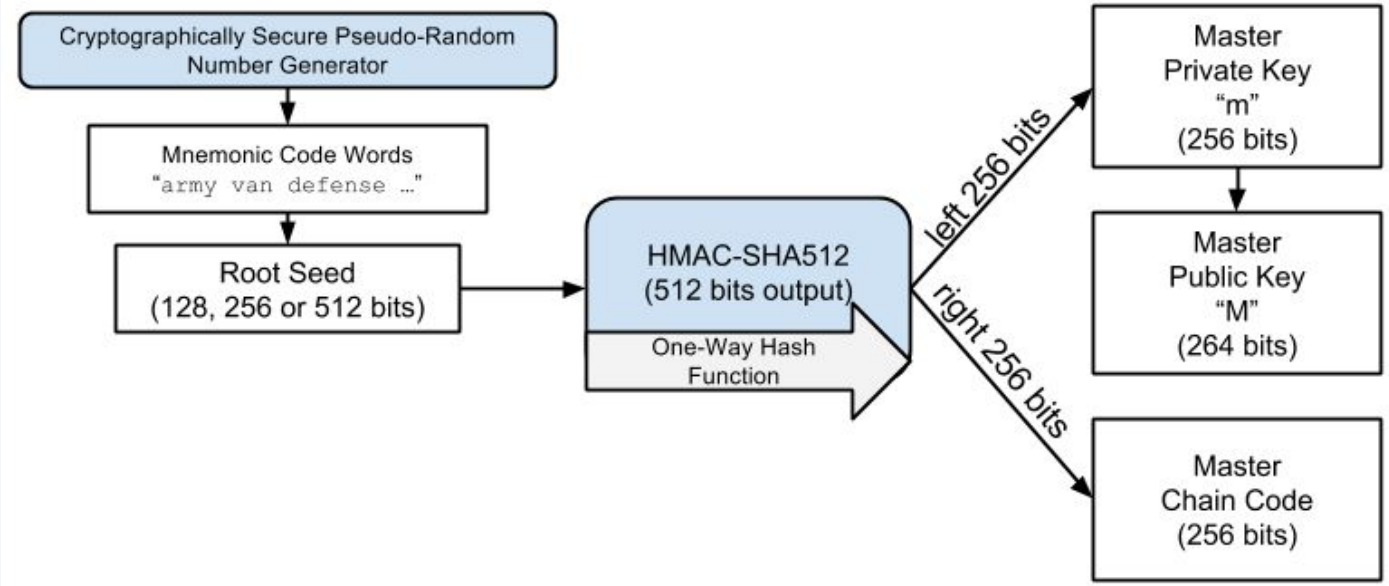
BIP 32

HD Wallets structure BIP 43/44

Recovery of a wallet from its seed:

- Two pieces of information are generated:
  - Master private key, to manage funds with an initial address.
  - Master chain code, to introduce entropy and generate subsequent keys.

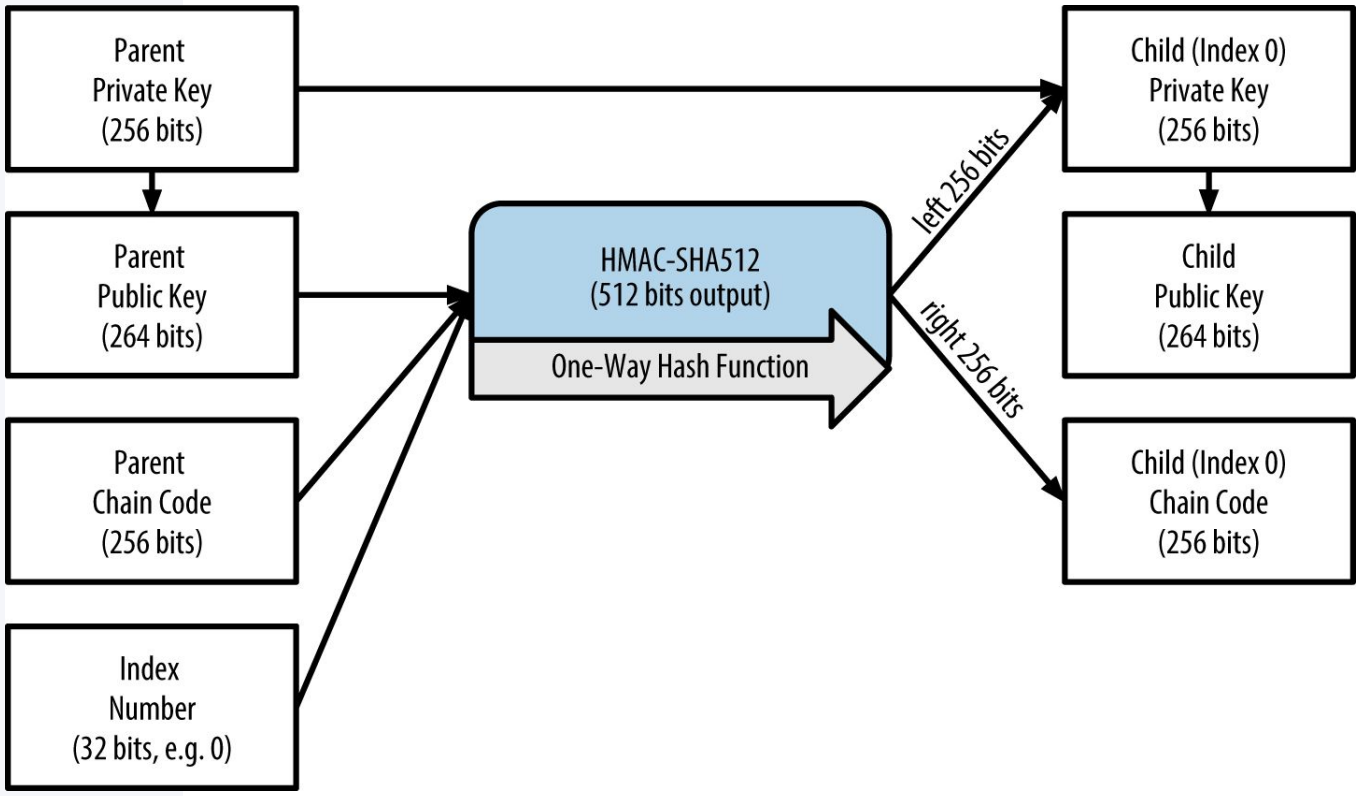
- Extended public key: Private/public key + chain code.



HD Wallets structure BIP 43/44

Child Key Generation

- Each generation has attributes that allow for the creation of a subsequent generation.
- It's possible to generate the tree from an Xpub, in a secure manner (only the pubKeys/addresses).



HD Wallets structure BIP 43/44

- https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki
- https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch05.asciidoc

Tasks list

BIP 39

- Create a GitHub repository and share it with me.
- Create an interactive Python, Rust, TS or JS command-line program (2pts).
- Generate a random integer that can serve as a safe seed for a wallet (2pts).
- Represent this seed in binary/bytes/hex and divide it into lots of 11 bits (2pts).
- Assign a word to each lot according to the BIP 39 list and display the seed in mnemonic form (2pts).
- Allow the import of a mnemonic seed (2pts).
- Verify the keys you generate on https://iancoleman.io/bip39/.

BIP 32

- Extract the master private key and the chain code (2pts).
- Extract the master public key (2pts).
- Generate a child key (2pts).
- Generate a child key at index N (2pts).
- Generate a child key at index N at derivation level M (2pts).

Contraintes

- You can do it in groups
- Python, Rust, TS or JS
- Code executable from the command line.
- No precompiled imported libraries (e.g., Bitcoin lib), just math (HMAC 256, ECDSA, etc).
- Importing libraries is ok to generate public keys from private keys, and for code verification.
- Instructions to run you program in your readme.md.
- write you report in the readme.md

Ressources utiles

- https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06.asciidoc
- Electrum