mobilefish.com

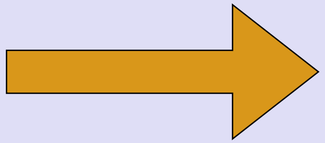
IOTA TUTORIAL 33

Restore IOTA Seed From Ledger Nano S Recovery Phrase

MY RECOVERY PHRASE

A list of 24 words will be displayed on your device when it is initialized. Make sure to copy each word below, it is a full backup of your accounts and configuration.

1. blast	13. genre
2. derive	14. elephant
3. number	15. dinosaur
4. just	16. roof
5. topic	17. raccoon
6. write	18. unfold
7. man	19. health
8. dinner	20. stereo
9. access	21. three
10. crouch	22. place
11. staff	23. cheap
12. arctic	24. rhythm



```
GHSGIU9SMCXPMBLFSKHV
DKHDKXPRKDVROTLKYOWZ
NIGJFTSA9GYWJVRJTCNH
YMSNJUNRAMYBIXQAFDDXW
```

INTRO

- In this tutorial I will demonstrate how to restore the IOTA seed using the Ledger Nano S recovery phrase.

WARNING

- **The purpose of a Ledger Nano S device is that seeds and private keys NEVER leaves the device.**

- **Restoring the IOTA seed from the recovery sheet does in fact the opposite and is therefore NOT recommended.**

- **If you do restore the IOTA seed from the recovery sheet, your recovery phrase and IOTA seed gets exposed outside the Ledger device and you create a security risk.**
  Your recovery phrase and/or extracted seed may get stolen and thus all your IOTA funds.

TWO METHODS TO RECOVER THE IOTA SEED

- In this tutorial I will demonstrate two methods to recover an IOTA seed from a Ledger Nano S recovery phrase.

- The first method is a python application created by the developer MuXxer, see: https://github.com/muXxer/recover-iota-seed-from-ledger-mnemonics

- The second method is a web application created by Mobilefish (Robert Lie) and it is based on the work created by MuXxer, see: https://github.com/robertlie/ledger_recovery_words_to_iota_seed

- **Use both applications at your own risk!**

METHOD 1: MUXXER PYTHON APPLICATION

- The muXxer python application uses the following libraries:
  See: https://github.com/muXxer/recover-iota-seed-from-ledger-mnemonics/blob/master/requirements.txt
  Use the https://pypi.org/ to get more information about these libraries.

  - mnemonic - Implementation of Bitcoin BIP-0039.
    More info: https://github.com/trezor/python-mnemonic

  - bip32utils - Provides a small set of utilities for generating Bitcoin Hierarchical Deterministic Wallet addresses using BIP0032.
    More info: https://github.com/muXxer/bip32utils

  - PyOTA - The IOTA Python API Library.
    More info: https://github.com/iotaledger/iota.lib.py

METHOD 2: MOBILEFISH WEB APPLICATION

- The Mobilefish web application uses the following JavaScript libraries:

  - jsBIP39 - Implementation of Bitcoin BIP-0039.
    The python mnemonic library has been ported to JavaScript and is called jsBIP39
    More info: https://github.com/iancoleman/jsbip39
    Version: Latest commit 01be853 on 8 May
    This library works in a browser and does not need to be browserified.

  - bip32-utils - Provides a small set of utilities for use with BIP32 HD key nodes.
    More info: https://github.com/bitcoinjs/bip32-utils
    Version: 0.11.1
    It is a Node.JS module. How this module is browserified is explained in
    https://www.mobilefish.com/developer/nodejs/nodejs_quickguide_browserify_bip32_utils.html

METHOD 2: MOBILEFISH WEB APPLICATION

- iota.lib.js - The IOTA JavaScript library.
  I am not using the new iota.js (beta, Nov 2018) Typescript port of iota.lib.js.
  More info: https://github.com/iotaledger/iota.js/releases/tag/v0.5.0
  Version: 0.5.0
  This library works in a browser and does not need to be browserified.

- Kerl - A hashing function, based on Keccak, with conversion to ternary.
  More info: https://github.com/iotaledger/kerl
  Version: Latest commit 7ca94d3 on 22 Sep 2017
  It is a Node.JS module. How this module is browserified is explained in
  https://www.mobilefish.com/developer/nodejs/nodejs_quickguide_browserify_kerl.html

HOW IT WORKS

- In the following slides a technical explanation will be given, how the IOTA seed is restored using the Ledger Nano S recovery phrase.

- I assume you know what BIP-32, BIP-39 and BIP-44 are.
  If not, please watch these two blockchain tutorials:

  - Blockchain tutorial 28: Bitcoin Improvement Proposal 39 (BIP-39) mnemonic words
    https://youtu.be/hRXcY_tIlrw

  - Blockchain tutorial 29: Hierarchical Deterministic wallet - BIP32 and BIP44
    https://youtu.be/2HrMlVr1QX8

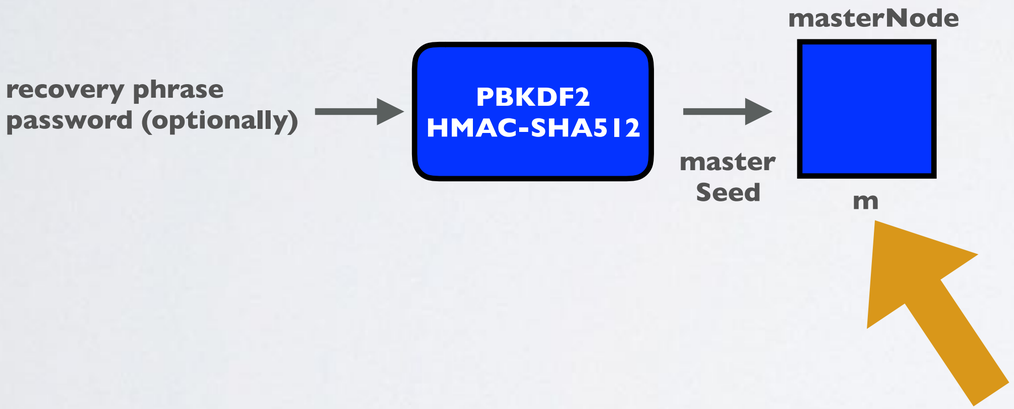
HOW IT WORKS

- Based on this code https://github.com/robertlie/ledger_recovery_words_to_iota_seed/blob/master/iota_seed_mnemonic_words.html I will explain how it all works.

- The recovery phrase and optional password (=salt) are used as input for the Password-Based Key Derivation Function 2 (PBKDF2) which outputs a 512 bits seed:
  See: https://www.mobilefish.com/download/blockchain/blockchain_mnemonic_words_bip39_part28.pdf#page=17
  const masterSeed = m.toSeed(words, passphrase);

- Create a masterNode.
  See: https://www.mobilefish.com/download/blockchain/blockchain_hd_wallets_part29.pdf#page=20
  const masterNode = Bip32JS.bitcoin.HDNode.fromSeedHex(masterSeed);

HOW IT WORKS



The letter m denotes the master node.

HOW IT WORKS

- The IOTA Trinity Wallet implements BIP44 to generate accounts and addresses.

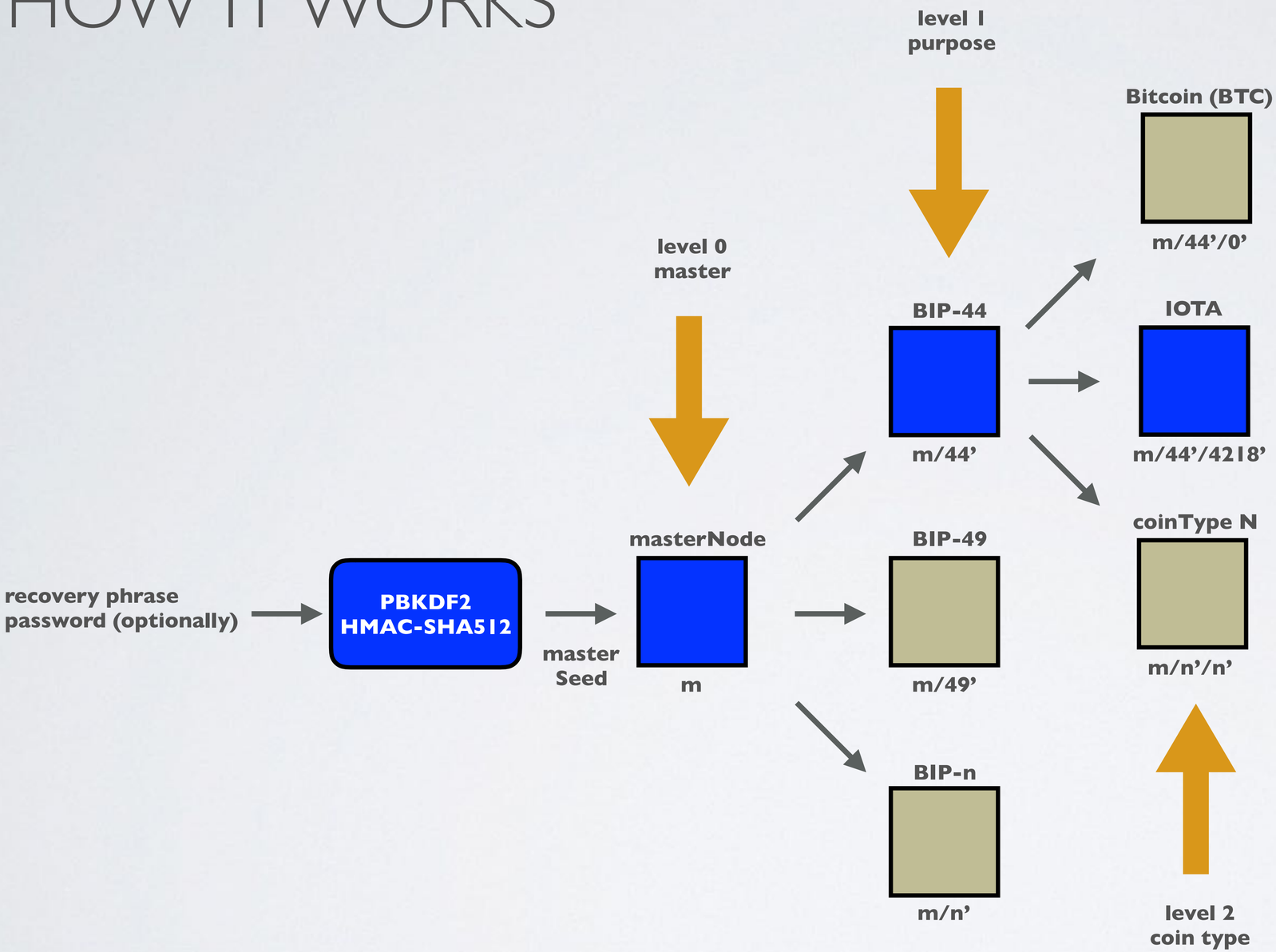
- BIP-44 defines a standard method for deriving many keys for different uses from a single mnemonic seed in an organised way.

- BIP-44 uses the following derivation path:
  m/purpose'/coin_type'/account'/change'

- purpose = 44', referring to BIP-44.
  coin_type = 4218' (IOTA), 0' (Bitcoin)
  The registered coin types can be found at:
  https://github.com/satoshilabs/slips/blob/master/slip-0044.md

- The apostrophe indicates hardened derivation.

HOW IT WORKS


level 1
purpose
Bitcoin (BTC)
m/44'/0'
level 0
master
BIP-44
IOTA
m/44'
m/44'/4218'
recovery phrase
password (optionally)
PBKDF2
HMAC-SHA512
masterNode
BIP-49
coinType N
master
Seed
m
m/49'
m/n'/n'
BIP-n
m/n'
level 2
coin type

HOW IT WORKS

- In https://github.com/satoshilabs/slips/blob/master/slip-0044.md
  the IOTA hexa value = 0x8000107a

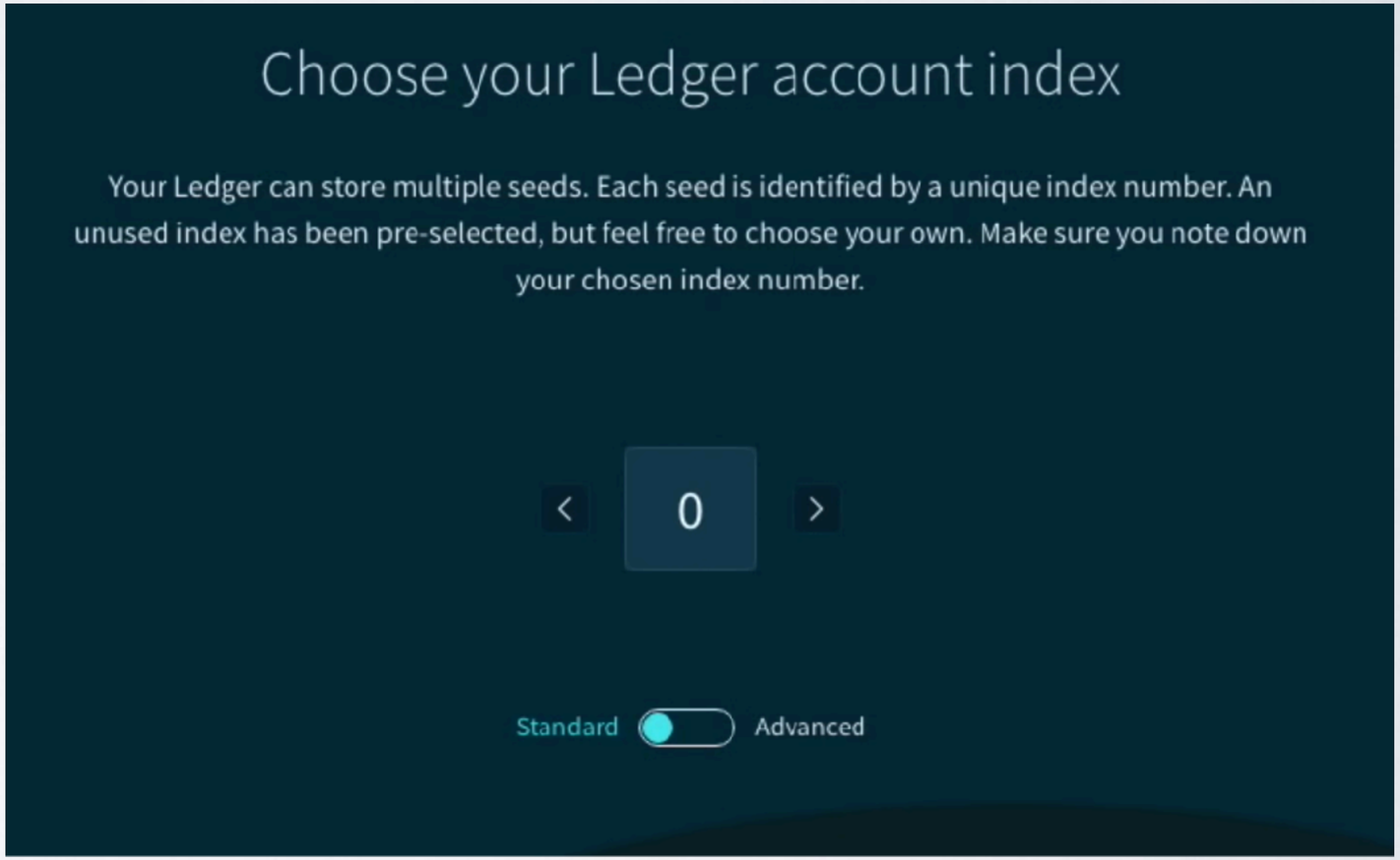
- This constant value is a "hardened" value, which means:
  IOTA hexa value = 0x80000000 + 0x107a

- This is the same as:
  4218' = 0x8000107a = 0x80000000 + 0x107a
  4218' = 2^{31} + 4218

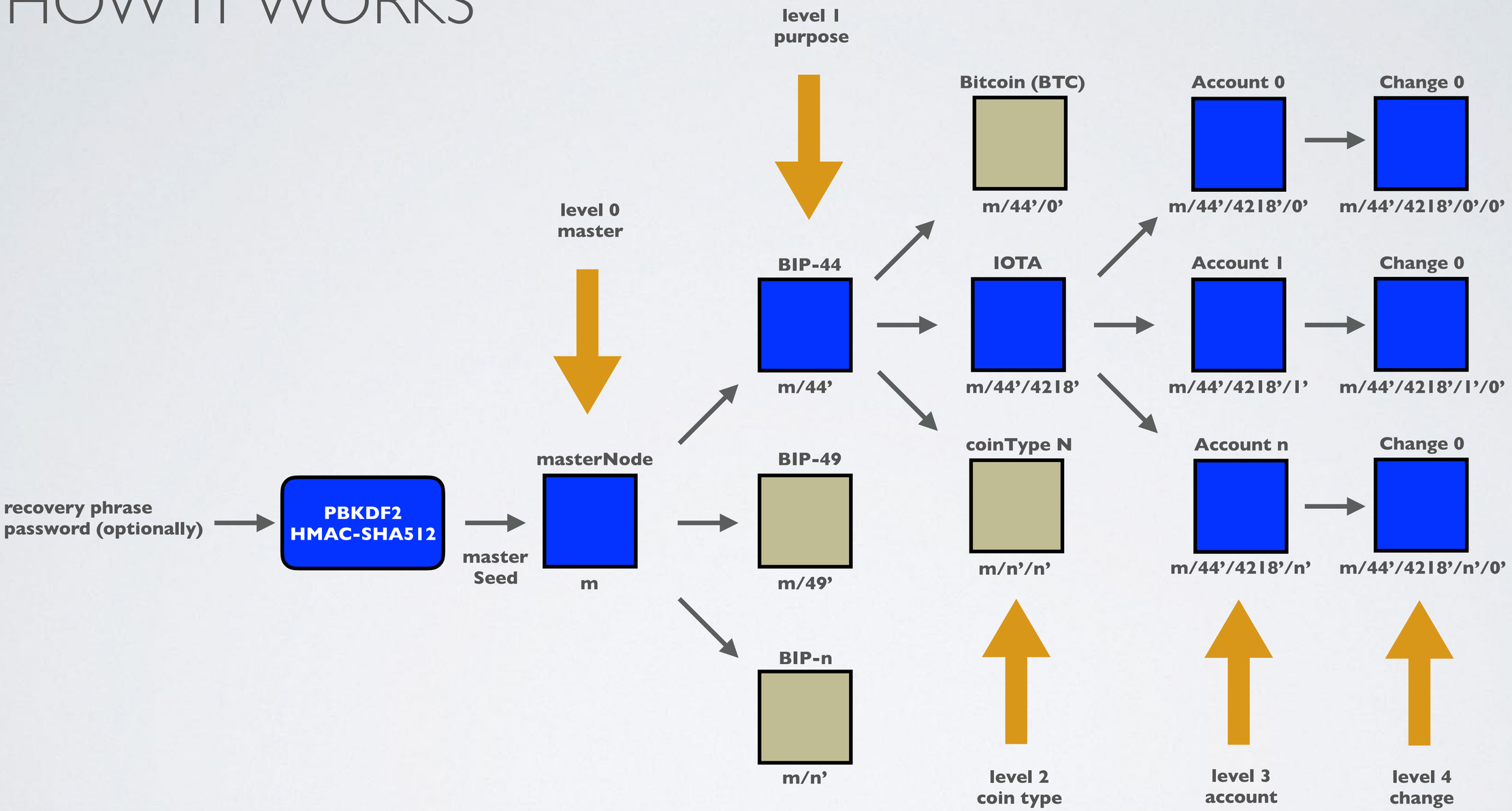
HOW IT WORKS

- **m/44’/4218’/account’/change’**
  account = Ledger Account Index

- The change level is 0’ because the addresses are meant to be visible outside the wallet.
  change = 0’



HOW IT WORKS

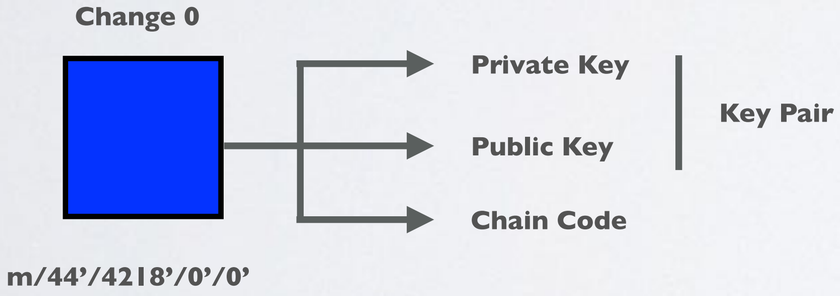

level 1
purpose
Bitcoin (BTC)
Account 0
Change 0
m/44'/0'
m/44'/4218'/0'
m/44'/4218'/0'/0'
level 0
master
BIP-44
IOTA
Account 1
Change 0
m/44'
m/44'/4218'
m/44'/4218'/1'
m/44'/4218'/1'/0'
recovery phrase
password (optionally)
PBKDF2
HMAC-SHA512
masterNode
BIP-49
coinType N
Account n
Change 0
master
Seed
m
m/49'
m/n'/n'
m/44'/4218'/n'
m/44'/4218'/n'/0'
BIP-n
m/n'
level 2
coin type
level 3
account
level 4
change

HOW IT WORKS

- The IOTA wallet uses the following derivation path m/44'/4218'/[0-n]'/0'

```
const path = "m/44'/4218'/"+ledgerAccountIndex+"'/0'";
const node = masterNode.derivePath(path);
```

- Each node contains three pieces of information: a private key, a public key, and a chain code. The chain code is to prevent someone from determining the children of a node using only the node's public and private keys.
  See: https://www.mobilefish.com/download/blockchain/blockchain_hd_wallets_part29.pdf#page=10

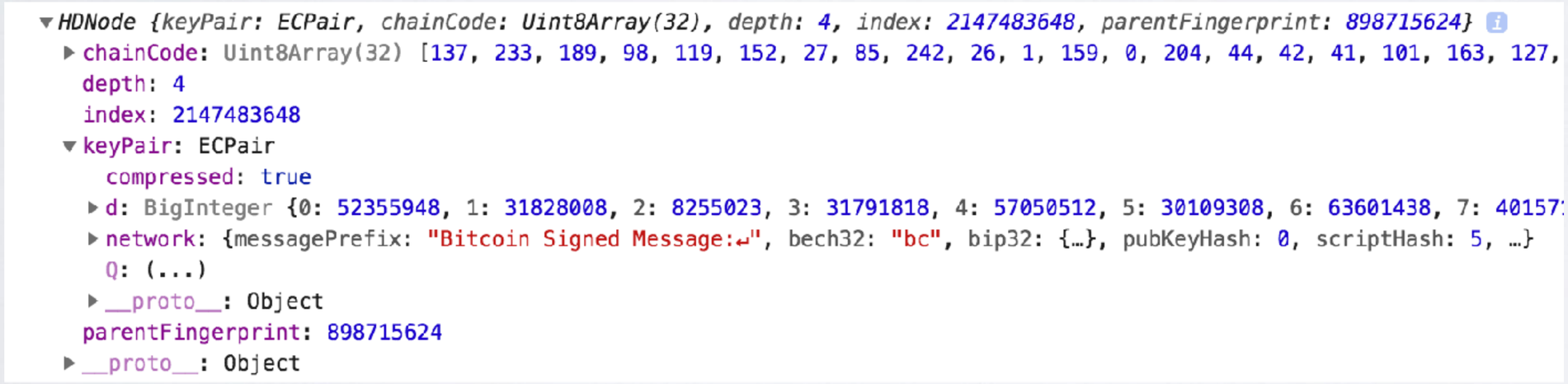


HOW IT WORKS

- Get the node chainCode.
  const chainCode = node.chainCode;

- Get the node key pair.
  const keyPair = node.keyPair;

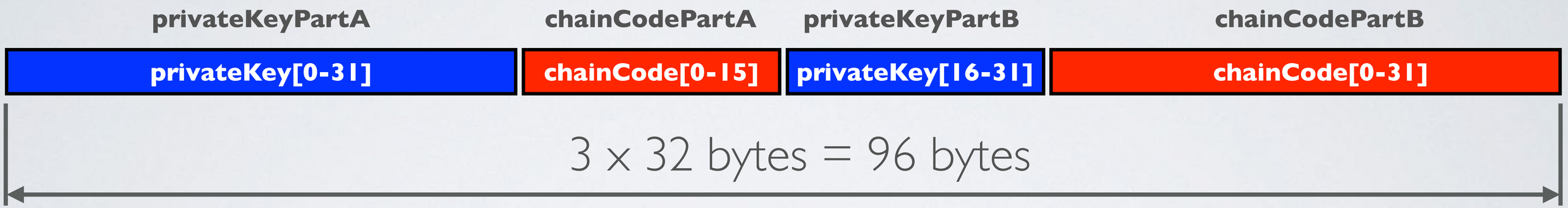
- Extract the private key from the key pair.
  const privateKey = keyPair.d.toBuffer(32);



See:
iota_seed_mnemonic_words.html
(debug = true)

HOW IT WORKS

- Create a new byte array called "dataBytes":




```
const privateKeyPartA = privateKey.slice(0, 32);
const privateKeyPartB = privateKey.slice(16, 32);
const chainCodePartA = chainCode.slice(0, 16);
const chainCodePartB = chainCode.slice(0, 32);

let dataBytes = new Uint8Array(privateKeyPartA.byteLength +
chainCodePartA.byteLength + privateKeyPartB.byteLength +
chainCodePartB.byteLength);
```

HOW IT WORKS

```
dataBytes.set(new Uint8Array(privateKeyPartA),0);

dataBytes.set(new Uint8Array(chainCodePartA), privateKeyPartA.byteLength);

dataBytes.set(new Uint8Array(privateKeyPartB),
(privateKeyPartA.byteLength+chainCodePartA.byteLength));

dataBytes.set(new Uint8Array(chainCodePartB),
(privateKeyPartA.byteLength+chainCodePartA.byteLength+
privateKeyPartB.byteLength));
```

For example:
dataBytes = [31, 96, 121, 61,153,...]
dataBytes = [00011111, 01100000, 01111001, 00111101, 10011001,...]
dataBytes is an array consisting of 96 bytes (3 x 32 bytes = 768 bits)

HOW IT WORKS

- Create a Kerl instance.
  const kerl = new KerlJS.Kerl();

- Convert dataBytes (96 bytes) to a CryptoJS compatible word array.
  Each word consists of 32 bits.
  wordArr is an array consisting of 24 words (24 x 32 bits = 768 bits)
  const wordArr = KerlJS.Converter.wordsFromBytes(dataBytes);

  For example:
  wordArr = [00011111101100000001111001 00111101, 10011001…]

HOW IT WORKS

- Create a CryptoJS wordArray object.
  More information about CryptoJS, see IOTA Tutorial 21.
  const wordArrObj = KerlJS.CryptoJS.lib.WordArray.create(wordArr, dataBytes.length);

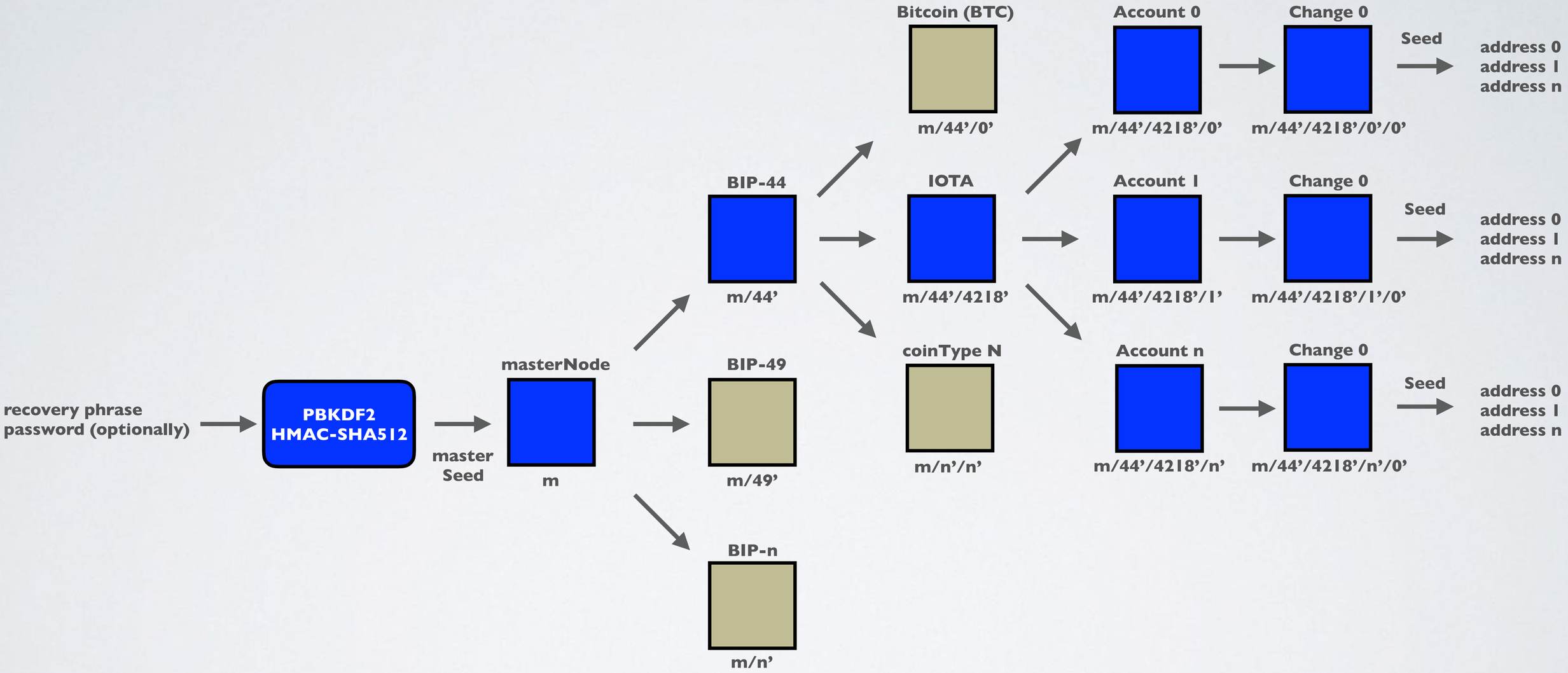
- kerl.k executes CryptoJS.algo.SHA3.create() which creates a sha3 instance.
  kerl.k.update reads in the CryptoJS wordArray object that will be SHA3 hashed.
  PLEASE NOTE: CryptoJS.algo.SHA3 implemented Keccak and NOT SHA3.
  kerl.k.update(wordArrObj);

- Create the IOTA seed
  let trits_out = [];
  kerl.squeeze(trits_out,0,kerl.HASH_LENGTH);
  const iota_seed = KerlJS.Converter.trytes(trits_out);

HOW IT WORKS

- With the IOTA seed, the addresses can be calculated.
  More information how addresses are calculated, see IOTA Tutorial 9.1

HOW IT WORKS


recovery phrase
password (optionally)
PBKDF2
HMAC-SHA512
master
Seed
masterNode
m
BIP-44
m/44'
BIP-49
m/49'
BIP-n
m/n'
Bitcoin (BTC)
m/44'/0'
IOTA
m/44'/4218'
coinType N
m/n'/n'
Account 0
m/44'/4218'/0'
Account 1
m/44'/4218'/1'
Account n
m/44'/4218'/n'
Change 0
m/44'/4218'/0'/0'
Change 0
m/44'/4218'/1'/0'
Change 0
m/44'/4218'/n'/0'
Seed
address 0
address 1
address n
Seed
address 0
address 1
address n
Seed
address 0
address 1
address n

HOW IT WORKS

MY RECOVERY PHRASE

A list of 24 words will be displayed on your device when it is initialized. Make sure to copy each word below, it is a full backup of your accounts and configuration.

1. blast	13. genre
2. derive	14. elephant
3. number	15. dinosaur
4. just	16. roof
5. topic	17. raccoon
6. write	18. unfold
7. man	19. health
8. dinner	20. stereo
9. access	21. three
10. crouch	22. place
11. staff	23. cheap
12. arctic	24. rhythm

account index 0

Always write down you Ledger Account Index