Building your own

Hardware wallet

crypto advance.

@StepanSnigirev

stepan@cryptoadvance.io

Workshop plan

Overview

- Why DIY makes sense
- QR code & SD card demo
- Hardware platforms and architectures
- Available languages and frameworks
- Bitcoin libraries for embedded devices

Key management

- Mnemonic phrase (BIP-39)
- Hierarchical Deterministic Wallets (BIP-32)
- Address types (BIP-44, BIP-49, BIP-84)

Transactions

- Segwit vs Legacy
- Partially Signed Bitcoin Transactions (BIP-174)
- Change detection and other checks

Overview

Why DIY makes sense

Security

- Better security in multisig
- No supply chain attacks
- Reduced risk of mass attacks
- Anti-tamper measures

User experience

- Any communication channel
- Custom authentication
- Screen, form-factor, interface

Functionality

- Check multisig with external xpubs
- Custom scripts (timelocks)
- Chosen nonce attack mitigations

Toolbox

- Physical RNG with dices
- Watch-only ePaper wallet
- Experiments with new tech (CoinJoin, LN)

Overview

Why DIY makes sense

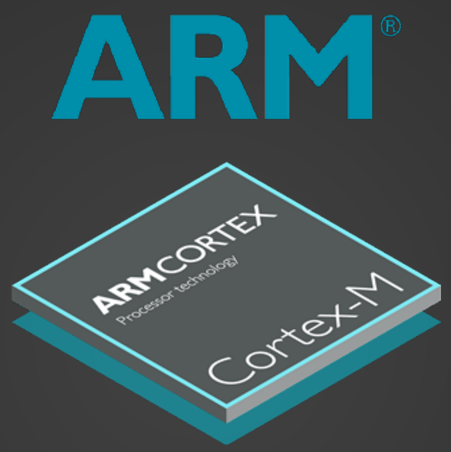
How I use it

Demo time!

Airgapped hardware wallet

Overview

Platforms and architectures



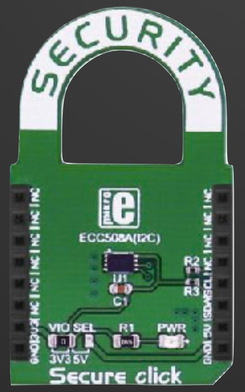
Overview

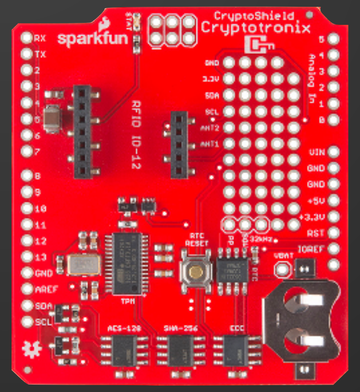
Secure hardware for tinkerers

Infineon

MICROCHIP

crypto advance.





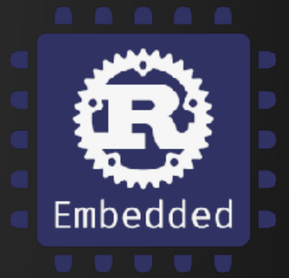
Blockchain to go :(

work in progress

Overview

Languages and frameworks

MicroPython



Overview

Bitcoin libraries

trezor-crypto

secp256k1

libwally

work in progress

micro-bitcoin (?)

Let’s get started!

Setting everything up

mbed.com

https://os.mbed.com/platforms/ST-Discovery-F469NI/

https://os.mbed.com/users/stepansnigirev/code/workshop_template/

Mnemonic phrase

Recovery seed, BIP-39

Take a random number:

f34b3e256b8b8bb9cf2f3e73e423521a

Convert to binary, add checksum:

11110011010 01011001111 10001001010 11010111000 10111000101 11011100111
00111100101 11100111110 01110011111 00100001000 11010100100 0011010**1100**

Split, convert to words:

viable fly matter strike reward table device treat initial canal stand **culture**

Mnemonic phrase

Password and master key derivation

Take the mnemonic:

viable fly matter strike reward table device treat initial canal stand culture

Hash it 2048 times with the passphrase:

PBKDF2(password = mnemonic, salt = “mnemonic”+password, 2048).read(64)

Use the result as master private key:

chain code: 93fb9d28d8f8e60f0298f638b1c7340bb014f708daca29d47535dc0339b1ebd1

private key: ab819774d0cf931676302cc3b79d5e01127e91472543be4e84ebc5f7ff5676e4

Mnemonic phrase

Problems and discussions

Depends on the dictionary:

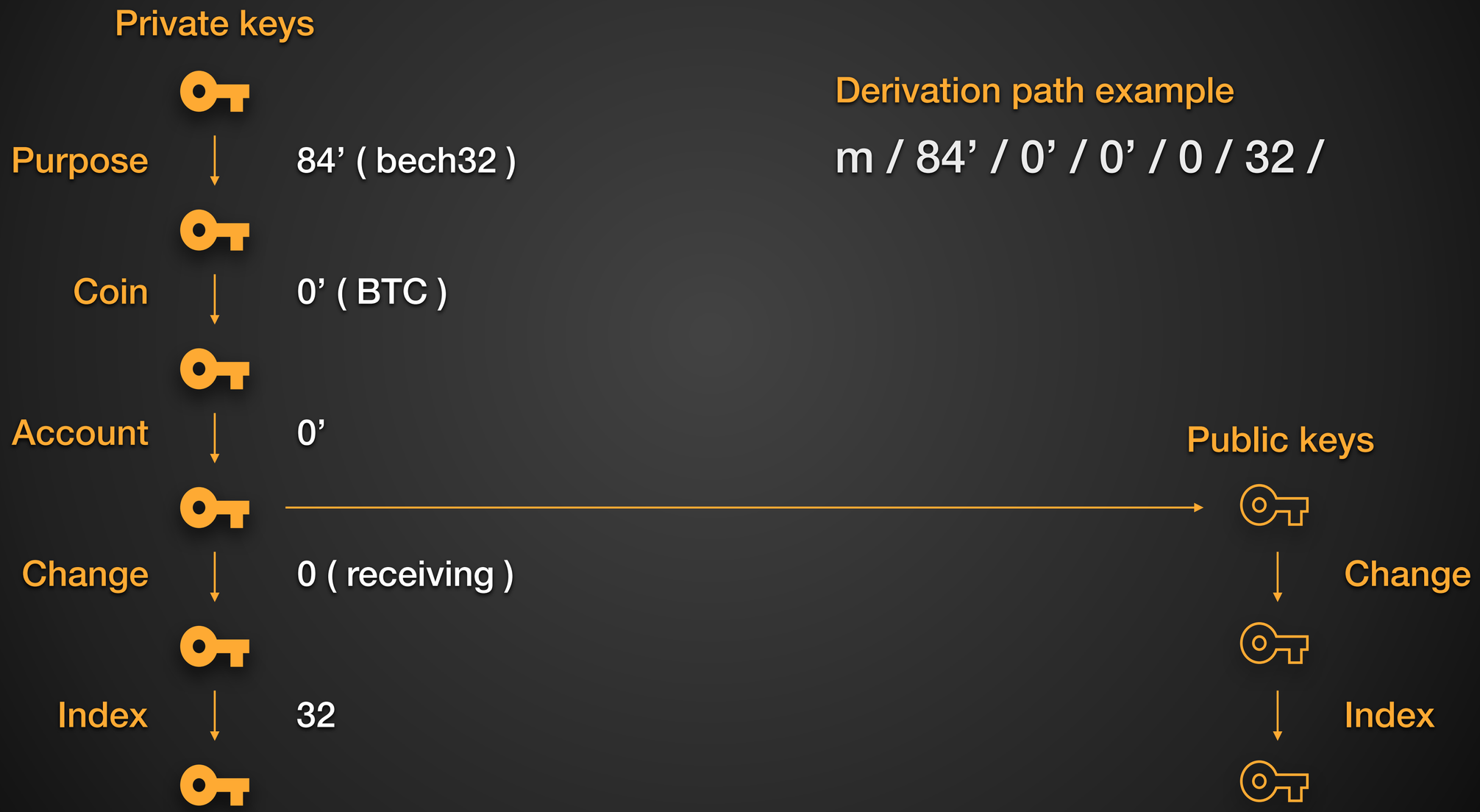
- Limited set of languages
- Only English is widely supported

Hash-based checksum:

- Impossible to generate by a human
- Checksum is based on entropy

HD wallets

Master keys: BIP-32, BIP-44, BIP-49, BIP-84

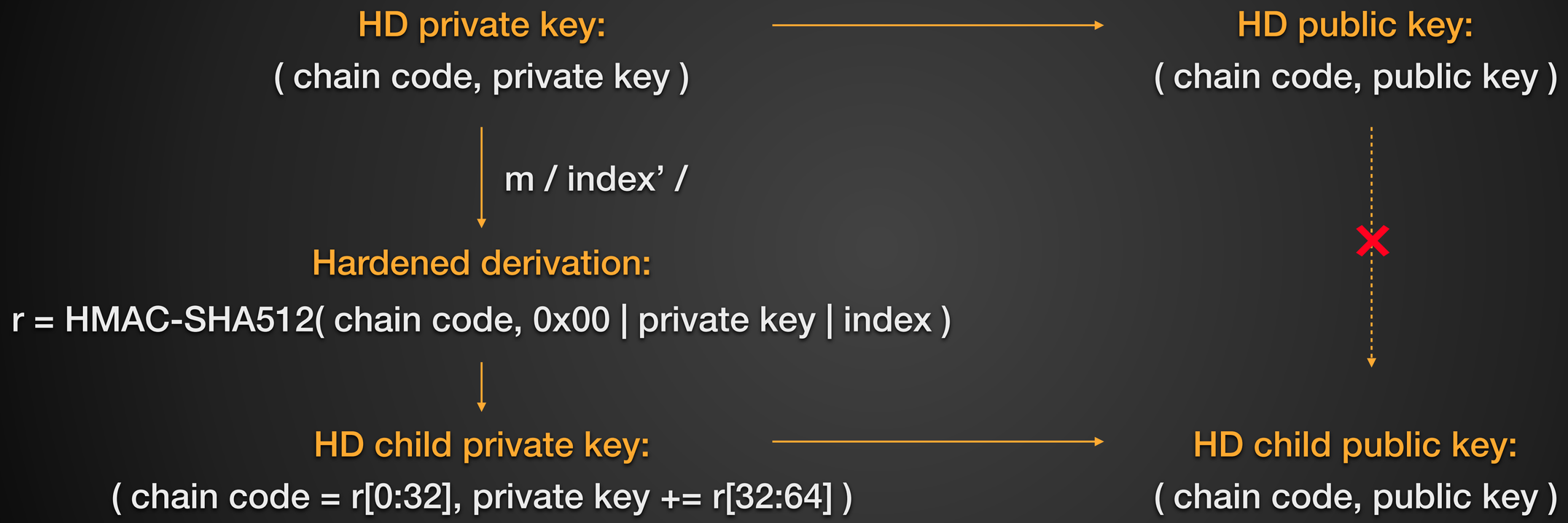


Derivation path example

m / 84’ / 0’ / 0’ / 0 / 32 /

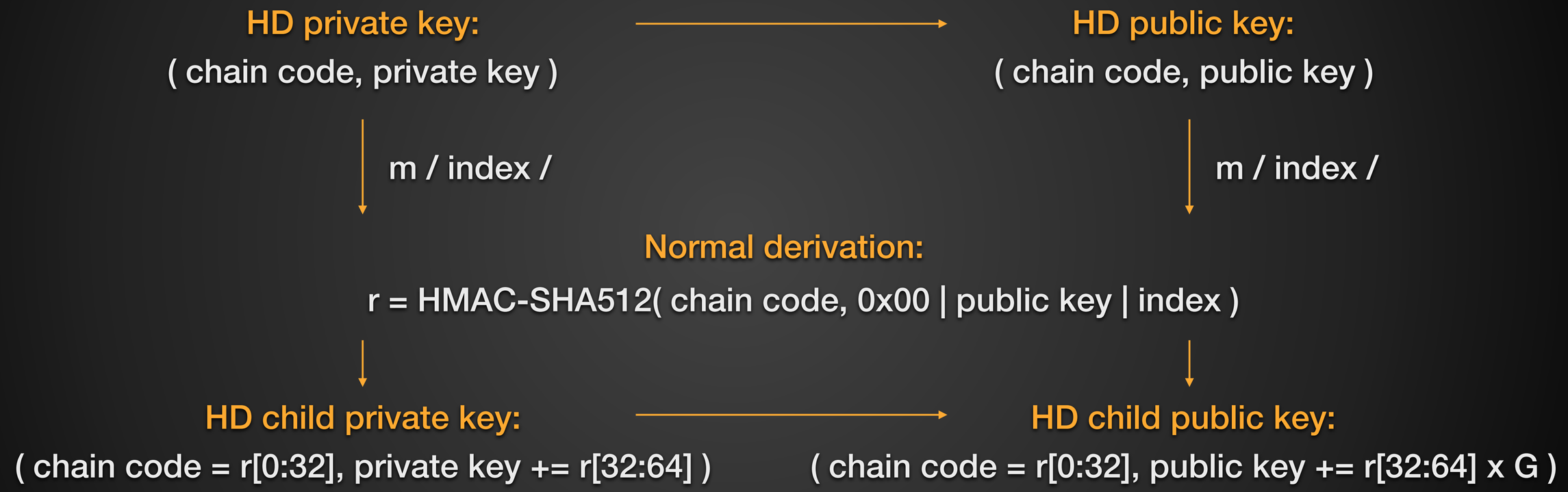
HD wallets

Hardened derivation



HD wallets

Non-hardened derivation



HD wallets

Master keys: BIP-32, BIP-44, BIP-49, BIP-84

< prefix > < depth > < parent fingerprint > < chain code > < key >

BIP 44 - legacy:

xprv 9zGySLGC7bW76AxZEsfgzWv…yQRXHChHXgejzT8XEZnQX731Ah1tzUjR

xpub 6DGKqqo5wy4QJf32LuChMer…T56mMcnQ2fQeh57DGA2mpFyyjMCf7SQ

BIP 49 - Nested segwit:

yprv AK7Ejzw7GH3awU9g5ETKCc1…Hi8qvPhYUtRbKUDf9hE1Nm4p5dnDMyh

ypub 6Y6b9WU16ebt9xE9BFzKZjxS…bLMiL5KCQpu6FZQodr4rG5z3d81Apgc1

BIP 84 - Native segwit:

zprv AdwW3fc2Qxb4nmLnubEwQh7…5AzkQe67ZH87CBov5jc3VeV8TU7yJJYp

zpub 6rvrTB8vFL9N1FRG1cmwmq3…eKto1cRd8mrRkW4RyfkMi7GWVRAETZ

Addresses

Legacy, nested segwit, native segwit

Private key: < prefix > < 32-byte secret > < compressed ?>

KwnbUMi7EMhPdeWn2i9fQasfFvjQRY3dvsDDKLYQkhP6aSfYP4Km

Public key: < even / odd y ?> < 32-byte x coordinate >

02763bb01ec889b77fb81871c88262ce0e34b1c56b5321163125aa5b6e3a46099f

Legacy address: < prefix > < hash160(pubkey) >

1FymoUmHWeCKvJQvtdeszMnHNkNLubmA5c

Nested segwit: < prefix > < hash160(segwit_version | hash160(pubkey)) >

3M8AuZSJUUmRgccjZjpQWVLJe4Yw93oVkp

Native segwit: < prefix > < segwit_version | hash160(pubkey) >

bc1q53888l7suc2d4a2586tykx7uzzga7rdpk5wjf8

Transactions

Segwit vs Legacy

Legacy transaction

< version > < N in > < input > … < input > < N out > < output > … < output > < lock time >

Legacy input

< previous tx > < output number > < script sig > < sequence number >

Segwit transaction

< version > < 0 > < segwit flag > < N in > < inputs > < N out > < outputs > < witnesses > < lock time >

Segwit input

< previous tx > < output number > < empty | hash(witness script) > < sequence number >

Transactions

Signing inputs

Legacy transaction

Depends on:

- raw transaction we are signing
- script of previous output

Goes to: scriptsig of input

Segwit transaction

Depends on:

- raw transaction we are signing
- script of previous output
- amount of previous output

Goes to: witness (not in transaction id)

Transactions

BIP-174: Partially Signed Bitcoin Transaction (PSBT) format

Global scope:

- raw transaction we are signing
- list of extended public keys (for multisig)

For every input:

- derivation path
- previous outputs (script and amount)
- redeem script (if any)

For every change output:

- derivation path
- redeem script