Binary → 24 words

256 bits break up into 11 bits

01010101010, ..., ..., ... 23 times + 101
36 bits for 24th word

⇓

decimal #, 0–2048

⇓

find dec # on list
of 2049 words

example 10110111101 = 1469 = response

⇓

find first 23 words

⇓

sha256 the binary & take the first 8 bits
of the result to append to make the
last word 256+8 = 264 bits / 11 = 24 words

Converting Words → Public key

Entropy — 256 bits ⇄ 24 words

↓

"mnumonic" + passphrase + { empty str if no passphrase }

↓ salt

PBKDF2
HMAC-SHA512

2048 iterations
length = 64 bytes

Master Seed

"Bitcoin Seed" in hex format

↓ key ↓ data

HMAC
SHA512

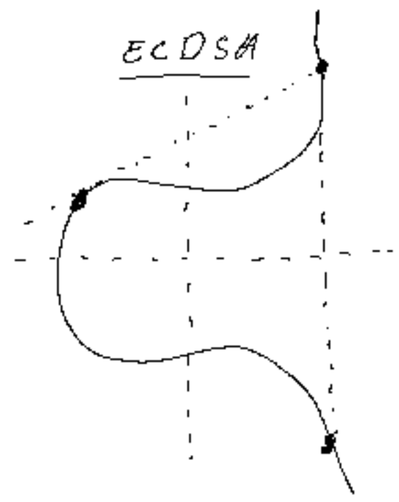
↓

Ext. Priv Key

Mast. Priv. Key [64 bytes] | Mas. Chain. Code [64 bytes]

↓

ECDSA ⇒ x & y if y is odd → "03", if y is even → "02"



↓

Ext. Public Key

"04" + public key

uncompressed public key

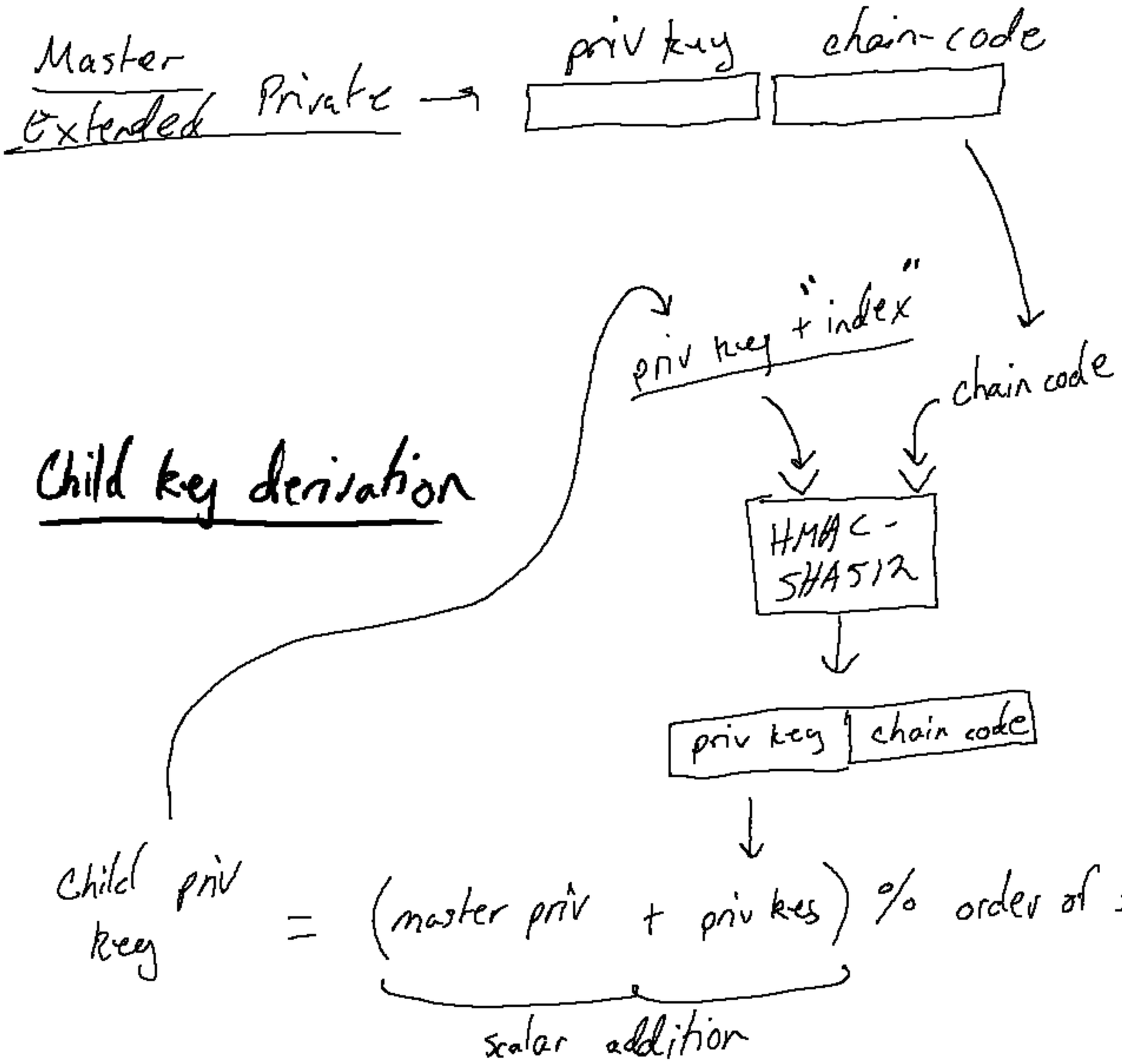
uncompressed pub key

"03" or "02" + [64 bytes]

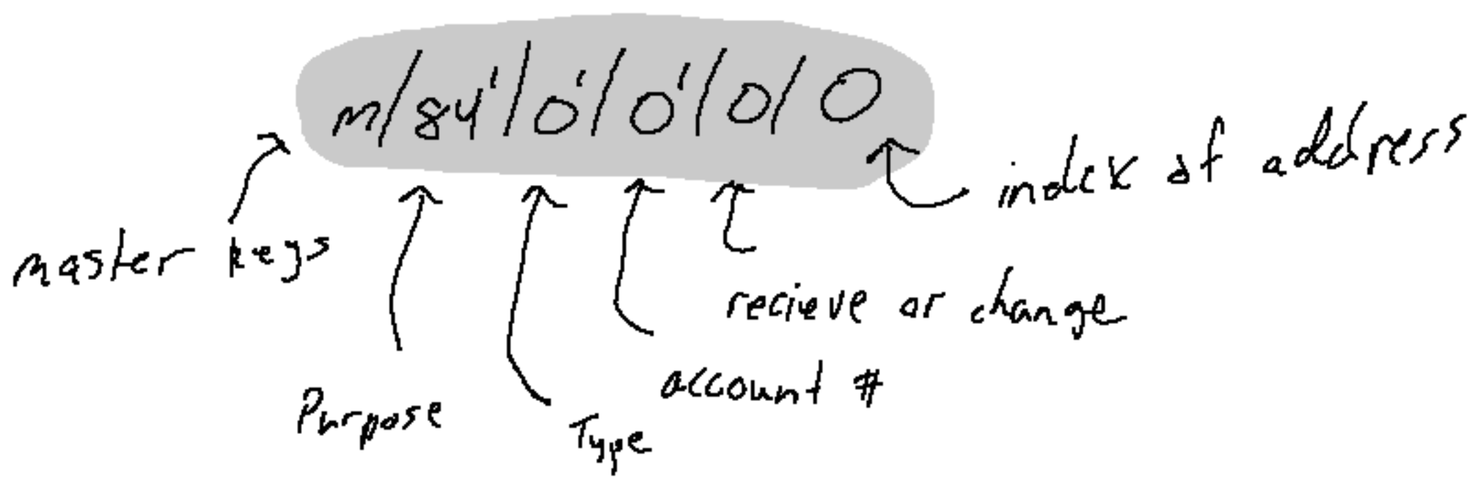
Compressed Public key

HD Wallet Generation

Ext-priv key → Child Priv key

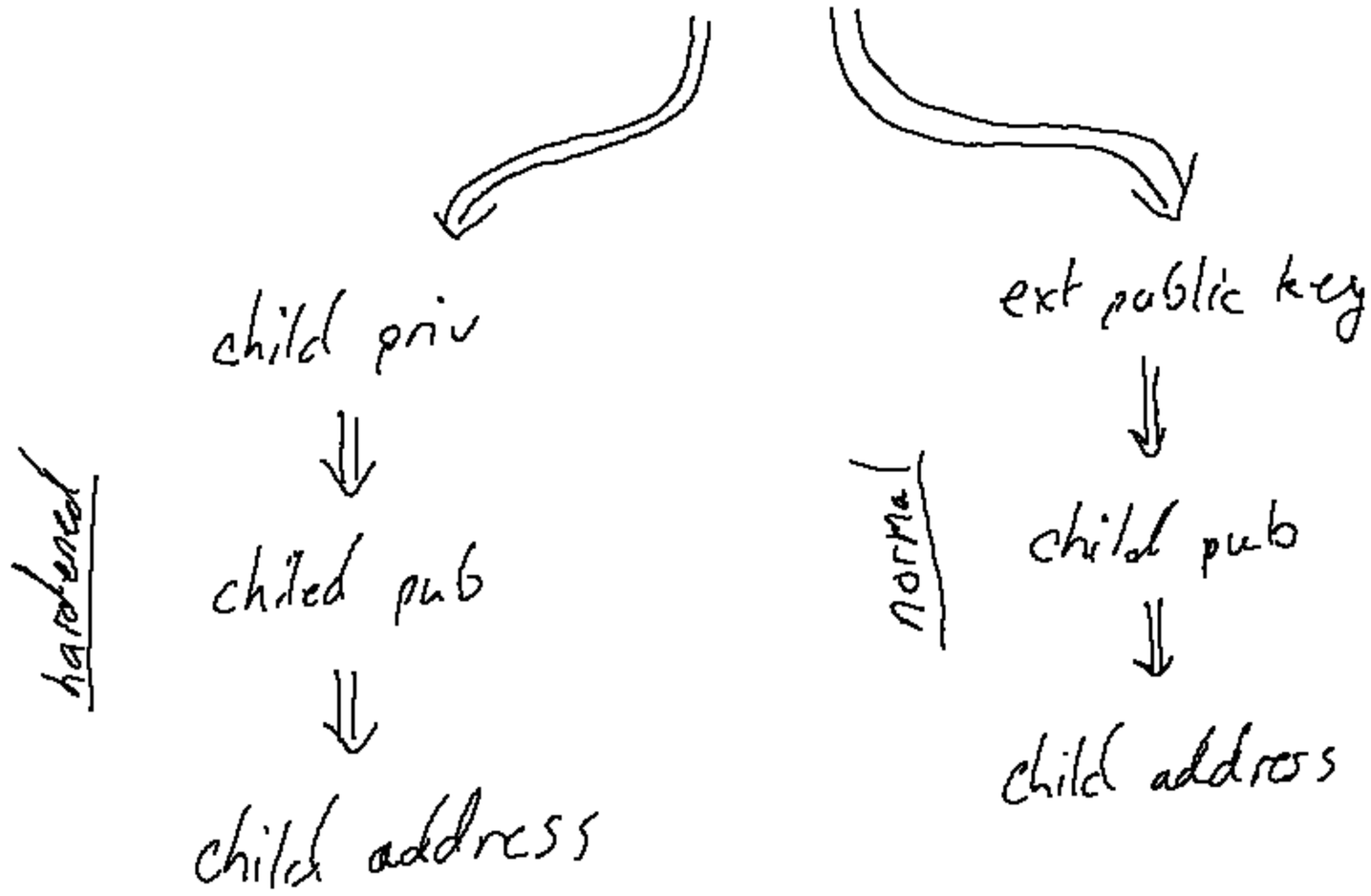


- At the start the priv key & chain code is always from parent
- This is a "hardened" derivation from private key.
- To compute "normal" derivation, use public key in place of private



Hardened vs. Unhardened

256 bits ⇌ 24 words → seed → extended private key



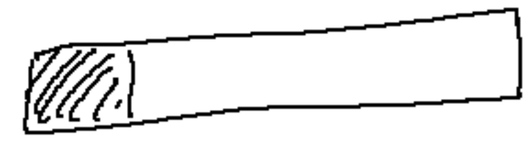
Normal - If xpub if found with child private & index, then the rest of the child private keys can be found.

Hardened - xpriv is used to generate all child keys & addresses it starts at index 2147483648 where a public key can no longer generate bc of larges 4 byte # possible

WIP

(main net) (compression = True)

sha256 ["80" + master private + "01"] = hash

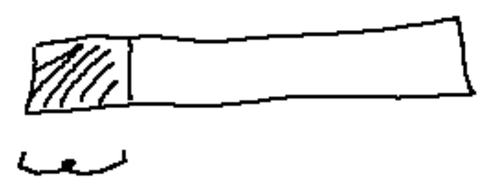


4 byte checksum

WIF = base 58 ("80" + master priv + "01" + checksum)

Master Fingerprint

Fingerprint = RipeMD160 [sha256 (compressed public)] =



first 4 bytes checksum