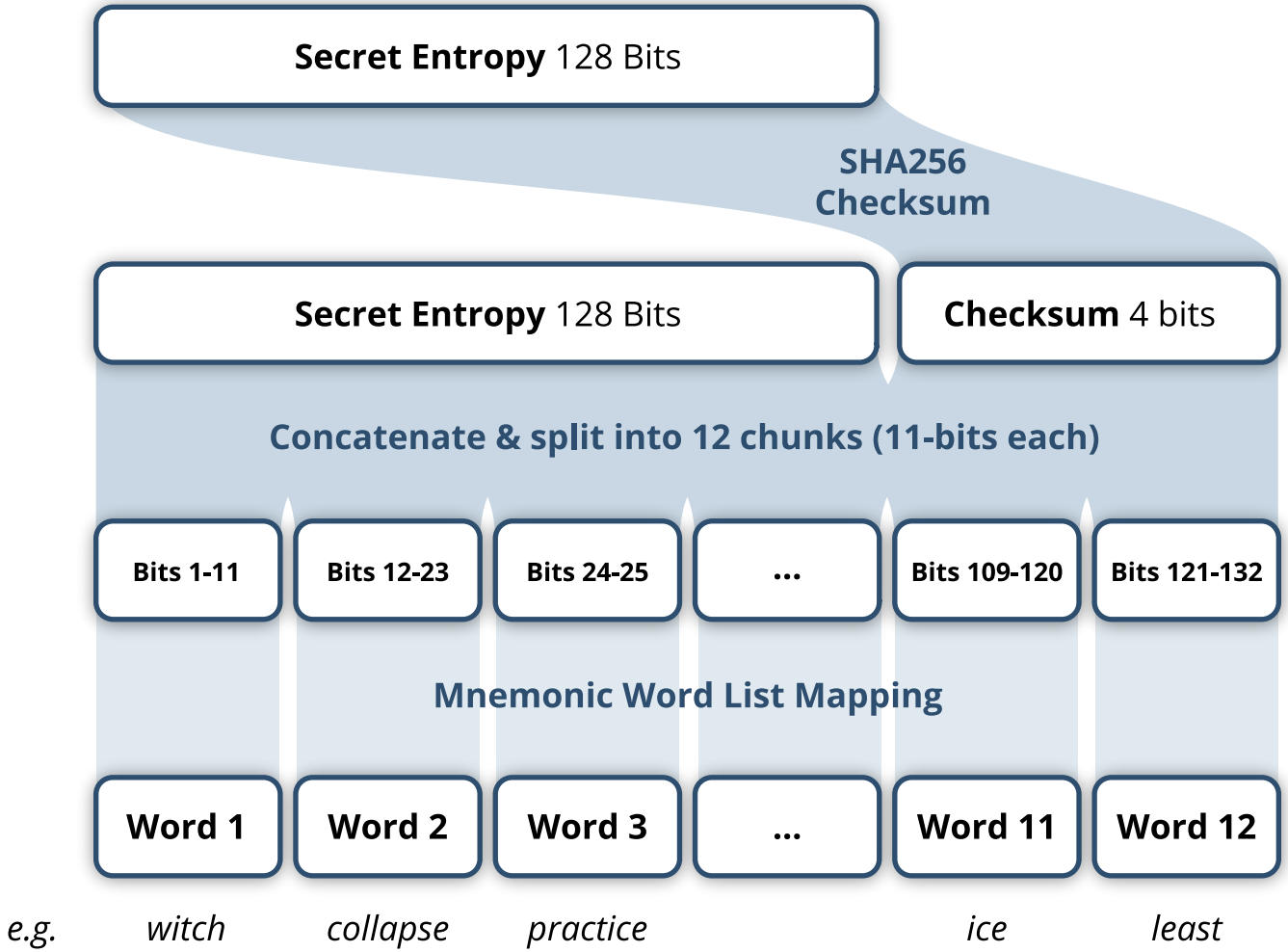
# Wallets

- Good terminology?
- A collection of private keys
- Bitcoin Core: Independently randomly drawn
  - Backup problems
- BIP 39
  - How to backup initial entropy, and how to derive a 512-bit seed from it
- BIP 32
  - How to generate unlimited private keys from the seed

# Mnemonic Key Words (BIP39)




Mnemonics are a user-friendly way to encode a secret root seed for a wallet.

**Generate SHA256 checksum**

- Secret lengths: 128, 160, 192, 224, 256 bits
- Checksum lengths: 4, 5, 6, 7, 8 bits
- Checksum length = secret length / 32

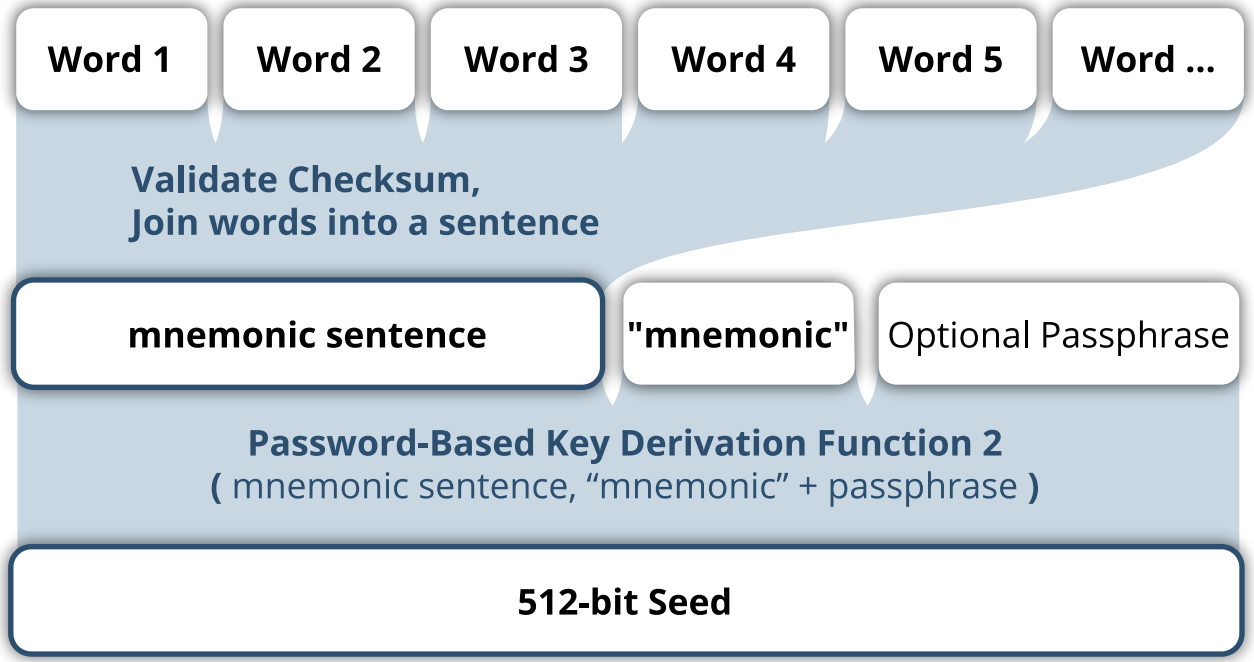
**Split into 11-bit chunks**

Sequence of 11-bit chunks needs to be maintained.

**11-bit to word mapping**

There are multiple languages mappings available to translate 11-bit chunks into words (BIP39).

# Mnemonics-to-Seed (BIP39)



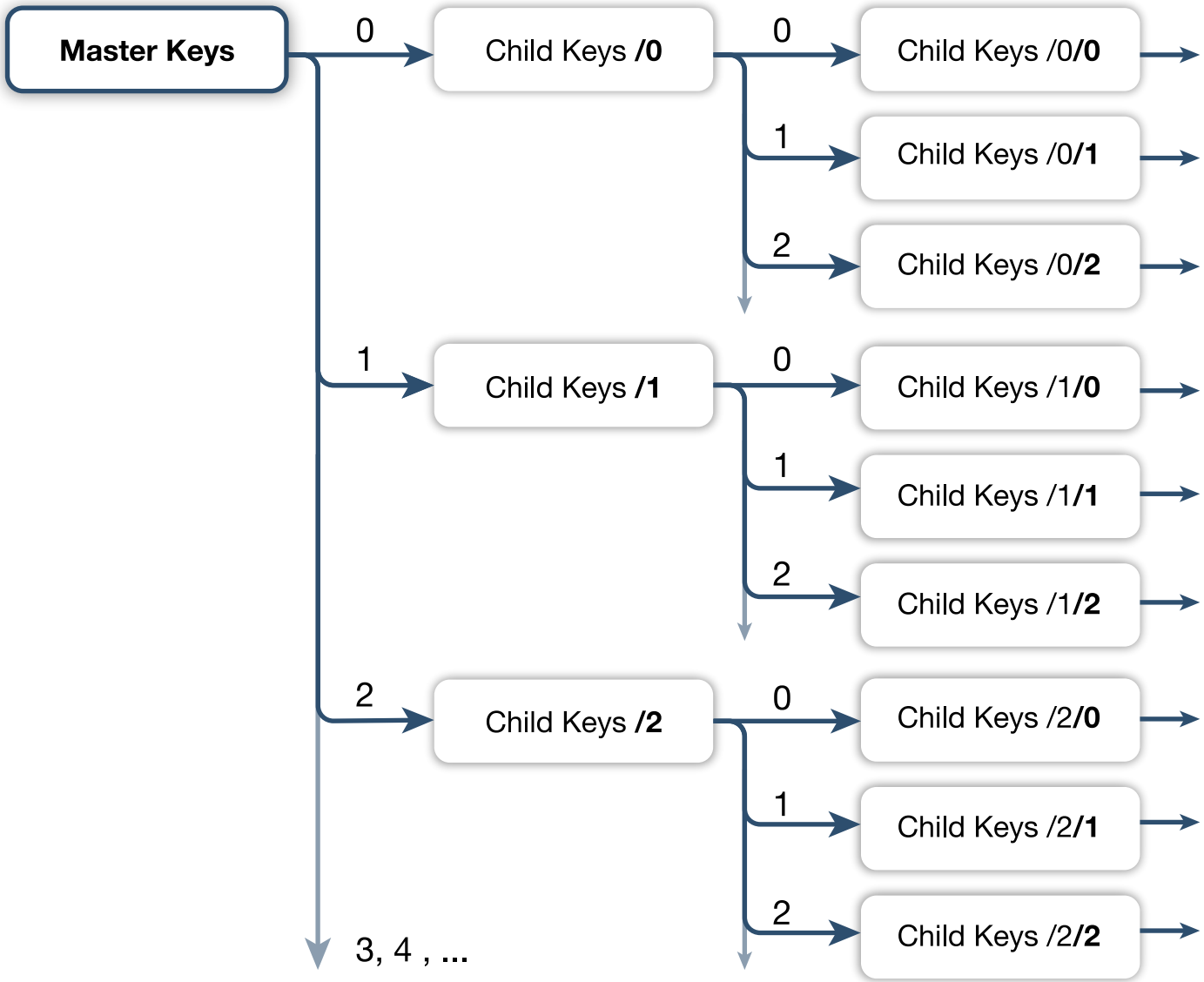

**Password-Based Key Derivation Function 2**

- 2048 rounds of HMAC-SHA512 (keyed hash function)
- Password: mnemonic sentence
- Salt: "mnemonic" + passphrase
- (Password & salt encoded in UTF-8 NFKD)

**512-bit hash digest**

- Seed for the creation of a wallet

# Hierarchical Deterministic Wallets (BIP32)




HD wallets (BIP32) can deterministically derive an indefinite number of fresh addresses from a single wallet secret.

**HD Tree**

- Fresh addresses to improve privacy.
- HD Tree is derived from Master Keys.
- HD Tree can be reconstructed from master Keys (given tree structure).

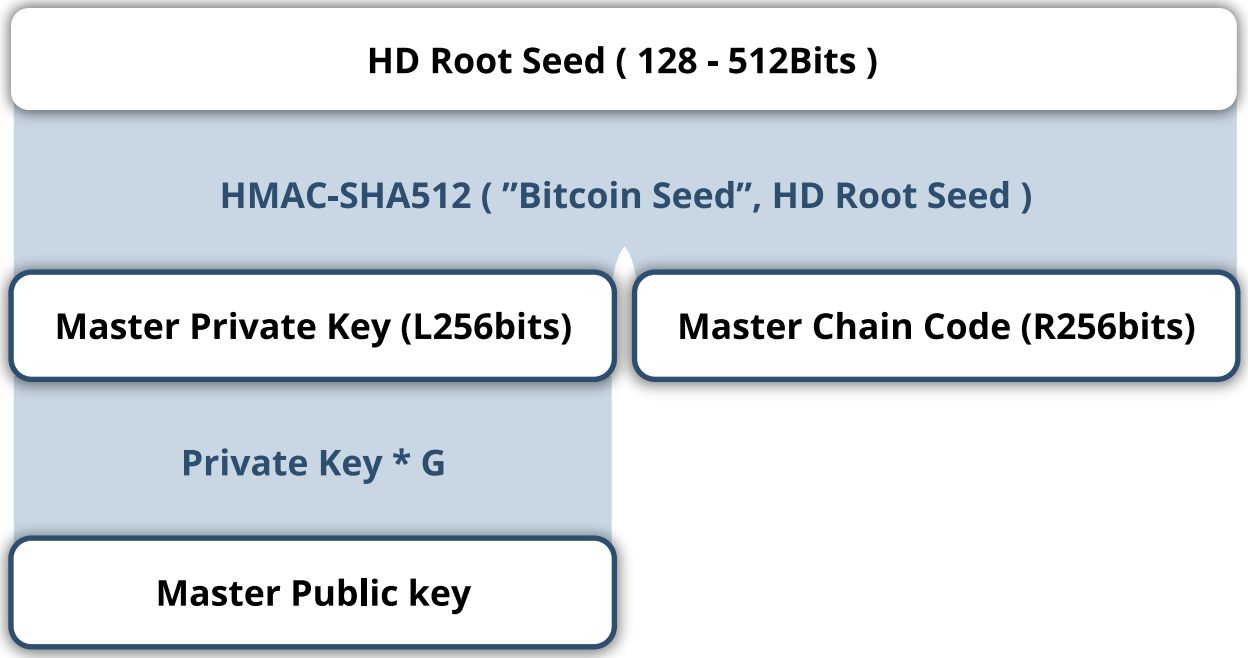
**Master keys**

- Derived from HD root secret.

**Subtrees**

- Allow separation of keys for accounts/usages.
- Selective key sharing.

# Master Key Pair Derivation

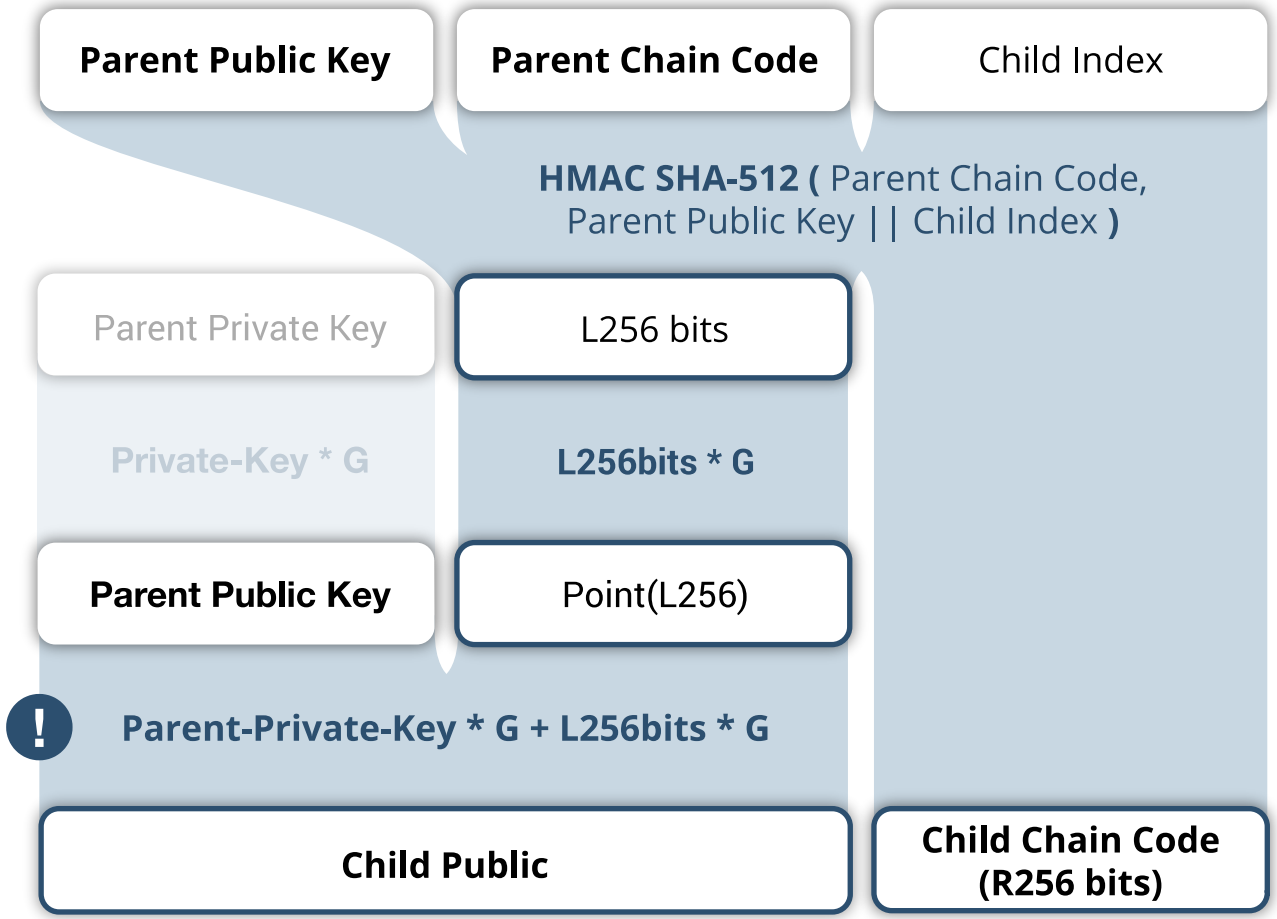



The master key pair is dervied from the HD root secret, and together with the chaincode, provides the basis for deriving subsequent child key generations.

**HMAC-SHA512**

- 512 bit hash digest is split into left and right 256 bits.
- Right 256 bits are chaincode, used in child key derivation.

# Child Key Pair Derivation




Hierarchical deterministic (child) private keys are derived from parent private keys.

**HMAC SHA512**

- Key: Parent chaincode
- Data: Parent public Key || Index

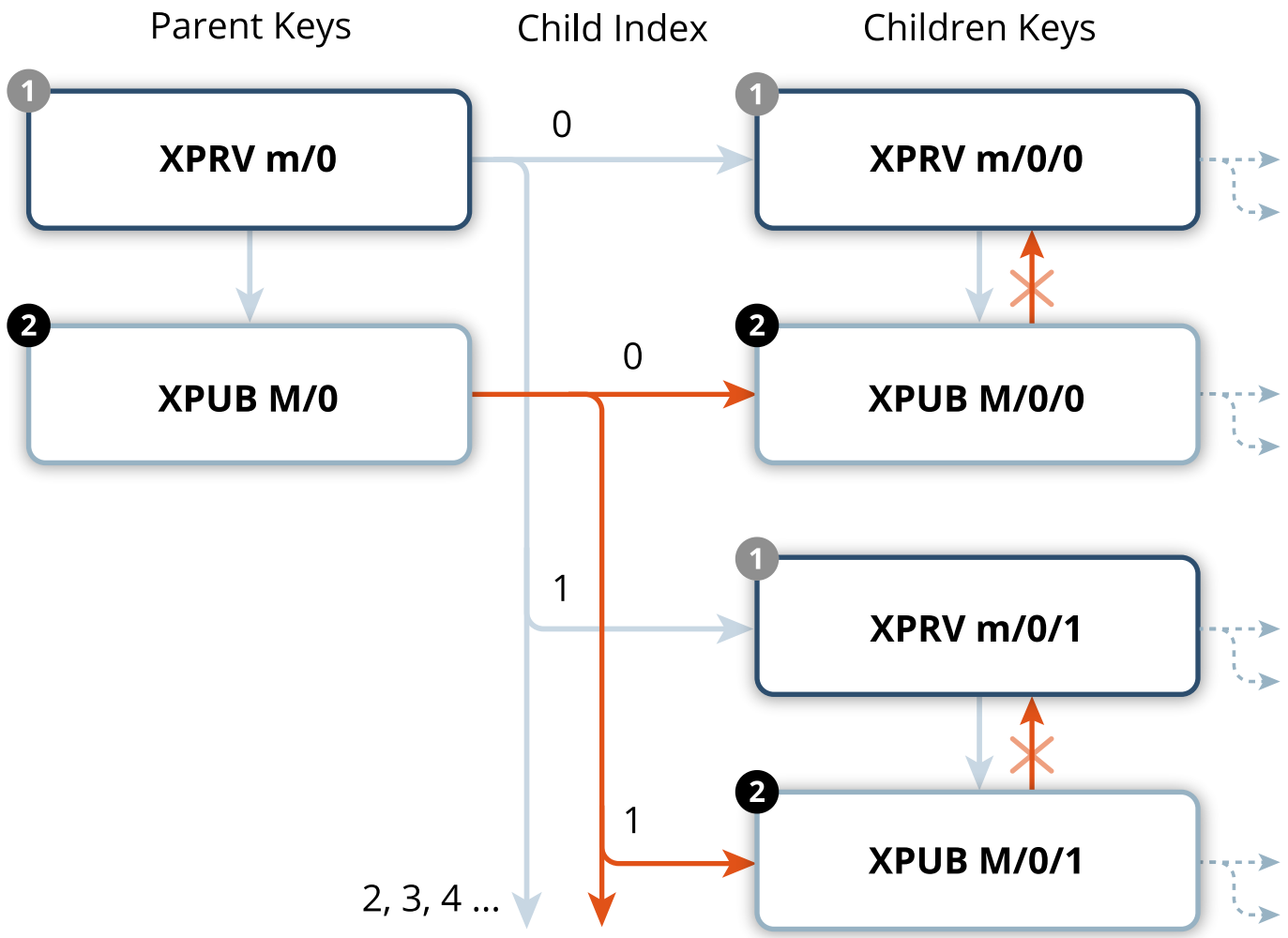
**Addition of two 256bit scalars**

- Private key + L256
- Result: Child private key

**Parent public key to child public key**

- HD child public key derivation without parent private key.

# HD Derivation Paths




**Parent-to-child derivation**

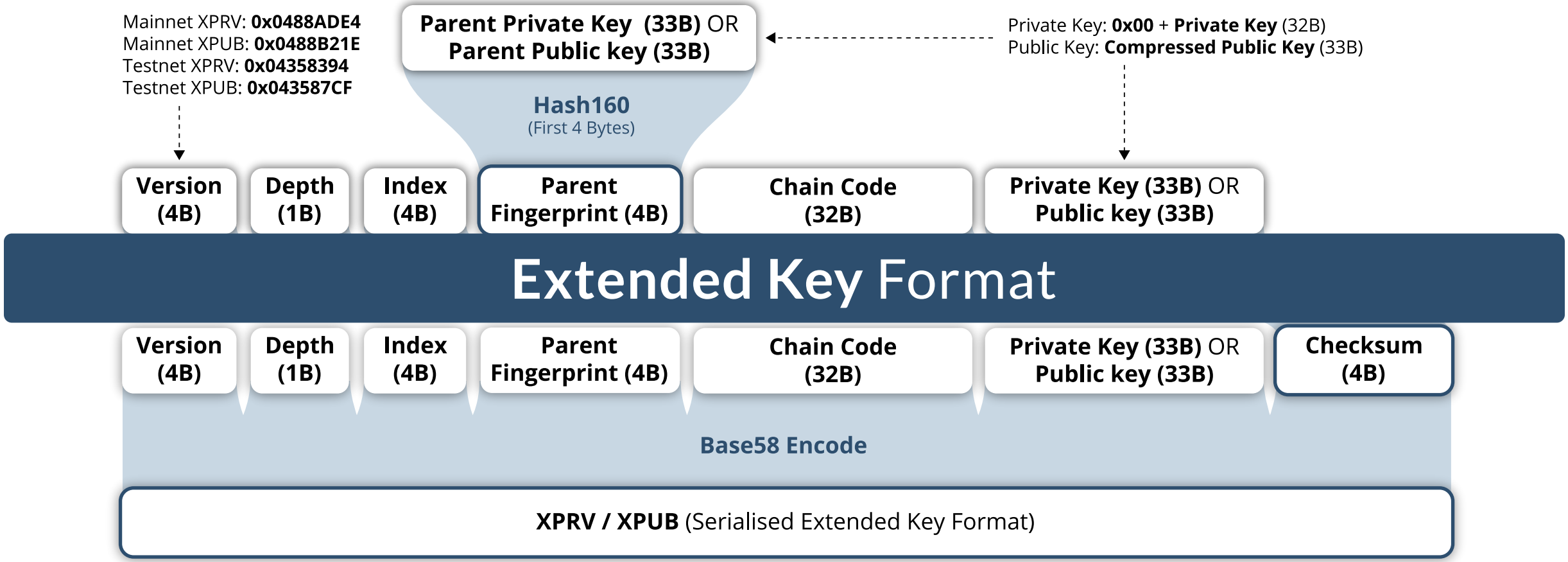
- Parent private to child private key
- Parent public to child public key

**Firewall between private and public key derivation paths**

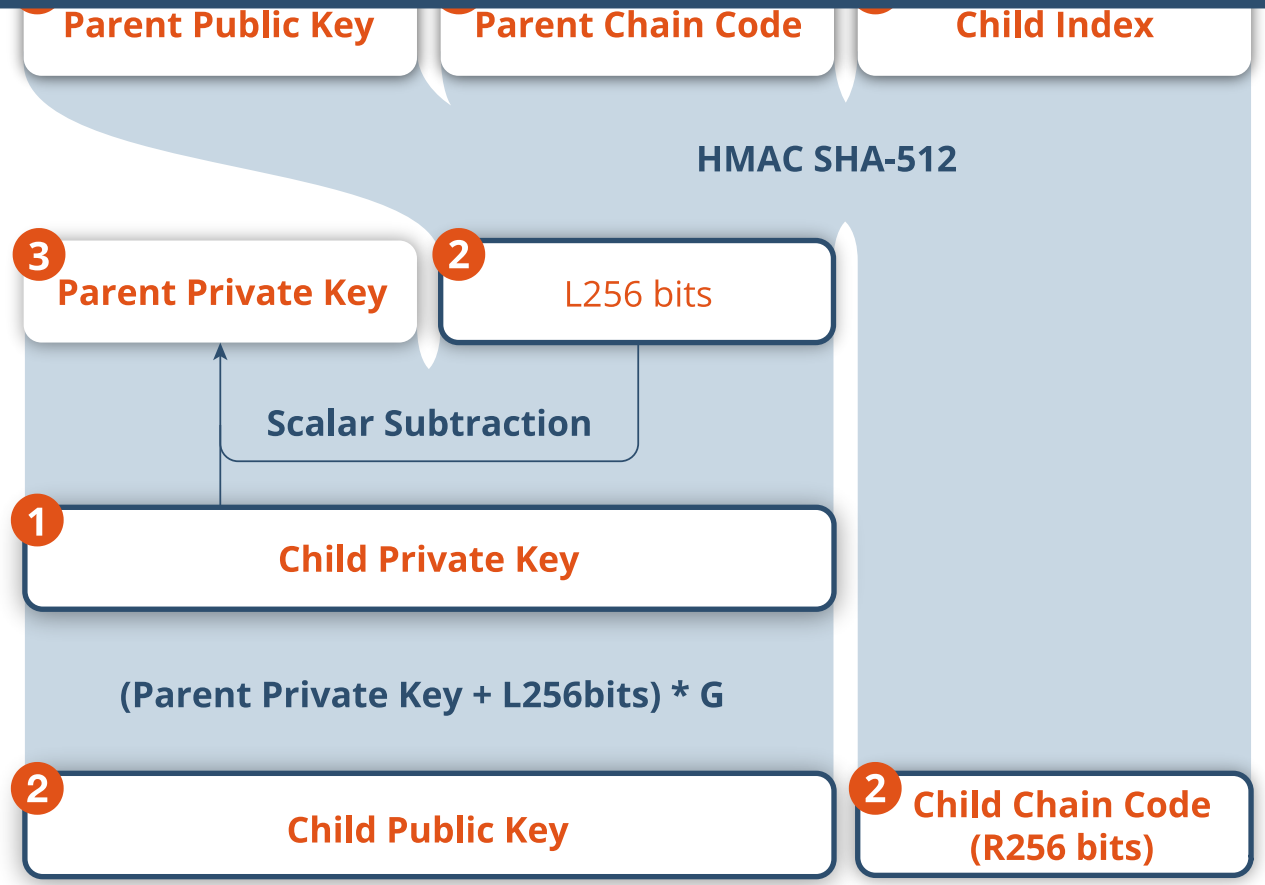
- Creation of new addresses can be delegated safely.
- Example: Creation of new receiving addresses by frontend.

**XPRV & XPUB keys**

- Chaincode
- + Private or public key


Mainnet XPRV: 0x0488ADE4
Mainnet XPUB: 0x0488B21E
Testnet XPRV: 0x04358394
Testnet XPUB: 0x043587CF
Parent Private Key (33B) OR Parent Public key (33B)
Private Key: 0x00 + Private Key (32B)
Public Key: Compressed Public Key (33B)
Hash160
(First 4 Bytes)
Version (4B)
Depth (1B)
Index (4B)
Parent Fingerprint (4B)
Chain Code (32B)
Private Key (33B) OR Public key (33B)
Extended Key Format
Version (4B)
Depth (1B)
Index (4B)
Parent Fingerprint (4B)
Chain Code (32B)
Private Key (33B) OR Public key (33B)
Checksum (4B)
Base58 Encode
XPRV / XPUB (Serialised Extended Key Format)

# Upstream **Private Key Exposure**




**1) Parent XPUB + private child exposure.**

- Public keys are potentially public.
- Child private key is exposed.

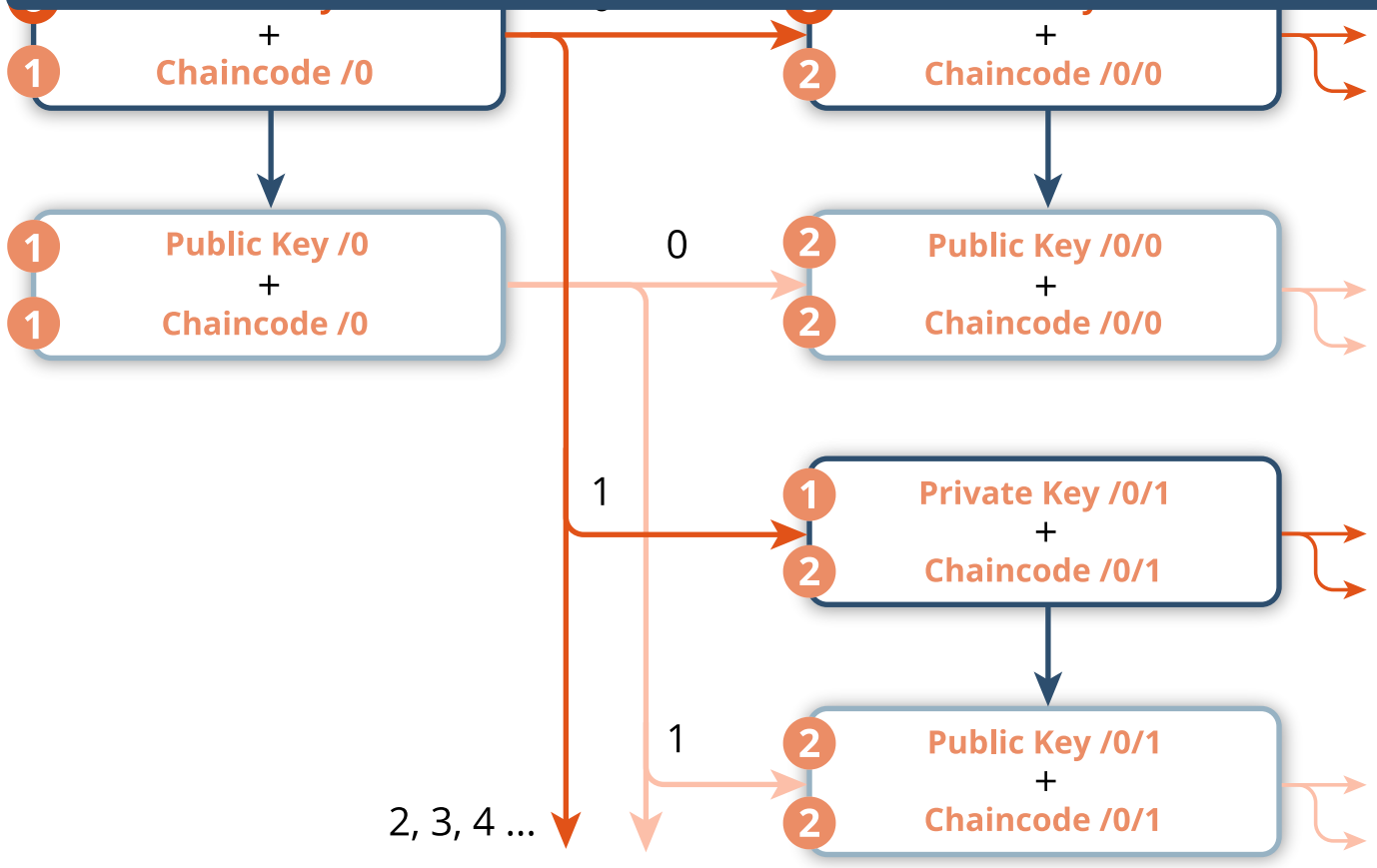
**2) Child XPUB is exposed.**

- L256 is derived.
- Both child XPRV and XPUB are now exposed.

**3) Parent private key is exposed.**

- Complete HD subtree is exposed.

# Upstream **Private Key Exposure 2**




**1) Parent XPUB + private child exposure.**

- Chaincode is identical for all keys of the same generation.

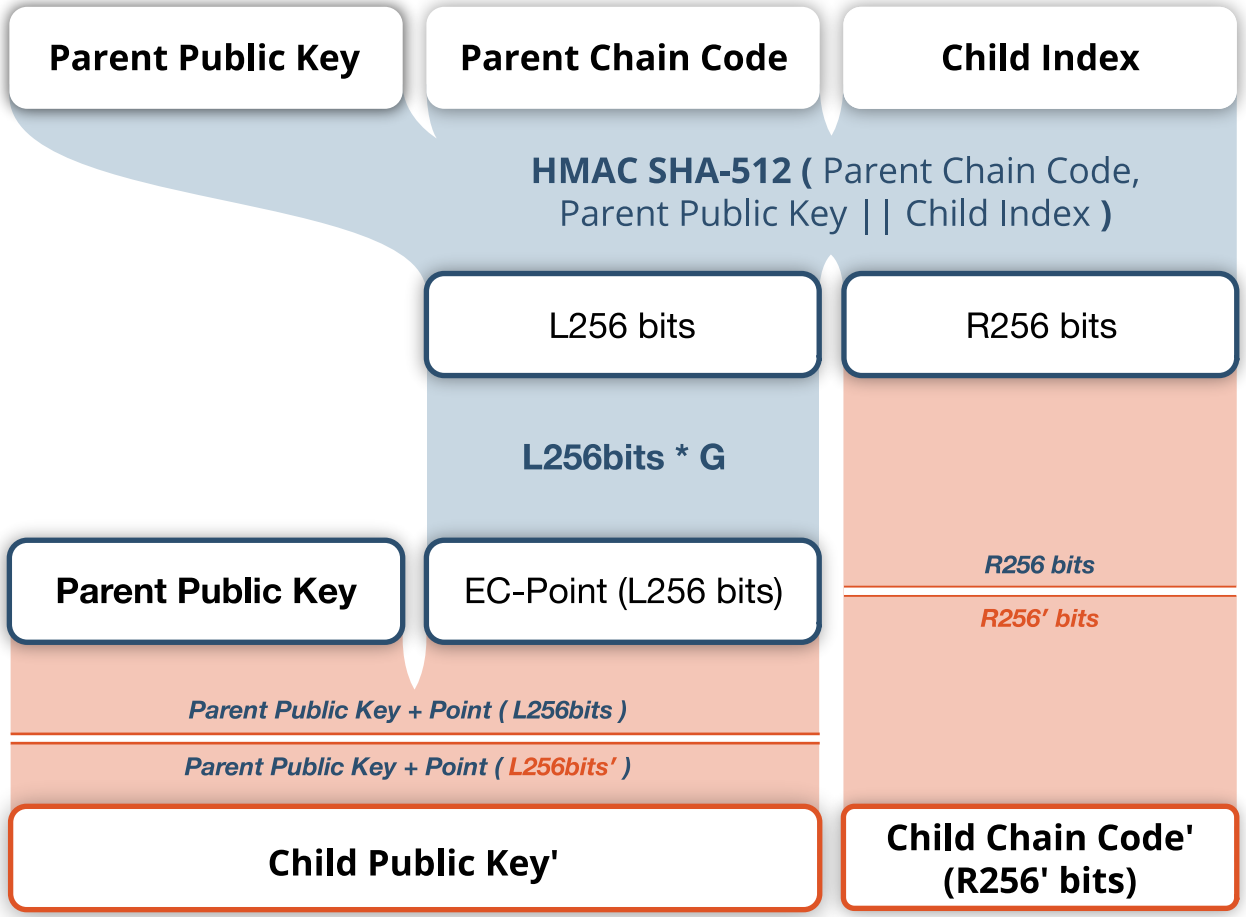
**2) All child public keys are exposed.**

- All child chain codes are exposed.

**3) Parent Private Key is exposed.**

- Complete HD subtree is exposed.

# Hardened HD Children



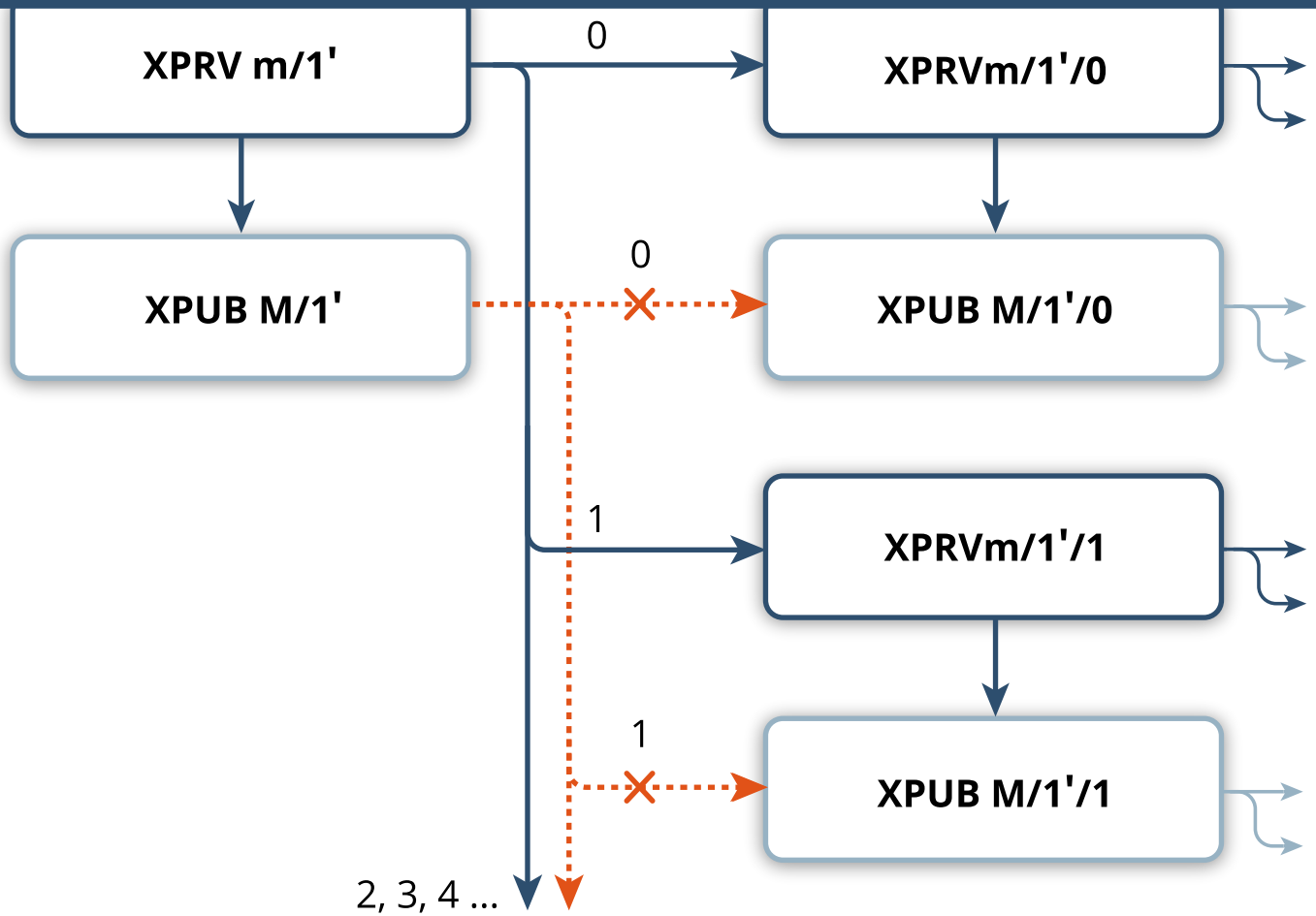

**Child private key hardening**

- Parent public key replaces private key.
- HMAC512:
  - Key: parent chaincode
  - Data: 0x00 || private key || index
- Hardened Index Notation:
  - i' = i + 2^31

**Hardened public keys.**

- Cannot derive any children.
- Derived only from hardened parent child key.

# Hardened HD Key Path




**Hardened Child Keys break XPUB derivation paths**

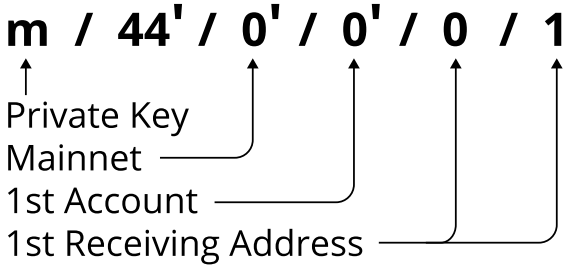
- Hardened keys denoted with prime′
- Although the parent keys are hardened, note that children keys mustn't necessarily be hardened, as shown here.
- This means the child XPUB keys in this example can derive grandchildren XPUB keys.

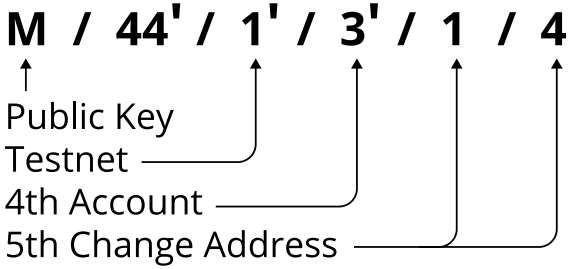
**Note:** In the case of key exposure in any of the subsequent child generations, the upstream key exposure cannot propgate up to the hardened parent key.

# HD Wallet Tree Structure (BIP44/43)

| Depth | ❶ | ❷ | ❸ | ❹ | ❺ |
|---|---|---|---|---|---|
| **m** or **M** | / **44'** | / **0'** | / **0'** | / **0** | / **0** |

Examples:




❶ **Purpose**

- Always set to a hardened 44′ (BIP44/43).

❷ **Network**

- Mainnet: 0′
- Testnet: 1′

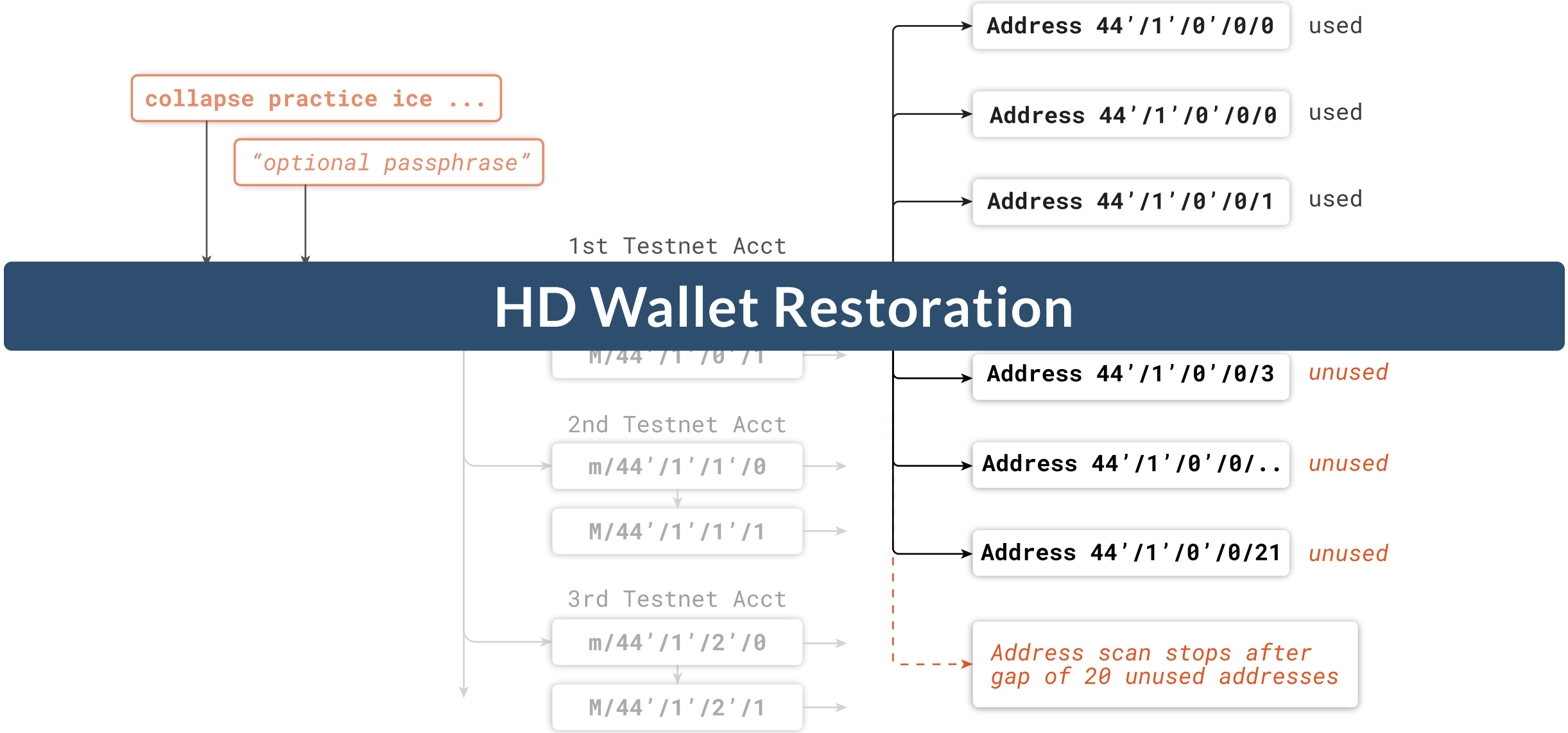
❸ **Account**

- Individual Wallet Accounts.

❹ **Receiving/Change Addresses**

- Keys of receiving address: 0 (unhardened)
- Keys of change address: 1 (unhardened)

❺ **Address Index**

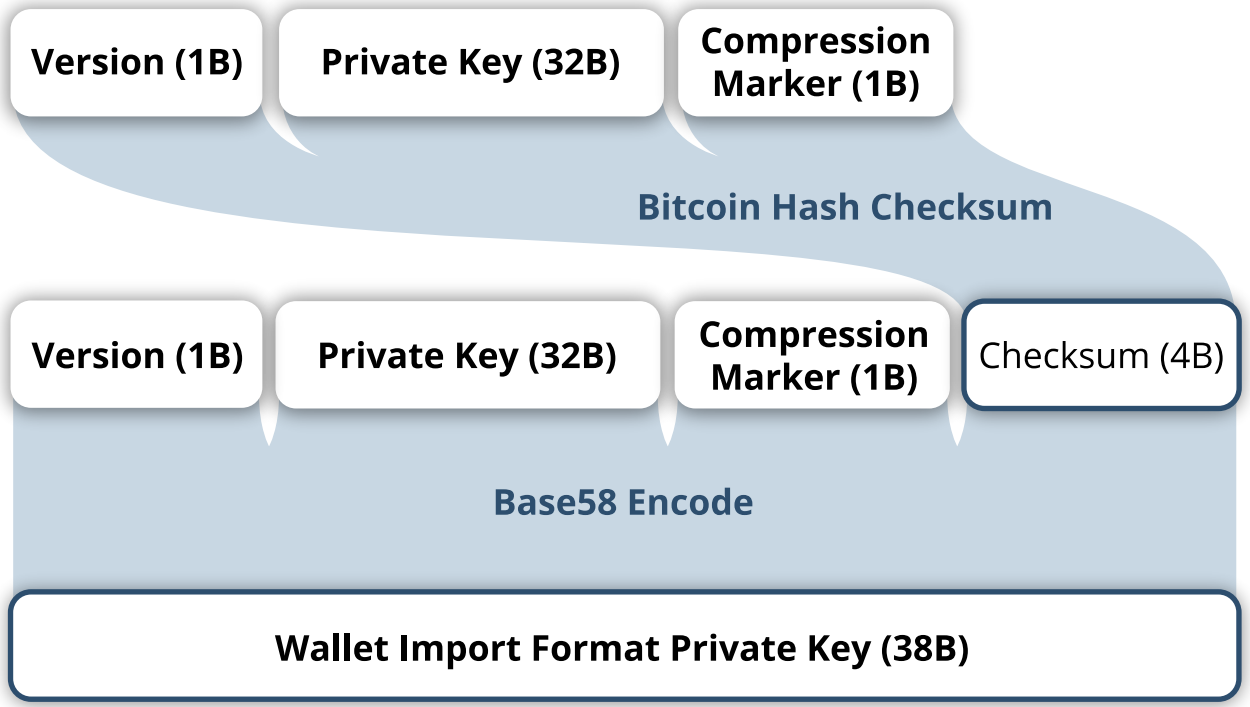

collapse practice ice ...
"optional passphrase"
1st Testnet Acct
Address 44'/1'/0'/0/0
used
Address 44'/1'/0'/0/0
used
Address 44'/1'/0'/0/1
used
HD Wallet Restoration
M/44'/1'/0'/1
Address 44'/1'/0'/0/3
unused
2nd Testnet Acct
m/44'/1'/1'/0
Address 44'/1'/0'/0/..
unused
M/44'/1'/1'/1
Address 44'/1'/0'/0/21
unused
3rd Testnet Acct
m/44'/1'/2'/0
Address scan stops after
gap of 20 unused addresses
M/44'/1'/2'/1

# Beyond BIP44 & P2PKH

- BIP 49 introduced yprv/ypub
  - m / 49' / ...
  - Wallets construct P2SH-P2WPKH outputs
- BIP 84 introduced zprv/zpub
  - m / 84' / ...
  - Wallets construct P2WPKH outputs
- not scalable
- ⇒ Output descriptors
  - sh(wpkh(03fff97bd5755eeea420453a14355235d382f6472f8568a18b2f057a1460297556))
  - pkh(xpub68G...GDnw/1/2)
  - wsh(multi(1,xpub661...uduB/1/0/*,xpub69H...QTPH/0/0/*))

# Private Key - **Wallet Import Format**

The WIF private key provides information to recreate a private/public key pair.




**Bitcoin Hash Checksum**

- `double SHA256(input)`
- First 4 bytes of digest.
- Version byte:
  - Mainnet: 0x80
  - Testnet: 0xEF
- Compression Marker (0x01) ommited if associated public key is uncompressed

**Base58 Encoding**

- Base58 encoded WIF begins with:
  - Mainnet/Compressed: K/L
  - Mainnet/Uncompressed: 5
  - Testnet/Compressed: C
  - Testnet/Uncompressed: 9