```
BIP: 49
Layer: Applications
Title: Derivation scheme for P2WPKH-nested-in-P2SH based accounts
Author: Daniel Weigl <DanielWeigl@gmx.at>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0049
Status: Final
Type: Informational
Created: 2016-05-19
License: PD
```

Abstract

This BIP defines the derivation scheme for HD wallets using the P2WPKH-nested-in-P2SH (BIP 141) serialization format for segregated witness transactions.

Motivation

With the usage of P2WPKH-nested-in-P2SH (BIP 141) transactions it is necessary to have a common derivation scheme. It allows the user to use different HD wallets with the same masterseed and/or a single account seamlessly.

Thus the user needs to create dedicated segregated witness accounts, which ensures that only wallets compatible with this BIP will detect the accounts and handle them appropriately.

Considerations

Two generally different approaches are possible for current BIP44 capable wallets:

1) Allow the user to use the same account(s) that they already use, but add segregated witness encoded addresses to it.

1.1) Use the same public keys as defined in BIP44, but in addition to the normal P2PKH address also derive the P2SH address from it.

1.2) Use the same account root, but branch off and derive different external and internal chain roots to derive dedicated public keys for the segregated witness addresses.

2) Create dedicated accounts used only for segregated witness addresses.

The solutions from point 1 have a common disadvantage: if a user imports/recovers a BIP49-compatible wallet masterseed into/in a non-BIP49-compatible wallet, the account might show up but also it might miss some UTXOs.

Therefore this BIP uses solution 2, which fails in a more visible way. Either the account shows up or not at all. The user does not have to check his balance after using the same seed in different wallets.

Specifications

This BIP defines the two needed steps to derive multiple deterministic addresses based on a BIP 32 root account.

Public key derivation

To derive a public key from the root account, this BIP uses the same account-structure as defined in BIP 44, but only uses a different purpose value to indicate the different transaction serialization method.

`m / purpose' / coin_type' / account' / change / address_index`

For the ‘purpose‘-path level it uses ‘49'‘. The rest of the levels are used as defined in BIP44.

Address derivation

To derive the P2SH address from the above calculated public key, we use the encapsulation defined in BIP 141:

```
   witness:
scriptSig:    <0 <20-byte-key-hash>>
(0x160014{20-byte-key-hash})
scriptPubKey: HASH160 <20-byte-script-hash> EQUAL
(0xA914{20-byte-script-hash}87)
```

Extended Key Version

When serializing extended keys, this scheme uses alternate version bytes. Extended public keys use `0x049d7cb2` to produce a "ypub" prefix, and private keys use `0x049d7878` to produce a "yprv" prefix. Testnet uses `0x044a5262` "upub" and `0x044a4e28` "uprv."

Additional registered version bytes are listed in SLIP-0132.

Backwards Compatibility

This BIP is not backwards compatible by design as described under considerations. An incompatible wallet will not discover accounts at all and the user will notice that something is wrong.

Test vectors

```
masterseedWords = abandon abandon abandon abandon abandon abandon abandon abandon abandon
masterseed = uprv8tXDerPXZ1QsVNjUJWTurs9kA1KGfKUAts74GCkcXtU8GwnH33GDRbNJpEqTvipfCyycARtQ

// Account 0, root = m/49'/1'/0'
account0Xpriv = uprv91G7gZkzehuMVxDJTYE6tLivdF8e4rvzSu1LFfKw3b2Qx1Aj8vpoFnHdfUZ3hmi9jsvPi
```

```
account0Xpub = upub5EFU65HtV5TeiSHmZZm7FUffBGy8UKeqp7vw43jYbvZPpoVsgU93oac7Wk3u6moKegAEWt

// Account 0, first receiving private key = m/49'/1'/0'/0/0
account0recvPrivateKey = cULrpoZGXiuC19Uhvykx7NugygA3k86b3hmdCeyvHYQZSxojGyXJ
account0recvPrivateKeyHex = 0xc9bdb49cfbaedca21c4b1f3a7803c34636b1d7dc55a717132443fc3f4c58
account0recvPublickKeyHex = 0x03a1af804ac108a8a51782198c2d034b28bf90c8803f5a53f76276fa69a4

// Address derivation
keyhash = HASH160(account0recvPublickKeyHex) = 0x38971f73930f6c141d977ac4fd4a727c854935b3
scriptSig = <0 <keyhash>> = 0x001438971f73930f6c141d977ac4fd4a727c854935b3
addressBytes = HASH160(scriptSig) = 0x336caa13e08b96080a32b5d818d59b4ab3b36742

// addressBytes base58check encoded for testnet
address = base58check(prefix | addressBytes) = 2Mww8dCYPUpKHofjgcXcBCEGmniw9CoaiD2 (testne
```

Reference

- BIP16 - Pay to Script Hash
- BIP32 - Hierarchical Deterministic Wallets
- BIP43 - Purpose Field for Deterministic Wallets
- BIP44 - Multi-Account Hierarchy for Deterministic Wallets
- BIP141 - Segregated Witness (Consensus layer)

Copyright