```
BIP: 87
Layer: Applications
Title: Hierarchy for Deterministic Multisig Wallets
Author: Robert Spigler <RobertSpigler@ProtonMail.ch>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0087
Status: Proposed
Type: Standards Track
Created: 2020-03-11
License: BSD-2-Clause
```

Abstract

This BIP defines a sane hierarchy for deterministic multisig wallets based on an algorithm described in BIP-0032 (BIP32 from now on), purpose scheme described in BIP-0043 (BIP43 from now on), and multi-account hierarchy described in BIP-0044 (BIP44 from now on).

This BIP is a particular application of BIP43.

Copyright

Motivation

With the increase of more user friendly (offline) multisignature wallets, and adoption of new technologies such as the descriptor language and BIP-0174 (Partially Signed Bitcoin Transactions), it is necessary to create a common derivation scheme that makes use of all new technologies.

As background, BIP 44/49/84 specifies:

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

where the BIP43 `purpose'` path is separate for each script (P2PKH, P2WPKH-in-P2SH, and P2WPKH respectively). Having a script-per-derivation for single sig wallets allows for easy backup and restore, with just the private key information.

Multisignature wallets need more information to backup and restore (such as all cosigner public keys), and these per-script derivations are made redundant with descriptors, which provide that information (while also specifying a collection of output scripts). A modern standardization is needed for multisig derivation paths. There are some in existence, but all have issues. For example, BIP45 specifies:

`m / purpose' / cosigner_index / change / address_index`

BIP45 unecessarily demands a single script type (here, P2SH). In addition, BIP45 sets `cosigner_index` in order to sort the `purpose'` public keys of each cosigner. This too is redundant, as descriptors can set the order of the public keys with `multi` or have them sorted lexicographically (as described in BIP67) with `sortedmulti`. Sorting public keys between cosigners in order to create the full derivation path, prior to sending the key record to the coordinator to create the descriptor, merely adds additional unnecessary communication rounds.

The second multisignature "standard" in use is m/48', which specifies:

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

Rather than following in BIP 44/49/84's path and having a separate BIP per script after P2SH (BIP45), vendors decided to insert `script_type'` into the derivation path (where P2SH-P2WSH=1, P2WSH=2, Future_Script=3, etc). As described previously, this is unnecessary, as the descriptor sets the script. While it attempts to reduce maintainence work by getting rid of new BIPs-per-script, it still requires maintaining an updated, redundant, `script_type` list.

The structure proposed later in this paper solves these issues and is quite comprehensive. It allows for the handling of multiple accounts, external and internal chains per account, and millions of addresses per chain, in a multi-party, multisignature, hierarchical deterministic wallet regardless of the script type [1].

This paper was inspired from BIP44.

Specification

Key sorting

Any wallet that supports descriptors inherently supports deterministic key sorting as per BIP67 (through the `sortedmulti` function) so that all possible multisignature addresses/scripts are derived from deterministically sorted public keys.

Path levels

We should not be mixing keys and scripts in the same layer. The wallet should create extended private/public keys independent of the script type, whereas the descriptor language tells wallets to watch the multisig outputs with the specified public keys.

We define the following 5 levels in the BIP32 path:

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

[1]**Why propose this structure only for multisignature wallets?** Currently, single-sig wallets are able to restore funds using just the master private key data (in the format of BIP39 usually). Even if the user doesn't recall the derivation used, the wallet implementation can iterate through common schemes (BIP44/49/84). With this proposed hierarchy, the user would either have to now backup additional data (the descriptor), or the wallet would have to attempt all script types for every account level when restoring. Because of this, even though the descriptor language handles the signature type just like it does the script type, it is best to restrict this script-agnostic hierarchy to multisignature wallets only.

`h` or `'` in the path indicates that BIP32 hardened derivation is used.

Each level has a special meaning, described in the chapters below.

Purpose

Purpose is a constant set to `87'` following the BIP43 recommendation. It indicates that the subtree of this node is used according to this specification.

Hardened derivation is used at this level.

Coin type

One master node (seed) can be used for multiple Bitcoin networks. Sharing the same space for various networks has some disadvantages.

This level creates a separate subtree for every network, avoiding reusing addresses across networks and improving privacy issues.

Coin type `0` for mainnet and `1` for testnets (testnet, regtest, and signet).

Hardened derivation is used at this level.

Account

This level splits the key space into independent user identities, following the BIP44 pattern, so the wallet never mixes the coins across different accounts.

Users can use these accounts to organize the funds in the same fashion as bank accounts; for donation purposes (where all addresses are considered public), for saving purposes, for common expenses, etc.

Accounts are numbered from index `0` in sequentially increasing manner. This number is used as child index in BIP32 derivation.

Hardened derivation is used at this level.

It is crucial that this level is increased for each new wallet joined or private/public keys created; for both privacy and cryptographic purposes. For example, before sending a new key record to a coordinator, the wallet must increment the `account'` level. This prevents key reuse - across ECDSA and Schnorr signatures, across different script types, and inbetween the same wallet types.

Change

Constant `0` is used for external chain and constant `1` for internal chain (also known as change addresses). External chain is used for addresses that are meant to be visible outside of the wallet (e.g. for receiving payments). Internal chain is used for addresses which are not meant to be visible outside of the wallet and is used for return transaction change.

Public derivation is used at this level.

Index

Addresses are numbered from index `0` in sequentially increasing manner. This number is used as child index in BIP32 derivation.

Public derivation is used at this level.

Address Discovery

The multisig descriptors or descriptor template that is generated from the cosigners' combined key records should be used to generate and discover addresses.

Please see BIP-0129 (Bitcoin Secure Multisig Setup) for an introduction on descriptor templates. The descriptor or descriptor template should contain the key origin information for maximum compatibility with BIP-0174.

For example:

The following descriptor template and derivation path restrictions:

```
wsh(sortedmulti(2,[xfpForA/87'/0'/0']XpubA/**,[xfpForB/87'/0'/0']XpubB/**))
```

```
/0/*,/1/*
```

Expands to the two concrete descriptors:

```
wsh(sortedmulti(2,[xfpForA/87'/0'/0']XpubA/0/*,[xfpForB/87'/0'/0']XpubB/0/*))
```

```
wsh(sortedmulti(2,[xfpForA/87'/0'/0']XpubA/1/*,[xfpForB/87'/0'/0']XpubB/1/*))
```

To discover addresses, import both the receiving and change descriptors; respect the gap limit described below.

Address Gap Limit

Address gap limit is currently set to 20. If the software hits 20 unused addresses in a row, it expects there are no used addresses beyond this point and stops searching the address chain.

Wallet software should warn when the user is trying to exceed the gap limit on an external descriptor by generating multiple unused addresses.

Backwards Compatibility

Any script that is supported by descriptors (and the specific wallet implementation) is compatible with this BIP.

As wallets complying with this BIP are descriptor wallets, this therefore necessitates that the cosigners backup their private key information and the descriptor, in order to properly restore at a later time. This shouldn't be a user burden, since (to much user surprise), all cosigner public keys need to be supplied in addition to `M` seeds in any `M` of `N` multisig restore operation. The descriptor provides this information in a standardized format, with key origin information and error detection.

Rationale

Examples

network	account	chain	address	path
mainnet	first	external	first	m / 87' / 0' / 0' / 0 / 0
mainnet	first	external	second	m / 87' / 0' / 0' / 0 / 1
mainnet	first	change	first	m / 87' / 0' / 0' / 1 / 0
mainnet	first	change	second	m / 87' / 0' / 0' / 1 / 1
mainnet	second	external	first	m / 87' / 0' / 1' / 0 / 0
mainnet	second	external	second	m / 87' / 0' / 1' / 0 / 1
testnet	first	external	first	m / 87' / 1' / 0' / 0 / 0
testnet	first	external	second	m / 87' / 1' / 0' / 0 / 1
testnet	first	change	first	m / 87' / 1' / 0' / 1 / 0
testnet	first	change	second	m / 87' / 1' / 0' / 1 / 1
testnet	second	external	first	m / 87' / 1' / 1' / 0 / 0
testnet	second	external	second	m / 87' / 1' / 1' / 0 / 1
testnet	second	change	first	m / 87' / 1' / 1' / 1 / 0
testnet	second	change	second	m / 87' / 1' / 1' / 1 / 1

Reference Implementation

None at the moment.

Acknowledgement

Special thanks to SomberNight, Craig Raw, David Harding, Jochen Hoenicke, Sjors Provoost, and others for their feedback on the specification.

References

Original mailing list thread: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-March/018630.html

- BIP-0032 (Hierarchical Deterministic Wallets)
- BIP-0043 (Purpose Field for Deterministic Wallets)
- BIP-0044 (Multi-Account Hierarchy for Deterministic Wallets)
- Output Descriptors
- BIP-0174 (Partially Signed Bitcoin Transaction Format)
- BIP-0067 (Deterministic Pay-to-script-hash multi-signature addresses through public key sorting)
- BIP-0129 (Bitcoin Secure Multisig Setup)