```
BIP: 86
Layer: Applications
Title: Key Derivation for Single Key P2TR Outputs
Author: Andrew Chow <andrew@achow101.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0086
Status: Draft
Type: Standards Track
Created: 2021-06-22
License: BSD-2-Clause
```

Abstract

This document suggests a derivation scheme for HD wallets whose keys are involved in single key P2TR (BIP 341) outputs as the Taproot internal key.

Copyright

Motivation

With the usage of single key P2TR transactions, it is useful to have a common derivation scheme so that HD wallets that only have a backup of the HD seed can be likely to recover single key Taproot outputs. Although there are now solutions which obviate the need for fixed derivation paths for specific script types, many software wallets and hardware signers still use seed backups which lack derivation path and script information. Thus we largely use the same approach used in BIPs 49 and 84 for ease of implementation.

Specifications

This BIP defines the two needed steps to derive multiple deterministic addresses based on a BIP 32 master private key.

Public key derivation

To derive a public key from the root account, this BIP uses the same account-structure as defined in BIPs 44, 49, and 84, but with a different purpose value for the script type.

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

For the `purpose`-path level it uses `86'`. The rest of the levels are used as defined in BIPs 44, 49, and 84.

A key derived with this derivation path pattern will be referred to as `derived_key` further in this document.

Address derivation

BIP 341 states: "If the spending conditions do not require a script path, the output key should commit to an unspendable script path instead of having no script path. This can be achieved by computing the output key point as $Q = P + int(hash_{TapTweak}(bytes(P)))G$." Thus:

```
internal_key:       lift_x(derived_key)
32_byte_output_key: internal_key + int(HashTapTweak(bytes(internal_key)))G
```

In a transaction, the scripts and witnesses are as defined in BIP 341:

```
witness:      <signature>
scriptSig:    (empty)
scriptPubKey: 1 <32_byte_output_key>
              (0x5120{32_byte_output_key})
```

Backwards Compatibility

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

However this BIP uses the same method used in BIPs 44, 49, and 84, so it should not be difficult to implement.

Test vectors

```
mnemonic = abandon abandon abandon abandon abandon abandon abandon abandon abandon abandon a
rootpriv = xprv9s21ZrQH143K3GJpoapnV8SFfukcVBSfeCficPSGfubmSFDxo1kuHnLisriDvSnRRuL2Qrg5ggqHK
rootpub  = xpub661MyMwAqRbcFkPHucMnrGNzDwb6teAX1RbKQmqtEF8kK3Z7LZ59qafCjB9eCRLiTVG3uxBxgKvRg

// Account 0, root = m/86'/0'/0'
xprv = xprv9xgqHN7yz9MwCkxsBPN5qetuNdQSUttZNKw1dcYTV4mkaAFiBVGQziHs3NRSWMkCzvgjEe3n9xV8oYyw
xpub = xpub6BgBgsespWvERF3LHQu6CnqdvfEvtMcQjYrcRzx53QJjSxarj2afYWcLteoGVky7D3UKDP9QyrLprQ3VC

// Account 0, first receiving address = m/86'/0'/0'/0/0
xprv         = xprvA449goEeU9okwCzzZaxiy475EQGQzBkc65su82nXEvcwzfSskb2hAt2WymrjyRL6kpbVTGL3c
xpub         = xpub6H3W6JmYJXN49h5TfcVjLC3onS6uPeUTTJoVvRC8oG9vsTn2J8LwigLzq5tHbrwAzH9DGo6Th
internal_key = cc8a4bc64d897bddc5fbc2f670f7a8ba0b386779106cf1223c6fc5d7cd6fc115
output_key   = a60869f0dbcf1dc659c9cecbaf8050135ea9e8cdc487053f1dc6880949dc684c
scriptPubKey = 5120a60869f0dbcf1dc659c9cecbaf8050135ea9e8cdc487053f1dc6880949dc684c
address      = bc1p5cyxnuxmeuwuvkwfem96lqzszd02n6xdcjrs20cac6yqjjwudpxqkedrcr

// Account 0, second receiving address = m/86'/0'/0'/0/1
xprv         = xprvA449goEeU9okyiF1LmKiDaTgeXvmh87DVyRd35VPbsSop8n8uALpbtrUhUXByPFKK7C2yuqrE
xpub         = xpub6H3W6JmYJXN4CCKUSnriaiQRCZmG6aq4sCMDqTu1ACyngw7HShf59hAxYjXgKDuuHThVEUzdH
internal_key = 83dfe85a3151d2517290da461fe2815591ef69f2b18a2ce63f01697a8b313145
```

```
output_key   = a82f29944d65b86ae6b5e5cc75e294ead6c59391a1edc5e016e3498c67fc7bbb
scriptPubKey = 5120a82f29944d65b86ae6b5e5cc75e294ead6c59391a1edc5e016e3498c67fc7bbb
address      = bc1p4qhjn9zdvkux4e44uhx8tc55attvtyu358kutcqkudyccelu0was9fqzwh

// Account 0, first change address = m/86'/0'/0'/1/0
xprv         = xprvA3Ln3Gt3aphvUgzgEDT8vE2cYqb4PjFfpmbiFKphxLg1FjXQpkAk5M1ZKDY15bmCAHA35jTia
xpub         = xpub6GL8SnQwRCGDhB59LEz9HMyM6sRYoByXBzXK3iEKWgCz8XrZNHUzd9L3AUBELW5NzA7dEFvMa
internal_key = 399f1b2f4393f29a18c937859c5dd8a77350103157eb880f02e8c08214277cef
output_key   = 882d74e5d0572d5a816cef0041a96b6c1de832f6f9676d9605c44d5e9a97d3dc
scriptPubKey = 5120882d74e5d0572d5a816cef0041a96b6c1de832f6f9676d9605c44d5e9a97d3dc
address      = bc1p3qkhfews2uk44qtvauqyr2ttdsw7svhkl9nkm9s9c3x4ax5h60wqwruhk7
```

Reference

- BIP32 - Hierarchical Deterministic Wallets
- BIP43 - Purpose Field for Deterministic Wallets
- BIP44 - Multi-Account Hierarchy for Deterministic Wallets
- BIP49 - Derivation scheme for P2WPKH-nested-in-P2SH based accounts
- BIP84 - Derivation scheme for P2WPKH based accounts
- BIP341 - Taproot: SegWit version 1 spending rules