
# LNP/BP Core Library – BP module

a better library for bitcoin wallets
or what’s missing in rust-bitcoin, miniscript & BDK

**LNP/BP Standards Association**
Prepared & supervised by **Dr Maxim Orlovsky**

Sponsored by Pandora Core AG

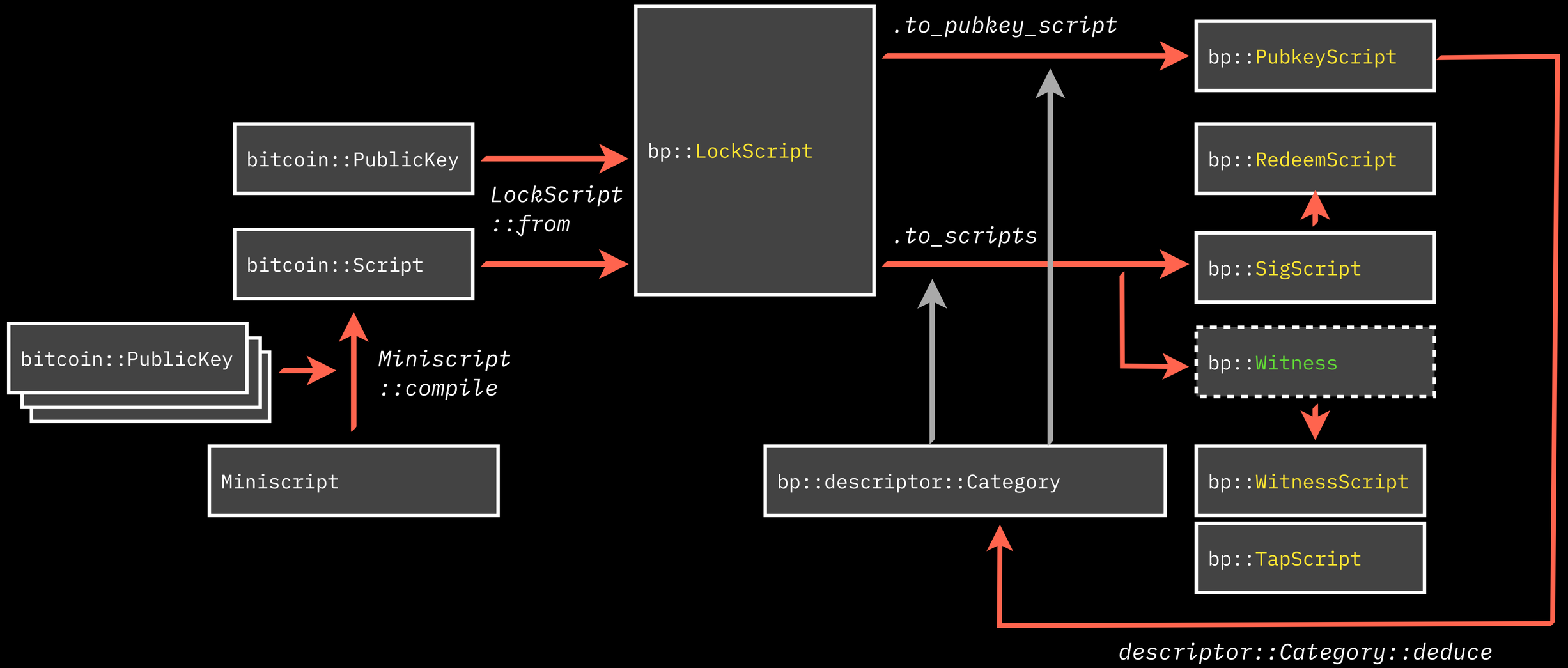
# Main components

- **Deterministic bitcoin commitments**: public key pay-to-contract tweaks
  - based on LNPBP1-3 standards
  - support for arbitrary complex scripts / tx formats
  - ready for Taproot
- **Strict scripting types** & their conversions
- Better **BIP32** + SLIP32 bitcoin extensions
- Bitcoin **descriptors** beyond what miniscript allows
- **PSBT** improvements (migrated to rust-bitcoin already)
- Universal **bitcoin invoices**

# Other components

- Short bitcoin ids (LNPBP-5): will be used by BP Node
- Lexicographic ordering primitives
- Hash-lock data types
- Blind UTXOs (used in new invoices)

# bp::scripts – strict script type system



# BIP32/SLIP32 improvements (bp::bip32, slip32)

- DerivationComponents: complete information on derivation when extended private key is not required
- Safe hardened and unhardened derivation path steps
- Multiple routines for working with complex derivation path and fail-safe derivations
- Derivation templates constructed of derivation steps
- Xpub/xpriv serialization (migrated to rust-bitcoin)
- Support for other xpub/xpriv formats: y, z, t, u, Y, Z, T, U

# Derivation components

- Allows to skip hardened part of the derivation path so that derivation may happen without extended private key available
- Used in Bitcoin Pro, MyCitadel
- Can be serialized as a standard string & be placed inside miniscript or other type of descriptor

```
[master_xpub]/
branch_path=[branch_xpub]/
terminal_path/index_ranges
```

- Master Extended public key
- Hardened part of derivation path (“branch path”)
- Extended public key at the end of branch path
- Unhardened part of the derivation path (“terminal path”)
- Optional list of ranges that has being already used

# Descriptors: more than in miniscript

- Based on strict script type system (bp::scripts mod)
- Descriptor categories (will be migrated to miniscript)
- Compact & extended forms of descriptors
  - Compact: have only hash of the data
  - Extended: complete source code
    (correspond to Bitcoin Core;
    can work w/o miniscript, which is required for LN)
- Deducting descriptors from a given pubkey script

# Descriptor categories

Descriptor alternative when we need to abstract from whether we use a single public key or a script

- **Bare**: ancient bare scripts (including OP_RETURN) of P2PK
- **Hashed**: P2PKH & P2SH
- **SegWit**: P2WPKH & P2WSH
- **Nested**: P2WPKH & P2WSH inside P2SH
- **Taproot**: P2T

# Script templates & generators

- We can’t be satisfied with miniscript only, since it does not cover 100% of bitcoin scripts existing today
- Script template is an extension of miniscript for such cases
- Script generator is a factory for building pubkey scripts for a given *script construction* & a ***set*** of descriptor categories

# Script construction structure

- **SingleSig**: either
  - Miniscript “**DescriptorSinglePub**” - a pubkey with optional key derivation source
  - **XpubDerived** instructions in a form of **DerivationComponents**
- **MultiSig**: threshold + a set of SingleSig pubkeys and optional lexicographic ordering flag
- **Scripted**:
  - Script binary data in form of either:
    - Script template: bitcoin script source allowing instructions for key derivation
    - Miniscript with SingleSig keys
    - Miniscript concrete policy with SingleSig keys
  - Optional text script source for code formatting
- **MuSigBranched**: for the future MuSig & TapScript branch selection