Taproot & Schnorr: status update

Building layer 1, 2 & 3 implementation with
LNP/BP Standard Association, funded by Pandora Core AG

Dr Maxim Orlovsky

Text version

Taproot activation

Overview About Taproot Settings

LOCKED IN!

Taproot has been locked in!

Taproot state: we are at very early stage

- ☑ Schnorr signatures: Bitcoin Core, libsecp256k1
- ☑ SegWit v1 outputs: Bitcoin Core
- ☑ Tapscript: Bitcoin Core?
- ☑ Bech32m: Bitcoin Core, Rust Bech32m, Rust Bitcoin (PR WIP)
- ☑ Addresses (P2TR): Bitcoin Core
- ☑ Descriptors `` `tr(…)` ``: only public key in Bitcoin Core
- ☑ PSBT: none, zero standard proposals
- ☑ HD derivation: none, one standard proposal
- ☑ Multisig: MuSig2 in libsecp256k1-zpk (PR WIP)

Need broader support:

- Amendments into BIPs for PSBTs, HD derivation paths
- BIP standards for key tweaking and client-side data handling
- Miniscript support for tr descriptors and Tapscript
- Rust versions for: bech32m, bitcoin, secp256k1, miniscript
- Support by hardware wallets
- Critical software infrastructure: Electrum Server, Esplora, HWI…
- Lightning network support

LNP/BP Standards Association and Pandora Core are one of main contributors into Taproot implementation and ecosystem at layer 1, 2 & 3

LNP/BP Association Efforts:

- Completing rust secp256k1 implementation for BIP-340 keys
- Driving rust bitcoin taproot implementation for last half of year
  github.com/orgs/rust-bitcoin/projects/3
- Initiates work on bringing Taproot & miniscript to LN
- Making RGB taproot-ready from day 0
- Universal LNP/BP invoices (LNPBP-38) providing pay-to-descriptor option

MyCitadel wallet & Citadel Runtime will support Taproot single-key outputs with the next release this summer

brought by Pandora Core AG

General Taproot pending TODOs

- ☐ Adding descriptors for Taproot output capable of working with Tapscript (BIP-342)
- ☐ Tapscript modifications to miniscript standard
- ☐ Standard for HD wallet key derivation using Schnorr signatures
- ☐ PSBT support for Schnorr signatures / keys
- ☐ Safe MuSig standard & its implementation
- ☐ PSBT support for MuSig schemes

Roadmap for Taproot in Rust Bitcoin

github.com/orgs/rust-bitcoin/projects/3

github.com/rust-bitcoin/rust-bitcoin/issues/503

- ☑ BIP-350 Bech32m encoding in **bech32** and **bitcoin-bech32**
- ☑ BIP-340 tagged hashes in **bitcoin_hashes**
- ☑ BIP-340 Schnorr keys & signatures in **secp256k1** and **bitcoin**
- ■ BIP-341 outputs with SegWit v1 (+addresses) in **bitcoin**
- ☐ BIP-342 signing process: **bitcoin**

MVP

Needed later

- ☐ BIP-342 tapscript support: **bitcoin**
- ☐ Descriptors support **miniscript**
- ☐ PSBTs support in **miniscript**, **bitcoin**
- ☐ MuSig2: one day, first in **secp256k1-zpk**, than in **bitcoin** & **miniscript**
- ☐ Signature validation in **bitcoinconsensus**

Client-side-validated Tapscript handling

Very much inline with RGB requirements

- [ ] Public key tweak inclusion into all layers
- [ ] Support by hardware wallets
- [ ] Backup infrastructure (also related to LN watchtowers)

Protocols requiring client-side data

- Lightning network before Eltoo (storing signatures for revoked transactions)
- RGB (storing client-side-validated data)
- Taproot (storing Tapscripts)

- Data storage in wallet
- Data backup (critical as for private keys)
- Watchtowers must account for these data
- Hardware wallets must support client-side key tweaks
- Need for custom derivation schemes with dedicated “change” path segments

Lightning network + Taproot = ❤️

Lightning network + Taproot = ❤️

… and miniscript, RGB, DLCs

What Minscript gives to LN

- Smaller tx size:

  for offered HTLC output we decrease ***scriptPubkey*** from 156 to 131 bytes (-16%) and ***witness*** from 104 down to 68 bytes (-34%) in cooperative cases

- Compatibility with descriptor-based wallets

- Ability to negotiate custom tx inputs/outputs as Miniscript descriptors, not arbitrary bitcoin script

  enables deterministic analysis

- Simpler backups for channel state when custom outputs or tweaks are present

What Taproot & Schnorr gives to LN

- Even smaller tx size:
  for offered HTLC output we decrease ***scriptPubkey*** from 156 to 34 bytes (-78%) and ***witness*** up to 33 bytes (except penalty transactions)

- Onchain privacy:
  non-penalty channel openings and closings are not seen

- Faster signatures for channel updates

- Ability to move from vulnerable HTLC to PTLCs with adaptor signatures

What Taproot & Schnorr gives to RGB & LN

- No need to store scripts in RGB data for LN outputs (still need to store in Taproot client-side data)
- Public key tweaks become a part of the common wallet infrastructure

What we customize in LN tx structure with L3

- RGB: Adding tweaks

  both sides of the same channel must support RGB

- DLC:

  - adaptor signatures

  - custom outputs (already WIP with bi-directional funding PR)

LN upgrade for Miniscript, Taproot, RGB, DLC

We can do four upgrades at once

- Decide on combined miniscript/taproot LN update
- Propose feature flag(s) for their support
  + propose extendable feature flag standard
- Propose new script output structure
  + propose pubkey tweak negotiation standard
- Finalize custom tx in/output negotiation
  + propose negotiation of custom tx spending those outputs

Relevant PRs to BOLT’s

Not directly Taproot/miniscript related, but may help in adoption

- Interactive tx protocol
  https://github.com/lightningnetwork/lightning-rfc/pull/851
- Quiescence channel updates
  https://github.com/lightningnetwork/lightning-rfc/pull/869/files
- Channel upgrades
  https://github.com/lightningnetwork/lightning-rfc/pull/868/files

Aiming to make LNP Node the first LN implementation supporting both Taproot & RGB