```
BIP: 372
Layer: Applications
Title: Pay-to-contract tweak fields for PSBT
Author: Maxim Orlovsky <orlovsky@lnp-bp.org>
Discussions-To: <bitcoin-dev@lists.linuxfoundation.org>
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0372
Status: Draft
Type: Standards Track
Created: 2022-01-16
License: BSD-2-Clause
Requires: BIP-174
```

Introduction

Abstract

This document proposes additional fields for BIP 174 PSBTv0 and BIP 370 PSBTv2 that allow for pay-to-contract key tweaking data data to be included in a PSBT of any version. These will represent an extra-transaction information required for the signer to produce valid signatures spending previous outputs.

Copyright

Background

Key tweaking is a procedure for creating a cryptographic commitment to some message using elliptic curve properties. The procedure uses the discrete log problem (DLP) to commit to an extra-transaction message. This is done by adding to a public key (for which the output owner knows the corresponding private key) a hash of the message multiplied on the generator point G of the elliptic curve. This produces a tweaked public key, containing the commitment. Later, in order to spend an output containing P2C commitment, the same commitment should be added to the corresponding private key.

This type of commitment was originally proposed as a part of the pay to contract concept by Ilja Gerhardt and Timo Hanke in [1] and later used by Eternity Wall [2] for the same purpose. Since that time multiple different protocols for P2C has been developed, including OpenTimeStamps [3], Elements sidechain P2C tweaks [4] and LNPBP-1 [5], used in for constructing Peter Todd's single-use-seals [6] in client-side-validation protocols like RGB.

Motivation

P2C outputs can be detected onchain and spent only if the output owner not just knows the corresponding original private key, but also is aware about P2C tweak applied to the public key. In order to produce a valid signature, the same

tweak value must be added (modulo group order) to the original private key by a signer device. This represents a challenge for external signers, which may not have any information about such commitment. This proposal addresses this issue by adding relevant fields to the PSBT input information.

The proposal abstracts details of specific P2C protocols and provides universal method for spending previous outputs containing P2C tweaks, applied to the public key contained within any standard form of the `scriptPubkey`, including bare scripts and P2PK, P2PKH, P2SH, witness v0 P2WPKH, P2WSH, nested witness v0 P2WPKH-P2SH, P2WSH-P2SH and witness v1 P2TR outputs.

Design

P2C-tweaked public keys are already exposed in the `PSBT_IN_REDEEM_SCRIPT`, `PSBT_IN_WITNESS_SCRIPT`, `PSBT_IN_TAP_INTERNAL_KEY` and `PSBT_IN_TAP_LEAF_SCRIPT` fields; the only information signer is needed to recognize which keys it should sign with is from which of the original keys they were generated. This is achieved by introducing new 'PSBT_IN_P2C_TWEAK' field which has the original key as a field key and the tweak as a field value. The signer will recognize the keys which are available to it, apply the tweak to them and see in which scripts it was used -- and use this information to apply tweaks for the corresponding private keys and produce valid signatures.

Specification

The new per-input type is defined as follows:

Name		Description
P2C Key Tweak	`PSBT_IN_P2C_TWEAK = 0x19`	33 bytes of compact public key serialization specifying to

Security considerations

The scope of this proposal is deliberately kept narrow; it addresses only spending of transaction outputs containing P2C tweaks - and does not addresses construction of a new P2C commitments or transactions containing them in their outputs.[2]

[1] **Why compressed public keys are not distinguished from BIP-340 public keys**We follow the logic of BIP32 key derivation which does not performs that distinguishment. The type of the key is defined by the input type, and adding additional PSBT field type will just create the need for handling errors when the input type does not match the provided key type.

[2] **Why only spending of P2C tweaked outputs is covered** P2C tweaks commit to external data, some of which may represent certain value (like in some sidechains, single-use-seal applications like RGB etc). Creation of such outputs much allow hardware devices to understand the structure of such extra-transaction data, which may be in different formats and constantly involve. Thus, this should be addresses with a separate standards (or be a vendor-based). The current proposal only touches the question of spending an output which

Rationale

Compatibility

The proposal is compatible with the existing consensus rules and does not require any of their modifications.

The proposed P2C PSBT fields provides sufficient information for creating a valid signatures for spendings of the following output types containing tweaked public keys: - bare scripts, - P2PK, - P2PKH, - P2SH, - witness v0 P2WPKH and P2WSH, - nested witness v0 P2WPKH-P2SH and P2WSH-P2SH,

Post-0 witness versions, including taproot outputs and future witness versions, may not be supported or covered by this BIP and may require addition of new fields to the PSBT inputs.

Reference implementation

WIP

Acknowledgements

Author is grateful to Andrew Poelstra, who provided an initial set of ideas and information on his previous work on the topic basing on which this standard was designed.

Test vectors

TBD

References

[1] Ilja Gerhardt, Timo Hanke. Homomorphic Payment Addresses and the

Pay-to-Contract Protocol. arXiv:1212.3257

$cs.CR$

<https://arxiv.org/pdf/1212.3257.pdf>

[2] Eternity Wall's "sign-to-contract" article.

<https://blog.eternitywall.com/2018/04/13/sign-to-contract/>

[3] Peter Todd. OpenTimestamps: Scalable, Trust-Minimized, Distributed

Timestamping with Bitcoin.
<https://petertodd.org/2016/opentimestamps-announcement>

contained previously created P2C commitment, which does not creates a new commitment and does not provides that kind of risk of extra-blockchain value loses.

[4] Adam Back, Matt Corallo, Luke Dashjr, et al. Enabling Blockchain

Innovations with Pegged Sidechains (commit5620e43). Appenxix A. <https://blockstream.com/sidechains.pdf>;.

[5] Maxim Orlovsky, Rene Pickhardt, Federico Tenga, et al. Key

tweaking: collision- resistant elliptic curve-based commitments. LNPBP-1 Standard.
<https://github.com/LNP-BP/LNPBPs/blob/master/lnpbp-0001.md>

[6] Peter Todd. Single-use-seals. LNPBP-8 Standard.

<https://github.com/LNP-BP/LNPBPs/blob/master/lnpbp-0008.md>