What OP_CANT We Do With OP_CAT

Andrew Poelstra
Director, Blockstream Research
March 12, 2024

Blockstream

Contents

- Covenants.
- Everything else.

Covenants

Covenants, broadly, are about constraining the transaction context.

- `CSV` and `CLTV` do this in a super limited way.
- `CHECKSIG` et al appear to be even more limited.
- . . . but are they?

Covenants

(Tap)script is a language with 255 opcodes, in principle.

- Roughly 90 just push stuff, 90 are insta-fail/insta-suceed.
- Of the 80 or so “real” opcodes, exactly 5 access the transaction context:
  - `CHECKSIG`, `CHECKSIGVERIFY`, `CHECKSIGADD`
  - `CHECKSEQUENCEVERIFY`, `CHECKLOCKTIMEVERIFY`
  - `CAT` would not.

Covenants

In a Blockchain context, we don't compute but verify. There are not inputs and outputs but statments and witnesses.

Covenants

So `CHECKSIGVERIFY` doesn't take a signature/message/pubkey and output pass/fail.

Instead it makes a claim about the relationship between these three objects, and the objects themselves are the witness.

The claim is not just "this key signed this message" but something algebraically specific.

And `OP_CAT` lets us be more specific.

Covenants

The Schnorr signing equation is $s = k + ex$, where x is a secret key, k is the "secret nonce" and e is a hash of our transaction data.

- Setting the pubkey to G forces $x = 1$.
- With `CAT` we can also force $k = 1$.
- Then $s = e + 1$.

Using `CAT` again we can reconstruct e using explicit transaction data, and compensate for that annoying $+1$.

Everything Else

`CAT` can also be used to implement:

- Big-integer arithmetic from 4-byte arithmetic (`ADD` and `SUB` anyway).
- Verifying PoW in Script.
- Lexicographic comparisons of arbitrary-length strings; simulating `LEXCAT`.
- One-time Schnorr signatures (which reveal secret keys).

Everything Else

It can be used to implement Lamport or Winternitz one-time signatures.

- Lamport public keys consist of 256 pairs of hashes.
- Signatures hash a message and reveal one preimage for each bit of the hash.
- Can be done in Script with `CAT`. (Jeremy Rubin 2021)
- Can sign transaction data *or* arbitrary messages.

Everything Else

Can be used to implement Merkle trees, which have some direct applications:

- Tree signatures (multisigs with millions or billions of keys).
- Trees of verification conditions (using `SIZE` to distinguish keys, hashes, etc).
- Compressing BitVM into fewer transactions.
- c.f. `MERKLEBRANCHVERIFY` (BIP 116) (Friedenbach, Alm, BtcDrak 2017)

Everything Else

Can be used to implement Merkle trees, which have some indirect applications:

- Any function with a small enough domain can be precomputed as a Merkelized lookup table (e.g. 16-bit multiplication/division).
- Can extend 16-bit math to bignum math by splitting large values.
- Modular arithmetic? EC ops? ZK verifiers?
- Cut-and choose protocols?

Thank you

Original blog post about CAT covenants

`https://wpsoftware.net/andrew/blog/cat-and-schnorr-tricks-ii.html`

Implementation by rot13maxi

`https://github.com/taproot-wizards/purrfect_vault`

I am Andrew Poelstra `catman@wpsoftware.net`