BitcoinのOP CHECKTEMPLATEVERIFY

bips/bip-0119.mediawiki at master · bitcoin/bips

OP_CHECKTEMPLATEVERIFY - Bitcoin Optech

ChatGPT

`OP_CHECK～VERIFY` 系の提案である。

2020年6月に BIP-119 になっていて、現在も Draft のままである。

シンプルな支出制限といっているが、なんか私にはわかりにくい。。。

スタックにある 32バイトデータと「テンプレート」のハッシュ値を比較する。

BIP-119 の擬似コードから引用した「テンプレート」の作り方。

なんとなく意味が分かると思う。

`scriptSigs` が入っていて `witness` が入っていないのは、segwit 以降だけ有効にしたいためらしい。

`template`

```
   # Serialize as 4 byte unsigned integer (little endian)
   r += struct.pack("<i", self.nVersion)
   # Serialize as 4 byte unsigned integer (little endian)
   r += struct.pack("<I", self.nLockTime)
   # we do not include the hash in the case where there is no
   # scriptSigs
   if "scriptSigs" in precomputed:
       r += precomputed["scriptSigs"]
   # Serialize as 4 byte unsigned integer (little endian)
   r += struct.pack("<I", len(self.vin))
   r += precomputed["sequences"]
   # Serialize as 4 byte unsigned integer (little endian)
   r += struct.pack("<I", len(self.vout))
   r += precomputed["outputs"]
   # Serialize as 4 byte unsigned integer (little endian)
   r += struct.pack("<I", nIn)

   SHA256(r)
```

これがあると、 `OP_CAT` でのコベナンツでやろうとしていたことが面倒なスクリプトを書かずにできるようになる。
コベナンツの目的は `OP_CAT` を使うことでも `e` を算出することでもなく、出力先を絞りたいというところにあるので、こちらの方が柔軟性はないものの用途が限定される分、BIPとして通りやすいと思う。
BitcoinのOP_CAT#6caa29

ただ、用途が限定されるトランザクションはそもそもよいのか、というところはどうしても残る。