APO Vault

Covenants For Dummies

Goals

- Explore how BIP-118 could be used to enable covenants.
- Build a simplified Vault system derived from current Vault proposals combined with BIP-118 sighashes and Taproot tapscripts.
- Get feedback on how this approach compares with other covenant proposals and if it can be improved - possibly through tweaks to BIP-118.

DISCLAIMER: I am not an expert, or even well versed in the covenant and vaults literature!

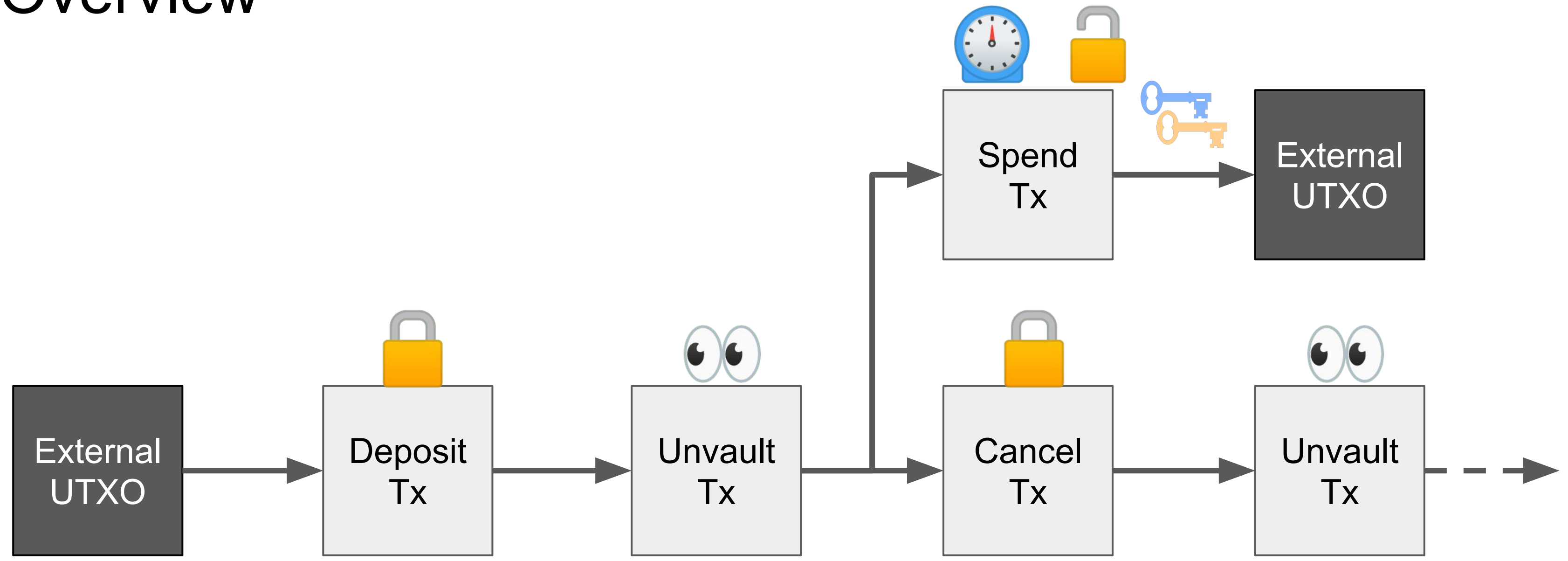
The vault scheme I describe here is for research and educational purposes only. Links to more information are given at the end of this presentation.

How It Works

1. Funds are sent to a Taproot output that can only ever be spent by an Unvault Tx.
2. The Unvault Tx is a covenant that has a Taproot output that can only ever be spent by a Spend Tx or a Cancel Tx.
3. The Spend Tx can only be spent after a delay from when the Unvault Tx is committed.
4. During this delay, the Cancel Tx can instead be spent from the Unvault Tx, without any delay.
5. The Cancel Tx Taproot output can only ever be spent by the Unvault Tx.
6. The Spend Tx output can be any external address.

Loosely Based on github.com/revault/practical-revault/blob/master/revault.pdf

Overview



Loosely Based on github.com/revault/practical-revault/blob/master/revault.pdf

Covenant Key 🗝️

(X,x) is a Schnorr public/private key pair used to sign covenant transactions

Pubkey X could be created with Musig2(A,B,C…)

Privkey x could be deleted after the vault transactions are created

Pubkey X is the internal public key used to create all Taproot covenant outputs

Privkey x is used to sign all Taproot covenant transactions

Signatures created with privkey x are included in the actual tapscripts themselves instead of the witness script. This enables covenants because outputs commit to a specific transaction, **including the transaction’s outputs and CSV spending delay**.

Covenant Script

1. Check that Tx was signed with internal covenant pubkey X
2. APO Tapscript replaces `0x01` with the internal Taproot pubkey X
3. `[Signature]` is created from a Tx signed with the covenant privkey x using `ANYPREVOUTANYSCRIPT`
4. A covenant Tx prepends `[Signature]` to the script itself instead of revealing in the witness
5. Because of `ANYPREVOUTANYSCRIPT`, the `[Signature]` is valid for any Tx with the same outputs, timelocks and Taproot internal pubkey, **even though the taproot scripts are different**.

```
[Signature]
[Leaf Script]
[Leaf Control Block]
```

Taproot Witness

```
0x01
OP_CHECKSIGVERIFY
```

Taproot Leaf Script

```
[Covenant Leaf Script]
[Leaf Control Block]
```

Covenant Taproot Witness

```
[Signature]
0x01
OP_CHECKSIGVERIFY
```

Covenant Taproot Leaf Script

Spending Key

(Y,y) is a Schnorr public/private key pair used to spend value out of the vault

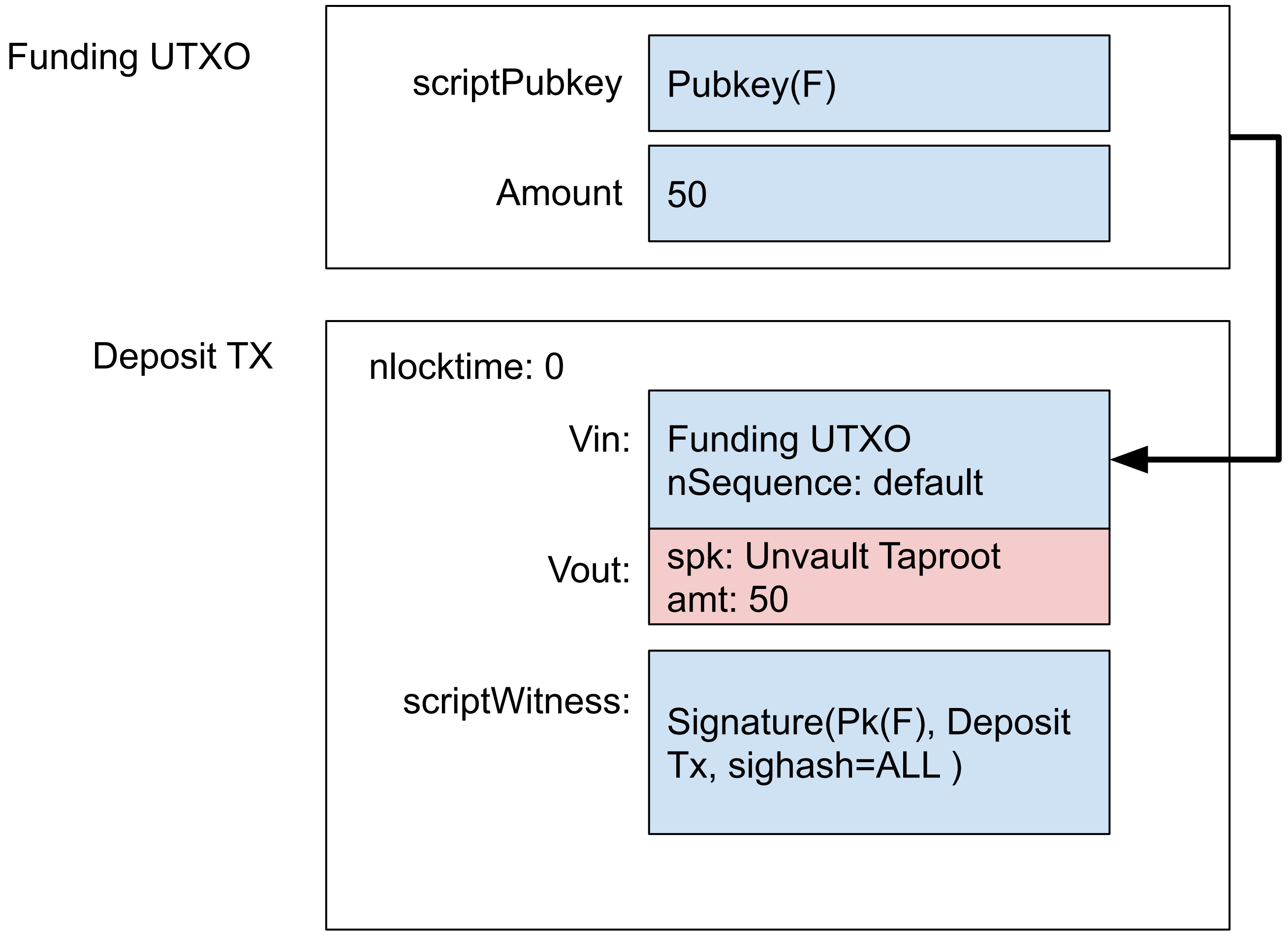
Pubkey Y could be created with Musig2(A,B)

Privkey y only needs a hot-wallet level of security used for routine spending

Privkey y is only used to sign transactions that spend from the vault to an externally owned UTXO

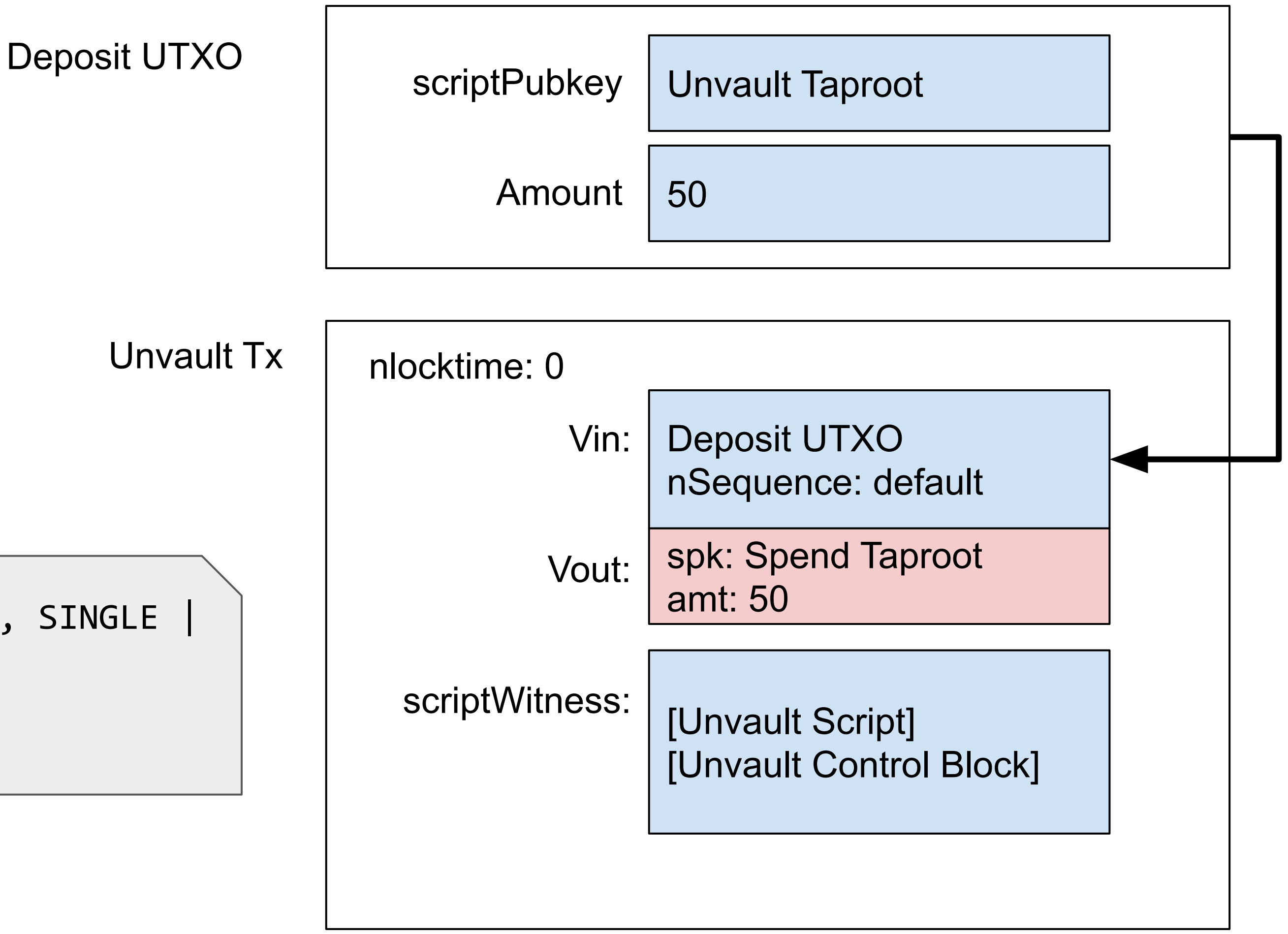
Signatures created with privkey y are added to the witness script **at spending time**, not when the vault is setup and are not included in the script itself like the vault covenant signature.

Fund Vault


Funding UTXO
scriptPubkey
Pubkey(F)
Amount
50
Deposit TX
nlocktime: 0
Vin:
Funding UTXO
nSequence: default
Vout:
spk: Unvault Taproot
amt: 50
scriptWitness:
Signature(Pk(F), Deposit Tx, sighash=ALL)

Unvault

👀



```
[Signature(X, Unvault Tx, SINGLE |
ANYPREVOUTANYSCRIPT)]
0x01
OP_CHECKSIGVERIFY
```

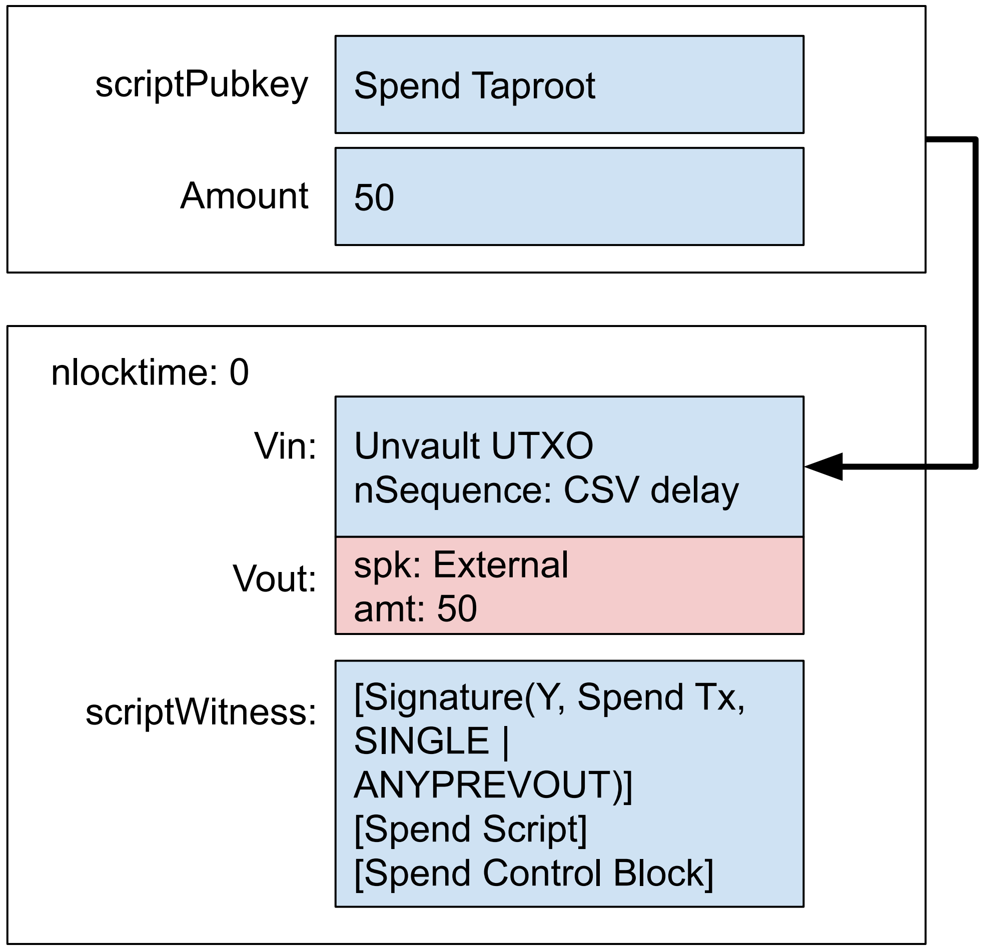
Covenant Taproot Leaf Script
(Unvault)

Spend

🕛 🔓

Unvault UTXO

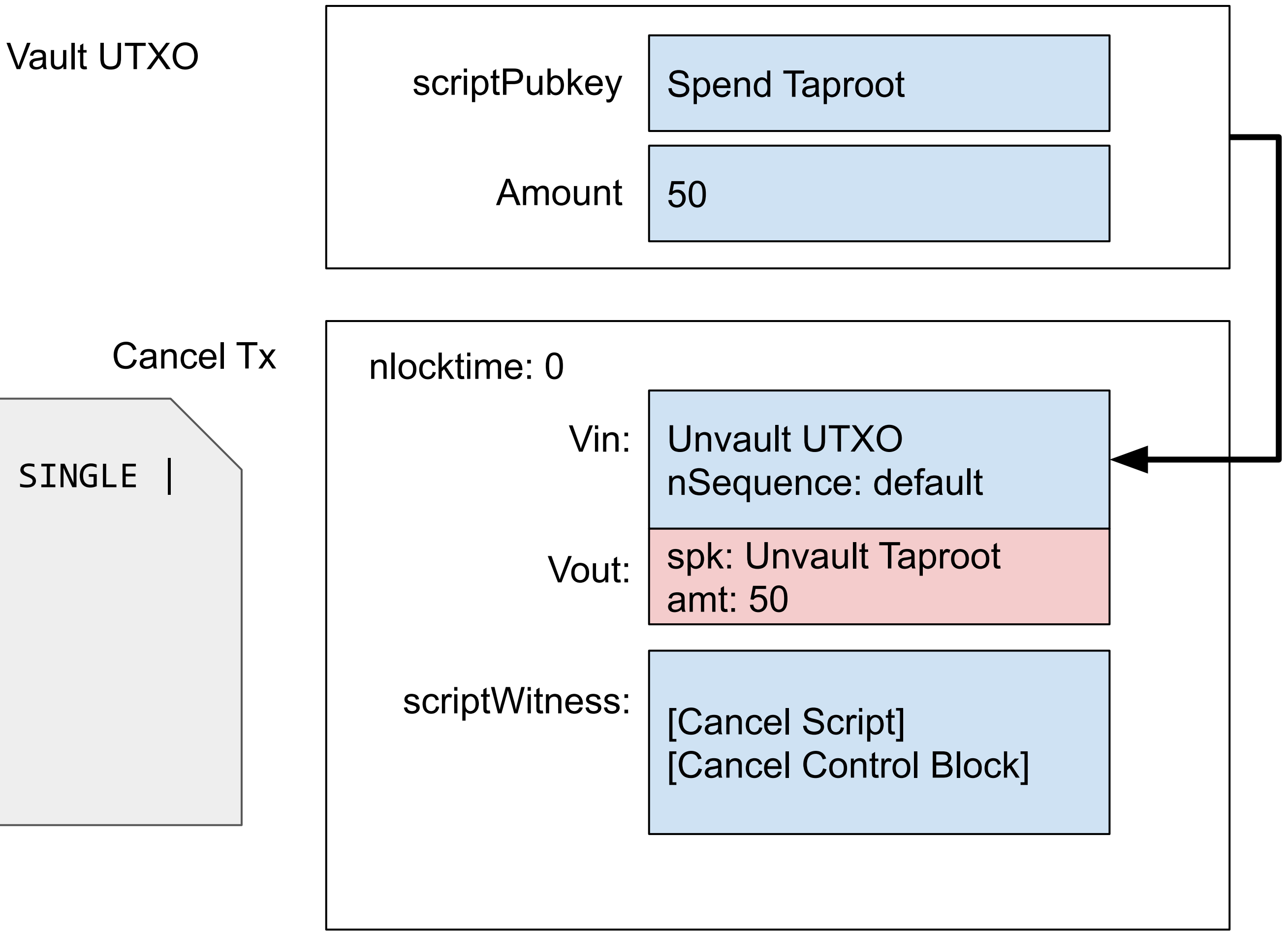
Spend Tx



```
[Signature(X, Spend Tx, SINGLE |
ANYPREVOUTANYSCRIPT)]
0x01
OP_CHECKSIGVERIFY
[Pubkey(Y)]
OP_CHECKSIGVERIFY
CSV_DELAY
OP_CHECKSEQUENCEVERIFY
```

Covenant Taproot Leaf Script
(Spend)

Cancel



```
[Signature(X, Cancel Tx, SINGLE |
ANYPREVOUTANYSCRIPT)]
0x01
OP_CHECKSIGVERIFY
```

Covenant Taproot Leaf Script
(Cancel)

Further Reading

- Kanzure described a scheme in 2019 on the bitcoin-dev mailing list for creating vaults that do not require, but would benefit from, SIGHASH_NOINPUT.
- The Revault team has proposed a similar scheme for vaults that do not require covenants, but could potentially benefit from them.
- You can read more about covenants in the 2020 paper, Bitcoin Covenants: Three Ways to Control the Future.
- Python code to test this scheme can be found here (incomplete, still in progress!)