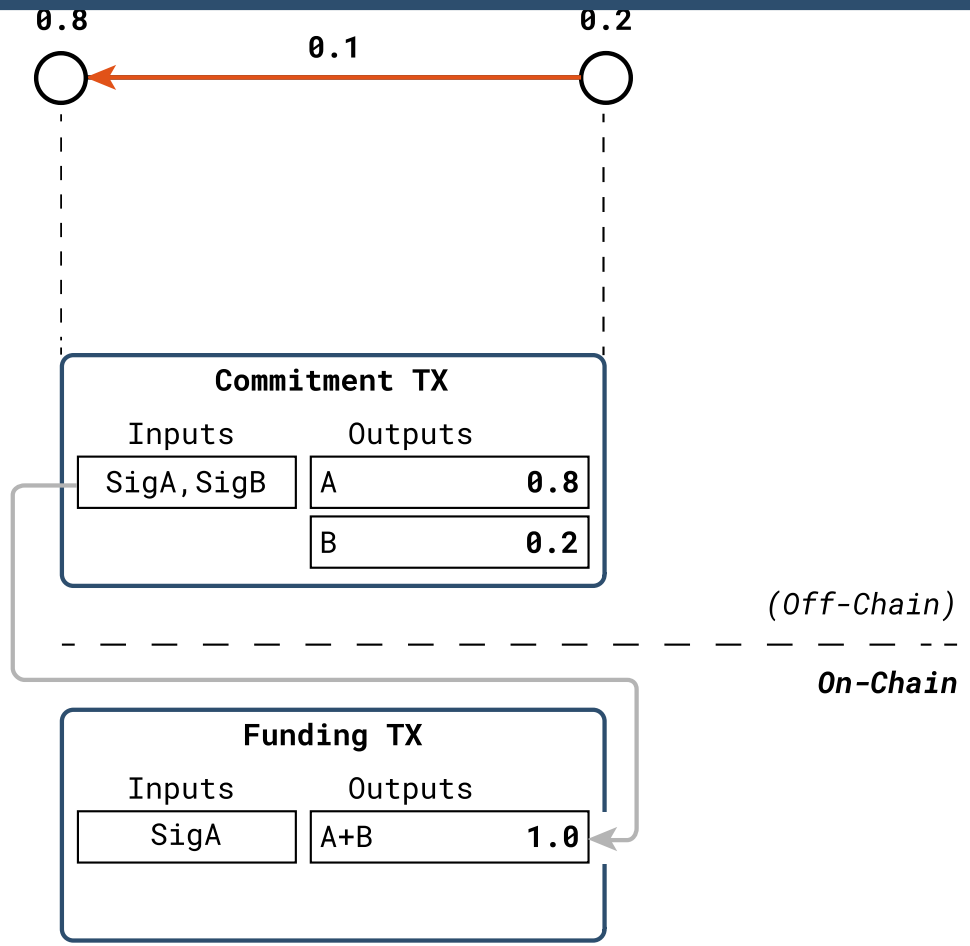
Introduction to **Payment Channels**



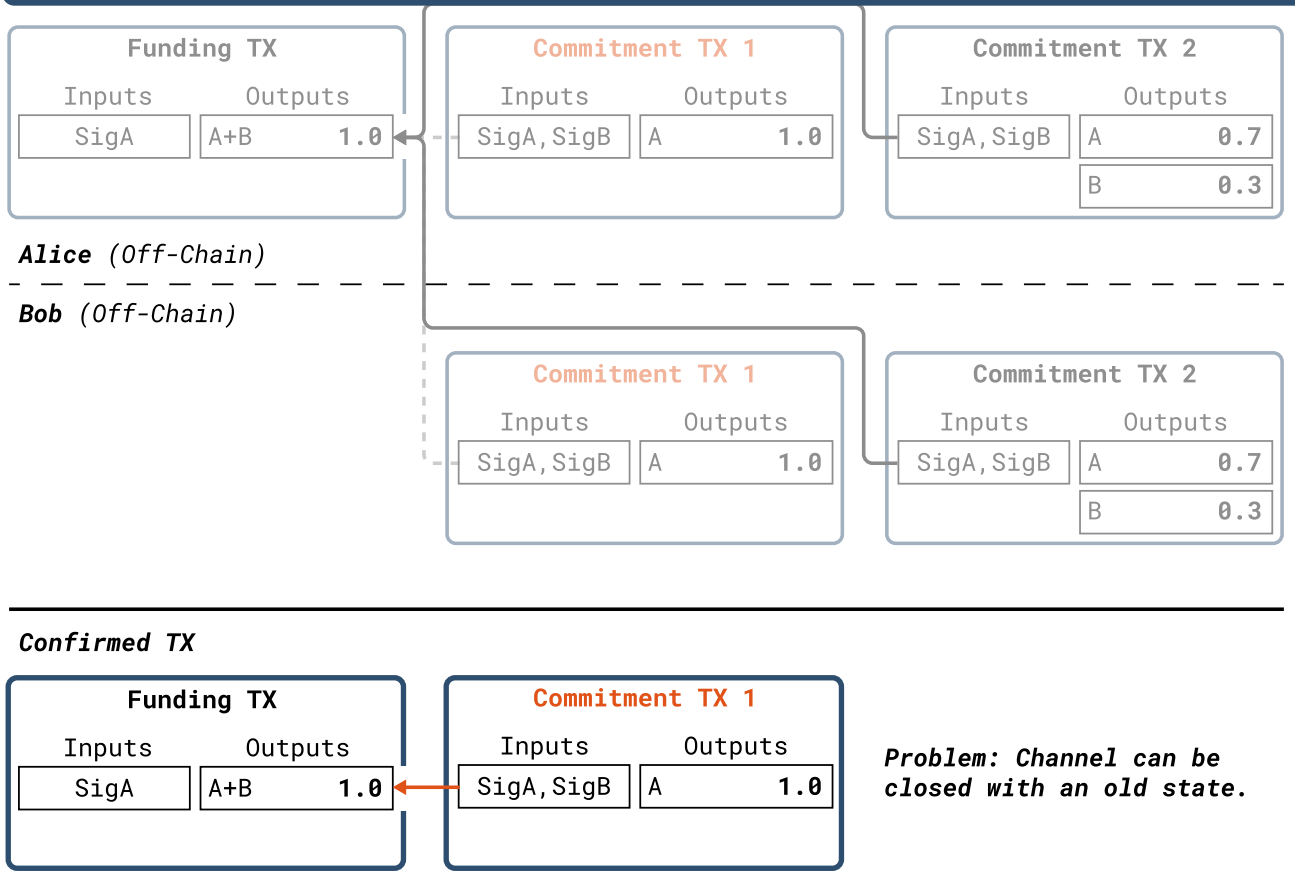
Alice has funded the channel

- Alice signs a funding transaction which spends to a 2-of-2 multisig output.
- Once the funding transaction is broadcast, the payment channel is open.

Both parties sign the commitment TX.

- Requires signatures of both parties.
- Commitment captures channel balance.
- Is updated with each new channel payment.
- Is not broadcasted until channel closure.

Payment Channel: Naive Implementation



Funding TX and Commitment TX 1

- Funding TX signed by A
- Commitment TX 1 signed by both parties

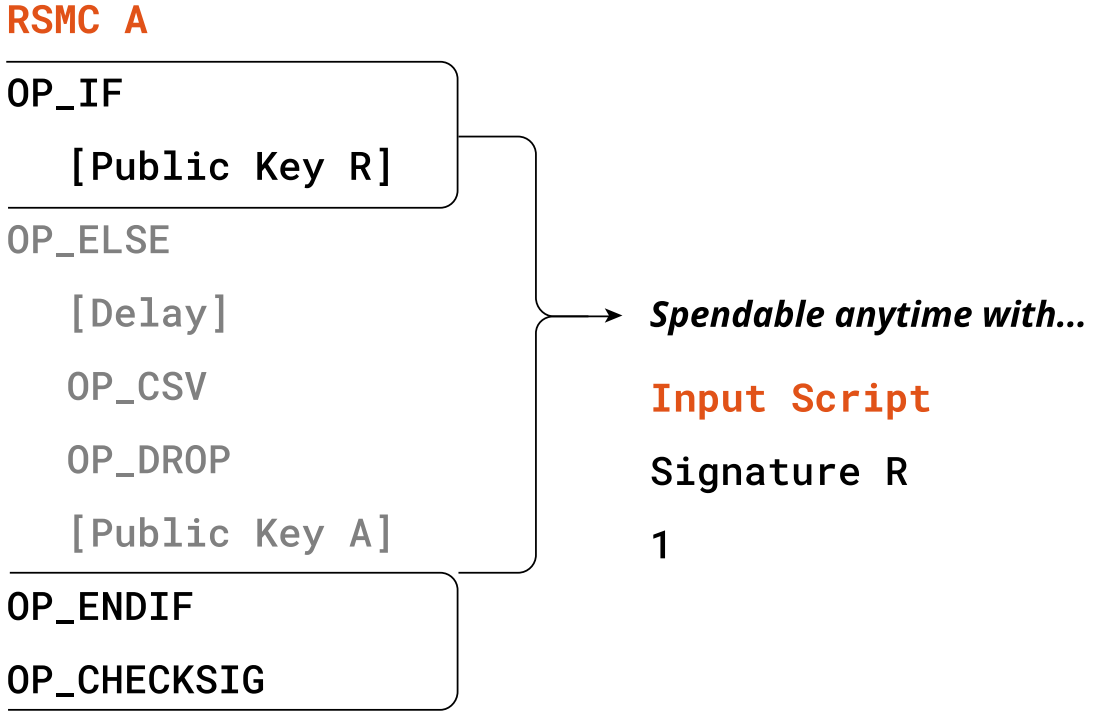
Valid Channel Close

- Latest commitment TX is broadcast by either party

Problem: Invalid Channel Close

- Nothing prevents either party from broadcasting old state

Revocable Sequence Maturity Contract



Misnomer: not really revocable

Spendable after time-out

- RSMC A is spendable by A after a delay

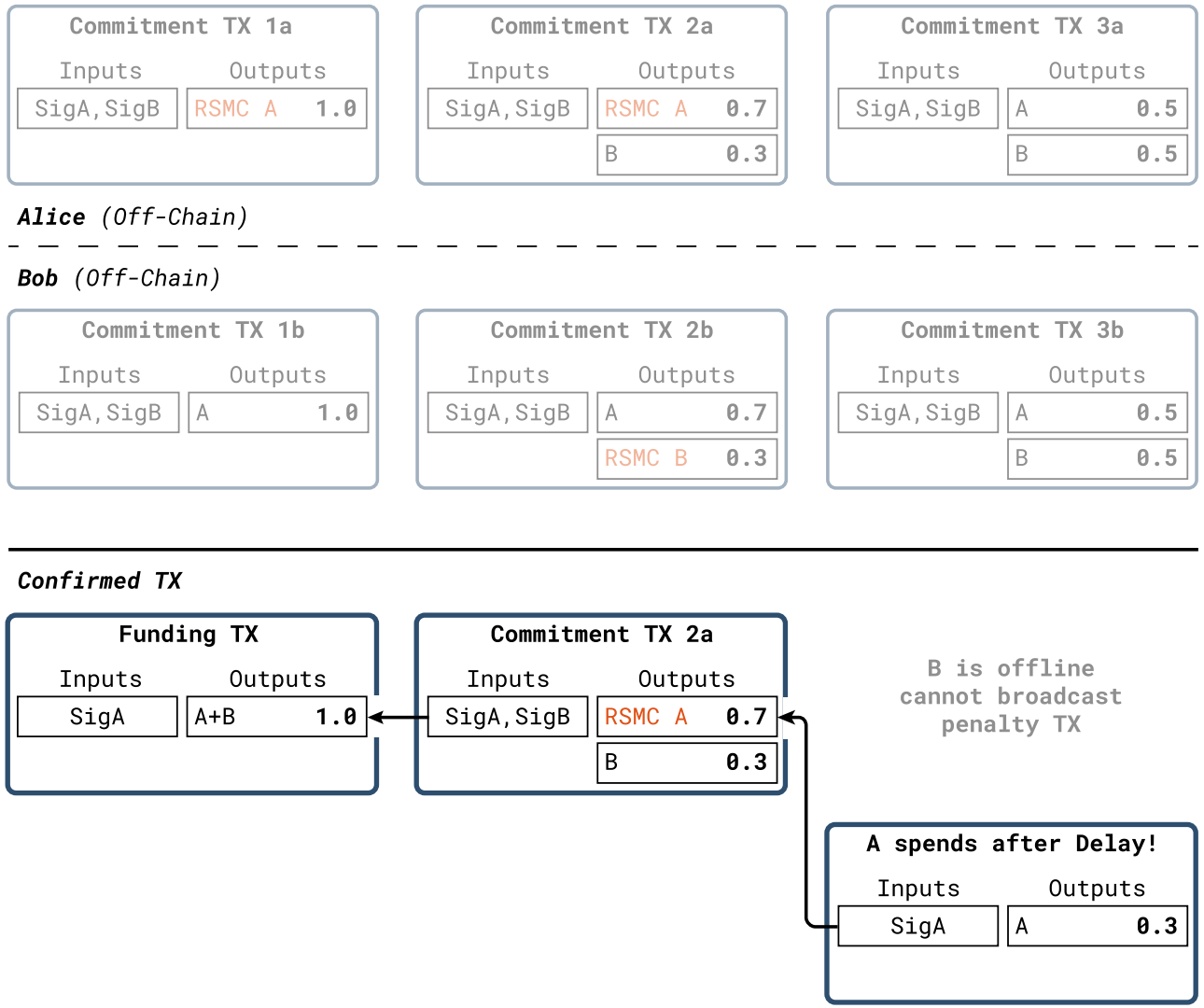
Spendable by secret R

- Public key R is derived as: $R = R_{\text{loc}} + R_{\text{rem}} = (r_{\text{loc}} + r_{\text{rem}})\text{G}$
- Public key R is spendable with: $r_{\text{loc}}, r_{\text{rem}}$
- r_{rem} is revealed by counterparty as revocation

A party can revoke its RSMC output:

- By sharing its secret with counterparty.

Payment Channel with RSMC's



Old states are revoked

- RSMC can be spent by counterparty

Old broadcast states are swept

- Penalty TX by counterparty broadcast

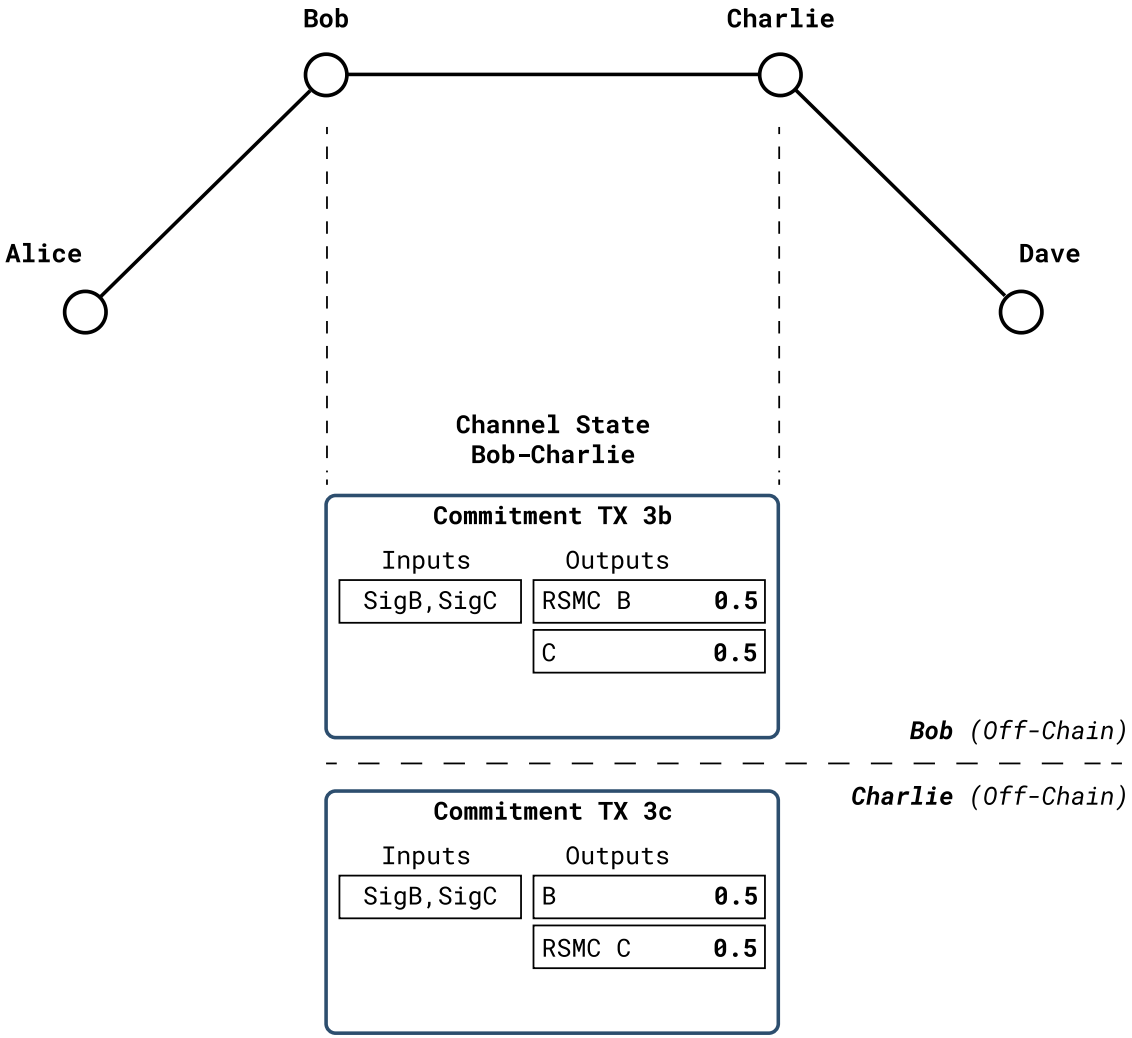
Chain needs to be watched

- Both parties must monitor chain for invalid commitment transactions

Watchtower

- Channel participant outsources the monitoring of commitment transactions

Introduction to **Payment Channel Routing**



1) Alice wants to send a payment to Dave.

- Dave sends `HASH160(payment_preimage)` to Alice, for a unique `payment_preimage`

2) Alice constructs a route to Dave.

- Onion routes `HASH160(payment_preimage)` along route.

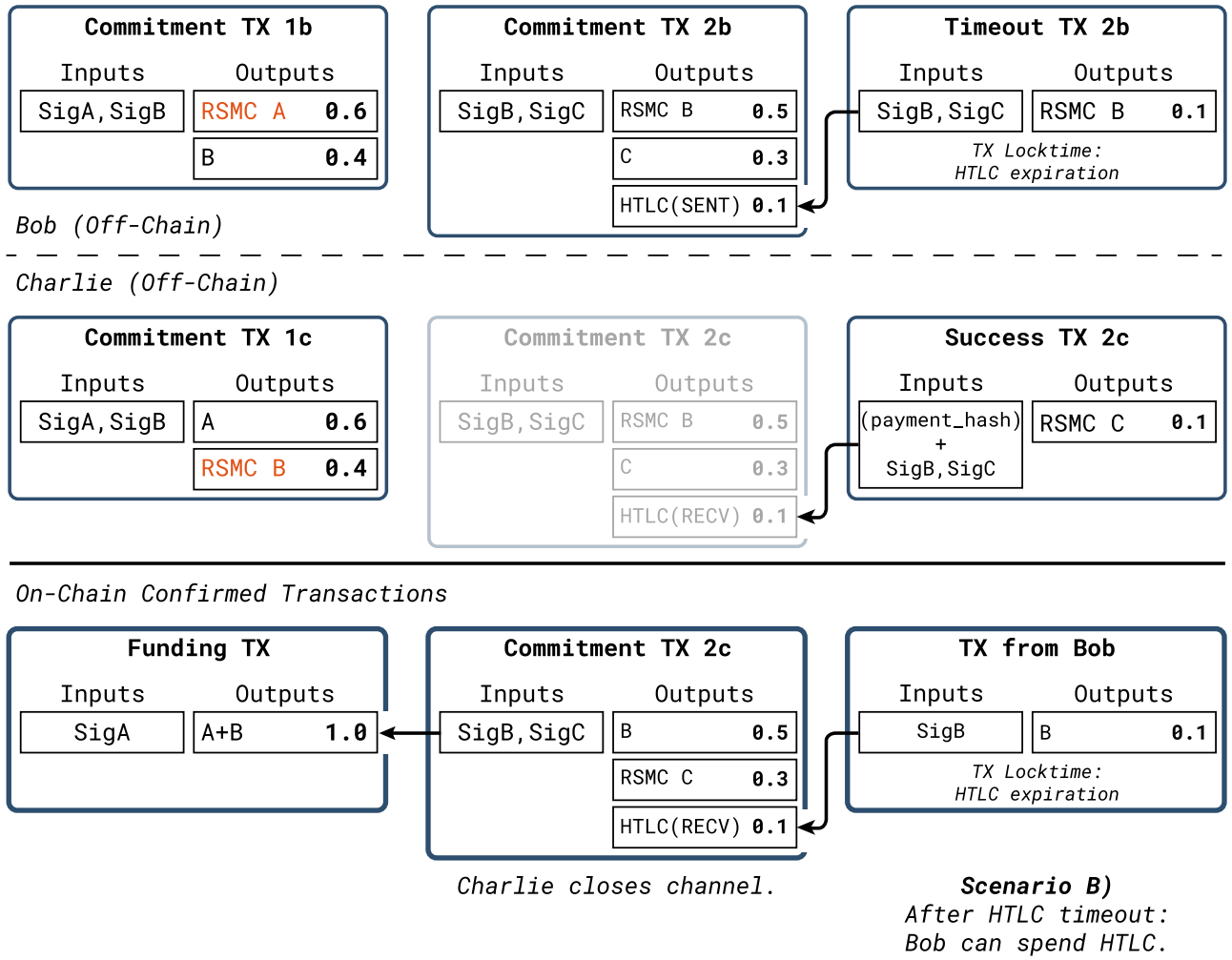
3) Channels update commitment with `htlc`.

- Hash time locked contract represents a "pending" amount/output. Should be reconciled back into new commitment transactions.

4) Dave unlocks `htlc` with `payment_preimage`.

- Preimage is propgated backwards through payment route: atomicity!
- `htlc`'s are settled and commitment tx is updated.

Closing a channel with **HTLC outputs**



Bob closes channel with sent HTLC's

- A) HTLC spendable by timeout transaction after htlc timeout.
  - *HTLC & RSMC B revocable with* `secret r`
- B) HTLC immediately spendable by Charlie with `payment_preimage`

Charlie closes channel with received HTLC's

- A) HTLC spendable with success transaction with `payment_preimage`.
  - *HTLC & RSMC C revocable with* `secret r`
- B) HTLC spendable by Bob after HTLC timeout.

Timeout & Success Transactions:

- Two-stage design.
- Enables both HTLC timeout and RSMC delay.

Closer look: **HTLC scripts**

	Success Transaction
HTLC(RECV) is spent by:	Bob's Signature after Timeout
	Revocation Signature & Public Key

```
Input Script                                                    Top Stack Element
<signature_revocation>
<pubkey_revocation>

HTLC(Sent) Output Script
OP_DUP OP_HASH160 <revocation_pubkeyhash> OP_EQUAL                          1
OP_IF                                                   <pubkey_revocation>
    OP_CHECKSIG                                                             1
OP_ELSE
    <pubkey_bob> OP_SWAP OP_SIZE 32 OP_EQUAL
    OP_IF
        OP_HASH160 <payment_hash> OP_EQUALVERIFY
        2 OP_SWAP <pubkey_charlie> 2 OP_CHECKMULTISIG
    OP_ELSE
        OP_DROP <htlc_expiry> OP_CHECKLOCKTIMEVERIFY OP_DROP
        OP_CHECKSIG
    OP_ENDIF
OP_ENDIF
```

HTLC(SENT) output spendable by:

- Bob: After HTLC timeout: Presigned timeout TX
- Charlie: Payment preimage & HTLC receiver's signature
- Charlie: Revocation signature & public key

HTLC(RECV) output spendable by:

- Charlie: Payment preimage & success TX
- Bob: After HTLC timeout: HTLC sender's signature
- Bob: Revocation signature & public key