
MODULE *LNContractsUsingBitcoinTransactions*

This spec captures the behaviour of commitment transactions on the two sides of a Lightning channel.

We model the various kinds of outputs a commitment transactions will have over its lifetime.

The state of the commitment transaction changes in response to the various actions like supersede, publish, etc are taken by parties.

We also do not deal with the communication protocol between nodes for creating and updating commitment transactions. This spec will only focus on the various commitment transaction and their lifecycle in response to interaction between parties and the blockchain.

We ignore the details of how transactions are signed and just mark transactions as signed. This lets us focus on the specifying the behaviour of the commitment transactions without dealing with lower level complexities.

The model defines the initial balance from alice to bob. TLA+ will handle situations where channels are balanced and when all the balance is on the other side.

TODO: Add actions for closing channels. Currently we only have support for breach *tx* and the corresponding breach remedy txs.

TODO: Add *HTLCs*.

EXTENDS *Integers*,
 TLC,
 Sequences,
 FiniteSets,
 BitcoinTransactions

CONSTANTS
 INITIAL_BALANCE, Initial balances for alice and bob
 CHANNEL_PARTY Channel between parties

VARIABLES
 funding_txs, The *TXID* for the channel funding txs
 commitment_txs, Commitment txs held by each party. Not yet broadcast.
 breach_remedy_txs *BR* txs held by each party. Not yet broadcast.

$SeqToSet(s) \triangleq \{s[i] : i \in \text{DOMAIN } s\}$

$vars \triangleq \langle chain_height, transactions, mempool, published, \\ commitment_txs, breach_remedy_txs, funding_txs \rangle$

$Init \triangleq$
 $\wedge transactions = [id \in TXID \mapsto [inputs \mapsto \langle \rangle, outputs \mapsto \langle \rangle]]$
 $\wedge funding_txs = [channel \in CHANNEL_PARTY \mapsto NoTxid]$
 $\wedge commitment_txs = [channel \in CHANNEL_PARTY \mapsto [p \in channel \mapsto NoTxid]]$

$$\begin{aligned}
& \wedge \text{breach_remedy_txs} = [\text{channel} \in \text{CHANNEL_PARTY} \mapsto [p \in \text{channel} \mapsto \text{NoTxid}]] \\
& \wedge \text{chain_height} = 0 \\
& \wedge \text{mempool} = \{\} \\
& \wedge \text{published} = [\text{id} \in \text{TXID} \mapsto \text{NoSpendHeight}]
\end{aligned}$$

$\text{TypeOK} \triangleq$

$$\begin{aligned}
& \wedge \text{transactions} \in [\text{TXID} \rightarrow [\text{inputs} : \text{Seq}(\text{Input}), \text{outputs} : \text{Seq}(\text{Output})]] \\
& \wedge \text{funding_txs} \in [\text{CHANNEL_PARTY} \rightarrow \text{TXID} \cup \{\text{NoTxid}\}] \\
& \wedge \text{commitment_txs} \in [\text{CHANNEL_PARTY} \rightarrow [\text{PARTY} \rightarrow \text{TXID} \cup \{\text{NoTxid}\}]] \\
& \wedge \text{breach_remedy_txs} \in [\text{CHANNEL_PARTY} \rightarrow [\text{PARTY} \rightarrow \text{TXID} \cup \{\text{NoTxid}\}]] \\
& \wedge \text{mempool} \in \text{SUBSET } \text{TXID} \\
& \wedge \text{published} \in [\text{TXID} \rightarrow \text{Int}]
\end{aligned}$$

Choose keys for parties that have a channel.

The keys should have the same sequence number. This becomes important when parties create commitment transactions.

$\text{ChooseChannelKeys} \triangleq$

$$\begin{aligned}
& \text{CHOOSE } \langle j, k \rangle \in \text{Keys} \times \text{Keys} : \\
& \quad \wedge \{j[1], k[1]\} \in \text{CHANNEL_PARTY} \\
& \quad \wedge j[2] = k[2]
\end{aligned}$$

Choose parties that have a channel
Choose keys with same index

$\text{ChannelKeys}(\text{channel}) \triangleq \text{SetToSeq}(\{\langle p, 1 \rangle : p \in \text{channel}\})$

$\text{ChoosePartyKey}(\text{party}) \triangleq$

$$\text{CHOOSE } k \in \text{Keys} : k[1] = \text{party}$$

$\text{AllCommitmentsTxids} \triangleq$

$$\begin{aligned}
& \{\text{commitment_txs}[cp[1]][cp[2]] : \\
& \quad cp \in \{\langle \text{channel_party}, \text{party} \rangle \in \text{CHANNEL_PARTY} \times \text{PARTY} : \text{party} \in \text{channel_party}\}\}
\end{aligned}$$

$\text{AllBreachRemedyTxids} \triangleq$

$$\begin{aligned}
& \{\text{breach_remedy_txs}[cp[1]][cp[2]] : \\
& \quad cp \in \{\langle \text{channel_party}, \text{party} \rangle \in \text{CHANNEL_PARTY} \times \text{PARTY} : \text{party} \in \text{channel_party}\}\}
\end{aligned}$$

$\text{TXID } id$ has not been used yet.

TODO: We might need to rethink how we track txids, but for now, this is the solution.

$\text{IsUnused}(id) \triangleq$

$$\begin{aligned}
& \wedge id \notin \text{mempool} \\
& \wedge \text{published}[id] = \text{NoSpendHeight} \\
& \wedge id \notin \text{AllCommitmentsTxids} \\
& \wedge id \notin \text{AllBreachRemedyTxids} \\
& \wedge id \notin \{\text{funding_txs}[t] : t \in \text{CHANNEL_PARTY}\}
\end{aligned}$$

Confirm a transaction in *mempool*. This publishes the transaction.

We need to add a function like *IsOutputSpent(o)* which checks if there is any transaction in published with *o* as input.

$$\begin{aligned} \text{ConfirmTx}(id) &\triangleq \\ &\wedge \text{ConfirmMempoolTx}(id) \\ &\wedge \text{UNCHANGED } \langle \text{commitment_txs}, \text{breach_remedy_txs}, \text{funding_txs} \rangle \end{aligned}$$

We generate simple *p2wkh* transactions as inputs for funding transactions

Outputs both have *INITIAL_BALANCE * 2* so that later we can have bi-directional channel with *INITIAL_BALANCE*

$$\begin{aligned} \text{CreateInputsForFundingTx}(id, \text{channel}) &\triangleq \\ &\wedge \text{funding_txs}[\text{channel}] = \text{NoTxid} \quad \text{No funding tx for channel exists} \\ &\wedge \exists \text{party} \in \text{channel} : \\ &\quad \wedge \text{AddP2WKHCoinbaseToMempool}(id, \langle \text{ChoosePartyKey}(\text{party}) \rangle, \text{INITIAL_BALANCE} * 2) \\ &\wedge \text{UNCHANGED } \langle \text{commitment_txs}, \text{breach_remedy_txs}, \text{funding_txs} \rangle \end{aligned}$$

Create funding transaction. The funding *tx* is not locked at this point.

Once a commitment *tx* is signed then we can send this funding *tx* to the *mempool*.

Create a funding *tx* only if no funding *tx* exists for the channel

$$\begin{aligned} \text{CreateFundingTxByParty}(id, \text{channel}) &\triangleq \\ &\exists o \in \text{UnspentOutputs}, p \in \text{channel} : \\ &\quad \text{transaction with } id \text{ not created yet} \\ &\quad \wedge \text{IsUnused}(id) \\ &\quad \wedge \text{funding_txs}[\text{channel}] = \text{NoTxid} \quad \text{No funding tx exists for the channel} \\ &\quad \wedge \text{OutputOwnedByParty}(o, p) \\ &\quad \wedge \text{LET } \text{fundingTx} \triangleq \text{CreateUnsignedMultisigTx}(o, \text{ChannelKeys}(\text{channel}), \text{INITIAL_BALANCE}) \\ &\quad \text{IN} \\ &\quad \wedge \text{transactions}' = [\text{transactions} \text{ EXCEPT } ![id] = \text{fundingTx}] \\ &\quad \wedge \text{funding_txs}' = [\text{funding_txs} \text{ EXCEPT } ![channel] = id] \\ &\quad \wedge \text{UNCHANGED } \langle \text{commitment_txs}, \text{breach_remedy_txs}, \\ &\quad \quad \text{chain_height}, \text{published}, \text{mempool} \rangle \end{aligned}$$

$$\begin{aligned} \text{CreateCommitmentTxInput}(ftxid, \text{ftx}) &\triangleq \\ &\langle [txid \mapsto \text{ftxid}, \\ &\quad \text{index} \mapsto \text{ftx.outputs}[1].\text{index}, \quad \text{Assume we only have one output in ftx} \\ &\quad \text{sighash_flag} \mapsto \text{"all"}, \\ &\quad \text{signed_by} \mapsto \text{ftx.outputs}[1].\text{keys}, \\ &\quad \text{hash_preimage} \mapsto \text{NoHash}] \rangle \end{aligned}$$

$$\begin{aligned} \text{CreateCommitmentTxOutputs} &\triangleq \\ &\langle \rangle \end{aligned}$$

 Create a commitment tx for for a channel parties.

$CreateCommitmentTx(txid, channel, party) \triangleq$

$txid$ is not used
 $\wedge IsUnused(txid)$
 Funding tx for channel exists
 $\wedge funding_txs[channel] \neq NoTxid$
 Commitment tx for channel and party doesn't exist
 $\wedge commitment_txs[channel][party] = NoTxid$
 Create the commitment tx for party paying back to funder
 $\wedge LET \quad ftxid \triangleq funding_txs[channel]$
 $\quad ftx \triangleq transactions[ftxid]$
 $\quad ftx_input \triangleq transactions[ftx.inputs[1].txid]$
 Create commitment tx , that spends ftx and creates outputs for party and other party
 Use amounts based on $party =$ funding input key holder
 $tx \triangleq [inputs \mapsto CreateCommitmentTxInput(ftxid, ftx),$
 $\quad outputs \mapsto CreateCommitmentTxOutputs]$
 $tx \triangleq CreateP2WKHTx((ftxid, ftx.outputs[1].index),$
 $\quad ftx_input.outputs[1].keys,$
 $\quad ftx_input.outputs[1].amount)$

IN
 $\wedge commitment_txs' = [commitment_txs \text{ EXCEPT } ![channel][party] = txid]$
 $\wedge transactions' = [transactions \text{ EXCEPT } ![txid] = tx]$
 $\wedge UNCHANGED \langle breach_remedy_txs, funding_txs,$
 $\quad chain_height, published, mempool \rangle$

Add funding tx to $mempool$ once commitment transactions have been signed and shared.

$AddFundingTxToMempool(txid, channel) \triangleq$

LET
 $funding_txid \triangleq funding_txs[channel]$

IN
 $\wedge funding_txid \neq NoTxid$ A funding tx exists
 $\wedge \forall p \in channel :$
There is a commitment tx for both parties
 $\wedge commitment_txs[channel][p] \neq NoTxid$
both commitment tx spending the funding tx
 $\wedge transactions[commitment_txs[channel][p]].inputs[1].txid = funding_txid$
 $\wedge mempool' = mempool \cup \{funding_txid\}$
 $\wedge UNCHANGED \langle commitment_txs, breach_remedy_txs,$
 $\quad chain_height, published, funding_txs, transactions \rangle$

$Next \triangleq$
 $\vee \exists id \in TXID, channel \in CHANNEL_PARTY :$
 $\quad \vee CreateInputsForFundingTx(id, channel)$

$$\begin{aligned}
& \vee \exists id \in TXID : ConfirmTx(id) \\
& \vee \exists id \in TXID, channel \in CHANNEL_PARTY : \\
& \quad \vee CreateFundingTxByParty(id, channel) \\
& \quad \vee AddFundingTxToMempool(id, channel) \\
& \vee \exists channel \in CHANNEL_PARTY : \\
& \quad \exists \langle txid, party \rangle \in TXID \times channel : \\
& \quad \quad CreateCommitmentTx(txid, channel, party)
\end{aligned}$$

$$\begin{aligned}
Spec & \triangleq \\
& \wedge Init \\
& \wedge \Box [Next]_{\langle vars \rangle}
\end{aligned}$$
