MODULE *Contracts*

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 reponse 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 focusses 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.

TODO: We don't track balances yet. Need to do that and don't go into certain states depending on the balance checks

TODO: Add *HTLC*s! Now that will be fun!

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

CONSTANTS

CSV,	The csv value to use in contracts
$Height$,	The height up to which we run the spec
$NumTxs$	The number of commitment txs we want

Current channel contracts only ever have two parties

$Party \triangleq \{\text{"alice"}, \text{"bob"}\}$

For the first revocation we only need two keys per party

$NumKey \triangleq 2$

Set of all keys

$Key \triangleq \{\langle p, k\rangle : p \in Party, k \in 0 \,..\, NumKey\}$

Value to capture missing CSV in output

$NoCSV \triangleq \text{CHOOSE } c : c \notin 0 \,..\, CSV$

Multisig outputs without CSV encumberance

$MultiSig \triangleq Party \times Party \times \{NoCSV\}$

Multisig outputs with CSV encumberance

$MultiSigWithCSV \triangleq Party \times Party \times \{CSV\}$

P2PKH outputs, without encumbrance

$P2PKH \triangleq Key$

Set of all signatures for all commit txs. The signature in real world is related to the commit transaction. However, we leave out this complication of how the signature is generated. If there is a signature by a key on a tx, it is assumed it is correctly signed as per bitcoin's requirements

$Sig \triangleq \{\langle p, k\rangle : p \in Party, k \in 0 \,..\, NumKey - 1\}$

Value to capture unsigned transactions

$NoSig \triangleq \text{CHOOSE } s : s \notin Sig$

$$\begin{aligned} CT \triangleq\ & [index \mapsto 0 \,..\, NumTxs, \\ & multisig \mapsto MultiSigWithCSV,\ pk \mapsto P2PKH, \\ & local\_sig \mapsto Sig \cup \{NoSig\}, \\ & remote\_sig \mapsto Sig \cup \{NoSig\}] \end{aligned}$$

$PublishId \triangleq \{\langle p, i, h\rangle : p \in Party, i \in 0 \,..\, NumTxs, h \in 0 \,..\, Height\}$
$NoSpend \triangleq \langle\rangle$

VARIABLES

$alice\_cts$,	Commitment tx for *alice*
bob_cts,	Commitment tx for bob
$alice\_brs$,	Breach remedy transactions for *alice*
bob_brs,	Breach remedy transactions for bob
$mempool\_ct$,	The CT that has been broadcasted.
	$TODO$: Turn into Seq. More than one can be in mempool.
$published\_ct$	The CT that has been included in a block and confirmed.

$vars \triangleq \langle alice\_cts, bob\_cts, alice\_brs, bob\_brs, mempool\_ct, published\_ct\rangle$

Helper function to get other party

$OtherParty(party) \triangleq \text{CHOOSE } p \in Party : p \neq party$

Create a commitment transaction given the party, index and key to use.

$$\begin{aligned} CreateCT(party, index, key\_num) \triangleq\ & \\ & [index \mapsto index, \\ & multisig \mapsto \langle party, OtherParty(party), CSV\rangle, \\ & pk \mapsto \langle party, key\_num\rangle, \\ & local\_sig \mapsto NoSig, \\ & remote\_sig \mapsto \langle OtherParty(party), key\_num\rangle] \end{aligned}$$

$$\begin{aligned} Init \triangleq\ & \\ & \wedge alice\_cts = \{CreateCT(\text{“alice”}, 0, 0)\} \\ & \wedge bob\_cts = \{CreateCT(\text{“bob”}, 0, 0)\} \\ & \wedge alice\_brs = \{\} \end{aligned}$$

$\land\ bob\_brs = \{\}$
$\land\ mempool\_ct = NoSpend$
$\land\ published\_ct = NoSpend$

$TypeInvariant\ \triangleq$
$\land\ \forall\ ct \in alice\_cts \cup bob\_cts :$
$\land\ ct.index \in 0\ ..\ NumTxs$
$\land\ ct.local\_sig \in Sig \cup \{NoSig\}$
$\land\ ct.remote\_sig \in Sig \cup \{NoSig\}$
$\land\ ct.pk \in P2PKH$
$\land\ ct.multisig \in MultiSigWithCSV$
$\land\ \forall\ br \in alice\_brs \cup bob\_brs :$
$\land\ br.index \in 0\ ..\ NumTxs$
$\land\ br.pk \in P2PKH$
$\land\ mempool\_ct \in PublishId \cup \{NoSpend\}$
$\land\ published\_ct \in PublishId \cup \{NoSpend\}$

$MaxIndex(party\_cts)\ \triangleq$
$(\text{CHOOSE}\ x \in party\_cts : \forall\ y \in party\_cts : x.index \geq y.index).index$

Create first commitment transactions for given parties

Breach remedy transactions are pre-signed transactions instead of they private key being sent over to the other party.

$SupersedeCommitmentTx(index)\ \triangleq$
$\land\ mempool\_ct \neq NoSpend$ Stop creating new CTs once FT is spent
$\land$
LET $key\_index\ \triangleq\ 1$
IN
$\land\ index > MaxIndex(alice\_cts)$
$\land\ index > MaxIndex(bob\_cts)$
$\land\ alice\_cts' = alice\_cts \cup \{CreateCT(\text{“alice”}, index, key\_index)\}$
$\land\ bob\_cts' = bob\_cts \cup \{CreateCT(\text{“bob”}, index, key\_index)\}$
$\land\ alice\_brs' = alice\_brs \cup \{[index \mapsto index, pk \mapsto \langle \text{“bob”}, key\_index\rangle]\}$
$\land\ bob\_brs' = bob\_brs \cup \{[index \mapsto index, pk \mapsto \langle \text{“alice”}, key\_index\rangle]\}$
$\land$ UNCHANGED $\langle mempool\_ct, published\_ct\rangle$

Publish a commitment transaction to the blockchain. The commitment is first signed. The protocol allows all commitments to be published, what happens next depends on the status of the commitment transaction.

If the tx is the latest commitment transaction it is succesfuly spend.

If not, it gives the other party a chance to spend the breach remedy tx.

TODO: We only spec *CSV* (self) commitment transaction. We need to handle the non-*CSV* output being published and co-op closes.

$PublishCommitment(party,\ index,\ height) \triangleq$
$\quad \land\ mempool\_ct = NoSpend$
$\quad \land\ mempool\_ct' = \langle party,\ index,\ height \rangle$
$\quad \land$ UNCHANGED $\langle alice\_cts,\ bob\_cts,\ alice\_brs,\ bob\_brs,\ published\_ct \rangle$

Publish a breach remedy transaction in response to a commitment transaction.

This tx is immediately published on chain.

TODO: We skip the *BR* going through the mempool and confirm it immeidiately. This can be improved too.

$PublishBR(party,\ index,\ height) \triangleq$
$\quad$ LET $cts \triangleq$ IF $party =$ "alice" THEN $alice\_cts$ ELSE bob_cts
$\quad$ IN
$\quad\quad \land\ published\_ct = NoSpend$ — No CT is confirmed on chain yet
$\quad\quad \land\ mempool\_ct \neq NoSpend$ — Only if some CT has been published
$\quad\quad \land\ mempool\_ct[1] = OtherParty(party)$ — CT was published by the other party
$\quad\quad \land\ mempool\_ct[2] < MaxIndex(cts)$ — Revoked CT was published
$\quad\quad \land\ mempool\_ct[2] = index$ — We need to use the BR from the same index
$\quad\quad \land\ height - mempool\_ct[2] < CSV$ — Can only publish BR if CSV hasn't expired
$\quad\quad \land\ published\_ct' = \langle party,\ index,\ height \rangle$ — *TODO*: Pick the appropriate party's *BRS*
$\quad \land$ UNCHANGED $\langle alice\_cts,\ bob\_cts,\ alice\_brs,\ bob\_brs,\ mempool\_ct \rangle$

$Next \triangleq$
$\quad \lor \exists\, i \in 0\,..\,NumTxs : SupersedeCommitmentTx(i)$
$\quad \lor \exists\, i \in 0\,..\,NumTxs,\ p \in Party,\ h \in 0\,..\,Height : PublishCommitment(p,\ i,\ h)$
$\quad \lor \exists\, i \in 0\,..\,NumTxs,\ p \in Party,\ h \in 0\,..\,Height : PublishBR(p,\ i,\ h)$

$Spec \triangleq Init \land \Box[Next]_{\langle vars \rangle}$