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 supercede, spend, revoke etc are taken.

We also do not deal with the communication protocol between nodes for creating and updating commitment transactions. This spec only focusses on the various commitment transaction created, revoked, spent to open, close, force close or penalise.

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.

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

Sequences utility

$SeqOf(set, n) \triangleq$

All sequences up to length n with all elements in set. Includes empty sequence.

$\text{UNION } \{[1 \,..\, m \rightarrow set] : m \in 0 \,..\, n\}$

$ToSet(s) \triangleq$

The image of the given sequence s. $Cardinality(ToSet(s)) \leq Len(s)$ see https://en.wikipedia.org/wiki/Image_(mathematics)

$\{s[i] : i \in \text{DOMAIN } s\}$

$Contains(s, e) \triangleq$

TRUE iff the element $e \in ToSet(s)$.

$\exists\, i \in 1 \,..\, Len(s) : s[i] = e$

$Last(s) \triangleq s[Len(s)]$

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, 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

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

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

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

$vars \triangleq \langle alice\_cts, bob\_cts, alice\_brs, bob\_brs \rangle$

Helper function to get other party

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

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

$local\_sig \mapsto NoSig,$
$remote\_sig \mapsto \langle OtherParty(party),\ key\_num \rangle]$

$Init \triangleq$
$\land\ alice\_cts = \{CreateCT(\text{“alice”},\ 0,\ 0)\}$
$\land\ bob\_cts = \{CreateCT(\text{“bob”},\ 0,\ 0)\}$
$\land\ alice\_brs = \{\}$
$\land\ bob\_brs = \{\}$

We don't define transactions using a function because using variables as functions become hard to work with in TLA+

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

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.

$SupercedeCommitmentTx(index) \triangleq$
$\text{LET } key\_index \triangleq 1$
IN
$\land\ index = Cardinality(alice\_cts)$
$\land\ Cardinality(alice\_cts) > 0 \land Cardinality(bob\_cts) > 0$
$\land\ Cardinality(alice\_cts) < NumTxs \land Cardinality(bob\_cts) < NumTxs$
$\land\ alice\_cts' = alice\_cts \cup \{CreateCT(\text{“alice”},\ index,\ key\_index)\}$
$\land\ bob\_cts' = bob\_cts \cup \{CreateCT(\text{“bob”},\ index,\ key\_index)\}$
$\land\ alice\_brs' = alice\_brs \cup \{[index \mapsto index,\ pk \mapsto \langle \text{“bob”},\ key\_index \rangle]\}$
$\land\ bob\_brs' = bob\_brs \cup \{[index \mapsto index,\ pk \mapsto \langle \text{“alice”},\ key\_index \rangle]\}$

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.

$PublishCommitment(party,\ party\_cts,\ ct) \triangleq$
$\land\ \text{IF } ct.index = Cardinality(party\_cts)$
$\quad \text{THEN } SpendCommitment(ct)$

$\quad$ ELSE $PublishPenalty(ct)$

$Next \triangleq$

$\quad \land \exists\, index \in 0 \mathrel{..} NumTxs : SupercedeCommitmentTx(index)$

$\quad \land \exists\, ct \in alice\_cts: PublishCommitment(\text{"alice"}, alice\_cts, ct)$

$\quad \land \exists\, ct \in bob\_cts: PublishCommitment(\text{"bob"}, bob\_cts, ct)$

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