
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

<i>CSV</i> ,	The csv value to use in contracts
<i>Height</i> ,	The height up to which we run the spec
<i>NumTxs</i>	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.

UNION $\{[1 \dots m \rightarrow set] : m \in 0 \dots n\}$

$ToSet(s) \triangleq$

The image of the given sequence s . $Cardinality(ToSet(s)) \leq Len(s)$ see [https://en.wikipedia.org/wiki/Image_\(mathematics\)](https://en.wikipedia.org/wiki/Image_(mathematics))
 $\{s[i] : i \in DOMAIN\ s\}$

$Contains(s, e) \triangleq$

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

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

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

Current channel contracts only ever have two parties

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

For the first revocation we only need two keys per party

$NumKey \triangleq 2$

Set of all keys

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

Value to capture missing *CSV* in output

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

Multisig outputs without *CSV* encumbrance

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

Multisig outputs with *CSV* encumbrance

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

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

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

Value to capture unsigned transactions

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

$CT \triangleq [index \mapsto 0 \dots NumTxs,$
 $\quad multisig \mapsto MultiSigWithCSV, pk \mapsto P2PKH,$
 $\quad local_sig \mapsto Sig \cup \{NoSig\},$
 $\quad remote_sig \mapsto Sig \cup \{NoSig\}]$

VARIABLES

$alice_cts,$	Commitment tx for <i>alice</i>
$bob_cts,$	Commitment tx for bob
$alice_brs,$	Breach remedy transactions for <i>alice</i>
$bob_brs,$	Breach remedy transactions for bob
$funding_spent$	Is funding tx spent (we'll add spending txid later)

$vars \triangleq \langle alice_cts, bob_cts, alice_brs, bob_brs, funding_spent \rangle$

Helper function to get other party

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

$CreateCT(party, index, key_num) \triangleq$
 $\quad [index \mapsto index,$
 $\quad \quad multisig \mapsto \langle party, OtherParty(party), CSV \rangle,$
 $\quad \quad pk \mapsto \langle party, key_num \rangle,$

$local_sig \mapsto NoSig,$
 $remote_sig \mapsto \langle OtherParty(party), key_num \rangle]$

$Init \triangleq$
 $\wedge alice_cts = \{ CreateCT("alice", 0, 0) \}$
 $\wedge bob_cts = \{ CreateCT("bob", 0, 0) \}$
 $\wedge alice_brs = \{ \}$
 $\wedge bob_brs = \{ \}$
 $\wedge funding_spent = FALSE$

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

$TypeInvariant \triangleq$
 $\wedge \forall ct \in alice_cts \cup bob_cts :$
 $\quad \wedge ct.index \in 0 \dots NumTxs$
 $\quad \wedge ct.local_sig \in Sig \cup \{ NoSig \}$
 $\quad \wedge ct.remote_sig \in Sig \cup \{ NoSig \}$
 $\quad \wedge ct.pk \in P2PKH$
 $\quad \wedge ct.multisig \in MultiSigWithCSV$
 $\wedge \forall br \in alice_brs \cup bob_brs :$
 $\quad \wedge br.index \in 0 \dots NumTxs$
 $\quad \wedge br.pk \in P2PKH$
 $\wedge funding_spent \in BOOLEAN$

$MaxIndex(party_cts) \triangleq$
 $(CHOOSE x \in party_cts : \forall y \in party_cts : x.index \geq y.index).index$

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.

$SupcedeCommitmentTx(index) \triangleq$
 \wedge
 $\quad LET key_index \triangleq 1$
 $\quad IN$
 $\quad \wedge index > MaxIndex(alice_cts)$
 $\quad \wedge index > MaxIndex(bob_cts)$
 $\quad \wedge alice_cts' = alice_cts \cup \{ CreateCT("alice", index, key_index) \}$
 $\quad \wedge bob_cts' = bob_cts \cup \{ CreateCT("bob", index, key_index) \}$
 $\quad \wedge alice_brs' = alice_brs \cup \{ [index \mapsto index, pk \mapsto \langle "bob", key_index \rangle] \}$
 $\quad \wedge bob_brs' = bob_brs \cup \{ [index \mapsto index, pk \mapsto \langle "alice", key_index \rangle] \}$
 $\wedge UNCHANGED \langle funding_spent \rangle$

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 successfully spend.
 If not, it gives the other party a chance to spend the breach remedy tx.
TODO: Use $Party \rightarrow cts$ function so we can merge these two actions.
TODO: Track which CT is spent and also the output spent.

$$\begin{aligned} AlicePublishCommitmentWithoutPenalty(index) &\triangleq \\ &\wedge funding_spent = \text{FALSE} \\ &\wedge index = MaxIndex(alice_cts) \quad \text{Spend latest } CT \\ &\wedge funding_spent' = \text{TRUE} \\ &\wedge \text{UNCHANGED } \langle alice_cts, bob_cts, alice_brs, bob_brs \rangle \end{aligned}$$

$$\begin{aligned} BobPublishCommitmentWithoutPenalty(index) &\triangleq \\ &\wedge funding_spent = \text{FALSE} \\ &\wedge index = MaxIndex(bob_cts) \quad \text{Spend latest } CT \\ &\wedge funding_spent' = \text{TRUE} \\ &\wedge \text{UNCHANGED } \langle alice_cts, bob_cts, alice_brs, bob_brs \rangle \end{aligned}$$

$$\begin{aligned} Next &\triangleq \\ &\vee \exists index \in 0 \dots NumTxS : SupercedeCommitmentTx(index) \\ &\vee \exists index \in 0 \dots NumTxS : AlicePublishCommitmentWithoutPenalty(index) \\ &\vee \exists index \in 0 \dots NumTxS : BobPublishCommitmentWithoutPenalty(index) \\ &\vee \exists index \in 0 \dots NumTxS : AlicePublishCommitmentWithoutPenalty(index) \end{aligned}$$

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