Virtual payment channels for the Lightning Network

Aggelos Kiayias[1,2] and Orfeas Stefanos Thyfronitis Litos[1]

[1] University of Edinburgh
[2] IOHK
akiayias@inf.ed.ac.uk, o.thyfronitis@ed.ac.uk

Abstract. Virtual Lightning-like payment channels

1 Notation

We introduce the following notation to formally express Bitcoin transactions.

Basic building blocks

- signature (needed to spend): $\mathrm{player}_{\mathrm{sigName}}$, e.g. $Alice_{\mathrm{rev}}$
- value (in bitcoin): xɃ

Spending method – in transaction output (possibly named)

- n out of n multisig:
  $\texttt{AND}(\mathrm{sig}_1, \ldots, \mathrm{sig}_n)$, alternatively $\mathrm{sig}_1 \wedge \cdots \wedge \mathrm{sig}_n$
- relative delay – minimum blocks between current and spending transaction:
  $\texttt{rltvDelay}(n\text{-of-}n\texttt{-multisig}, blocks)$, e.g. $\texttt{rltvDelay}(Alice_F \wedge Bob_F, 3)$
- absolute delay – minimum block where current transaction can be spent:
  $\texttt{absDelay}(n\text{-of-}n\texttt{-multisig}, block)$, e.g. $\texttt{delayed} = \texttt{absDelay}(Alice_{\mathrm{htlc}}, 1005)$
- hashlock – a hash is provided here, its preimage must be provided by the spending transaction. Can be nested: TODO remove nesting if unneeded
  $\texttt{hashLock}(n\text{-of-}n\texttt{-multisig}, h)$, e.g. $\texttt{hashLock}(Alice_{\mathrm{htlc}} \wedge Charlie_{\mathrm{htlc}}, \texttt{0x9b4f})$

Spending methods set – each output contains one such set
$\texttt{OR}(method_1, \ldots, method_m)$, alternatively $method_1 \vee \cdots \vee method_m$,
e.g. $(\texttt{fulfill} = \texttt{hashLock}(Alice, \texttt{0x1bc6})) \vee (\texttt{refund} = \texttt{absDelay}(Bob, 1007))$

Output – each transaction contains one or more (possibly named)
$\texttt{txOut}(set\ of\ methods, value)$,
e.g. $\texttt{coins}_{Alice} = \texttt{txOut}(\texttt{normal} = \texttt{rltvDelay}(Alice, 10) \vee \texttt{revocation} = Bob_{\mathrm{rev}})$

Input – each transaction contains one or more, unambiguous arguments can be omitted
$\texttt{txIn}(method\ name, list\ of\ signatures, preimage)$ TODO or list of preimages if needed
e.g. $\texttt{txIn}(\texttt{comm}_{Alice}, \texttt{coins}_{Alice}, \texttt{revocation})$

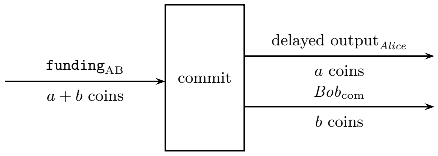

Fig. 1. *Alice*'s base channel tx

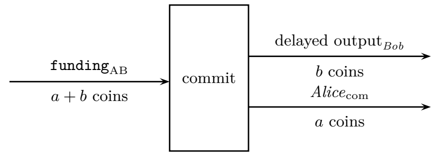

Fig. 2. *Bob*'s base channel tx

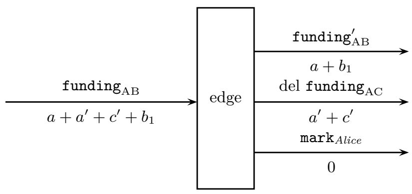

Fig. 3. *Alice*'s virtual channel tx

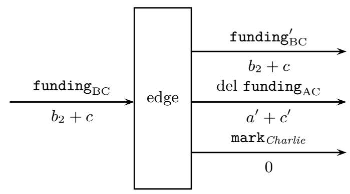

Fig. 4. *Charlie*'s virtual channel tx

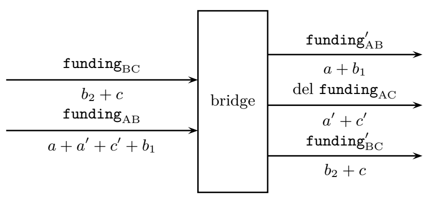

Fig. 5. *Bob*'s tx when others idle

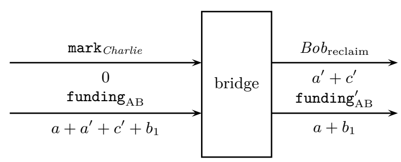

Fig. 6. *Bob*'s tx when only *Charlie* active

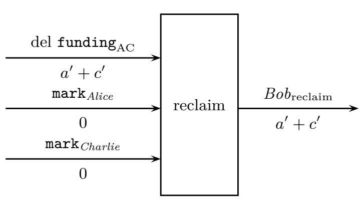

Fig. 7. *Bob*'s tx when others active

Transaction
$\texttt{tx}((\texttt{txIn}_1, \ldots, \texttt{txIn}_n), (\texttt{txOut}_1, \ldots, \texttt{txOut}_m))$,
e.g. $\texttt{rev}_{Bob} = \texttt{tx}((\texttt{txIn}(\texttt{comm}_{Alice}, \texttt{coins}_{Alice}, \texttt{revocation})), (\texttt{txOut}(Bob)))$

Protocol $\texttt{openChannel}_{\text{1-hop}}(Alice, Bob, pk_{Alice,F,1}, \text{coins})$

- Key circulation
  - *Alice*:
    - Generate the 4 $Alice_{F',\text{base/virt}}$, $Alice_{M,\text{source/dest}}$ keypairs
    - Send our public keys to *Bob* in (openVirtChan, `intermediary`, *Charlie*) message
  - *Bob*:
    - Generate TODO keypairs
    - Send TODO public keys to *Charlie*
  - *Charlie*:
    - Receive TODO keys from *Bob*
    - Generate TODO keypairs
    - Send TODO public keys to *Bob*
  - *Bob*:
    - Receive TODO keys from *Charlie*
    - Send TODO public keys to *Alice*
  - *Alice*:
    - Receive (virtChanKeys, `intermediary`, *Charlie*, $pk_{Bob,F,AB,\text{base}}, pk_{Bob,\text{rev},\text{virt}}, pk_{Bob,M,\text{source}}, pk_{Bob,\text{rev},\text{base}}, pk_{Bob,\text{dcom},\text{base}}$) from *Bob*
    - Send our public keys to *Charlie*in (openVirtChan, `counterparty`, *Bob*) message
  - *Charlie*:
    - Receive TODO keys from *Alice*
    - Send TODO public keys to *Alice*
- Signatures circulation:
  - a
- Revocations:
  - a

Fig. 8.

2 Virtual channel over 2 normal channels

We use the term "base channels" for the channels described in [**?**]. We adopt the notation of Perun [**?**] to differentiate base and virtual channels: *Peggy* $\Leftrightarrow$ *Quinney* refers to a base channel and *Peggy* $\leftrightarrow$ *Quinney* refers to a virtual channel.

Figures **??** and **??** show the transactions that two parties hold in an existing base channel between them, as described in [**?**][3]. Let existing $\mathit{Alice} \Leftrightarrow \mathit{Bob}$, $\mathit{Bob} \Leftrightarrow \mathit{Charlie}$ base channels, with an $\mathit{Alice} \leftrightarrow \mathit{Charlie}$ virtual channel on top. Fig. TODO reffig:virt shows the transactions that the three parties hold.

- On-chain outputs:
  - $\texttt{funding}_{\mathrm{AB}} = \texttt{txOut}(\mathit{Alice}_F \wedge \mathit{Bob}_{F,\mathrm{source}}, a + a' + c' + b_{\mathrm{w/src}})$
  - $\texttt{funding}_{\mathrm{BC}} = \texttt{txOut}(\mathit{Bob}_{F,\mathrm{dest}} \wedge \mathit{Charlie}_F, b_{\mathrm{w/dest}} + c)$
- Txs held by *Alice*:
  - $\texttt{baseCommitment}_{\mathit{Alice}} = \texttt{tx}($
    $(\texttt{txIn}(\texttt{funding}_{\mathrm{AB}}))$,
    $(\texttt{base} = \texttt{txOut}(\mathit{Alice}_{F,\mathrm{base}} \wedge \mathit{Bob}_{F,AB,\mathrm{base}}, a + b_{\mathrm{w/src}})$,
    $\texttt{virt} = \texttt{txOut}(\texttt{rltvDelay}(\mathit{Alice}_{F,\mathrm{virt}} \wedge \mathit{Charlie}_{F,\mathrm{virt}}, \texttt{bobDelay}) \vee$
    $\mathit{Bob}_{\mathrm{rev,virt}}, a' + c')$,
    $\texttt{mark}_{\mathit{Alice}} = \texttt{txOut}(\mathit{Alice}_{M,\mathrm{source}} \wedge \mathit{Bob}_{M,\mathrm{source}} \wedge \mathit{Charlie}_{M,\mathrm{source}}, 0)))$
  - $\texttt{ABcommitment}_{\mathit{Alice}} = \texttt{tx}($
    $(\texttt{txIn}(\texttt{baseCommitment}_{\mathit{Alice}}, \texttt{base}))$, TODO common keys make this incomplete - fix all inputs
    $(\texttt{txOut}(\texttt{rltvDelay}(\mathit{Alice}_{\mathrm{dcom,base}}, \texttt{bobDelay}) \vee \mathit{Bob}_{\mathrm{rev,base}}, a)$,
    $(\texttt{txOut}(\mathit{Bob}_{\mathrm{com,base}}, b_{\mathrm{w/src}}))))$
  - $\texttt{ACcommitment}_{\mathit{Alice}} = \texttt{tx}($
    $(\texttt{txIn}(\texttt{baseCommitment}_{\mathit{Alice}}, \texttt{virt}))$,
    $(\texttt{txOut}(\texttt{rltvDelay}(\mathit{Alice}_{\mathrm{dcom},2}, \texttt{charlieDelay}) \vee \mathit{Charlie}_{\mathrm{rev},1}, a')$,
    $(\texttt{txOut}(\mathit{Charlie}_{\mathrm{com},1}, c'))))$
- Txs held by *Bob*:
  -
  -
  -
- Txs held by *Charlie*:
  -

TODO choose and homogenize tx names (in list & figures)

Functionality $\mathcal{F}_{\mathrm{ch}}$

Initially `isOpen` is set to `false`.
Message (REGISTER, `otherPlayer`, *pk*, `myDelay`, `remoteDelay`):

- Until exactly two players are registered (call them *Alice* and *Bob*, in order of registration), ignore all other messages. After that, ignore all REGISTER messages.
- Store *pk* as pk_{Alice} or pk_{Bob} respectively.
- After receiving one REGISTER message from a player, ignore all further REGISTER messages from the same player.
- *Alice* must input *Bob* as `otherPlayer` and vice versa.

[3] In the lightning spec (https://github.com/lightningnetwork/lightning-rfc/) really

- *Alice*'s `myDelay` must match *Bob*'s `remoteDelay` and vice versa.

In all subsequent messages, denote P the sender (which must be either *Alice* or *Bob*) and P' the counterparty.
Upon receiving (OPEN, $(sk_{\text{in}}, pk_{\text{in}})$, c):

- $c_P \leftarrow c, c_{P'} \leftarrow 0$
- $(sk_{P,F}, pk_{P,F}) \leftarrow \texttt{KeyGen}(), (sk_{P',F}, pk_{P',F}) \leftarrow \texttt{KeyGen}()$
- $\texttt{openTX} \leftarrow \texttt{tx}((\texttt{txIn}(pk_{\text{in}}, \text{selfsig with } sk_{\text{in}})), (\texttt{txOut}(\texttt{msig} = pk_{P,F} \wedge pk_{P',F}, c)))$
- send (SUBMIT, `openTX`) to $\mathcal{G}_{\text{Ledger}}$as P

Upon receiving (ISOPEN):

- send (READ) to $\mathcal{G}_{\text{Ledger}}$ as P and assign reply to Σ_P
- if $\texttt{openTX} \in \Sigma_P$, set `isOpen` to `true` and return (ISOPEN, `true`), else return (ISOPEN, `false`)

Upon receiving (PAY, c):

- if $\texttt{isOpen} = \texttt{true}$ and $c_P \geq c$, decrease c_P by c, increase c'_P by c and return (PAID) to P
- else return (NOTPAID) to P

Upon receiving (GETBALANCE):

- if $\texttt{isOpen} = \texttt{true}$, return (BALANCE, $c_P, c_{P'}$) to P
- else return (NOTOPEN) to P

Upon receiving (CORRUPT, P) from $\mathcal{A}$, mark P as corrupted and send sk_P to $\mathcal{A}$
Upon receiving (CLOSE):

- $\texttt{txInput} \leftarrow \texttt{txIn}(\texttt{msig}, \text{selfsig with } sk_{P,F} \wedge sk_{P',F})$
- if P' is corrupted, send (COOPERATE, P') to $\mathcal{A}$
- if reply is (COOPERATE) or if P' is not corrupted, set `closeTX` to $\texttt{tx}((\texttt{txInput}), (\texttt{txOut}(pk_P, c_P), \texttt{txOut}(pk_{P'}, c_{P'})))$ (with signature)
- else if reply is (NOTCOOPERATE),
  - $(sk_{\text{rev}}, pk_{\text{rev}}) \leftarrow \texttt{KeyGen}()$
  - $\texttt{closeTX} \leftarrow \texttt{tx}((\texttt{txInput}), (\texttt{txOut}(\texttt{rltvDelay}(pk_P, \texttt{delay}_{P'}) \vee pk_{\text{rev}}, c_P), \texttt{txOut}(pk_{P'}, c_{P'})))$ (with signature)
- set $c_P, c_{P'}$ to 0 and `isOpen` to `false`
- send (SUBMIT, `closeTX`) to $\mathcal{G}_{\text{Ledger}}$ as P

Definition 1 ($\mathcal{F}_{\text{ch}}$ Funds retrievability).
Let Alice honest, Bob corrupted, $\mathcal{F}_{\text{ch}}$ parametrized by three protocols `openChannel`, `closeChannel`, and `checkOpen`, interacting with $\mathcal{G}_{\text{Ledger}}$ which in turn is parametrized by the `validate` predicate of Definition TODO . We say that this $\mathcal{F}_{\text{ch}}$ provides

funds retrievability *if for any PPT $\mathcal{A}$, after $\mathcal{F}_{\text{ch}}$ serves a sequence of*

$$\begin{gathered}((\textsc{open}, \text{keys}, \text{coins}_0, \mathit{Alice}), \texttt{paid} \leftarrow (\textsc{pay}, \text{coins}_1^{\mathit{Alice}}, \mathit{Alice}),\\ \texttt{paid} \leftarrow (\textsc{pay}, \text{coins}_1^{\mathit{Bob}}, \mathit{Bob}), \dots, \texttt{paid} \leftarrow (\textsc{pay}, \text{coins}_m^{\mathit{Bob}}, \mathit{Bob}),\\ \texttt{paid} \leftarrow (\textsc{pay}, \text{coins}_n^{\mathit{Alice}}, \mathit{Alice}), (\textsc{close}, \mathit{Charlie} \in \{\mathit{Alice}, \mathit{Bob}\}),\\ \mathit{sk}_{\mathit{Alice}} \leftarrow (\textsc{getSpendingKeys}, \mathit{Alice}))\end{gathered}$$

requests (possibly interspersed with any number of invalid requests by entities other than Alice) and after the response $\mathit{sk}_{\mathit{Alice}}$ from the (GETSPENDINGKEYS) *message is received, it can be used at any point in time to sign a valid transaction for $\mathcal{G}_{\text{Ledger}}$ that spends an existing, unspent output of value*

$$\text{coins}_0 + \sum_{i=1}^{m} \text{coins}_i^{\mathit{Bob}} - \sum_{j=1}^{n} \text{coins}_j^{\mathit{Alice}} \quad .$$

Definition 2 (Base channel).
A base channel *sends exactly one* SUBMIT *message to $\mathcal{G}_{\text{Ledger}}$ during the execution of* `openChannel` *and one more during* `closeChannel`.

Definition 3 (Virtual channel).
A virtual channel *does not send any* SUBMIT *message to $\mathcal{G}_{\text{Ledger}}$ during the execution of* `openChannel`. *Furthermore, if all participating parties in all implicated base channel functionalities are honest, a virtual channel does not send any* SUBMIT *message to $\mathcal{G}_{\text{Ledger}}$ during the execution of* `closeChannel` *either, whereas in case one party is dishonest it sends at most one such message.*

Theorem 1 (Basic Channel Functionality Security).

$$\forall \ \mathit{PPT} \ \mathcal{E}, \exists \, \mathcal{S} : \text{EXEC}^{\mathcal{G}_{\text{Ledger}}}_{\Pi_{\text{ch}}, \mathcal{A}_{\text{d}}, \mathcal{E}} \approx \text{EXEC}^{\mathcal{F}_{\text{ch}}, \mathcal{G}_{\text{Ledger}}}_{\mathcal{S}, \mathcal{E}}$$

Theorem 2 (Virtual Channel Functionality Security). *Let $n \in \mathbb{N}$ players.*

$$\forall \ \mathit{PPT} \ \mathcal{E}, \exists \, \mathcal{S} : \text{EXEC}^{\mathcal{F}_{\text{ch}}, \mathcal{G}_{\text{Ledger}}}_{\Pi_{\text{Vch},n}, \mathcal{A}_{\text{d}}, \mathcal{E}} \approx \text{EXEC}^{\mathcal{W}_n(\mathcal{F}_{\text{ch}}), \mathcal{G}_{\text{Ledger}}}_{\mathcal{S}, \mathcal{E}}$$

Theorem 3 (Virtual Payment Network Security). *Let $n \in \mathbb{N}$ players, $C = \mathcal{F}_{\text{ch}}, \mathcal{W}_{3,\dots,n}(\mathcal{F}_{\text{ch}}), \mathcal{W}_{3,\dots,n}(\mathcal{W}_{3,\dots,n}(\mathcal{F}_{\text{ch}})), \dots$.* *TODO check again*

$$\forall \ \mathit{PPT} \ \mathcal{E}, \exists \, \mathcal{S} : \text{EXEC}^{C, \mathcal{G}_{\text{Ledger}}}_{\Pi_{\text{Vnet}}, \mathcal{A}_{\text{d}}, \mathcal{E}} \approx \text{EXEC}^{\mathcal{F}_{\text{Vnet}}}_{\mathcal{S}, \mathcal{E}}$$