Protocol Π_{Chan}

Initialisation:
  $\mathit{State} \leftarrow \textsc{init}$

On $\textsc{top up}$, $\textsc{check top up}$ by $\mathcal{E}$, act as $\mathcal{F}_{\text{Chan}}$ (Fig. 3, lines 4-8 and 9-15 respectively)

On ($\textsc{open}$, c_F, $pk_{A,out}$, $pk_{B,out}$) by $\mathcal{E}$:
  ensure $\mathit{State} = \textsc{topped up}$
  $\mathit{State} \leftarrow \textsc{opening base channel}$
  do LN (other box)

On ($\textsc{check funding}$) by $\mathcal{E}$:
  ensure $\mathit{State} = \textsc{waiting for ledger}$
  send ($\textsc{read}$) to $\mathcal{G}_{\text{Ledger}}$ and assign reply to Σ
  ensure $F \in \Sigma$
  $c_A \leftarrow c$; $c_B \leftarrow 0$ // c received in $\textsc{open}$
  $\mathit{State} \leftarrow \textsc{open base}$
  output ($\textsc{open success}$) to $\mathcal{E}$

On ($\textsc{pay}$, x) by $\mathcal{E}$:
  ensure $\mathit{State} \in \{\textsc{open base}, \textsc{open virtual}\}$
  ensure $c_A \geq x$
  do LN payment (these channels won't be async) (balance change here)
  output ($\textsc{ok}$) to $\mathcal{E}$

On ($\textsc{balance}$) by $\mathcal{E}$:
  ensure $\mathit{State} \in \{\textsc{open base}, \textsc{open virtual}\}$
  output ($\textsc{balance}$, $(c_A, c_B, \text{locked}_A, \text{locked}_B)$) to $\mathcal{E}$

On ($\textsc{close}$) by $\mathcal{E}$:
  if $\mathit{State} = \textsc{open base}$ **then**
    prepare C TODO
    send ($\textsc{submit}$, C) to $\mathcal{G}_{\text{Ledger}}$
  else if $\mathit{State} = \textsc{open virtual}$ **then**
    TODO
  end if

Fig. 1.

Protocol Π_{Chan} – virtual

// notification to funder
// trust that *Alice* has c in her channel
On (FUND YOU, c, *Bob*) by *Charlie* as input:
    ensure $State = \textsc{init}$
    $State \leftarrow \textsc{opening virtual channel}$
    do LN with *Bob* – TODO
    output (OK) to *Charlie*

On (FUND, c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*) by $\mathcal{E}$:
    ensure $State \in \{\textsc{open base}, \textsc{open virtual}\}$
    do the same as in $\mathcal{F}_{\text{Chan}}$, Fig. 6, lines 9-18 TODO: make sure it makes sense
    do VChan() with hops – TODO // $P_{i-1}P_i, P_iP_{i+1}$ and all P_1P_n held by BOTH R_{i-1} and L_i. $P_{i-1}P_i$ held only by R_{i-1}, P_iP_{i+1} held only by L_i. This (probably) ensures that only relevant parties can close their channels (with the exception of honest R_{i-1} wanting to leave channels virtual but corrupted L_i demoting them to base, which however doesn't cost funds to anyone), but that they have minimal impact to the decisions of ajdacent channels. All $P_{i-1}P_i$ inputs must be signed by R_{i-1} and all P_iP_{i+1} inputs by L_i.
    do the same as in $\mathcal{F}_{\text{Chan}}$, Fig. 6, lines 20-24
    output (OK) to $\mathcal{E}$

// notification to fundee
On (ALLOW FUND, ...) by *Charlie*, act as $\mathcal{F}_{\text{Chan}}$ (Fig 6, line 26):

Fig. 2.

Functionality $\mathcal{F}_{\text{Chan}}$ – init & top up

Initialisation: // runs on first activation

  $\mathit{State} \leftarrow$ INIT

  $(\text{locked}_A, \text{locked}_B) \leftarrow (0, 0)$

On (TOP UP, $c_{\min}$) by *Alice*:

  ensure $\mathit{State} =$ INIT

  $\mathit{State} \leftarrow$ SENT KEY

  $(sk, pk) \leftarrow \textsc{KeyGen}()$

  output (PUBLIC KEY, pk) to *Alice*

On (CHECK TOP UP) by *Alice*:

  ensure $\mathit{State} =$ SENT KEY

  send (READ) to $\mathcal{G}_{\text{Ledger}}$ as *Alice* and assign reply to Σ

  ensure $\exists \text{tx} \in \Sigma, c_{\text{on}} : c_{\text{on}} \geq c_{\min} \wedge (c_{\text{on}}, pk) \in \text{tx.outputs}$

  `base_output` $\leftarrow (c_{\text{on}}, pk)$ of tx

  $\mathit{State} \leftarrow$ TOPPED UP

  output (TOPPED UP) to *Alice*

Fig. 3.

Functionality $\mathcal{F}_{\text{Chan}}$ – base

On (OPEN, c_F, $pk_{A,\text{out}}$, $pk_{B,\text{out}}$) by *Alice*:
    ensure $\mathit{State} = $ TOPPED UP
    ensure $c_F \geq c_{\text{on}}$
    $(sk_{A,F}, pk_{A,F}) \leftarrow$ KEYGEN(); $(sk_{B,F}, pk_{B,F}) \leftarrow$ KEYGEN()
    $F \leftarrow$ TX {input: `base_output`, output: $(c_F, 2/\{pk_{A,F}, pk_{B,F}\})$}
    $F \leftarrow F.\text{sign}(sk)$
    $\mathit{State} \leftarrow$ WAITING FOR LEDGER
    send (OPEN, $c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, F, \mathit{Alice}$) to $\mathcal{A}$ and ensure reply is OK
    output OK to *Alice*

On (CHECK FUNDING) by *Alice*:
    ensure $\mathit{State} = $ WAITING FOR LEDGER
    send (READ) to $\mathcal{G}_{\text{Ledger}}$ as *Alice* and assign reply to Σ
    ensure $F \in \Sigma$
    $c_A \leftarrow c$; $c_B \leftarrow 0$
    $\mathit{State} \leftarrow$ OPEN BASE
    output (OPEN SUCCESS) to *Alice*

On (PAY, x) by $\mathit{Dave} \in \{\mathit{Alice}, \mathit{Bob}\}$:
    ensure $\mathit{State} \in \{$OPEN BASE, OPEN VIRTUAL$\}$
    ensure $c_D - \text{locked}_D \geq x$
    send (PAY, x, *Dave*) to $\mathcal{A}$ and expect reply (OK)
    $c_D \leftarrow c_D - x; c_{\bar{D}} \leftarrow c_{\bar{D}} + x$ // $\bar{D}$ is *Alice* if D is *Bob* and vice-versa
    output (PAY SUCCESS) to *Dave*

On (BALANCE) by $\mathit{Dave} \in \{\mathit{Alice}, \mathit{Bob}\}$:
    ensure $\mathit{State} \in \{$OPEN BASE, OPEN VIRTUAL$\}$
    output (BALANCE, $(c_A, c_B, \text{locked}_A, \text{locked}_B)$) to *Dave*

Fig. 4.

Functionality $\mathcal{F}_{\text{Chan}}$ – close

On (CLOSE) by *Alice*:
  if $State =$ OPEN BASE **then**
    $C \leftarrow$ TX {input: F.out, outputs: $(c_A, pk_{A,\text{out}}), (c_B, pk_{B,\text{out}})$}
    $C \leftarrow C.\text{sign}(\text{sk}_{\text{A,F}}, \text{sk}_{\text{B,F}})$
    $State \leftarrow$ CLOSED
    input (SUBMIT, C) to $\mathcal{G}_{\text{Ledger}}$
  else if $State =$ OPEN VIRTUAL **then**
    $State \leftarrow$ CLOSED
    output (CLOSING, c_A, c_B) to `opener`
  end if

On (CLOSING, c_{left}, c_{right}) by $\mathcal{F}_{\text{Chan}}$:
  ensure $State \in \{$OPEN BASE, OPEN VIRTUAL$\}$
  ensure $((c_L, c_R), \text{hops}, (\textit{Charlie}, \textit{Dave}), (\textit{Frank}, \textit{George}), \text{id}) \in$ `funded` with *Frank* $\in \{\textit{Alice}, \textit{Bob}\}$
  ensure $c_{\text{left}} \leq c_L + c_R$
  remove entry from `funded`
  output (CLOSED VIRTUAL, c_{right}, id) to *Frank*

On (CLOSED VIRTUAL, c_{right}, id) by $\mathcal{F}_{\text{Chan}}$:
  ensure $State \in \{$OPEN BASE, OPEN VIRTUAL$\}$
  ensure $(\text{virtual}, c, \mathcal{F}_{\text{Chan}}, \textit{Dave}, \text{id}) \in$ `funded`
  ensure $c_{\text{right}} \leq c$
  send (CLOSED) to virtual and expect reply YES
  $c_D \leftarrow c_D + c_{\text{right}}$
  remove entry from `funded`

On (CLOSED) by P:
  if $State =$ CLOSED **then**
    send (YES) to P
  else
    send (NO) to P
  end if

Fig. 5.

Functionality $\mathcal{F}_{\text{Chan}}$ – virtual

On (FUND YOU, c, *Dave*) by *Charlie* as input to *Alice*: // *Alice* is funded by *Charlie*
  ensure *State* = INIT
  Bob ← *Dave*
  send (FUND YOU, c, *Bob*, *Charlie*, *Alice*) to $\mathcal{A}$ and ensure reply is (OK)
  $c_A \leftarrow c; c_B \leftarrow 0$
  `opener` ← *Charlie*
  State ← OPEN VIRTUAL
  output (OK) to *Charlie*

On (FUND, c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*)) by *Alice*: // we fund another channel
  ensure *State* ∈ {OPEN BASE, OPEN VIRTUAL}
  ensure $c_A - \text{locked}_A \geq c$
  input (FUND YOU, c, counterparty) to fundee as *Alice*, ensure output is (OK)
  $(L_0, R_0) \leftarrow (Alice, Bob)$
  generate random id
  for all $(L_i, R_i) \in$ hops **do** // $i \in \{1, \ldots, |\text{hops}|\}$
    ensure $R_{i-1} = L_i$
    send (ALLOW FUND, c, `sub_parties`, `local_funder` = L_i, id, $i \stackrel{?}{=} |\text{hops}|$) to L_i as *Alice* and ensure reply is (OK)
  end for
  send (FUND c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*), `funder` = *Alice*) to $\mathcal{A}$ and ensure reply is OK
  for all $(L_i, R_i) \in$ hops **do** // $i \in \{1, \ldots, |\text{hops}|\}$
    send (FUND DONE, id) to L_i as *Alice* and ensure reply is (OK)
  end for
  $c_A \leftarrow c_A - c$
  add ((c, 0), hops, `sub_parties`, `outer_parties`, id) to `funded`
  output (OK) to *Alice*

On (ALLOW FUND, c, `sub_parties`, *Dave*, id, `is_last`) by *Charlie*:
  ensure *State* ∈ {OPEN BASE, OPEN VIRTUAL}
  ensure *Dave* ∈ {*Alice*, *Bob*}
  ensure $c_D - \text{locked}_D \geq c$
  output received message to *Dave* and ensure reply is (OK)
  $\text{locked}_D \leftarrow \text{locked}_D + c$
  add (id, `is_last`, `sub_parties`, c, *Dave*) to `pending`
  send (OK) to *Charlie*

On (FUND DONE, id) by *Charlie*:
  ensure *State* ∈ {OPEN BASE, OPEN VIRTUAL}
  ensure (id, `is_last`, `sub_parties`,c, *Dave*) ∈ `pending`
  remove (id, `is_last`, `sub_parties`, c, *Dave*) from `pending`
  if `is_last` **then**
    add ((0, c), ⊥, `sub_parties`.reverse(), (*Dave*, ⊥), id) to `funded`
  end if
  send (OK) to *Charlie*

Fig. 6.

Simulator $\mathcal{S}$

On $(\textsc{open}, c_F, pk_{A,\mathrm{out}}, pk_{B,\mathrm{out}}, F, \mathit{Alice})$ by $\mathcal{F}_{\mathrm{Chan}}$:
    simulate *Alice* receiving input $(\textsc{open}, c_F, pk_{A,\mathrm{out}}, pk_{B,\mathrm{out}})$ by $\mathcal{E}$
    ensure simulated *Alice* outputs OK
    send OK to $\mathcal{F}_{\mathrm{Chan}}$

On $(\textsc{pay}, x, \mathit{Dave})$ by $\mathcal{F}_{\mathrm{Chan}}$:
    simulate *Dave* receiving input $(\textsc{pay}, x)$ by $\mathcal{E}$
    ensure simulated *Dave* outputs OK
    send OK to $\mathcal{F}_{\mathrm{Chan}}$

On $(\textsc{fund you}, c, \mathit{Bob}, \mathit{Charlie}, \mathit{Alice})$ by $\mathcal{F}_{\mathrm{Chan}}$:
    simulate *Alice* receiving input $(\textsc{fund you}, c, \mathit{Bob})$ by *Charlie*
    ensure simulated *Alice* outputs OK to *Charlie*
    send OK to $\mathcal{F}_{\mathrm{Chan}}$

On (FUND c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*), `funder` = *Alice*) by $\mathcal{F}_{\mathrm{Chan}}$:
    simulate receiving the message without the last element with *Alice*, executing only line 11 of Fig. 2
    ensure VChan() is successful
    send (OK) to $\mathcal{F}_{\mathrm{Chan}}$

Fig. 7.

1 Security Proof

When $\mathcal{E}$ sends (FUND, c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*)) to *Alice* in the real world, the same thing as in the ideal world happens first, then VChan() takes place, then again the same as in the ideal world. On the other hand, in the ideal world the same thing as in the real world happens first, then VChan() is simulated by $\mathcal{S}$, then again the same as in the real world. TODO: double-check that the above argument makes sense

References