Protocol Π_{Chan}

1: Initialisation:
2: $State \leftarrow$ INIT

3: On (OPEN, c, `tx_out`, sk, $pk_{A,out}$, $pk_{B,out}$) by $\mathcal{E}$:
4: ensure $State =$ INIT
5: $State \leftarrow$ OPENING BASE CHANNEL
6: do LN (other box)

7: On (CHECK) by $\mathcal{E}$:
8: ensure $State =$ WAITING FOR LEDGER
9: send (READ) to $\mathcal{G}_{\text{Ledger}}$ and assign reply to Σ
10: ensure $F \in \Sigma$
11: $c_A \leftarrow c$; $c_B \leftarrow 0$ // c received in OPEN
12: $State \leftarrow$ OPEN BASE
13: output (OPEN SUCCESS) to $\mathcal{E}$

14: On (PAY, x) by $\mathcal{E}$:
15: ensure $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
16: ensure $c_A \geq x$
17: do LN payment (these channels won't be async) (balance change here)
18: output (OK) to $\mathcal{E}$

19: On (BALANCE) by $\mathcal{E}$:
20: ensure $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
21: output (BALANCE, $(c_A, c_B, \text{locked}_A, \text{locked}_B)$) to $\mathcal{E}$

22: On (CLOSE) by $\mathcal{E}$:
23: **if** $State =$ OPEN BASE **then**
24: prepare C TODO
25: send (SUBMIT, C) to $\mathcal{G}_{\text{Ledger}}$
26: **else if** $State =$ OPEN VIRTUAL **then**
27: TODO
28: **end if**

Fig. 1.

Protocol Π_{Chan} – virtual

1: // notification to funder
2: // trust that *Alice* has c in her channel
3: On (FUND YOU, c, *Bob*) by *Charlie* as input:
4: ensure *State* = INIT
5: *State* ← OPENING VIRTUAL CHANNEL
6: do LN with *Bob* – TODO
7: output (OK) to *Charlie*

8: On (FUND, c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*) by $\mathcal{E}$:
9: ensure *State* ∈ {OPEN BASE, OPEN VIRTUAL}
10: do the same as in $\mathcal{F}_{\text{Chan}}$ (Fig. 5, lines 9-20) TODO: make sure it makes sense
11: do VChan() with hops – TODO
12: output (OK) to $\mathcal{E}$

13: // notification to fundee
14: On (ALLOW FUND, ...) by *Charlie*, act as $\mathcal{F}_{\text{Chan}}$ (Fig 5, line 22):

Fig. 2.

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

Initialisation: // runs on first activation
    $State \leftarrow$ INIT
    $(\text{locked}_A, \text{locked}_B) \leftarrow (0, 0)$

On (OPEN, c, `tx_out`, sk, $pk_{A,\text{out}}$, $pk_{B,\text{out}}$) by *Alice*:
    ensure $State =$ INIT
    $pk \leftarrow PK(sk)$; $(sk_{A,F}, pk_{A,F}) \leftarrow$ KEYGEN(); $(sk_{B,F}, pk_{B,F}) \leftarrow$ KEYGEN()
    $F \leftarrow$ TX {input: `tx_out`, output: $(c, 2/\{pk_{A,F}, pk_{B,F}\})$}
    $F \leftarrow F.\text{sign}(sk)$
    $State \leftarrow$ WAITING FOR LEDGER
    send (OPEN, F) to $\mathcal{A}$

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

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

Fig. 3.

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 $((c_L, c_R), \text{hops}, (\mathit{Charlie}, \mathit{Dave}), (\mathit{Frank}, \mathit{George}), \text{id}) \in$ `funded` with *Frank* $\in \{\mathit{Alice}, \mathit{Bob}\}$
  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 $(\text{virtual}, c, \mathcal{F}_{\text{Chan}}, \mathit{Dave}, \text{id}) \in$ `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. 4.

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
    $c_A \leftarrow c_A - c$
    add ((c, 0), hops, `sub_parties`, `outer_parties`, id) to `funded`
    output (OK) to *Alice*

On (ALLOW FUND, c, `sub_parties`, D, id, `is_last`) by *Charlie*:
    ensure *State* ∈ {OPEN BASE, OPEN VIRTUAL}
    ensure $D \in \{Alice, Bob\}$
    ensure $c_D - \text{locked}_D \geq c$
    output received message to D and ensure reply is (OK)
    $\text{locked}_D \leftarrow \text{locked}_D + c$
    if `is_last` **then**
        add ((0, c), $\perp$, `sub_parties`.reverse(), (*Dave*, $\perp$), id) to `funded`
    end if
    send (OK) to *Charlie*

Fig. 5.

Simulator $\mathcal{S}$

On (OPEN, F) by $\mathcal{F}_{\text{Chan}}$:
    TODO: how to trick $\mathcal{E}$? We don't have sk...

On (PAY, x, *Dave*) by $\mathcal{F}_{\text{Chan}}$:
    simulate receiving (PAY, x) with *Dave*
    ensure simulated *Dave* outputs OK
    send OK to $\mathcal{F}_{\text{Chan}}$

On (FUND YOU, c, *Bob*, *Charlie*, *Alice*) by $\mathcal{F}_{\text{Chan}}$:
    simulate *Alice* receiving input (FUND YOU, c, *Bob*) by *Charlie*
    ensure simulated *Alice* outputs OK to *Charlie*
    send OK to $\mathcal{F}_{\text{Chan}}$

Fig. 6.

References