Protocol Π_{Chan}

Initialisation:
    $\mathit{State} \leftarrow$ INIT

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

On (OPEN, c_F, $pk_{A,out}$, $pk_{B,out}$) by $\mathcal{E}$:
    ensure $\mathit{State} =$ TOPPED UP
    $\mathit{State} \leftarrow$ OPENING BASE CHANNEL
    do LN (other box)

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

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

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

On (CLOSE) by $\mathcal{E}$:
    if $\mathit{State} =$ OPEN BASE **then**
        prepare C TODO
        send (SUBMIT, C) to $\mathcal{G}_{\text{Ledger}}$
    else if $\mathit{State} =$ 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* = INIT
    State ← OPENING VIRTUAL CHANNEL
    do LN with *Bob* – TODO
    State ← OPEN VIRTUAL
    output (OK) to *Charlie*

On (FUND, c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*) by $\mathcal{E}$:
    do the same as in $\mathcal{F}_{\mathrm{Chan}}$, Fig. 6, lines 14-19, skipping line 14 // "as *Alice*" sender labels are applied anyway, since we *are Alice*
    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 $\mathcal{F}_{\mathrm{Chan}}$, Fig. 6, lines 21-25
    output (OK) to $\mathcal{E}$

// notification to fundee
On (ALLOW FUND, ...) by *Charlie*, act as $\mathcal{F}_{\mathrm{Chan}}$ (Fig 6, line 27):

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 $\exists \text{tx} \in \Sigma, c_{\text{on}} : c_{\text{on}} \geq c_{\text{min}} \land (c_{\text{on}}, pk) \in \text{tx.outputs}$
    `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 ($\textsc{open}$, c_F, $pk_{A,\text{out}}$, $pk_{B,\text{out}}$) by *Alice*:
  ensure $\mathit{State} = \textsc{topped up}$
  ensure $c_F \geq c_{\text{on}}$
  $(sk_{A,F}, pk_{A,F}) \leftarrow \textsc{KeyGen}()$; $(sk_{B,F}, pk_{B,F}) \leftarrow \textsc{KeyGen}()$
  $F \leftarrow$ TX {input: `base_output`, output: $(c_F, 2/\{pk_{A,F}, pk_{B,F}\})$}
  $F \leftarrow F.\text{sign}(sk)$
  $\mathit{State} \leftarrow \textsc{waiting for ledger}$
  send ($\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, F, \mathit{Alice}$) to $\mathcal{A}$ and ensure reply is ($\textsc{ok}$)
  output ($\textsc{ok}$) to *Alice*

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

On ($\textsc{pay}$, x) by $\mathit{Dave} \in \{\mathit{Alice}, \mathit{Bob}\}$:
  ensure $\mathit{State} \in \{\textsc{open base}, \textsc{open virtual}\}$
  ensure $c_D - \text{locked}_D \geq x$
  send ($\textsc{pay}$, x, *Dave*) to $\mathcal{A}$ and expect reply ($\textsc{ok}$)
  $c_D \leftarrow c_D - x; c_{\bar{D}} \leftarrow c_{\bar{D}} + x$ // $\bar{D}$ is *Alice* if D is *Bob* and vice-versa
  output ($\textsc{pay success}$) to *Dave*

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

Fig. 4.

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

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

On (CLOSING, c_{left}, c_{right}) by $\mathcal{F}_{\text{Chan}}$:
    ensure $\mathit{State} \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
    ensure $((c_L, c_R), \text{hops}, (\mathit{Charlie}, \mathit{Dave}), (\mathit{Frank}, \mathit{George}), \text{id}) \in$ `funded` with $\mathit{Frank} \in \{\mathit{Alice}, \mathit{Bob}\}$
    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 $\mathit{State} \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
    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 $\mathit{State} = \text{CLOSED}$ **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 $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
ensure $c_A - \text{locked}_A \geq c$
input (FUND YOU, c, counterparty) to fundee as *Alice*, ensure output is (OK)
generate random id
send (FUND c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*), `funder` = *Alice*, id) to $\mathcal{A}$ and ensure reply is (OK)
$(L_0, R_0) \leftarrow (Alice, Bob)$
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 (IS OPEN SUCCESSFUL, id) 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 $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
ensure $Dave \in \{Alice, Bob\}$
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 $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
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 $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, F, Alice)$ by $\mathcal{F}_{\text{Chan}}$:
    simulate *Alice* receiving input $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}})$ by $\mathcal{E}$
    ensure simulated *Alice* outputs (OK)
    send (OK) to $\mathcal{F}_{\text{Chan}}$

On (PAY, x, *Dave*) by $\mathcal{F}_{\text{Chan}}$:
    simulate *Dave* receiving input (PAY, x) by $\mathcal{E}$
    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 $\mathcal{F}_{\text{Chan}}$

On (FUND c, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Charlie*, *Dave*), `funder` = *Alice*, id) by $\mathcal{F}_{\text{Chan}}$:
    add the message data to `virtual_opening`
    simulate execution of line 15 of Fig. 6 with *Alice*// $\mathcal{S}$ knows *Bob* (*Alice*'s counterparty) through opening procedure
    send (OK) to $\mathcal{F}_{\text{Chan}}$

On (ALLOW FUND, c, `sub_parties`, `local_funder` = L_i, id, $i \stackrel{?}{=} |\text{hops}|$) by $\mathcal{F}_{\text{Chan}}$'s *Alice* to *Charlie*:
    simulate receiving message with *Charlie* by *Alice* and all subsequent communication
    ensure the simulated *Charlie* sends (OK) to the simulated *Alice*
    intercept this message and send it to $\mathcal{F}_{\text{Chan}}$'s *Alice*

On (IS OPEN SUCCESSFUL, id) by $\mathcal{F}_{\text{Chan}}$:
    retrieve and remove from `virtual_opening` the data marked with id
    simulate line 11 of Fig. 2 with *Alice* using this data
    ensure *Alice* completes execution of VChan() successfully
    send (OK) to $\mathcal{F}_{\text{Chan}}$

On (FUND DONE, id) by $\mathcal{F}_{\text{Chan}}$'s *Alice* to *Charlie*:
    simulate receiving message with *Charlie* by *Alice* and all subsequent communication
    ensure the simulated *Charlie* sends (OK) to the simulated *Alice*
    intercept this message and send it to $\mathcal{F}_{\text{Chan}}$'s *Alice*

Fig. 7.

1 Security Proof

When $\mathcal{E}$ sends (FUND, c, hops, (fundee, counterparty), (*Charlie*, *Dave*)) to *Alice* in the real world, lines 14-12 of Fig. 6 are executed and then control is handed over to the "fundee" ITI, which executes lines 3-8 of Fig. 2. This ITI will output (OK) if and only if line 6 of Fig. 2 succeeds.

When $\mathcal{E}$ sends (FUND, c, hops, (fundee, counterparty), (*Charlie*, *Dave*)) to *Alice* in the ideal world, lines 14-12 of Fig. 6 are executed and then control is handed over to the functionality that controls the "fundee", which executes lines 1-4 of Fig. 6 and then hands control over to $\mathcal{S}$. The latter in turn simulates lines 3-8 of Fig. 2, thus following the exact same steps as in the real world, therefore it will send (OK) to $\mathcal{F}_{\text{Chan}}$ if and only if the simulated line 6 of Fig. 2 succeeds. From this and the previous paragraph, we see that, up to this point, the two worlds are perfectly indistinguishable.

Moving on, in the ideal world subsequently lines 13-14 of Fig. 6 are executed, which results in $\mathcal{S}$ executing lines 13-16 of Fig. 7. During the latter steps, $\mathcal{S}$ simulates executing line 15 of Fig. 6 with *Alice*.

Similarly in the real world, *Alice* executes lines 13 and 15 of Fig. 6, therefore the two worlds still are perfectly indistinguishable.

The "for" loop of lines 16-19 of Fig. 6 is then executed in both the real and the ideal worlds. The message of line 18 results in the execution of lines 3-8 of Fig. 2 by L_i in both worlds: in the real world directly, in the ideal world simulated by $\mathcal{S}$.

In the ideal world, line 20 in Fig. 6 prompts $\mathcal{S}$ to simulate line 11 of Fig. 2 with *Alice*, which is exactly the code that would be directly run by *Alice* in the real world. Therefore the two worlds remain perfectly indistinguishable.

The "for" loop of lines 21-25 of Fig. 6 is also perfectly indistinguishable in the two worlds. With argumentation similar to that of the previous "for" loop, we conclude that the FUND message does not induce any chance of distinguishability between the two worlds.

References