**Protocol** $\Pi_{\text{Chan}}$

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 $\Sigma$
    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** $\Pi_{\text{Chan}}$ – 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` = (*Alice*, *Dave*), $pk_{VA,out}$, $pk_{VB,out}$) by $\mathcal{E}$:
    do the same as in $\mathcal{F}_{\text{Chan}}$, Fig. 8, lines 1-11, skipping line 6 // "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 $\mathcal{F}_{\text{Chan}}$, Fig. 8, lines 19-26
    output (OK) to $\mathcal{E}$

// notification to fundee
On (ALLOW FUND, ...) by *Charlie*, act as $\mathcal{F}_{\text{Chan}}$ (Fig 8, line 9):

**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
    $(sk, pk) \leftarrow$ KEYGEN()
    output (PUBLIC KEY, $pk$) to *Alice*

On (CHECK TOP UP) by *Alice*:

    ensure $\mathit{State} =$ SENT KEY
    input (READ) to $\mathcal{G}_{\text{Ledger}}$ as *Alice* and assign ouput to $\Sigma$
    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, *Alice*)
    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 $State = (\textsc{topped up}, Alice)$
- ensure $c_F = c_{\text{on}}$
- $(sk_{A,F}, pk_{A,F}) \leftarrow \textsc{keyGen}()$; $(sk_{B,F}, pk_{B,F}) \leftarrow \textsc{keyGen}()$
- $F \leftarrow$ TX {input: `base_output`, output: $(c_F, 2/\{pk_{A,F}, pk_{B,F}\})$}
- $\text{sig}_F \leftarrow \textsc{sign}(F, sk)$
- $State \leftarrow \textsc{waiting for ledger}$
- send $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, F, \text{sig}_F, Alice)$ to $\mathcal{A}$

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

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

**Fig. 4.**

**Functionality $\mathcal{F}_{\text{Chan}}$ – close Pt. 1**

On (CLOSE) by *Alice*:<br>
  **if** *State* = OPEN BASE **then**<br>
    **if** both channel parties are honest **then**<br>
      $C \leftarrow$ TX {input: $F$.output, outputs: $(c_A, pk_{A,\text{out}} \wedge t), (c_B, pk_{B,\text{out}})$}<br>
      $\text{sig}_{B,C} \leftarrow \text{SIGN}(C, sk_{B,F})$<br>
    **end if** // note that rest is run only if *Alice* is honest<br>
    $\text{sig}_{A,C} \leftarrow \text{SIGN}(C, sk_{A,F})$<br>
    *State* $\leftarrow$ CLOSED<br>
    input (SUBMIT, $(C, \text{sig}_{A,C}, \text{sig}_{B,C})$) to $\mathcal{G}_{\text{Ledger}}$<br>
  **else if** *State* = OPEN VIRTUAL **then** TODO: think more about closing w/ one corrupted party<br>
    *State* $\leftarrow$ CLOSED<br>
    output (CLOSING, $c_A$, $c_B$) to `opener`<br>
  **end if**

On (CLOSE) by $\mathcal{A}$:<br>
  ensure that exactly one party in {*Alice*, *Bob*} is corrupted<br>
  wlog, let *Alice* corrupted, *Bob* honest<br>
  $C' \leftarrow$ TX {input: $F$.output, outputs: $(c_B, pk_{B,\text{out}}), (c_A, pk_{A,\text{out}} \wedge t)$}<br>
  $\text{sig}_{B,C'} \leftarrow \text{SIGN}(C', sk_{B,F})$<br>
  *State* $\leftarrow$ CLOSED<br>
  send (C', $\text{sig}_{B,C'}$) to $\mathcal{A}$<br>
  TODO: asynchronously, if a lot of time passes and $C'$ not in $\mathcal{G}_{\text{Ledger}}$, submit $C$ to $\mathcal{G}_{\text{Ledger}}$ for good measure

**Fig. 5.**

**Functionality $\mathcal{F}_{\text{Chan}}$ – close Pt. 2**

On ($\textsc{closing}$, $c_{\text{left}}$, $c_{\text{right}}$) by $\mathcal{F}_{\text{Chan}}$:
ensure $\mathit{State} \in \{\textsc{open base}, \textsc{open virtual}\}$
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 ($\textsc{closed virtual}$, $c_{\text{right}}$, id) to *Frank*

On ($\textsc{closed virtual}$, $c_{\text{right}}$, id) by $\mathcal{F}_{\text{Chan}}$:
ensure $\mathit{State} \in \{\textsc{open base}, \textsc{open virtual}\}$
ensure $(\text{virtual}, c, \mathcal{F}_{\text{Chan}}, \mathit{Dave}, \text{id}) \in$ `funded`
ensure $c_{\text{right}} \leq c$
send ($\textsc{closed}$) to virtual and expect reply ($\textsc{yes}$)
$c_D \leftarrow c_D + c_{\text{right}}$
remove entry from `funded`

On ($\textsc{closed}$) by $P$:
**if** $\mathit{State} = \textsc{closed}$ **then**
send ($\textsc{yes}$) to $P$
**else**
send ($\textsc{no}$) to $P$
**end if**

**Fig. 6.**

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

On (FUND, $c$, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Alice*, *Dave*), $pk_{VA,out}$, $pk_{VB,out}$) by *Alice*: // we fund another channel TODO: use $pk_{VA}$ in virtual LN()

  ensure *State* $\in$ {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)

  generate random id

  send (FUND $c$, hops, `sub_parties` = (fundee, counterparty), `outer_parties` = (*Alice*, *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** // $i \in \{1, \dots, |\text{hops}|\}$

    ensure $R_{i-1} = L_i$

    send (ALLOW FUND, $c$, `sub_parties`, `local_funder` $\leftarrow L_i$, id, $i \stackrel{?}{=} |\text{hops}|$) to $L_i$ as *Alice* and ensure reply is (OK)

  **end for**

  **if** both channel parties are honest **then**

    send (IS OPEN SUCCESSFUL, id) to $\mathcal{A}$ and ensure reply is (OK)

  **else if** only *Alice* is honest **then**

    $(sk_{A,V}, pk_{A,V}) \leftarrow \text{KEYGEN}()$

    send (UPDATE TO VIRTUAL, $pk_{A,V}$) to $\mathcal{A}$ and assign reply to ($V$ = TX {input: $F$.output, outputs: $(c_A + c_B - c$, $2/\{pk_{A,V}, pk_{B,V}\})$, $(c$, $2/\{pk_{G,V}, pk_{A,V}\})$, $(0$, $|\text{hops}|/\{\text{hops}_i.pk\}_i)$}, $\text{sig}_{B,V}$, $C'$ = TX {input: $V$.outputs.0, outputs: $(c_A - \text{locked}_A - c, pk_{A,\text{out}} \wedge t), (c_B - \text{locked}_B, pk_{B,\text{out}})$}, $\text{sig}_{B,C'}$) TODO: think about locked coins

    ensure $\text{VERIFY}(V, \text{sig}_{B,V}, pk_{B,F}) = \text{VERIFY}(C', \text{sig}_{B,C'}, pk_{A,V}) = \text{True}$

  **end if**

  **for all** $(L_i, R_i) \in$ hops **do** // $i \in \{1, \dots, |\text{hops}|\}$

    send (FUND DONE, id) to $L_i$ as *Alice* and ensure reply is (OK)

  **end for**

  $c_A \leftarrow c_A - c$

  **if** only *Alice* is honest **then**

    $C \leftarrow C'$; $\text{sig}_{B,C} \leftarrow \text{sig}_{B,C'}$

  **end if**

  add $((c, 0)$, hops, `sub_parties`, `outer_parties`, id) to `funded`

  output (OK) to *Alice*

**Fig. 7.**

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

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

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*) $\in$ `pending`
    remove (id, `is_last`, `sub_parties`, $c$, *Dave*) from `pending`
    **if** `is_last` **then**
        add ((0, $c$), $\perp$, `sub_parties`.reverse(), (*Dave*, $\perp$), id) to `funded`
    **end if**
    send (OK) to *Charlie*

**Fig. 8.**

**Functionality $\mathcal{F}_{\text{Chan}}$ – corruption**

On $(\textsc{corrupt})$ by $P$, addressed to *Alice*:
    ensure $P \in \{\texttt{opener}, \mathcal{A}\}$
    **for all** $(\_, \_, (\text{fundee}, \_), (\mathit{Alice}, \_)) \in \texttt{funded}$ **do**
        send $(\textsc{corrupt})$ to fundee and ensure reply is $(\textsc{ok})$
    **end for**
    from now on, allow $\mathcal{A}$ to handle all *Alice*'s messages, i.e. act as a relay
    **if** *Bob* is not corrupted **then**
        from now on, override reactions to messages $(\textsc{open})$ and $(\textsc{pay})$ addressed to *Bob* with those defined in the current Figure:
    **end if**
    output $(\textsc{ok})$ to $P$

On $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}})$ by *Bob*:
    ensure $\mathit{State} = (\textsc{topped up}, \mathit{Bob})$ TODO: decide what happens when channel funded by corrupted party
    ensure $c_F = c_{\text{on}}$
    $(sk_{B,F}, pk_{B,F}) \leftarrow \textsc{KeyGen}()$
    send $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, pk_{B,F}, \mathit{Bob})$ to $\mathcal{A}$, assign reply to $(pk_{A,F}$, $C = \text{TX}$ $\{$input: $F$.output, outputs: $(c_F, pk_{B,\text{out}} \wedge t)$, $(0, pk_{A,\text{out}})\}$, $\text{sig}_{A,C})$
    ensure $\textsc{verify}(C, \text{sig}_{A,C}, pk_{A,F}) = \text{True}$
    $F \leftarrow \text{TX}$ $\{$input: `base_output`, output: $(c_F, 2/\{pk_{A,F}, pk_{B,F}\})\}$
    $\text{sig}_F \leftarrow \textsc{sign}(F, sk)$
    $\mathit{State} \leftarrow \textsc{waiting for ledger}$
    input $(\textsc{submit}, (F, \text{sig}_F))$ to $\mathcal{G}_{\text{Ledger}}$

On $(\textsc{pay}, x)$ by *Bob*:
    ensure $\mathit{State} \in \{\textsc{open base}, \textsc{open virtual}\}$
    ensure $c_B - \text{locked}_B \geq x$
    send $(\textsc{pay}, x, \mathit{Bob})$ to $\mathcal{A}$ and assign reply to $(C' = \text{TX}$ $\{$input: $F$.output, outputs: $(c_A + x, pk_{A,\text{out}}), (c_B - \text{locked}_B - x, pk_{B,\text{out}} \wedge t)\}$, $\text{sig}'_{A,C})$ TODO: think about locked coins again
    ensure $\textsc{verify}(C', \text{sig}'_{A,C}, pk_{A,F}) = \text{True}$
    $C \leftarrow C'$; $\text{sig}_{A,C} \leftarrow \text{sig}'_{A,C}$
    $c_B \leftarrow c_B - x; c_A \leftarrow c_A + x$
    output $(\textsc{pay success})$ to *Bob*

**Fig. 9.**

**Simulator $\mathcal{S}$– Pt. 1**

On $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, F, \text{sig}_F\, \textit{Alice})$ by $\mathcal{F}_{\text{Chan}}$: // both honest
    simulate *Alice* receiving input $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}})$ by $\mathcal{E}$
    ensure simulated *Alice* inputs $(\textsc{submit}, (F', \text{sig}_{F'}))$ to $\mathcal{G}_{\text{Ledger}}$
    input $(\textsc{submit}, (F, \text{sig}_F))$ to $\mathcal{G}_{\text{Ledger}}$

On $(\textsc{open}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, pk_{B,F}, \textit{Bob})$ by $\mathcal{F}_{\text{Chan}}$: // *Alice* corrupted
    send LN message $(\textsc{open}, pk_{B,F})$ to *Alice* and relay reply to $\mathcal{F}_{\text{Chan}}$ TODO: change msg to fit LN, ensure *Alice* doesn't see a difference from real world

On $(\textsc{pay}, x, \textit{Dave})$ by $\mathcal{F}_{\text{Chan}}$:
    **if** both channel parties are honest **then**
        simulate *Dave* receiving input $(\textsc{pay}, x)$ by $\mathcal{E}$
        ensure simulated *Dave* outputs $(\textsc{ok})$
        send $(\textsc{ok})$ to $\mathcal{F}_{\text{Chan}}$
    **else if** only *Dave*'s counterparty is corrupted **then** // else just relay to $\mathcal{A}$
        simulate *Dave* receiving input $(\textsc{pay}, x)$ by $\mathcal{E}$
        ensure simulated *Dave* outputs $(\textsc{ok})$
        extract the latest commitment transaction $C$ and its signature by *Dave*'s counterparty $\text{sig}_{\bar{D},C}$ from simulated *Dave*'s state
        send $(C, \text{sig}_{\bar{D},C})$ to $\mathcal{F}_{\text{Chan}}$
    **end if**

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

On ($\textsc{fund}$ $c$, 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 7 of Fig. 8 with *Alice*// $\mathcal{S}$ knows *Bob* (*Alice*'s counterparty) through opening procedure
    send $(\textsc{ok})$ to $\mathcal{F}_{\text{Chan}}$

On ($\textsc{allow fund}$, $c$, `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 $(\textsc{ok})$ to the simulated *Alice*
    intercept this message and send it to $\mathcal{F}_{\text{Chan}}$'s *Alice*

**Fig. 10.**

**Simulator** $\mathcal{S}$– Pt. 2

```
On (IS OPEN SUCCESSFUL, id) by F_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 F_Chan

On (UPDATE TO VIRTUAL ) by F_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
    extract from Alice's state the new virtual funding TX V for pre-existing channel
    extract from Alice's state the new commitment TX C that spends the on-chain funding TX
    send (V, C) to F_Chan

On (FUND DONE, id) by F_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 F_Chan's Alice
```

**Fig. 11.**

# 1 Security Proof

When $\mathcal{E}$ sends (FUND, $c$, hops, (fundee, counterparty), (*Charlie*, *Dave*), $pk_{VA,out}$, $pk_{VB,out}$) to *Alice* in the real world, lines 1-4 of Fig. 8 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 1-4 of Fig. 8 are executed and then control is handed over to the functionality that controls the "fundee", which executes lines 1-4 of Fig. 8 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 5-6 of Fig. 8 are executed, which results in $\mathcal{S}$ executing lines 22-25 of Fig. 10. During the latter steps, $\mathcal{S}$ simulates executing line 7 of Fig. 8 with *Alice*.

Similarly in the real world, *Alice* executes lines 5 and 7 of Fig. 8, therefore the two worlds still are perfectly indistinguishable.

The “for” loop of lines 8-11 of Fig. 8 is then executed in both the real and the ideal worlds. The message of line 10 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 18 in Fig. 8 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 19-26 of Fig. 8 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