

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

- 1: Initialisation:
- 2:    $State \leftarrow \text{INIT}$
  
- 3: On TOP UP, CHECK TOP UP by  $\mathcal{E}$ , act as  $\mathcal{F}_{\text{Chan}}$  (Fig. 3, lines 4-8 and 9-15 respectively)
  
- 4: On (OPEN,  $c_F$ ,  $pk_{A,out}$ ,  $pk_{B,out}$ ) by  $\mathcal{E}$ :
- 5:   ensure  $State = \text{TOPPED UP}$
- 6:    $State \leftarrow \text{OPENING BASE CHANNEL}$
- 7:   do LN (other box)
  
- 8: On (CHECK FUNDING) by  $\mathcal{E}$ :
- 9:   ensure  $State = \text{WAITING FOR LEDGER}$
- 10:   send (READ) to  $\mathcal{G}_{\text{Ledger}}$  and assign reply to  $\Sigma$
- 11:   ensure  $F \in \Sigma$
- 12:    $c_A \leftarrow c$ ;  $c_B \leftarrow 0$  //  $c$  received in OPEN
- 13:    $State \leftarrow \text{OPEN BASE}$
- 14:   output (OPEN SUCCESS) to  $\mathcal{E}$
  
- 15: On (PAY,  $x$ ) by  $\mathcal{E}$ :
- 16:   ensure  $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
- 17:   ensure  $c_A \geq x$
- 18:   do LN payment (these channels won't be async) (balance change here)
- 19:   output (OK) to  $\mathcal{E}$
  
- 20: On (BALANCE) by  $\mathcal{E}$ :
- 21:   ensure  $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
- 22:   output (BALANCE,  $(c_A, c_B, \text{locked}_A, \text{locked}_B)$ ) to  $\mathcal{E}$
  
- 23: On (CLOSE) by  $\mathcal{E}$ :
- 24:   **if**  $State = \text{OPEN BASE}$  **then**
- 25:     prepare  $C$  TODO
- 26:     send (SUBMIT,  $C$ ) to  $\mathcal{G}_{\text{Ledger}}$
- 27:   **else if**  $State = \text{OPEN VIRTUAL}$  **then**
- 28:     TODO
- 29:   **end if**

**Fig. 1.**

**Protocol  $\Pi_{\text{Chan}}$  – 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  $\leftarrow$  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  $\in$  {OPEN BASE, OPEN VIRTUAL}
10:  do the same as in  $\mathcal{F}_{\text{Chan}}$  (Fig. 6, 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 6, line 22):
```

**Fig. 2.**

**Functionality  $\mathcal{F}_{\text{Chan}} - \text{init}$**

- 1: Initialisation: // runs on first activation
- 2:      $State \leftarrow \text{INIT}$
- 3:      $(\text{locked}_A, \text{locked}_B) \leftarrow (0, 0)$
  
- 4: On (TOP UP,  $c_{\min}$ ) by *Alice*:
- 5:     ensure  $State = \text{INIT}$
- 6:      $State \leftarrow \text{SENT KEY}$
- 7:      $(sk, pk) \leftarrow \text{KEYGEN}()$
- 8:     output (PUBLIC KEY,  $pk$ ) to *Alice*
  
- 9: On (CHECK TOP UP) by *Alice*:
- 10:     ensure  $State = \text{SENT KEY}$
- 11:     send (READ) to  $\mathcal{G}_{\text{Ledger}}$  as *Alice* and assign reply to  $\Sigma$
- 12:     ensure  $\exists tx \in \Sigma, c_{\text{on}} : c_{\text{on}} \geq c_{\min} \wedge (c_{\text{on}}, pk) \in tx.\text{outputs}$
- 13:      $\text{base\_output} \leftarrow (c_{\text{on}}, pk)$  of tx
- 14:      $State \leftarrow \text{TOPPED UP}$
- 15:     output (TOPPED UP) to *Alice*

**Fig. 3.**

**Functionality  $\mathcal{F}_{\text{Chan} - \text{base}}$**

- 1: On (OPEN,  $c_F$ ,  $pk_{A,\text{out}}$ ,  $pk_{B,\text{out}}$ ) by *Alice*:
- 2:   ensure  $State = \text{TOPPED UP}$
- 3:   ensure  $c_F \geq c_{\text{on}}$
- 4:    $(sk_{A,F}, pk_{A,F}) \leftarrow \text{KEYGEN}()$ ;  $(sk_{B,F}, pk_{B,F}) \leftarrow \text{KEYGEN}()$
- 5:    $F \leftarrow \text{TX}$  {input: **base\_output**, output:  $(c_F, 2/\{pk_{A,F}, pk_{B,F}\})$ }
- 6:    $F \leftarrow F.\text{sign}(sk)$
- 7:    $State \leftarrow \text{WAITING FOR LEDGER}$
- 8:   send (OPEN,  $c_F$ ,  $pk_{A,\text{out}}$ ,  $pk_{B,\text{out}}$ ,  $F$ , *Alice*) to  $\mathcal{A}$  and ensure reply is OK
- 9:   output OK to *Alice*
  
- 10: On (CHECK FUNDING) by *Alice*:
- 11:   ensure  $State = \text{WAITING FOR LEDGER}$
- 12:   send (READ) to  $\mathcal{G}_{\text{Ledger}}$  as *Alice* and assign reply to  $\Sigma$
- 13:   ensure  $F \in \Sigma$
- 14:    $c_A \leftarrow c$ ;  $c_B \leftarrow 0$
- 15:    $State \leftarrow \text{OPEN BASE}$
- 16:   output (OPEN SUCCESS) to *Alice*
  
- 17: On (PAY,  $x$ ) by *Dave*  $\in \{Alice, Bob\}$ :
- 18:   ensure  $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
- 19:   ensure  $c_D - \text{locked}_D \geq x$
- 20:   send (PAY,  $x$ , *Dave*) to  $\mathcal{A}$  and expect reply (OK)
- 21:    $c_D \leftarrow c_D - x$ ;  $c_{\bar{D}} \leftarrow c_{\bar{D}} + x$  //  $\bar{D}$  is *Alice* if  $D$  is *Bob* and vice-versa
- 22:   output (PAY SUCCESS) to *Dave*
  
- 23: On (BALANCE) by *Dave*  $\in \{Alice, Bob\}$ :
- 24:   ensure  $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$
- 25:   output (BALANCE,  $(c_A, c_B, \text{locked}_A, \text{locked}_B)$ ) to *Dave*

**Fig. 4.**

**Functionality  $\mathcal{F}_{\text{Chan}} - \text{close}$**

```

1: On (CLOSE) by Alice:
2:   if State = OPEN BASE then
3:      $C \leftarrow \text{TX } \{\text{input: } F.\text{out}, \text{outputs: } (c_A, pk_{A,\text{out}}), (c_B, pk_{B,\text{out}})\}$ 
4:      $C \leftarrow C.\text{sign}(\text{sk}_{A,F}, \text{sk}_{B,F})$ 
5:     State  $\leftarrow$  CLOSED
6:     input (SUBMIT, C) to  $\mathcal{G}_{\text{Ledger}}$ 
7:   else if State = OPEN VIRTUAL then
8:     State  $\leftarrow$  CLOSED
9:     output (CLOSING,  $c_A, c_B$ ) to opener
10:  end if

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

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

24: On (CLOSED) by P:
25:   if State = CLOSED then
26:     send (YES) to P
27:   else
28:     send (NO) to P
29:   end if

```

**Fig. 5.**

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

```

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

9: On (FUND,  $c$ , hops, sub_parties = (fundee, counterparty), outer_parties =
   ( $Charlie$ ,  $Dave$ )) by  $Alice$ : // we fund another channel
10:  ensure  $State \in \{\text{OPEN BASE}, \text{OPEN VIRTUAL}\}$ 
11:  ensure  $c_A - \text{locked}_A \geq c$ 
12:  input (FUND YOU,  $c$ , counterparty) to fundee as  $Alice$ , ensure output is (OK)
13:   $(L_0, R_0) \leftarrow (Alice, Bob)$ 
14:  generate random id
15:  for all  $(L_i, R_i) \in \text{hops}$  do //  $i \in \{1, \dots, |\text{hops}|\}$ 
16:    ensure  $R_{i-1} = L_i$ 
17:    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)
18:  end for
19:   $c_A \leftarrow c_A - c$ 
20:  add  $((c, 0), \text{hops}, \text{sub_parties}, \text{outer_parties}, \text{id})$  to funded
21:  output (OK) to  $Alice$ 

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

```

**Fig. 6.**

**Simulator  $\mathcal{S}$** 

- 1: On  $(\text{OPEN}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}}, F, \text{Alice})$  by  $\mathcal{F}_{\text{Chan}}$ :
- 2:   simulate *Alice* receiving input  $(\text{OPEN}, c_F, pk_{A,\text{out}}, pk_{B,\text{out}})$  by  $\mathcal{E}$
- 3:   ensure simulated *Alice* outputs OK
- 4:   send OK to  $\mathcal{F}_{\text{Chan}}$
  
- 5: On  $(\text{PAY}, x, \text{Dave})$  by  $\mathcal{F}_{\text{Chan}}$ :
- 6:   simulate *Dave* receiving input  $(\text{PAY}, x)$  by  $\mathcal{E}$
- 7:   ensure simulated *Dave* outputs OK
- 8:   send OK to  $\mathcal{F}_{\text{Chan}}$
  
- 9: On  $(\text{FUND YOU}, c, \text{Bob}, \text{Charlie}, \text{Alice})$  by  $\mathcal{F}_{\text{Chan}}$ :
- 10:   simulate *Alice* receiving input  $(\text{FUND YOU}, c, \text{Bob})$  by *Charlie*
- 11:   ensure simulated *Alice* outputs OK to *Charlie*
- 12:   send OK to  $\mathcal{F}_{\text{Chan}}$

**Fig. 7.****References**