A Composable Security Treatment of the Lightning Network

Aggelos Kiayias

Orfeas Stefanos Thyfronitis Litos

CSF 2020

17/3/2022

VISA

20,000 tx/s

bitcoin

7 tx/s

VISA

instant*

bitcoin

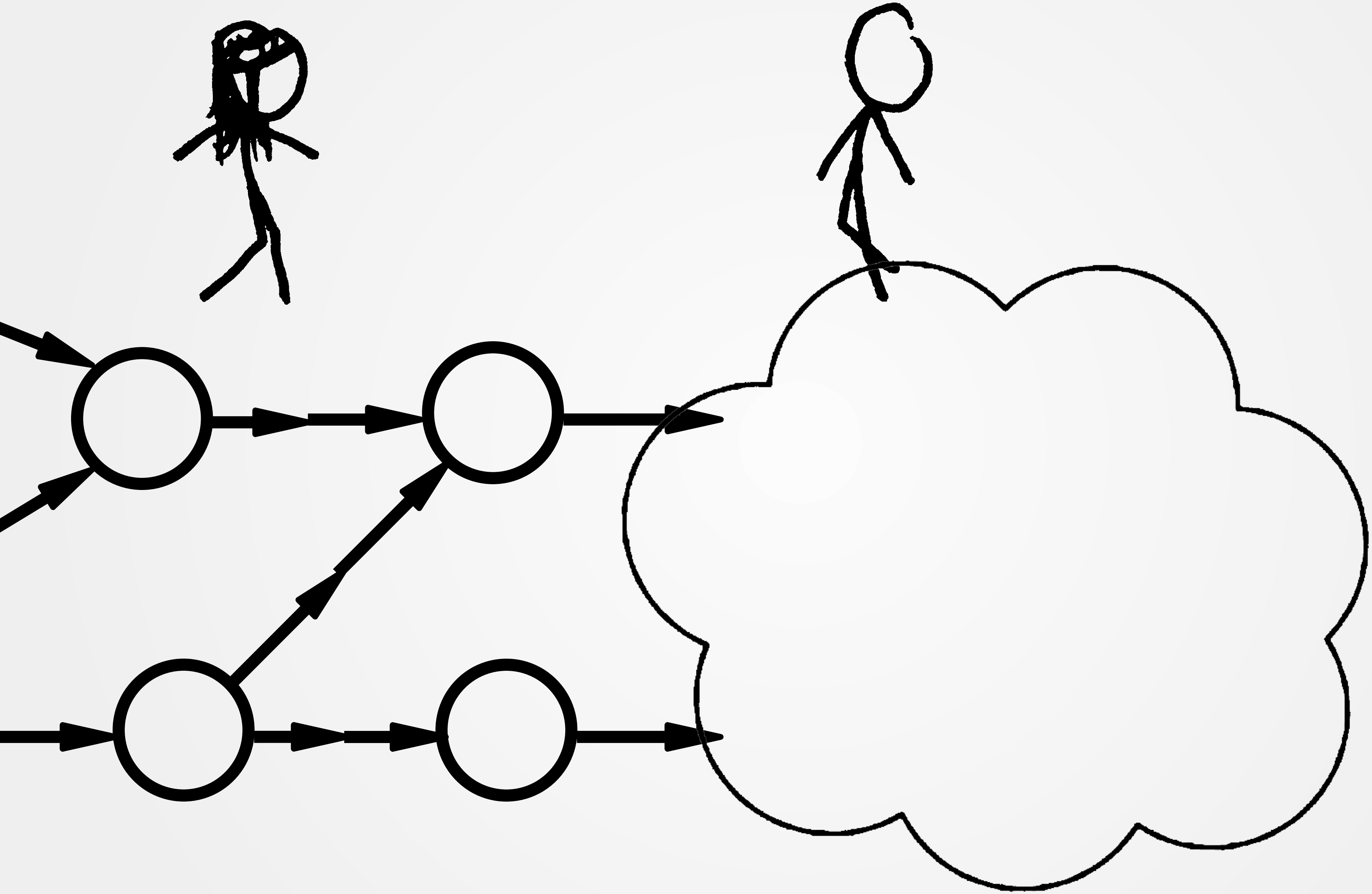
1 hour

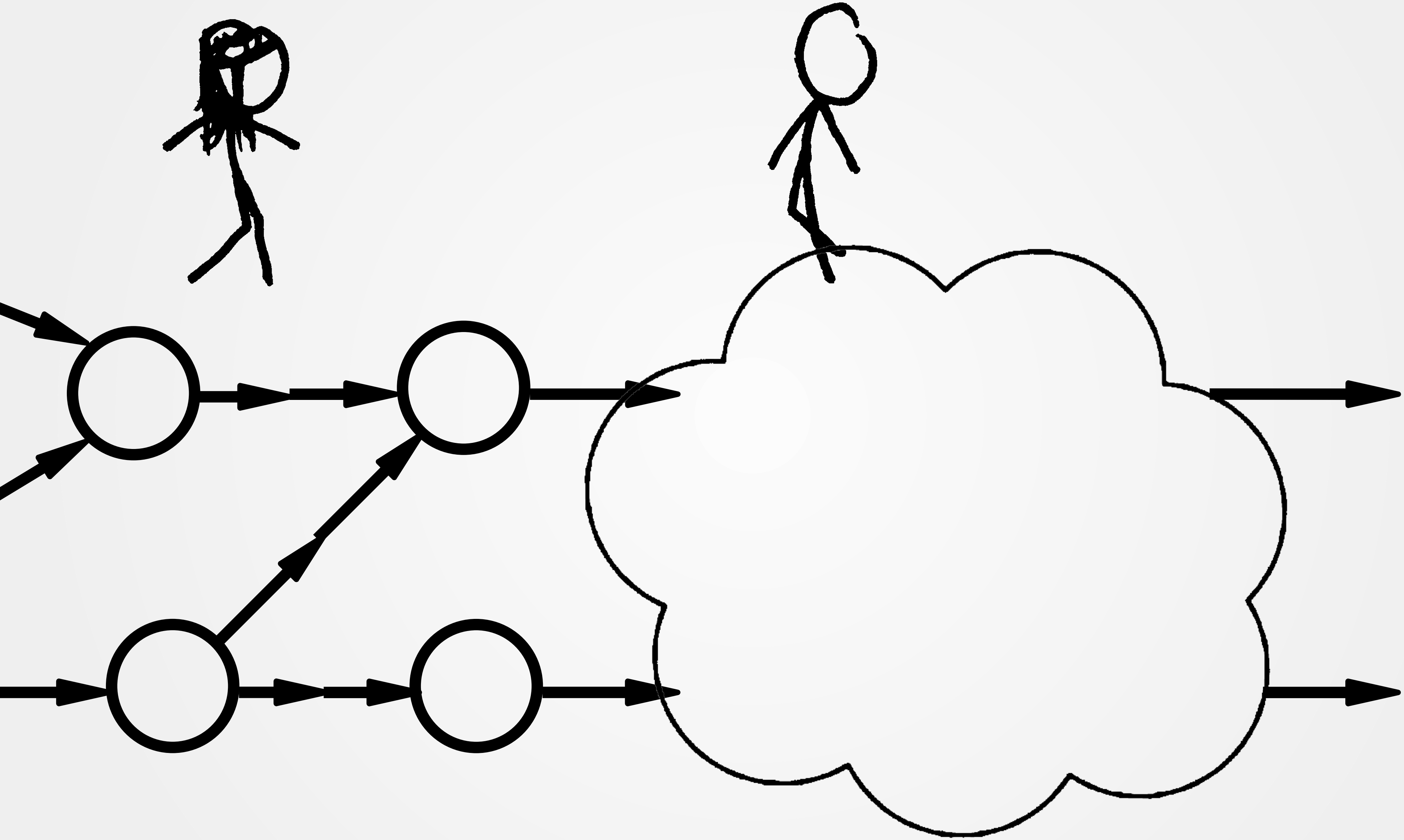
Problem

All txs validated by all wallets

Solution

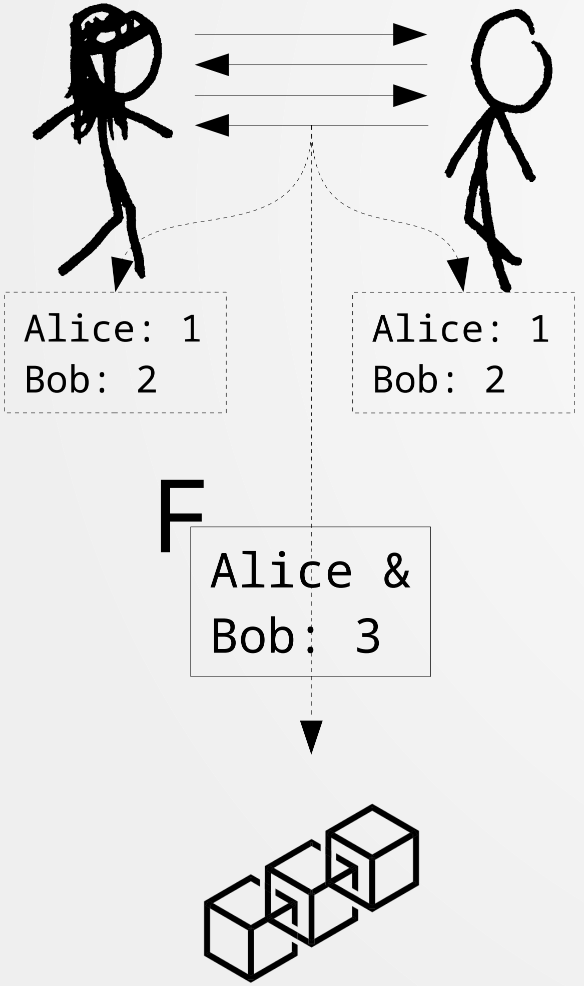
- Move most txs off-chain
- Resolve disputes on-chain





Lightning Channels

Open



Alice: 1
Bob: 2

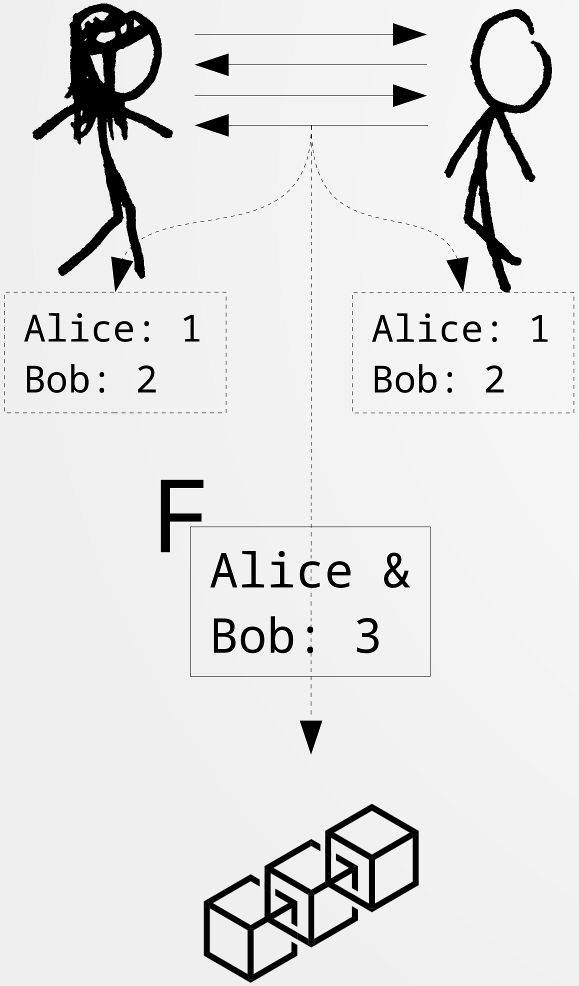
Alice: 1
Bob: 2

F

Alice &
Bob: 3

Lightning Channels

Open



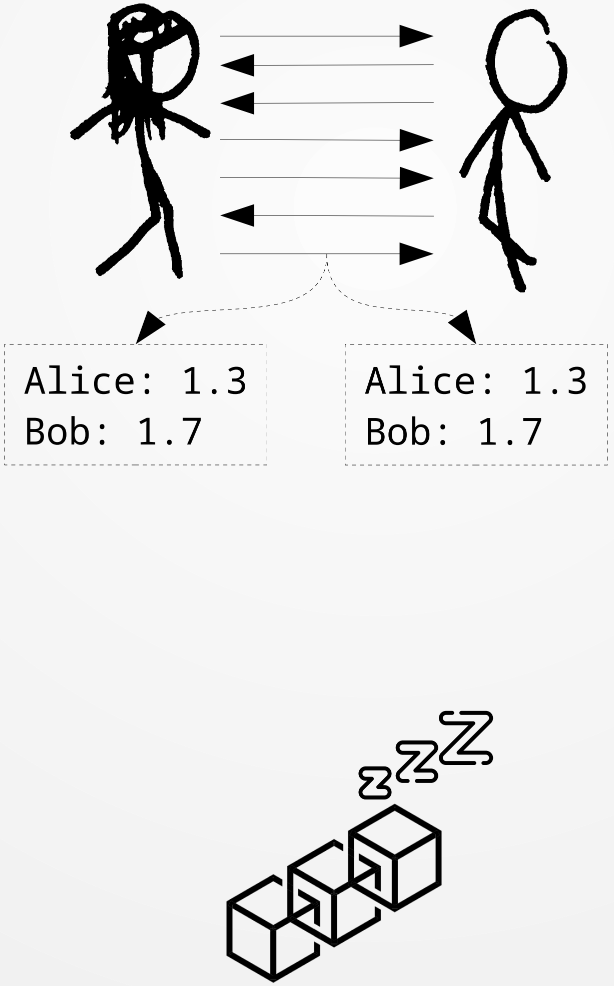
Alice: 1
Bob: 2

Alice: 1
Bob: 2

F

Alice &
Bob: 3

Pay

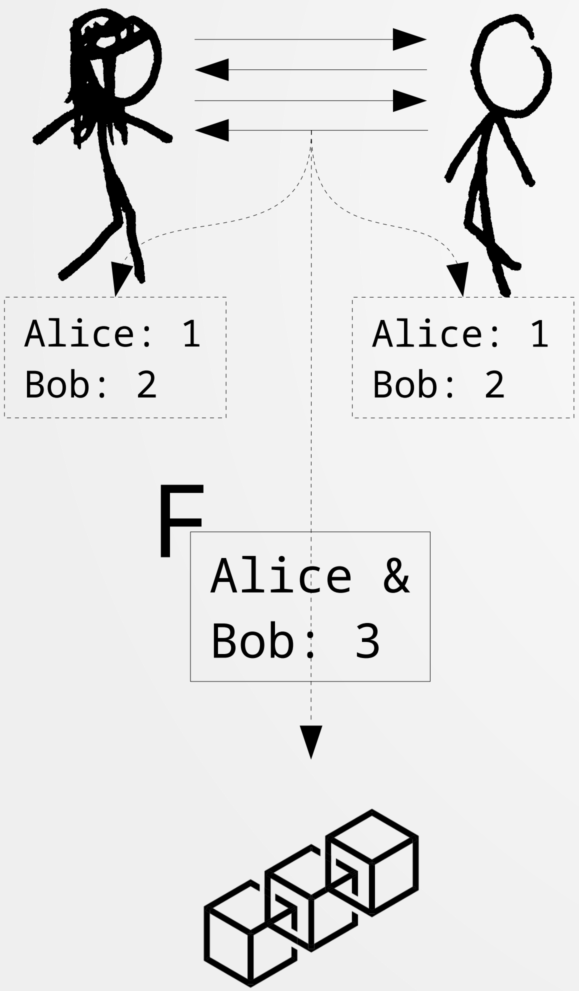


Alice: 1.3
Bob: 1.7

Alice: 1.3
Bob: 1.7

Lightning Channels

Open



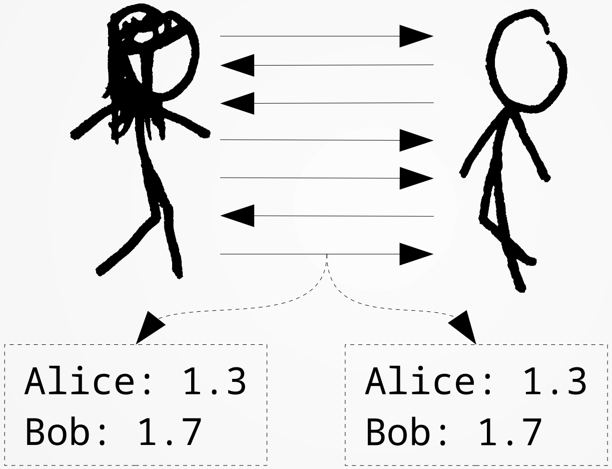
Alice: 1
Bob: 2

Alice: 1
Bob: 2

F

Alice &
Bob: 3

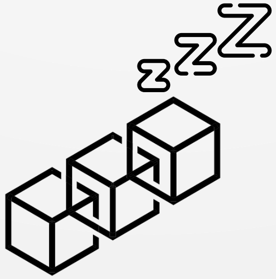
Pay



Alice: 1.3
Bob: 1.7

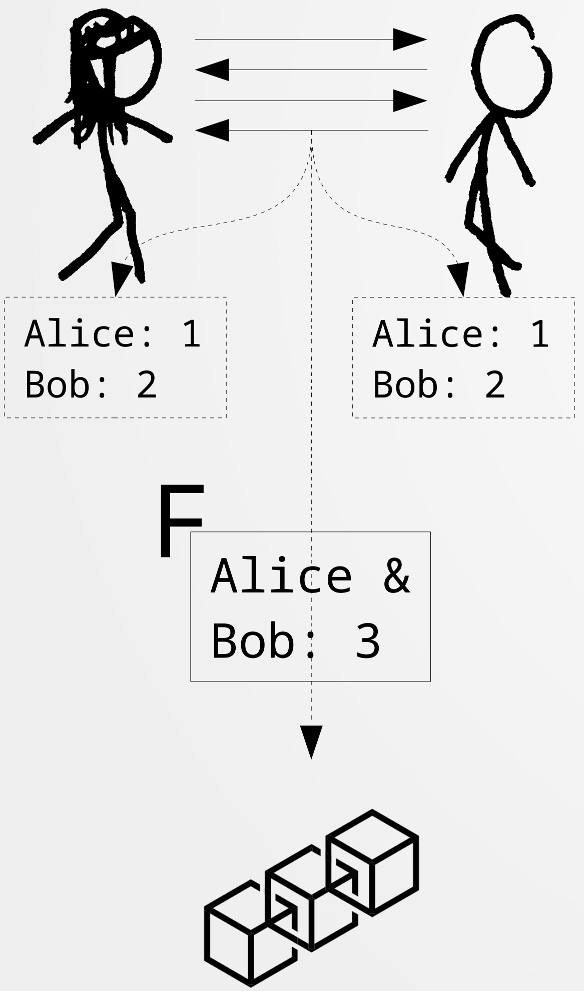
Alice: 1.3
Bob: 1.7

- Unlimited times
- No touching blockchain



Lightning Channels

Open



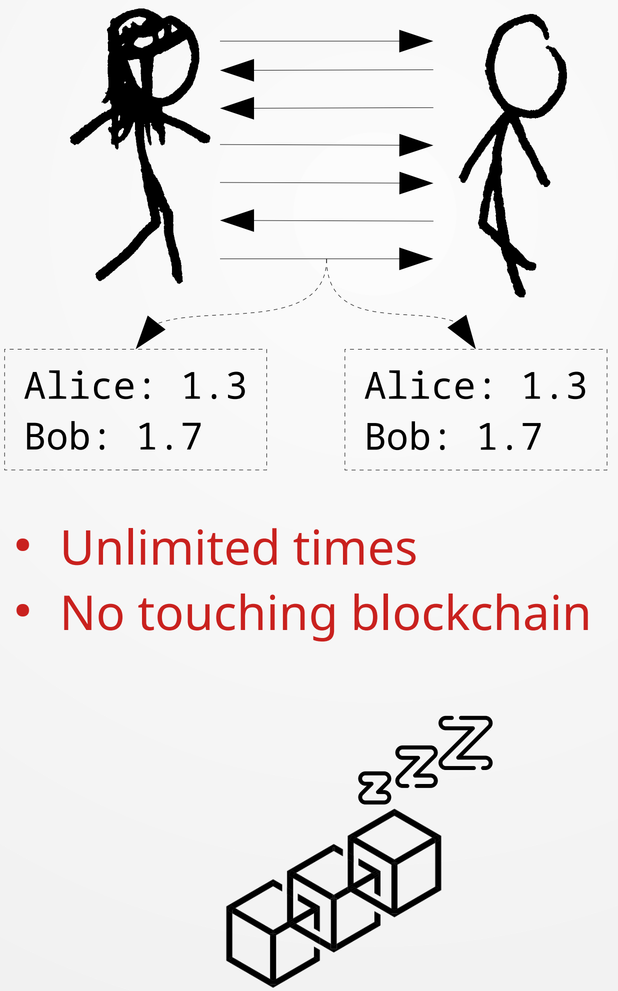
Alice: 1
Bob: 2

Alice: 1
Bob: 2

F

Alice &
Bob: 3

Pay

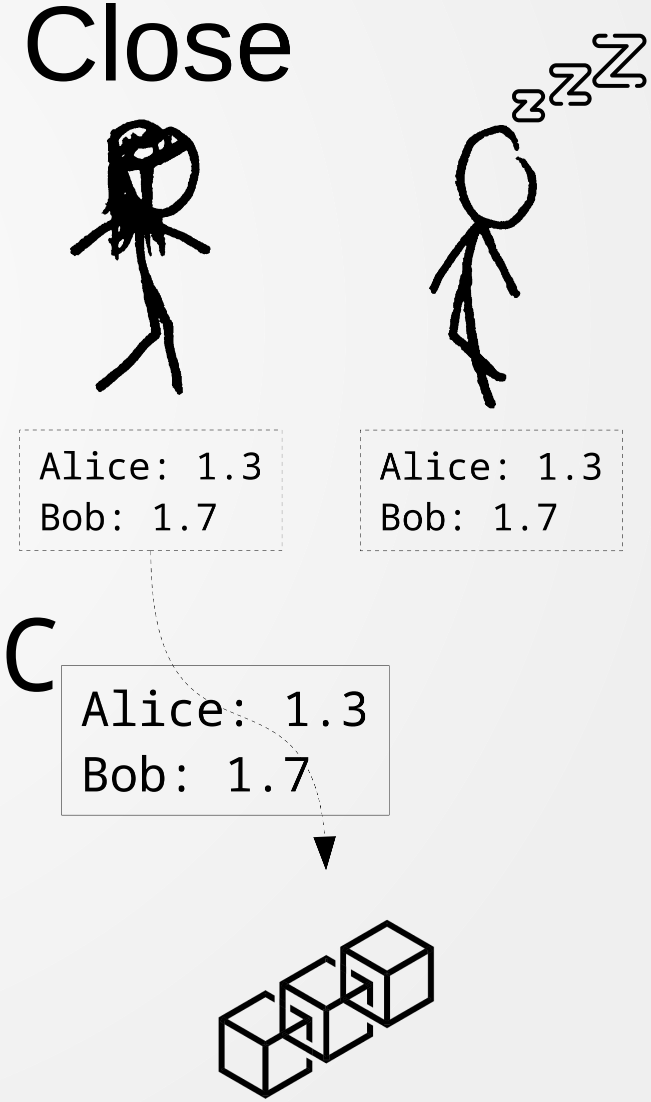


Alice: 1.3
Bob: 1.7

Alice: 1.3
Bob: 1.7

- Unlimited times
- No touching blockchain

Close



Alice: 1.3
Bob: 1.7

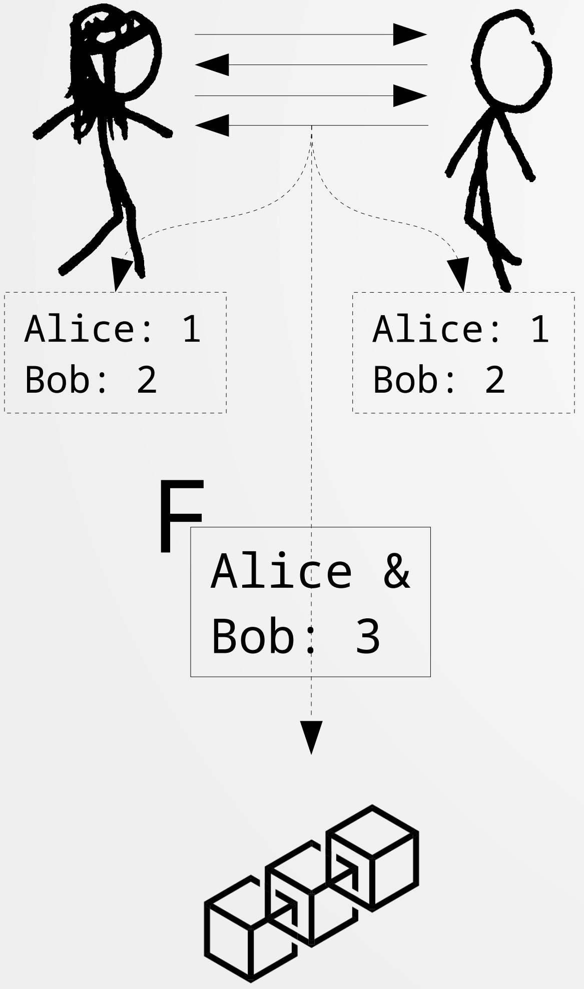
Alice: 1.3
Bob: 1.7

C

Alice: 1.3
Bob: 1.7

Lightning Channels

Open



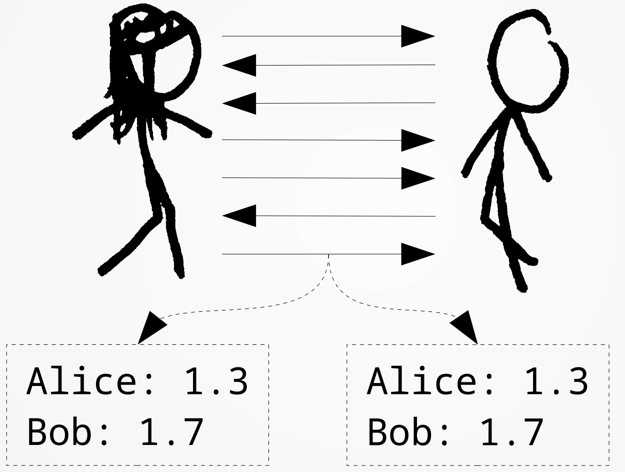
Alice: 1
Bob: 2

Alice: 1
Bob: 2

F

Alice &
Bob: 3

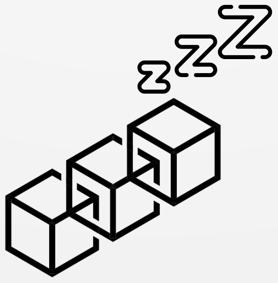
Pay



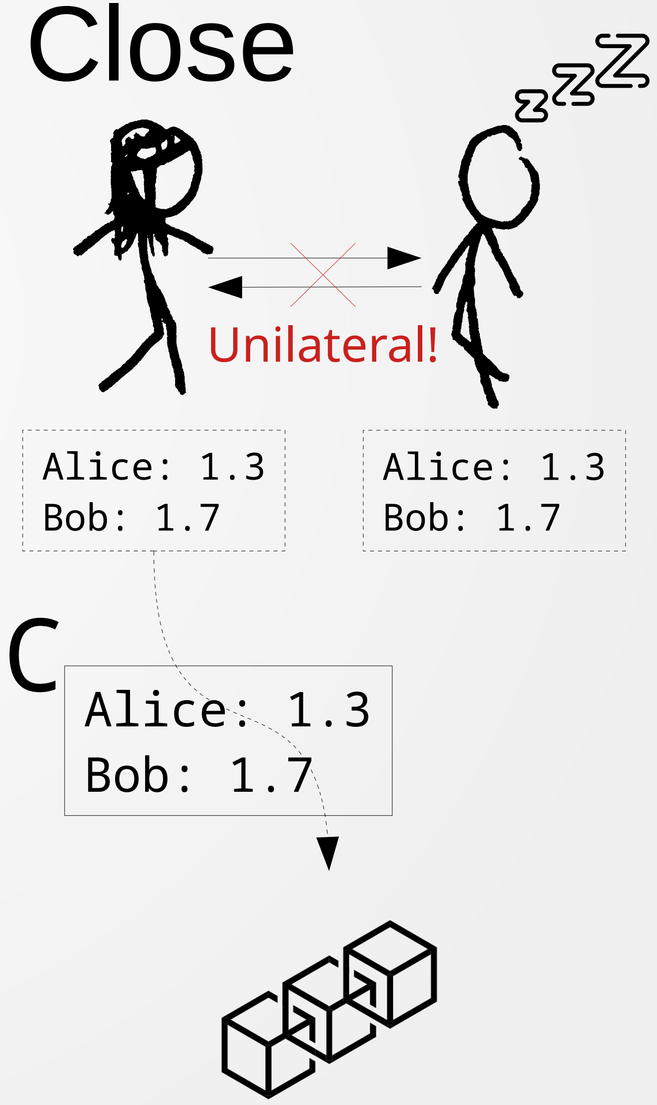
Alice: 1.3
Bob: 1.7

Alice: 1.3
Bob: 1.7

- Unlimited times
- No touching blockchain



Close



Unilateral!

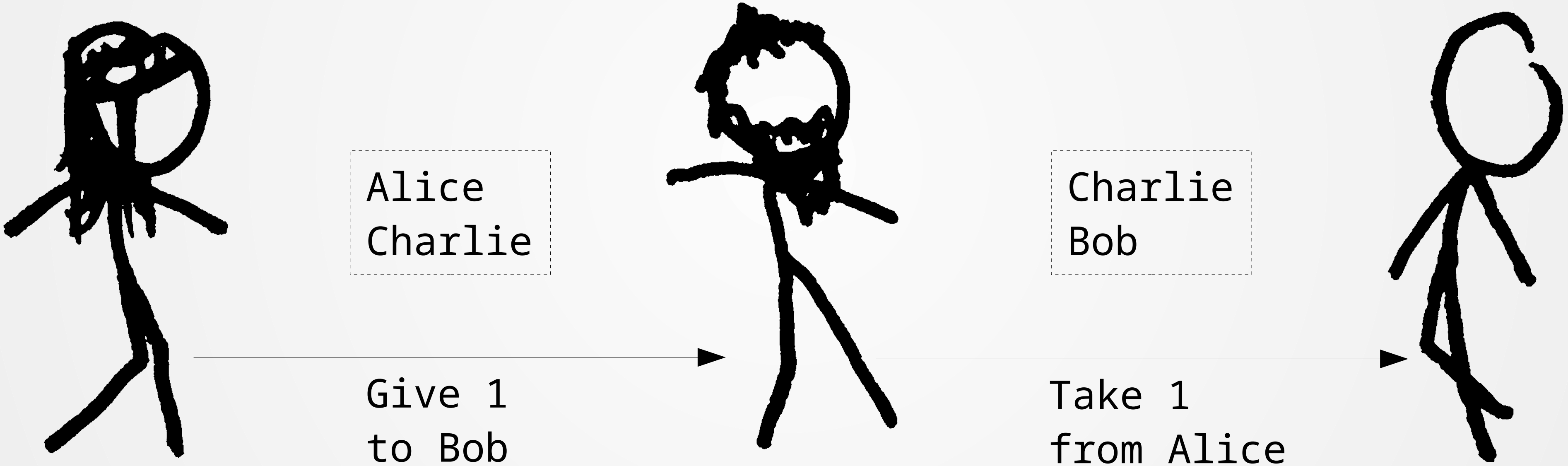
Alice: 1.3
Bob: 1.7

Alice: 1.3
Bob: 1.7

C

Alice: 1.3
Bob: 1.7

Multi-hop payments



Alice
Charlie

Give 1
to Bob

Charlie
Bob

Take 1
from Alice

From channels
to network!

Main result

Prove Lightning Network secure in the Universal Composability framework

Simulation-based Security

$$\forall \mathcal{A} \; \exists \mathcal{S} : \forall \mathcal{E}$$

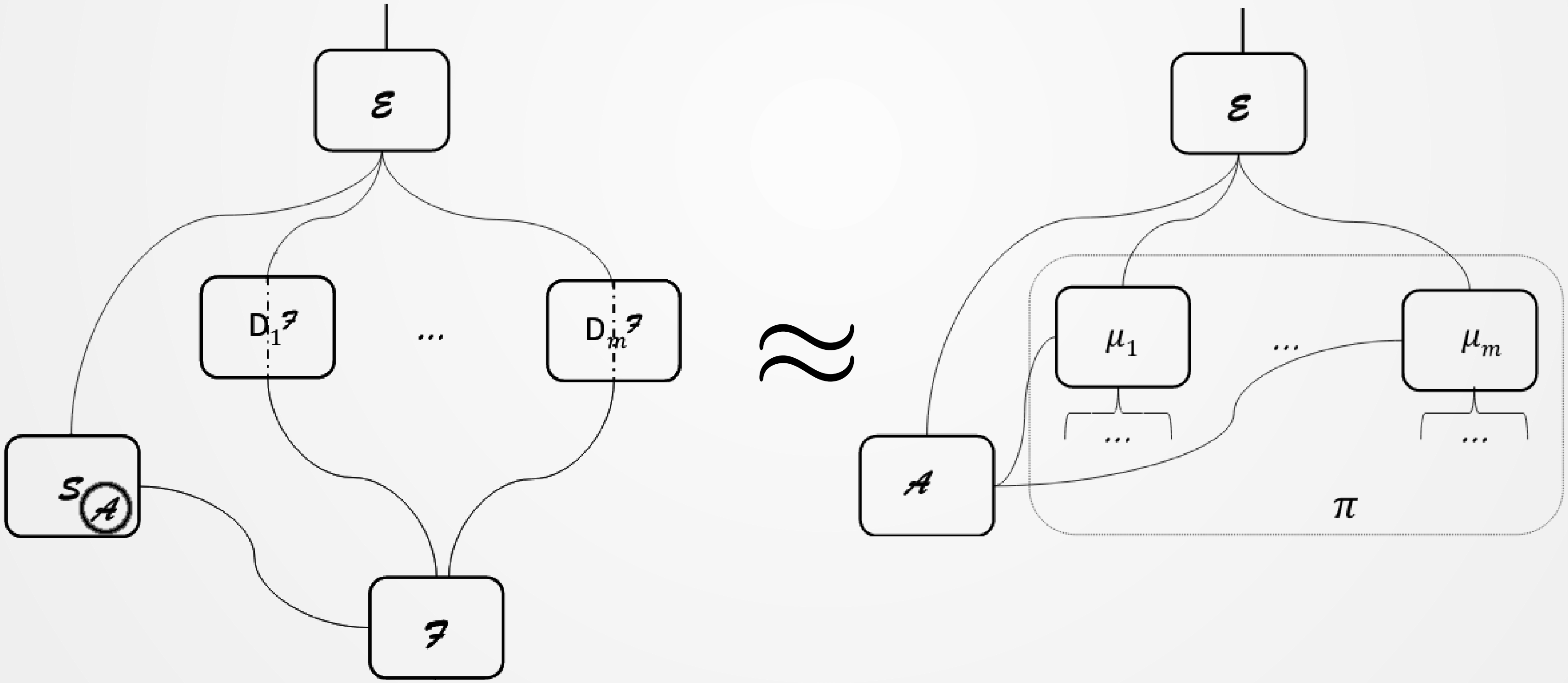
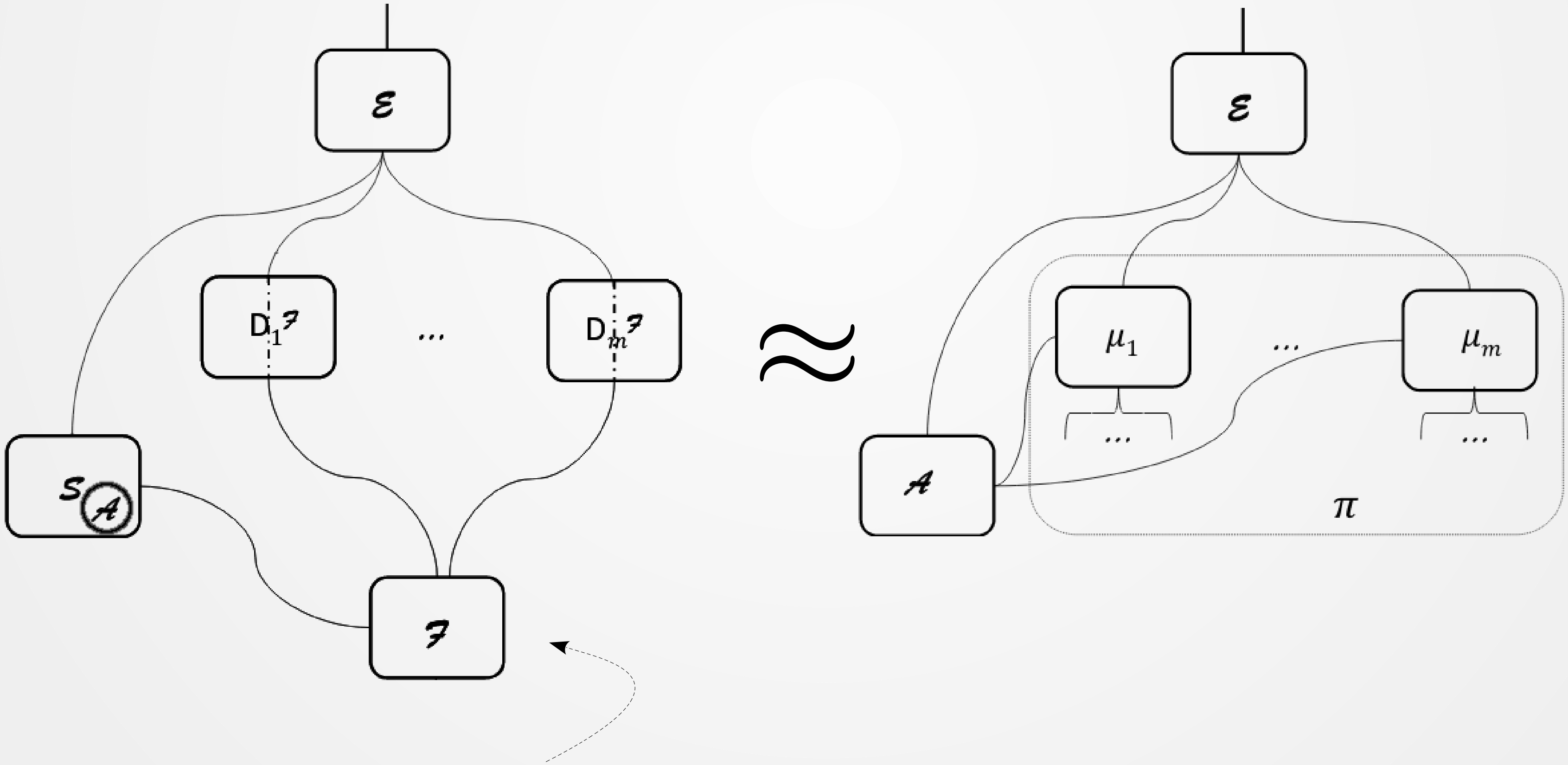


Figure: “Universally Composable Security”,
Ran Canetti https://eprint.iacr.org/2000/067

This work

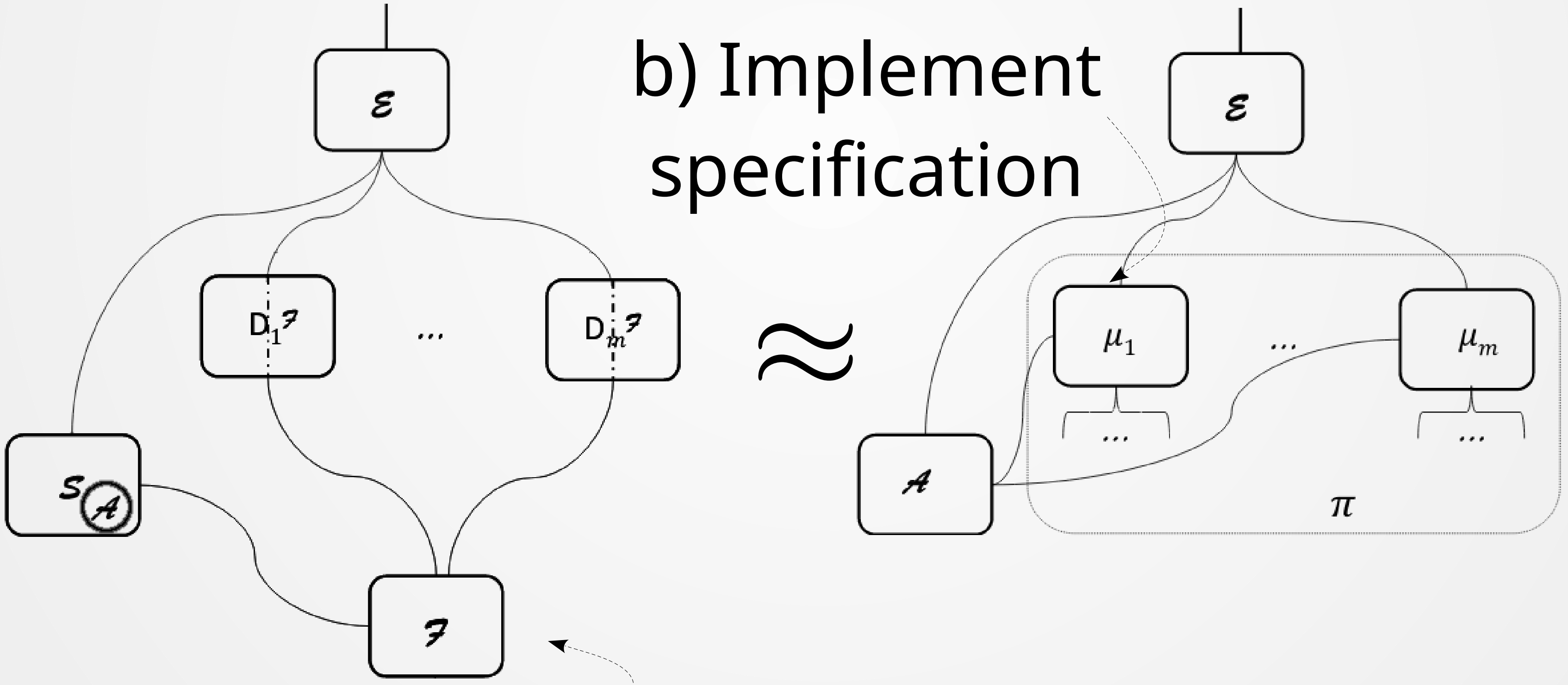
$$\forall \mathcal{A}\ \exists \mathcal{S} : \forall \mathcal{E}$$



a) Define functionality

This work

$$\forall \mathcal{A}\ \exists \mathcal{S} : \forall \mathcal{E}$$

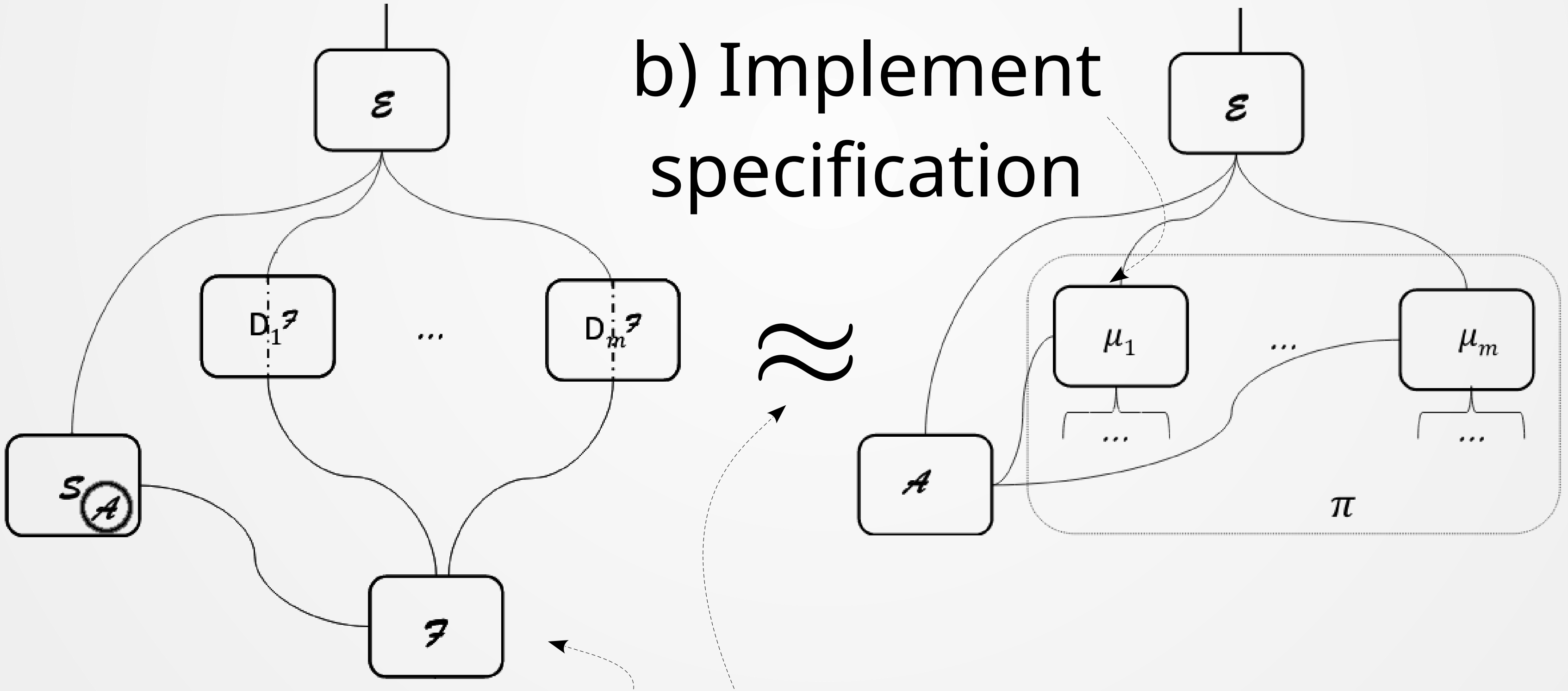


b) Implement specification

a) Define functionality

This work

$$\forall \mathcal{A}\ \exists \mathcal{S} : \forall \mathcal{E}$$



b) Implement specification

a) Define functionality

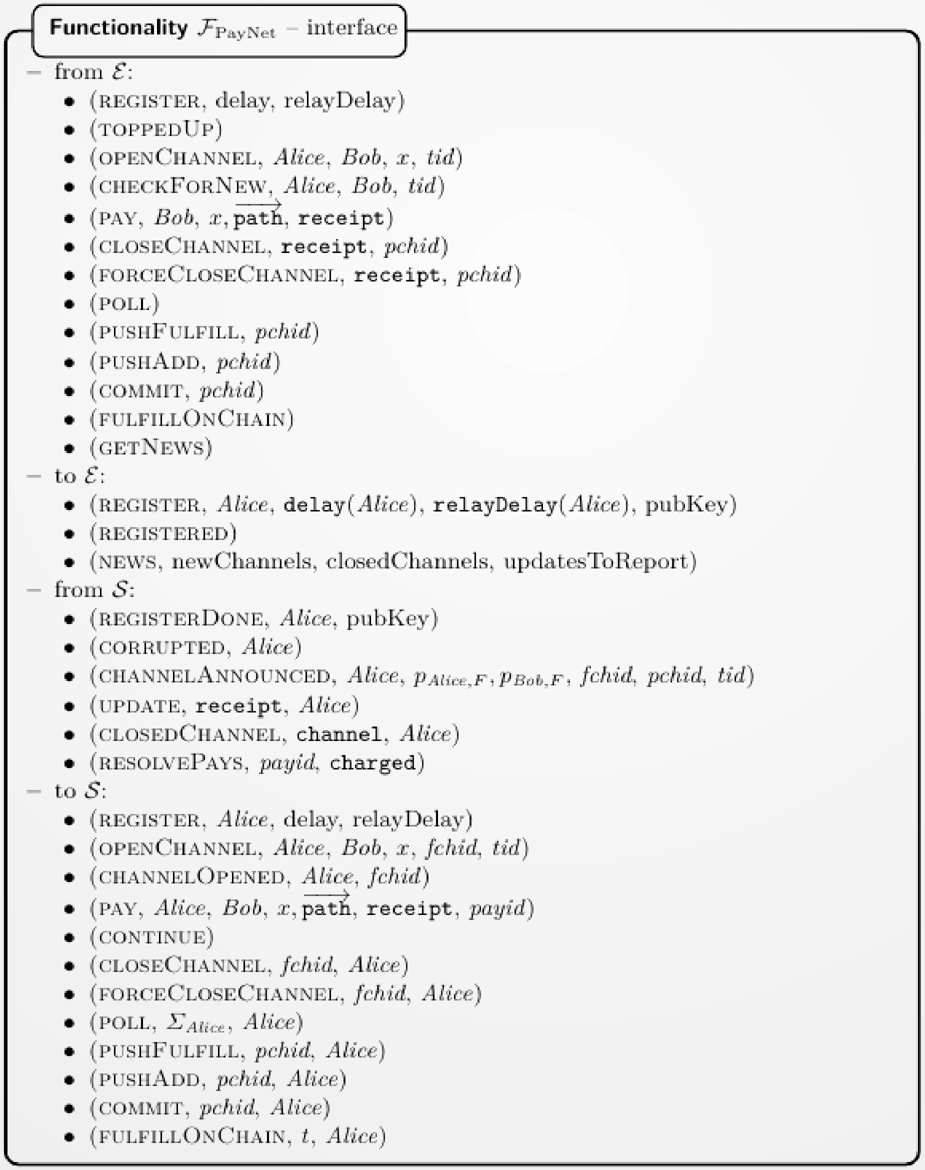
c) Prove indistinguishability

Blockchain Functionality

$\mathcal{G}_{\text{ledger}}$ [BMTZ'17, BGKRZ'18]

We prove that
a naive, instant-finality ledger
is unrealizable

Functionality



Functionality $\mathcal{F}_{\text{PayNet}}$ – interface

- from $\mathcal{E}$:
  - (REGISTER, delay, relayDelay)
  - (TOPPEDUP)
  - (OPENCHANNEL, *Alice*, *Bob*, x, *tid*)
  - (CHECKFORNEW, *Alice*, *Bob*, *tid*)
  - (PAY, *Bob*, x, $\overrightarrow{\texttt{path}}$, `receipt`)
  - (CLOSECHANNEL, `receipt`, *pchid*)
  - (FORCECLOSECHANNEL, `receipt`, *pchid*)
  - (POLL)
  - (PUSHFULFILL, *pchid*)
  - (PUSHADD, *pchid*)
  - (COMMIT, *pchid*)
  - (FULFILLONCHAIN)
  - (GETNEWS)
- to $\mathcal{E}$:
  - (REGISTER, *Alice*, `delay`(*Alice*), `relayDelay`(*Alice*), pubKey)
  - (REGISTERED)
  - (NEWS, newChannels, closedChannels, updatesToReport)
- from $\mathcal{S}$:
  - (REGISTERDONE, *Alice*, pubKey)
  - (CORRUPTED, *Alice*)
  - (CHANNELANNOUNCED, *Alice*, $p_{Alice,F}$, $p_{Bob,F}$, *fchid*, *pchid*, *tid*)
  - (UPDATE, `receipt`, *Alice*)
  - (CLOSEDCHANNEL, `channel`, *Alice*)
  - (RESOLVEPAYS, *payid*, `charged`)
- to $\mathcal{S}$:
  - (REGISTER, *Alice*, delay, relayDelay)
  - (OPENCHANNEL, *Alice*, *Bob*, x, *fchid*, *tid*)
  - (CHANNELOPENED, *Alice*, *fchid*)
  - (PAY, *Alice*, *Bob*, x, $\overrightarrow{\texttt{path}}$, `receipt`, *payid*)
  - (CONTINUE)
  - (CLOSECHANNEL, *fchid*, *Alice*)
  - (FORCECLOSECHANNEL, *fchid*, *Alice*)
  - (POLL, Σ_{Alice}, *Alice*)
  - (PUSHFULFILL, *pchid*, *Alice*)
  - (PUSHADD, *pchid*, *Alice*)
  - (COMMIT, *pchid*, *Alice*)
  - (FULFILLONCHAIN, t, *Alice*)

Our contributions

- Use a realistic ledger functionality
    - Prove naive ledger unrealizable
- Prove Lightning Network security in UC framework
- Derive exact time bounds for how often parties need to check the chain

Further work

- Virtual channels
  - Channels on top of channels
  - No on-chain txs for open/close
  - “Elmo: Recursive Virtual Payment Channels for Bitcoin”
    https://raw.githubusercontent.com/OrfeasLitos/virtual-payment-channels/master/virtual-channels.pdf

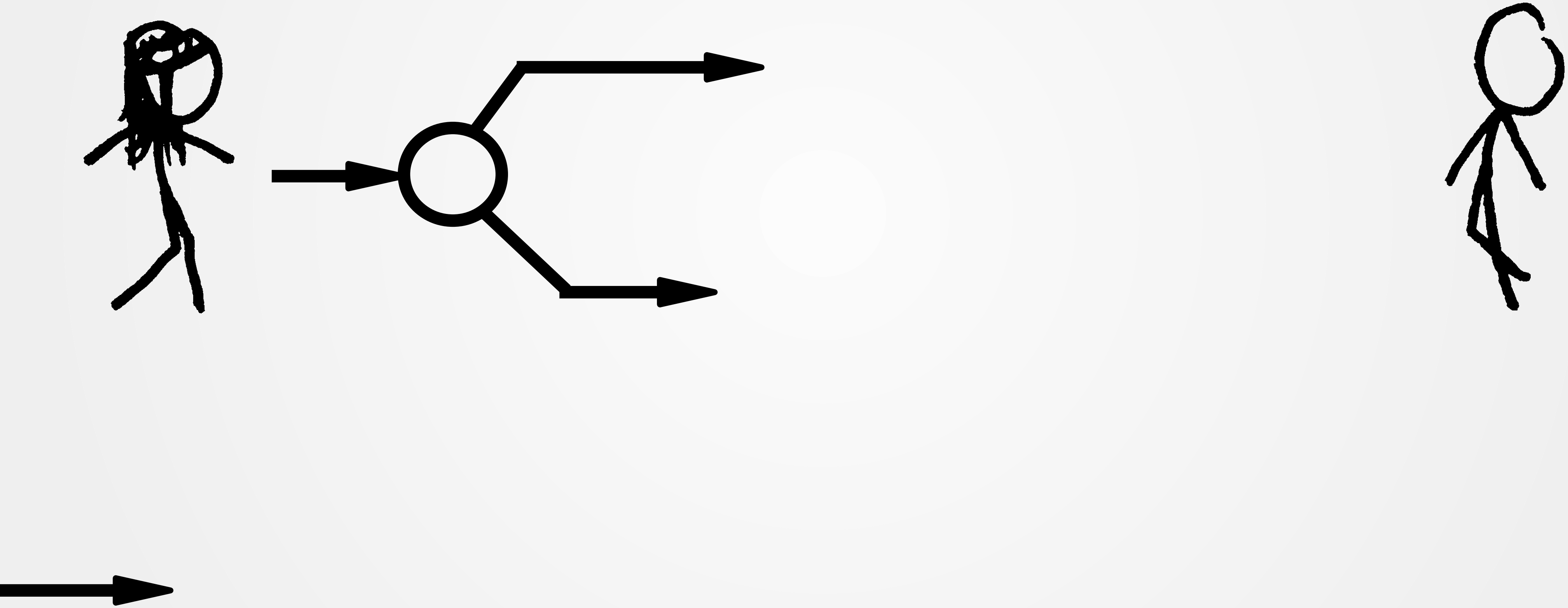
Further work

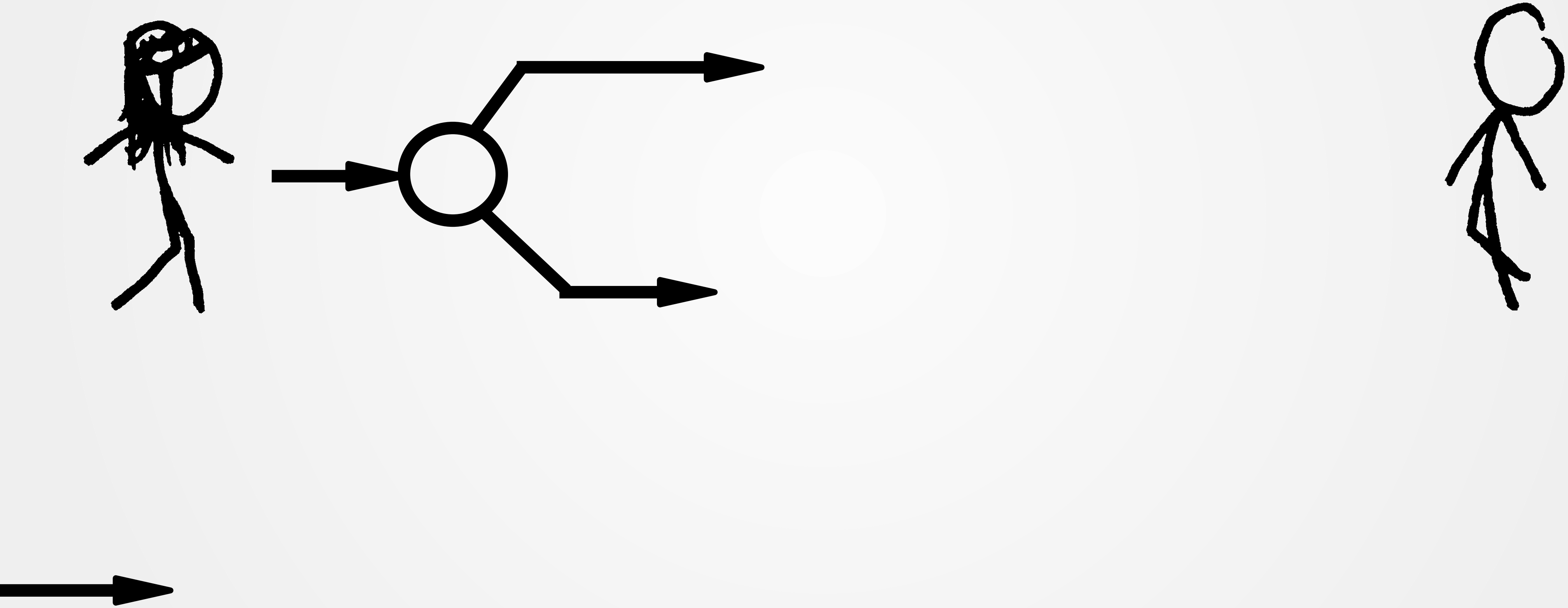
- Virtual channels
  - Channels on top of channels
  - No on-chain txs for open/close
  - “Elmo: Recursive Virtual Payment Channels for Bitcoin”
    https://raw.githubusercontent.com/OrfeasLitos/virtual-payment-channels/master/virtual-channels.pdf

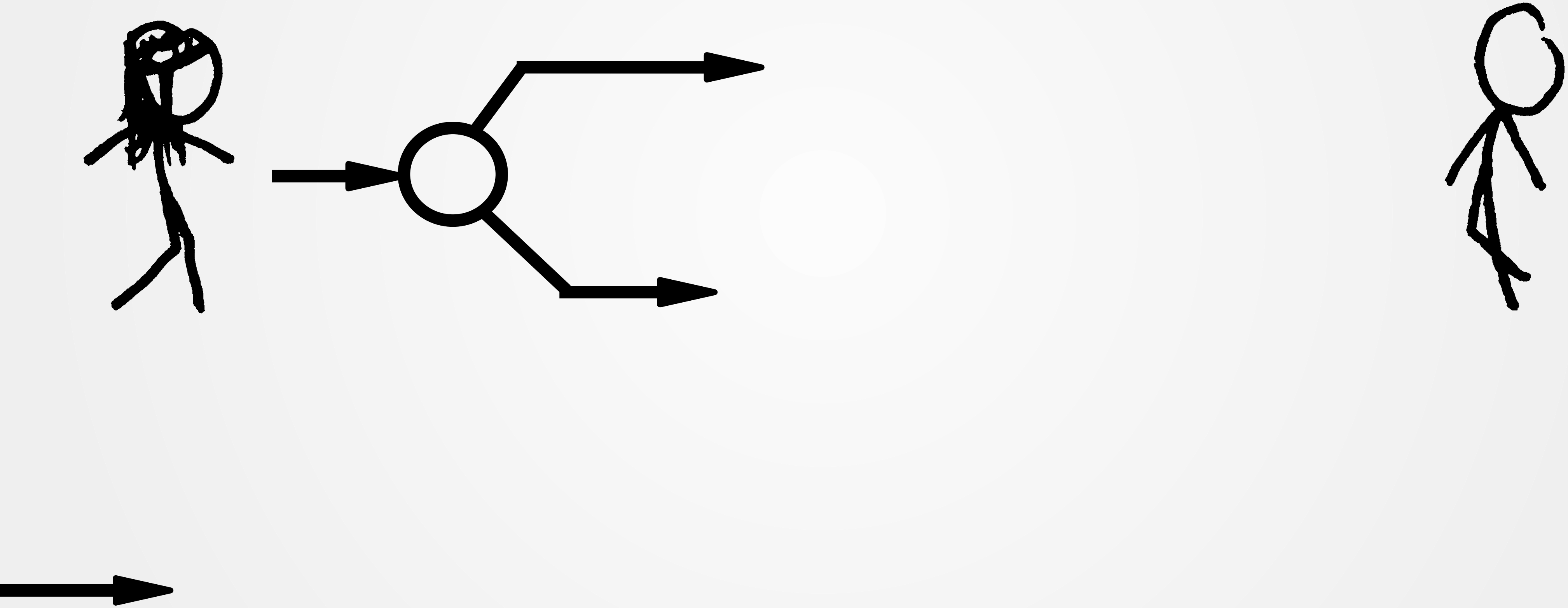
Thank you! Questions?

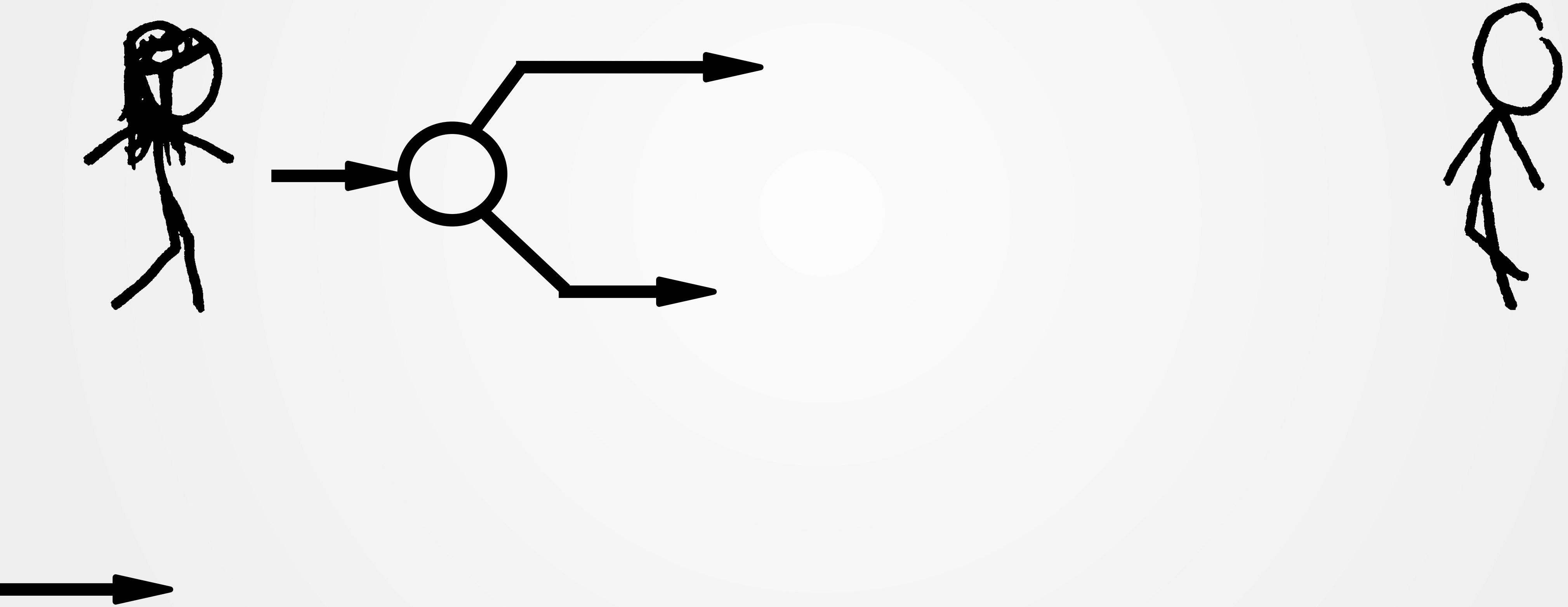
https://eprint.iacr.org/2019/778

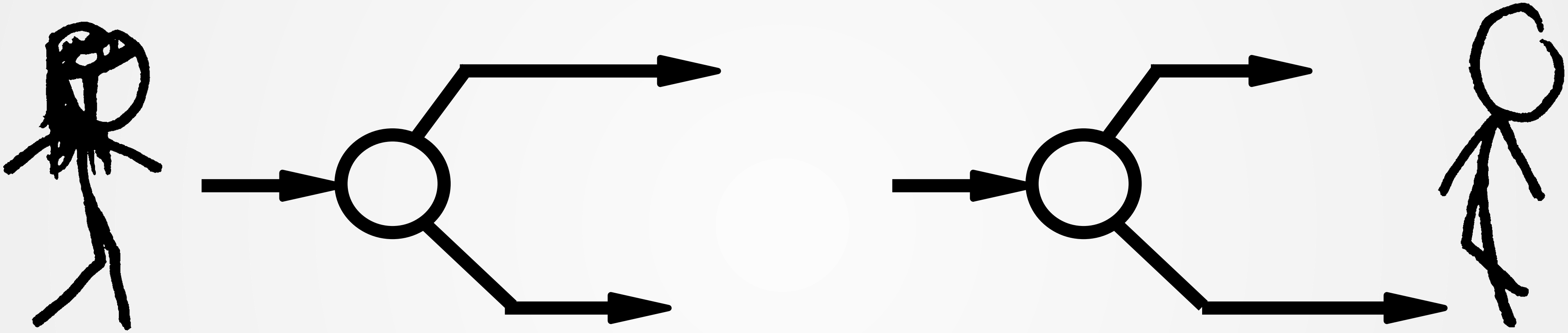
Bonus slides:
Protocol example

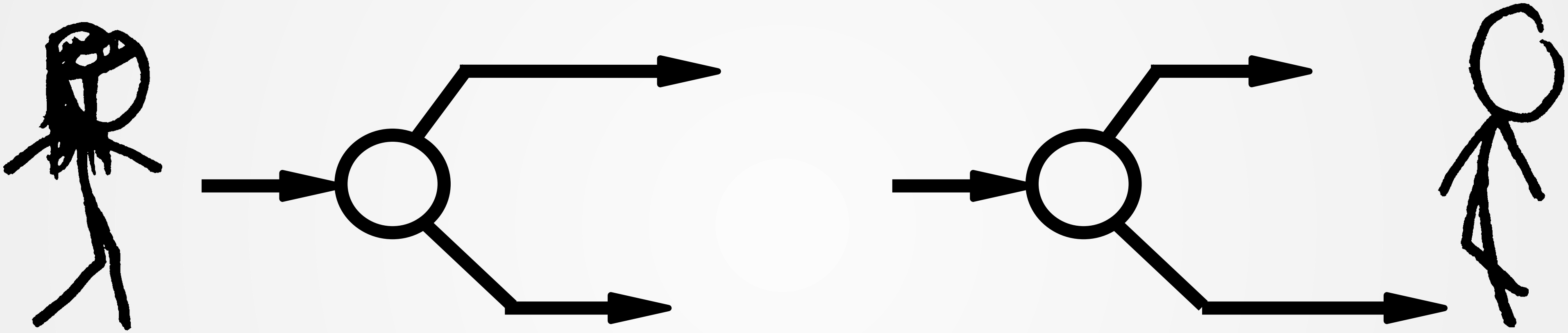


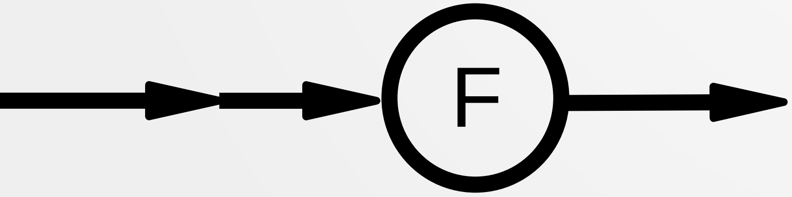


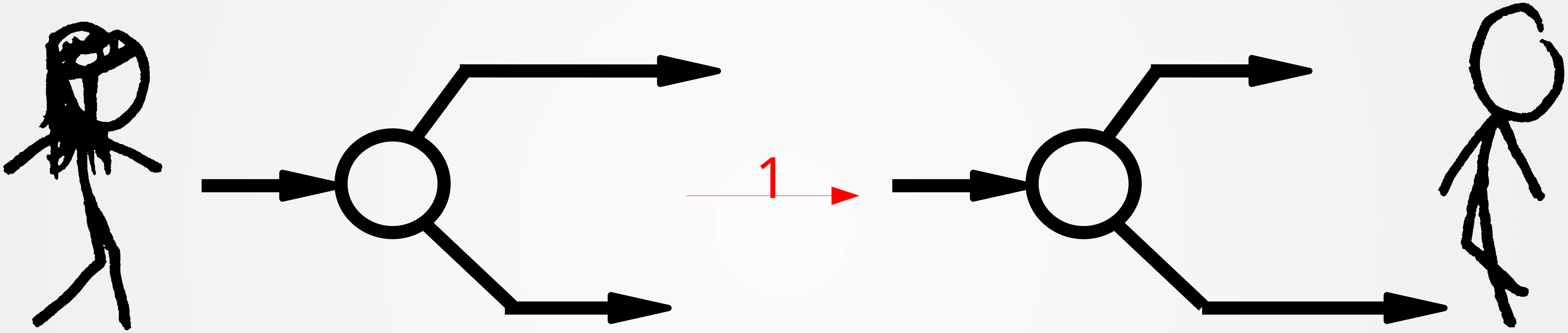


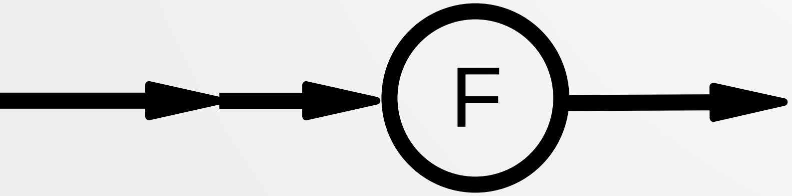


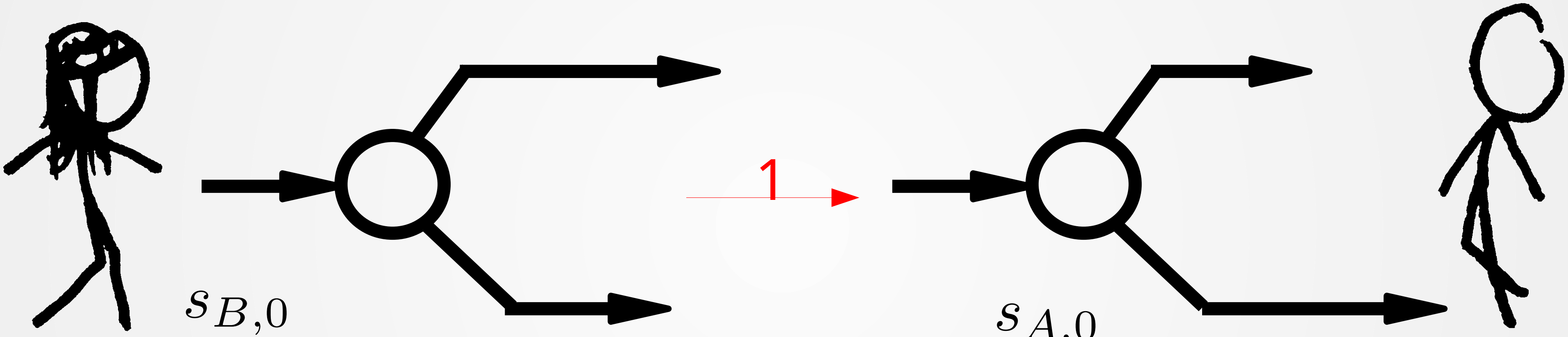


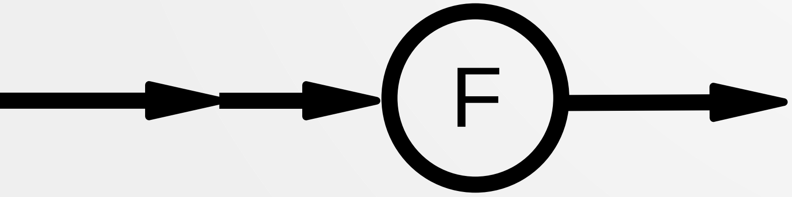


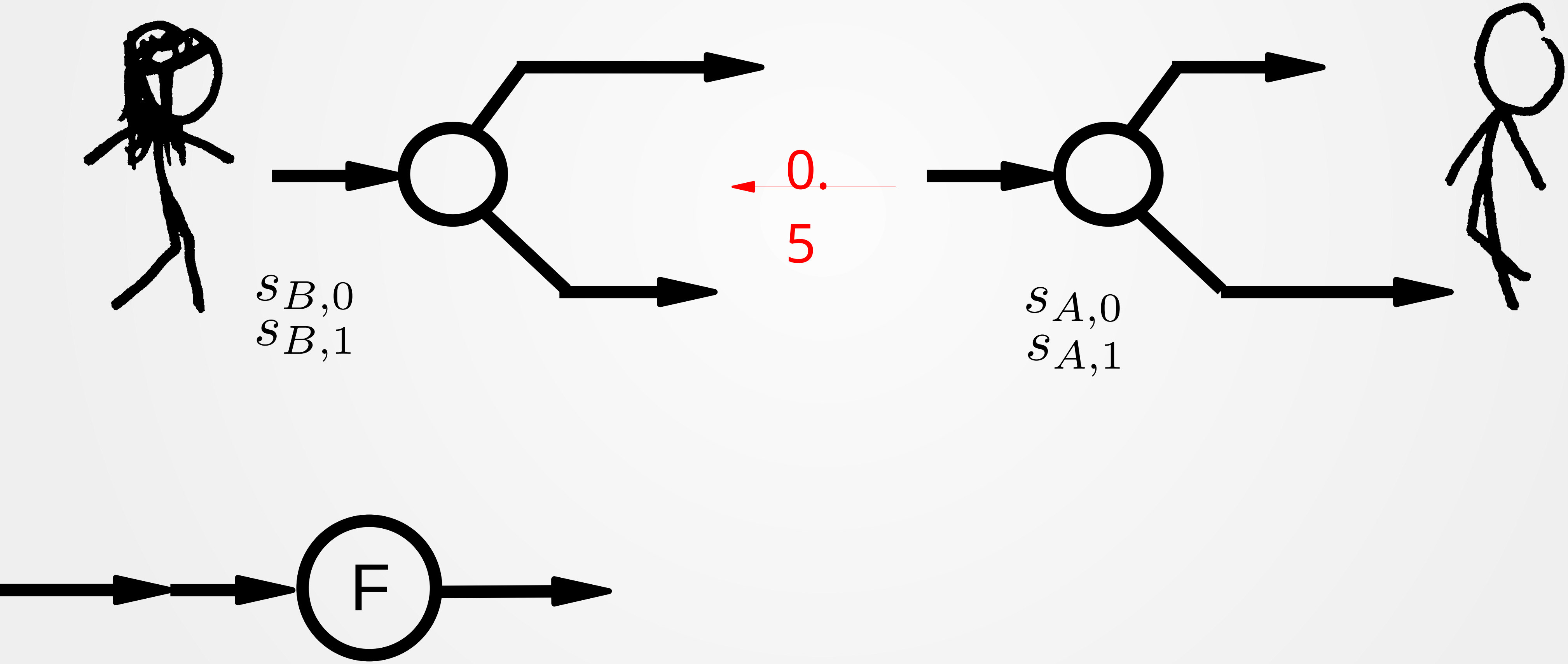

F

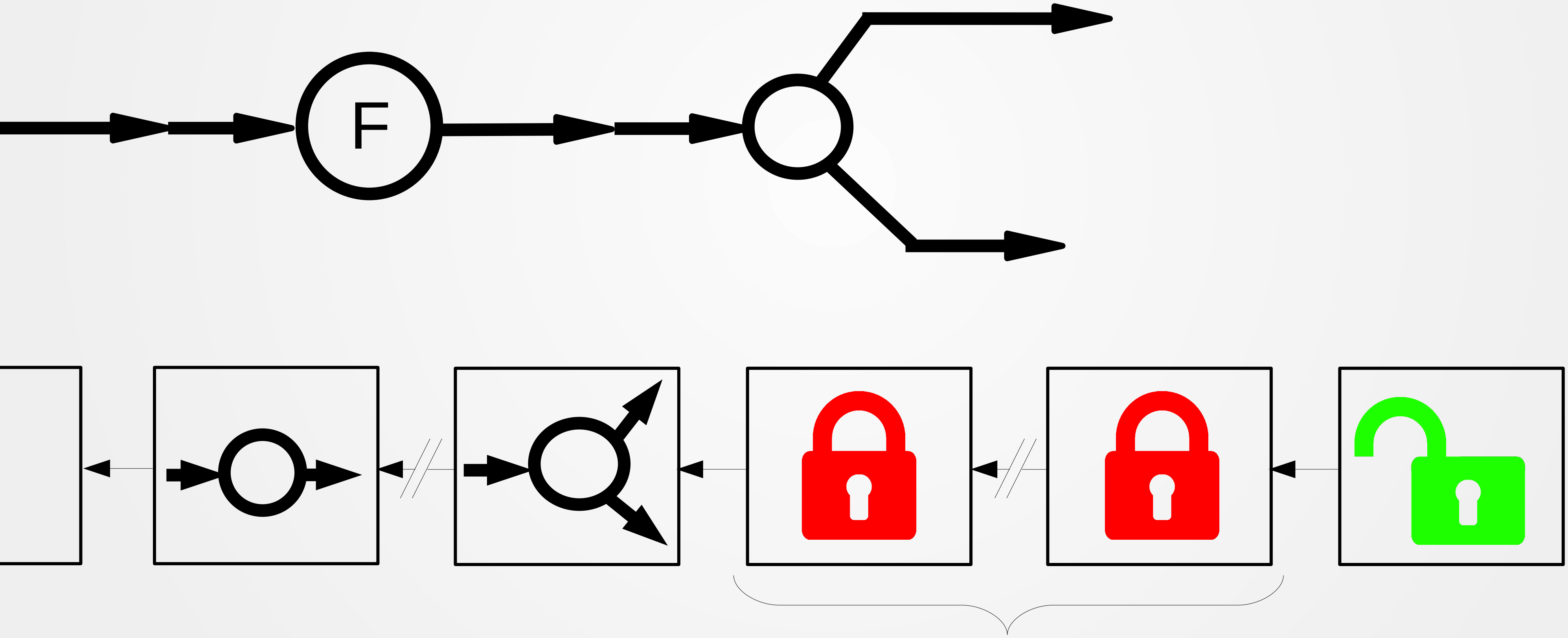

1


F

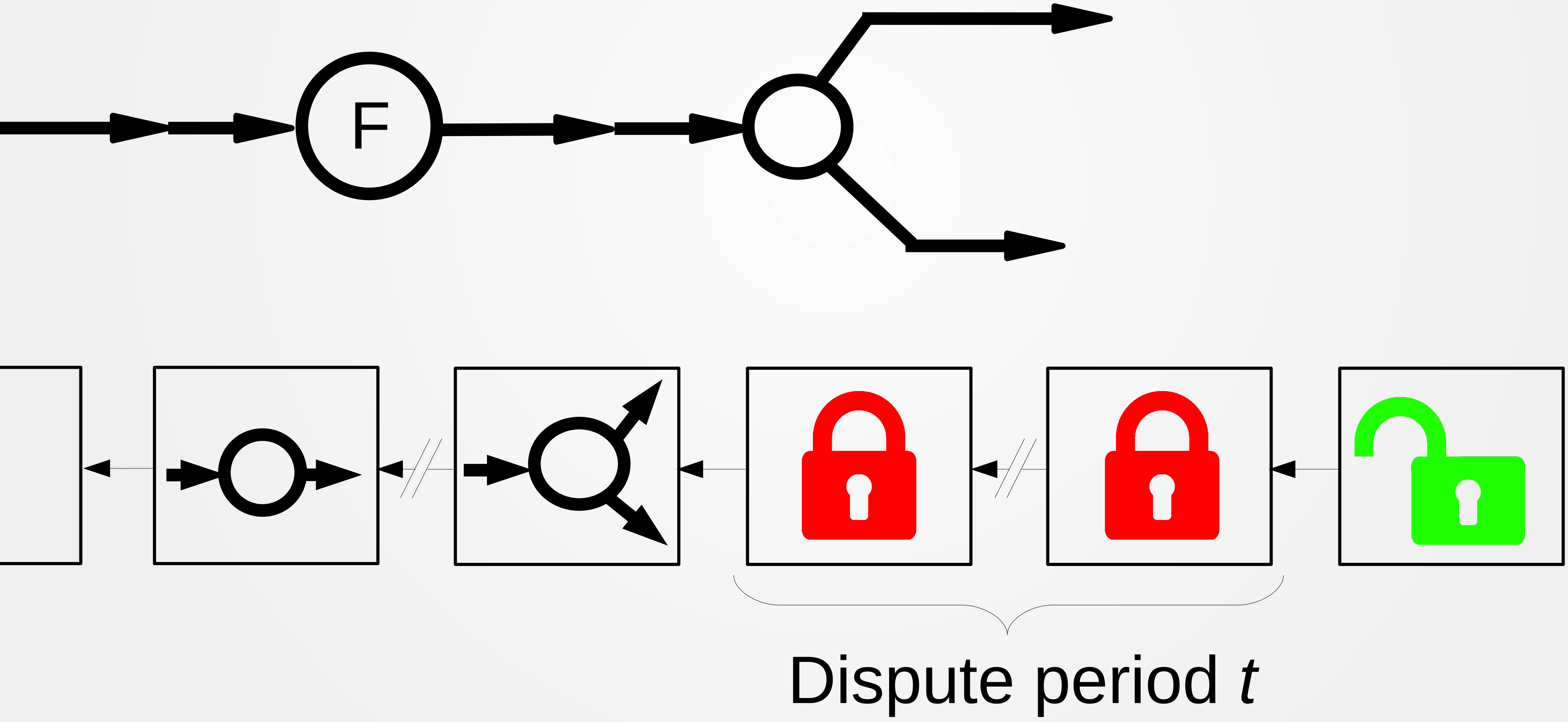

$s_{B,0}$
1
$s_{A,0}$


F

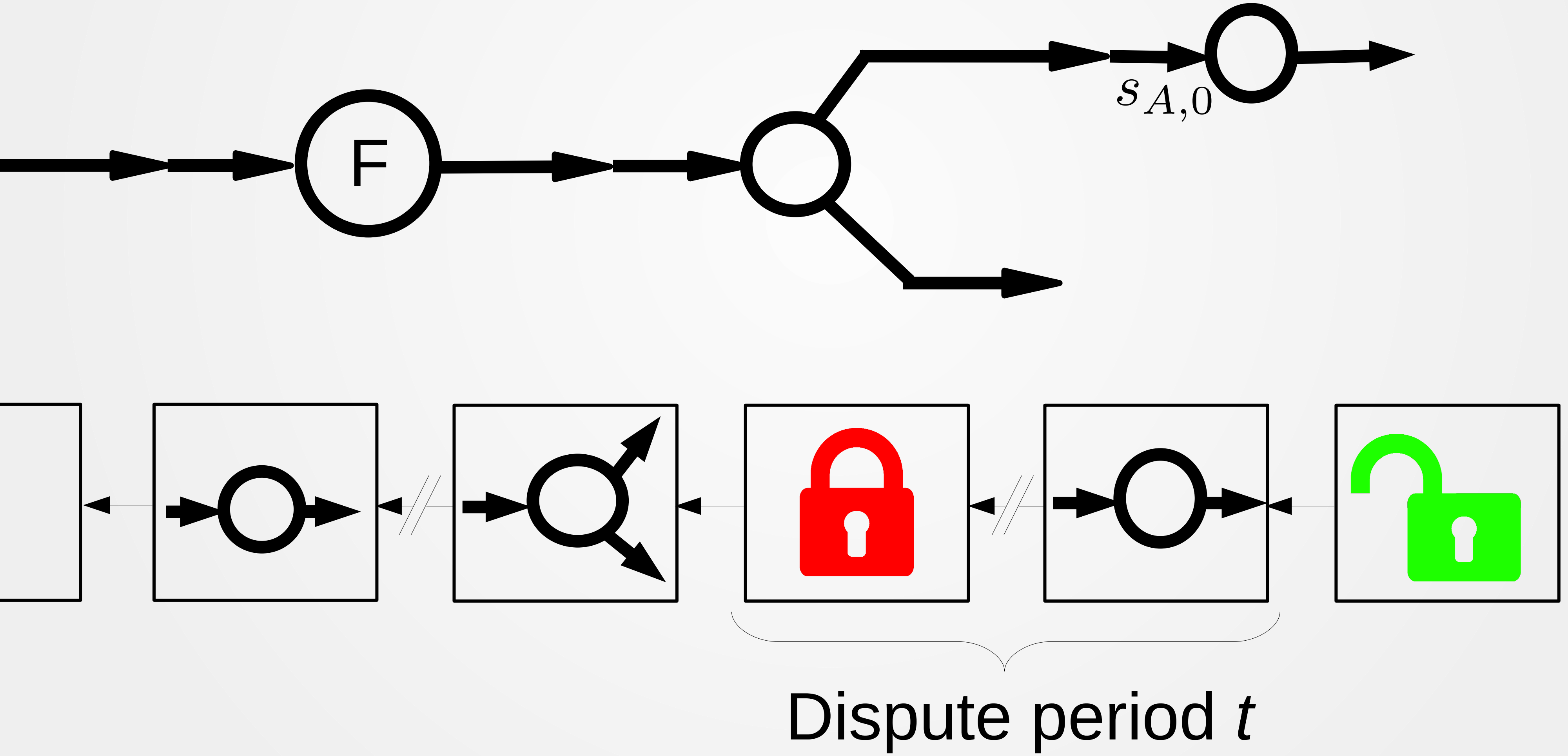

$s_{B,0}$
$s_{B,1}$
0.
5
$s_{A,0}$
$s_{A,1}$
F



Dispute period *t*



Dispute period *t*


F
$s_{A,0}$
Dispute period t