A Composable Security Treatment of the Lightning Network

Aggelos Kiayias

Orfeas Stefanos Thyfronitis Litos

University of Edinburgh

19/10/2019

Security

All about things NOT allowed

- Encryption DOESN’T leak plaintext
- Hash function DOESN’T leak preimage
- Signature CAN’T be made by random folks
- …
- **Lightning DOESN’T lose parties’ funds**

Simulation-based Security

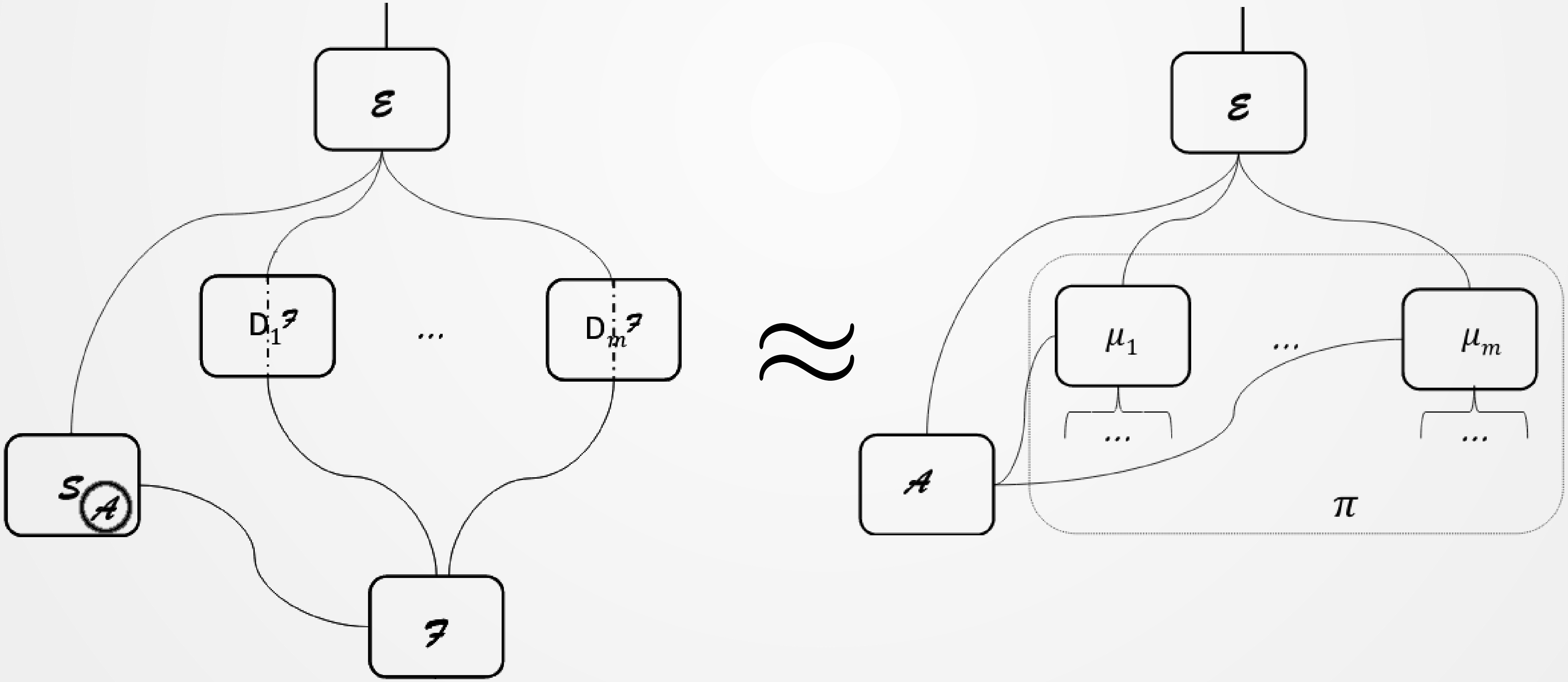
Approach

- Define ideal functionality
- Write protocol
- Prove protocol looks like functionality

Functionality captures everything

Simulation-based Security

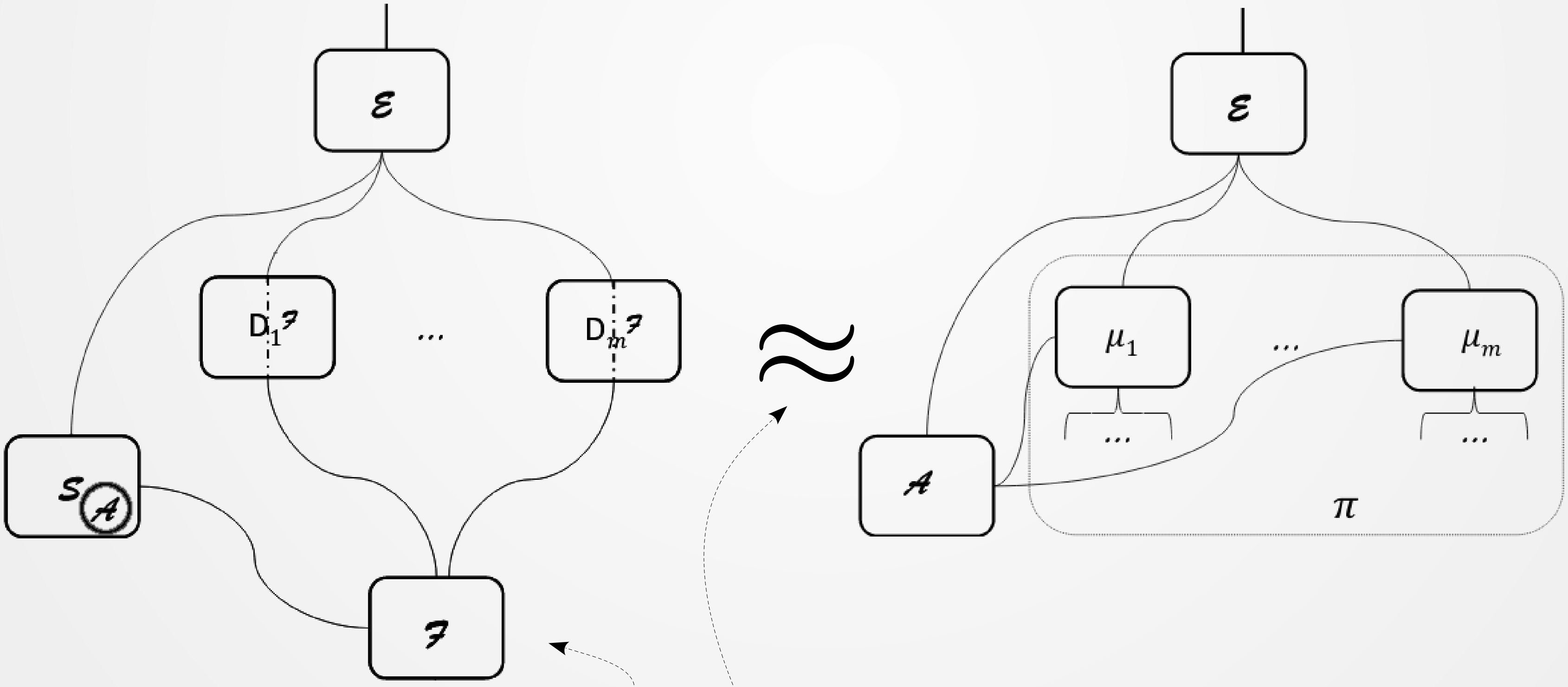
$$\forall \mathcal{E}, \mathcal{A}\ \exists \mathcal{S} :$$



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

Our paper

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



Define functionality

Prove indistinguishability

How often online?

- No in-flight payments
  - sync at least every `to_self_delay` blocks

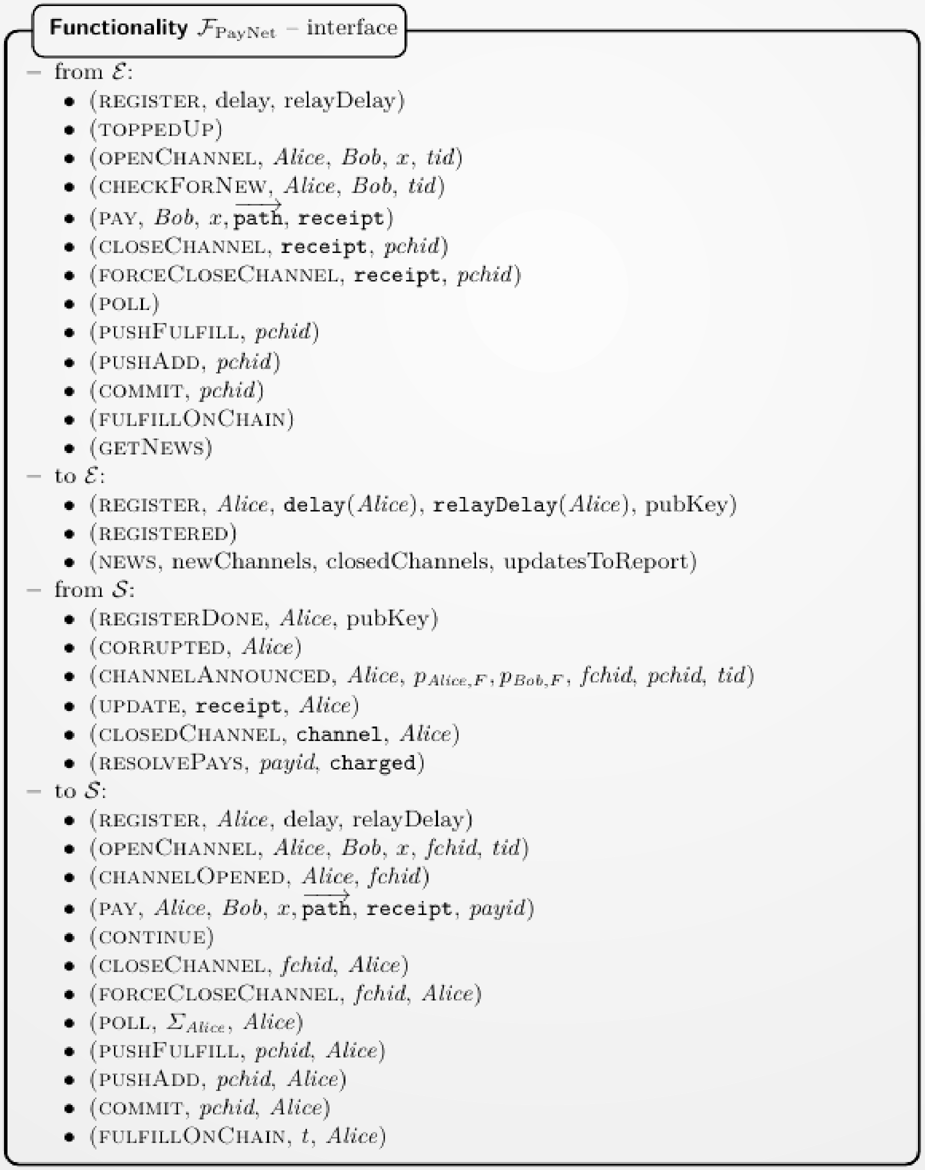
How often online?

- No in-flight payments
  - sync at least every `to_self_delay` blocks
- In-flight HTLC – intermediary
  - a = “max new blocks from tx bcast till settled”
  - Sync during [`out_cltv_exp`, `in_cltv_exp - 2a`]
  - try to publish HTLC-timeout
  - Sync again after `a`
  - If HTLC-success found, update or fulfill on-chain

How often online?

- No in-flight payments
  - sync at least every `to_self_delay` blocks
- In-flight HTLC – intermediary
  - a = “max new blocks from tx bcast till settled”
  - Sync during [`out_cltv_exp`, `in_cltv_exp - 2a`]
  - try to publish HTLC-timeout
  - Sync again after `a`
  - If HTLC-success found, update or fulfill on-chain
- In-flight HTLC – payee
  - Fulfill on-chain until `min_final_cltv_expiry - a`

Functionality



Functionality $\mathcal{F}_{\mathrm{PayNet}}$ – 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*)

Functionality

- Workhorse messages
  - (`open_channel`, Alice, Bob, x)
  - (pay, Bob, x, path, receipt)
  - (`{,force}_close_channel`, receipt, id)
- (`poll`) – sync and check for malicious closures
- (`resolve_pays`, charged) – HTLC resolutions
- `check_closed`(state)
- (`get_news`)

Thank you! Questions?

https://github.com/OrfeasLitos/PaymentChannels/