CARDANO

Cardano & Cardano Education

Crypto Hub Malta, September 2023

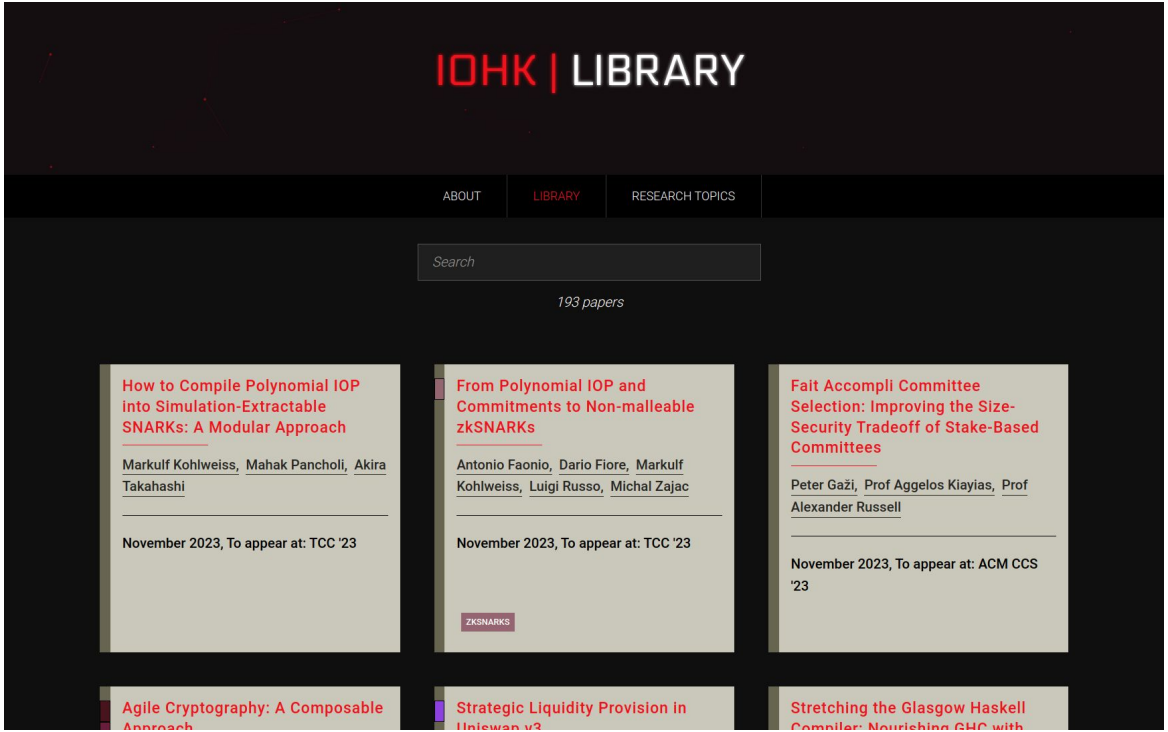
About myself

Dr. Lars Brünjes, Director of Education at IOG

- PhD in Pure Mathematics from Regensburg University.
- Postdoc at Cambridge University (UK).
- 40 years of programming experience.
- Joined IOG November 2016.

Cardano

Using the Scientific Method



From Mathematical Paper...

Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain

Bernardo David*, Peter Gaži**, Aggelos Kiayias***, and Alexander Russell†

October 6, 2017

Abstract. We present "Ouroboros Praos", a proof-of-st
the first time, provides security against *fully-adaptive cor
setting*: Specifically, the adversary can corrupt any partic
population of stakeholders at any moment as long the st
an honest majority of stake; furthermore, the protocol tol
message delivery delay unknown to protocol participants.
To achieve these guarantees we formalize and realize in th
suitable form of forward secure digital signatures and a new
that maintains unpredictability under malicious key genera
a general combinatorial framework for the analysis of se
may be of independent interest. We prove our protocol sec
assumptions in the random oracle model.

Protocol π_{SPoS}

The protocol π_{SPoS} is run by stakeholders $U_1, \ldots, U_n$ interacting among themselves and with ideal functionalities $\mathcal{F}_{\mathsf{INIT}}, \mathcal{F}_{\mathsf{VRF}}, \mathcal{F}_{\mathsf{KES}}, \mathcal{F}_{\mathsf{DSIG}}, \mathsf{H}$ over a sequence of slots $S = \{sl_1, \ldots, sl_R\}$. Define $T_i \triangleq 2^{\ell_{\mathsf{VRF}}}\phi_f(\alpha_i)$ as the threshold for a stakeholder U_i, where α_i is the relative stake of U_i, ℓ_{VRF} denotes the output length of $\mathcal{F}_{\mathsf{VRF}}$, f is the active slots coefficient and ϕ_f is the mapping from Definition 1. Then π_{SPoS} proceeds as follows:

1. **Initialization.** The stakeholder U_i sends $(\mathsf{KeyGen}, sid, U_i)$ to $\mathcal{F}_{\mathsf{VRF}}$, $\mathcal{F}_{\mathsf{KES}}$ and $\mathcal{F}_{\mathsf{DSIG}}$; receiving $(\mathsf{VerificationKey}, sid, v_i^{\mathrm{vrf}})$, $(\mathsf{VerificationKey}, sid, v_i^{\mathrm{kes}})$ and $(\mathsf{VerificationKey}, sid, v_i^{\mathrm{dsig}})$, respectively. Then, in case it is the first round, it sends $(\mathsf{ver\_keys}, sid, U_i, v_i^{\mathrm{vrf}}, v_i^{\mathrm{kes}}, v_i^{\mathrm{dsig}})$ to $\mathcal{F}_{\mathsf{INIT}}$ (to claim stake from the genesis block). In any case, it terminates the round by returning $(U_i, v_i^{\mathrm{vrf}}, v_i^{\mathrm{kes}}, v_i^{\mathrm{dsig}})$ to $\mathcal{Z}$. In the next round, U_i sends $(\mathsf{genblock\_req}, sid, U_i)$ to $\mathcal{F}_{\mathsf{INIT}}$, receiving $(\mathsf{genblock}, sid, \mathbb{S}_0, \eta)$ as the answer. U_i sets the local blockchain $\mathcal{C} = B_0 = (\mathbb{S}_0, \eta)$ and its initial internal state $st = H(B_0)$.
2. **Chain Extension.** After initialization, for every slot $sl_j \in S$, every online stakeholder U_i performs the following steps:
   (a) U_i receives from the environment the transaction data $d \in \{0,1\}^*$ to be inserted into the blockchain.
   (b) U_i collects all valid chains received via diffusion into a set $\mathbb{C}$, pruning blocks belonging to future slots and verifying that for every chain $\mathcal{C}' \in \mathbb{C}$ and every block $B' = (st', d', sl', B_\pi', \sigma_{\beta'}) \in \mathcal{C}'$ it holds that the stakeholder who created it is in the slot leader set of slot sl' (by parsing B_π' as (U_s, y', π') for some s, verifying that $\mathcal{F}_{\mathsf{VRF}}$ responds to $(\mathsf{Verify}, sid, \eta \| sl', y', \pi', v_s^{\mathrm{vrf}})$ by $(\mathsf{Verified}, sid, \eta \| sl', y', \pi', 1)$, and that $y' < T_s$), and that $\mathcal{F}_{\mathsf{KES}}$ responds to $(\mathsf{Verify}, sid, (st', d', sl', B_\pi'), sl', \sigma_{\beta'}, v_s^{\mathrm{kes}})$ by $(\mathsf{Verified}, sid, (st', d', sl', B_\pi'), sl', 1)$. U_i computes $\mathcal{C}' = \mathsf{maxvalid}(\mathcal{C}, \mathbb{C})$, sets $\mathcal{C}'$ as the new local chain and sets state $st = H(\mathrm{head}(\mathcal{C}'))$.
   (c) U_i sends $(\mathsf{EvalProve}, sid, \eta \| sl_j)$ to $\mathcal{F}_{\mathsf{VRF}}$, receiving $(\mathsf{Evaluated}, sid, y, \pi)$. U_i checks whether it is in the slot leader set of slot sl_j by checking that $y < T_i$. If yes, it generates a new block $B = (st, d, sl_j, B_\pi, \sigma)$ where st is its current state, $d \in \{0,1\}^*$ is the transaction data, $B_\pi = (U_i, y, \pi)$ and σ is a signature obtained by sending $(\mathsf{USign}, sid, U_i, (st, d, sl_j, B_\pi), sl_j)$ to $\mathcal{F}_{\mathsf{KES}}$ and receiving $(\mathsf{Signature}, sid, (st, d, sl_j, B_\pi), sl_j, \sigma)$. U_i computes $\mathcal{C}' = \mathcal{C}|B$, sets $\mathcal{C}'$ as the new local chain and sets state $st = H(\mathrm{head}(\mathcal{C}'))$. Finally, if U_i has generated a block in this step, it diffuses $\mathcal{C}'$.
3. **Signing Transactions.** Upon receiving $(\mathsf{sign\_tx}, sid', tx)$ from the environment, U_i sends $(\mathsf{Sign}, sid, U_i, tx)$ to $\mathcal{F}_{\mathsf{DSIG}}$, receiving $(\mathsf{Signature}, sid, tx, \sigma)$. Then, U_i sends $(\mathsf{signed\_tx}, sid', tx, \sigma)$ back to the environment.

Fig. 4: Protocol π_{SPoS}.

- Written in English.
- Written by Mathematicians.
- Very abstract.

...To Efficient Code

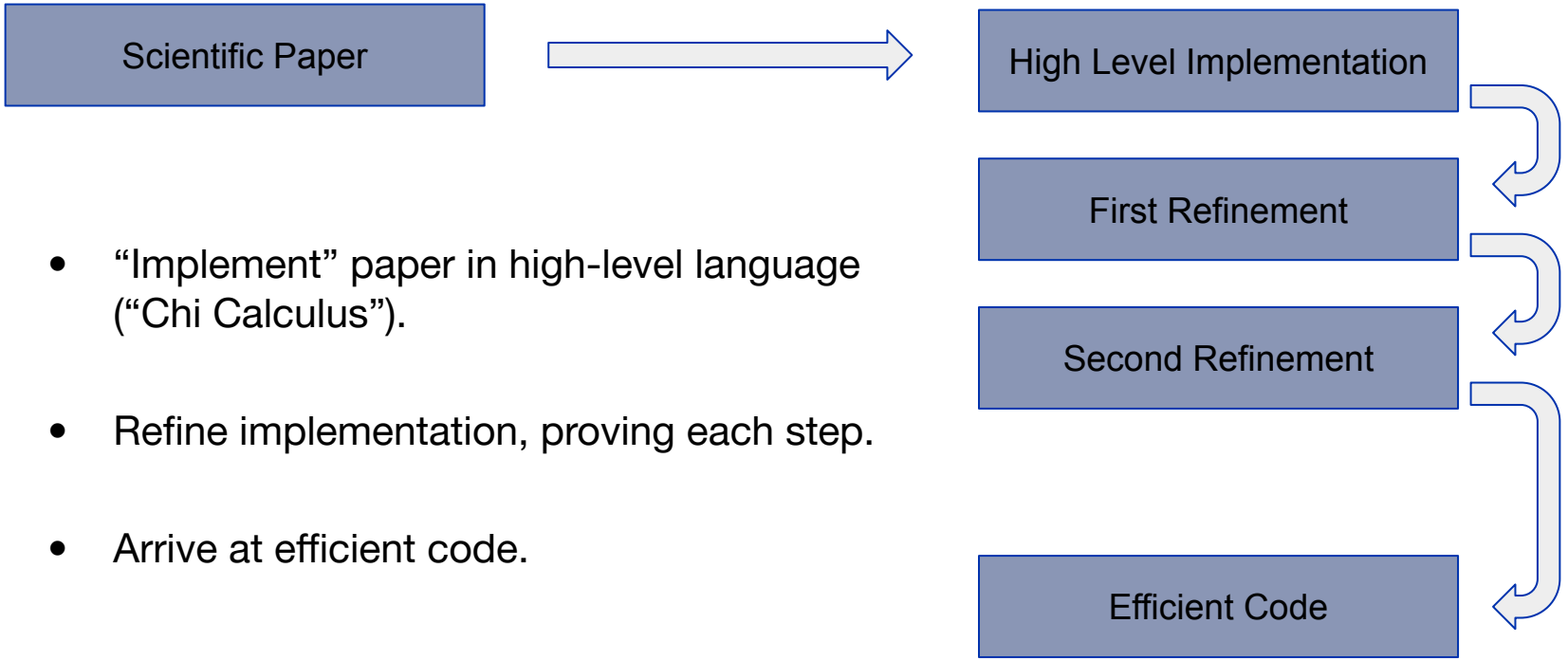
- Written in Haskell.
- Written by Software Engineers.
- Efficient code.

```haskell
-- CHECK: @verifyEncShare
-- | Verify encrypted shares
verifyEncShares
    :: MonadRandom m
    => SecretProof
    -> Scrape.Threshold
    -> [(VssPublicKey, EncShare)]
    -> m Bool
verifyEncShares SecretProof{..} threshold (sortWith fst -> pairs)
    | threshold <= 1     = error "verifyEncShares: threshold must be > 1"
    | threshold >= n - 1 = error "verifyEncShares: threshold must be < n-1"
    | otherwise =
          Scrape.verifyEncryptedShares
              spExtraGen
              threshold
              spCommitments
              spParallelProofs
              (coerce $ map snd pairs)  -- shares
              (coerce $ map fst pairs)  -- participants
  where
    n = fromIntegral (length pairs)
```

The Problem

- We start from a *mathematical* paper written by *mathematicians*.
- The paper will undergo rigid *peer review* and contain mathematical *proofs of correctness*.
- The outcome should be *correct* and *efficient* Haskell code.
- Our mathematicians don’t know Haskell, our engineers don’t know cryptography.
- How can we guarantee we deploy code that faithfully implements the original paper?

The Solution: Formal Methods



- “Implement” paper in high-level language (“Chi Calculus”).
- Refine implementation, proving each step.
- Arrive at efficient code.

Incentives

Desired Behavior

- A solid majority of stake should be delegated to 1000 stake pools of equal size.

- The stake pool operators should be online when needed.

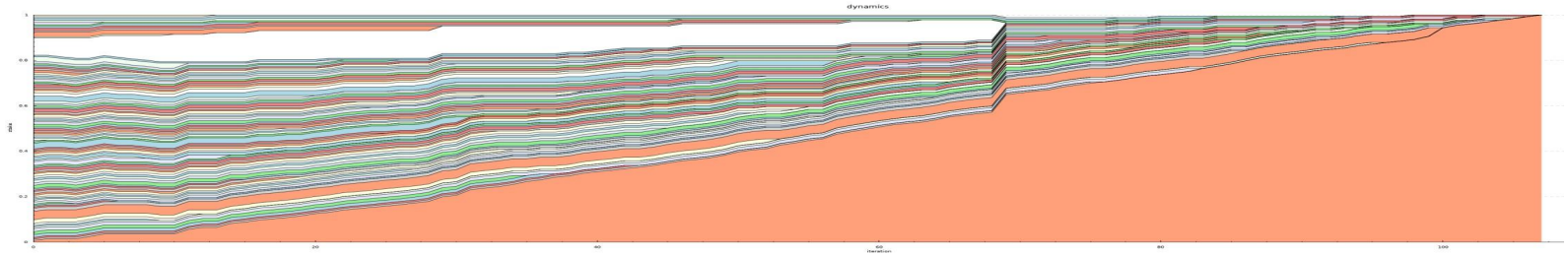
Basic Idea

The rewards pool from one epoch (five days) is distributed amongst stake pools and individual protocol participants according to their stake.

Problem with the Basic Idea

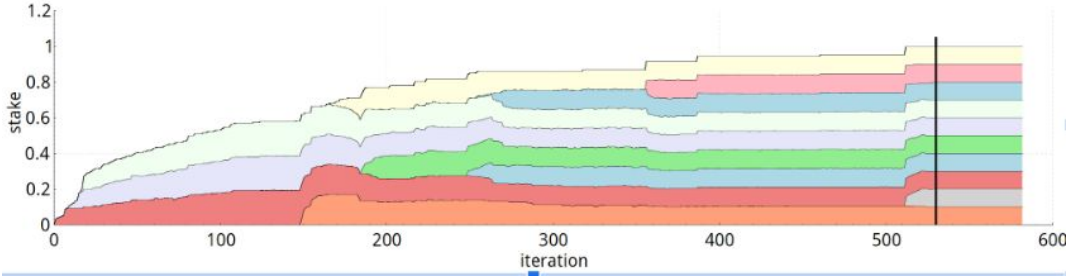
The basic idea is a good guideline, but too naive: The fewer pools there are, the lower total costs will be, the higher everybody's rewards will be.

So the system will tend towards a single dictatorial pool that everybody else delegates to.



Refinements

- Pool rewards are capped at 1‰ of available rewards.
- Pools are penalized for not being online when it is their turn.
- Pool leaders are compensated for their cost and effort by keeping a margin.
- Pools increase rewards by pledging.
- Pool members are rewarded proportional to the stake they delegated to the pool.



Cardano Education

Our Team

Lars Brünjes
Director of Education

Niamh Ahern
Education Manager

Alejandro Garcia
Project Manager

Karina Lopez
Plutus Specialist

Tuvshintsenguun Erdenejargal
Plutus Specialist

Antonio Ibarra
Plutus Specialist

Robertino Martinez
Education Assistant

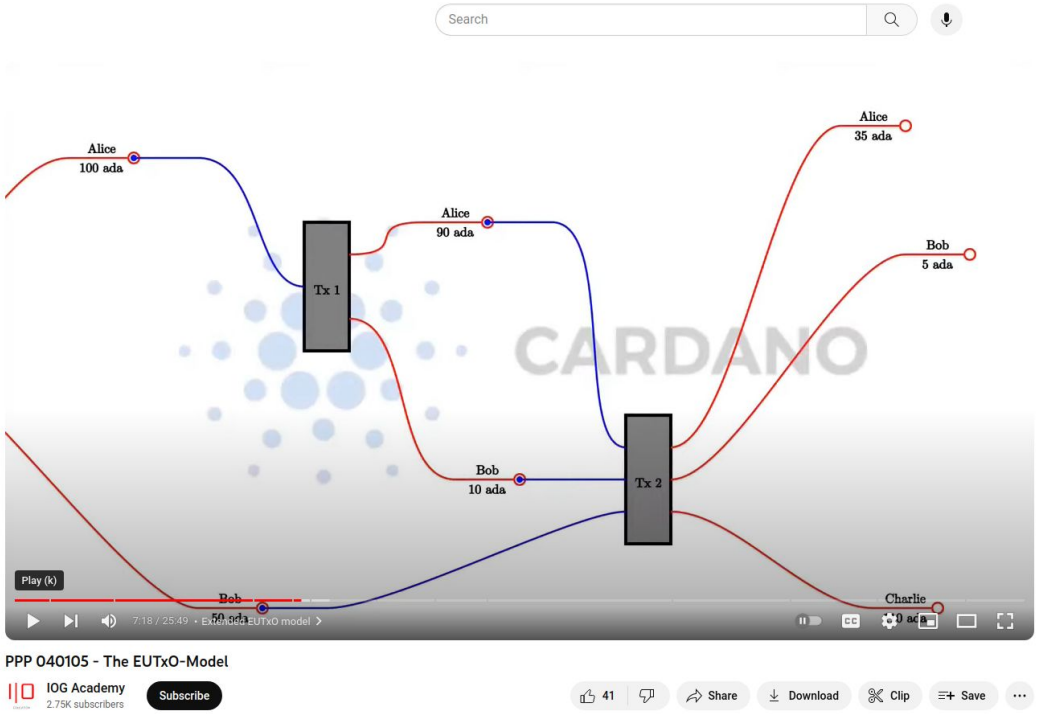
Thomas Vellekoop
Education Assistant

Alexey Sorokin
Education Assistant

Conferences & Meetups

- Meetups in Hamburg, London, Rotterdam, Zug, Tel Aviv, Malta, Ulaanbaatar, Buenos Aires…
- Conferences in Hamburg, London, Basel…
- Various interviews & podcasts.

Plutus Pioneer Program



- Freely available on YouTube & GitHub
- Reaching thousands of students.
- Recently completed iteration #4.

2017 - Athens, Greece

- Seven students.
- Hired most of them.

2018 - Barbados

- Ten students, most of them IOHK employees.
- Hired all external students.

2019 - Addis Ababa, Ethiopia

- 22 students, 18 from Ethiopia, four from Uganda.
- All women.
- Made an offer to all of them.

2023 - Nairobi, Kenya

- Hybrid course - two weeks life in Nairobi, rest online.
- Haskell, Marlowe & Plutus.
- Collaboration with the African Blockchain Center.

CARDANO

Thank you