Cardano

The Secret of Success of one of the Leading Cryptocurrencies

DISTRIBUTE Hamburg
2018-06-28

Well-Typed
The Haskell Consultants

Predictable Network Solutions

About myself

Dr. Lars Brünjes, Director of Education at IOHK

- PhD in Pure Mathematics from Regensburg University.
- Postdoc at Cambridge University (UK).
- Ten years working in Software Development prior to joining IOHK.
- Haskell enthusiast for more than 15 years.
- Joined IOHK November 2016.
- Taught Haskell courses (Athens, Barbados, Addis Ababa,...), internal and external trainings,...
- Working with Formal Methods team.
- Leading Incentives team.

Agenda

- IOHK
- Cardano
- Formal Methods
- Incentives

IOHK

Providing financial services to the three billion people that don't have them.

- Founded 2015 by Charles Hoskinson and Jeremy Wood.
- Company building Cardano.
- Distributed around the globe.
- Invested in functional programming (Haskell, Scala,...).
- **Research focused (peer-reviewed research, research centers, ...)**

Cardano

- Proof of Stake blockchain
- Cryptocurrency Ada
- Roadmap: https://cardanoroadmap.com/
- Smart Contracts: Plutus, IELE VM
- Sidechains
- Treasury

PoW vs PoS

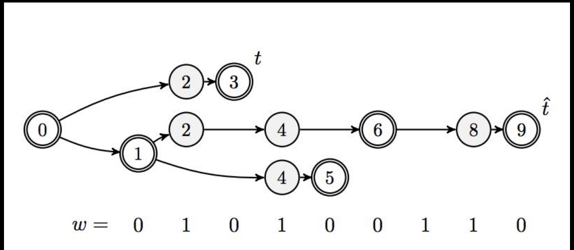
- Leader selection based on Hashing Power: “One CPU, one vote!”
- Huge energy consumption to guarantee security.
- Well established and provably secure.

- Leader selection based on Stake: “Follow the Satoshi!”
- Consensus is relatively cheap.
- Provably secure, but hotly debated.

Ouroboros

First Provably Secure PoS Protocol

- Elect leader for each time-slot based on stake.
- Stakeholders agree on randomness for next epoch.
- Running in production in Cardano since October 2017.
- Provably secure against adversary with less than 50% stake.



Adversary	BTC	OB Covert	OB General
0.10	50	3	5
0.15	80	5	8
0.20	110	7	12
0.25	150	11	18
0.30	240	18	31
0.35	410	34	60
0.40	890	78	148
0.45	3400	317	663

Ouroboros Praos

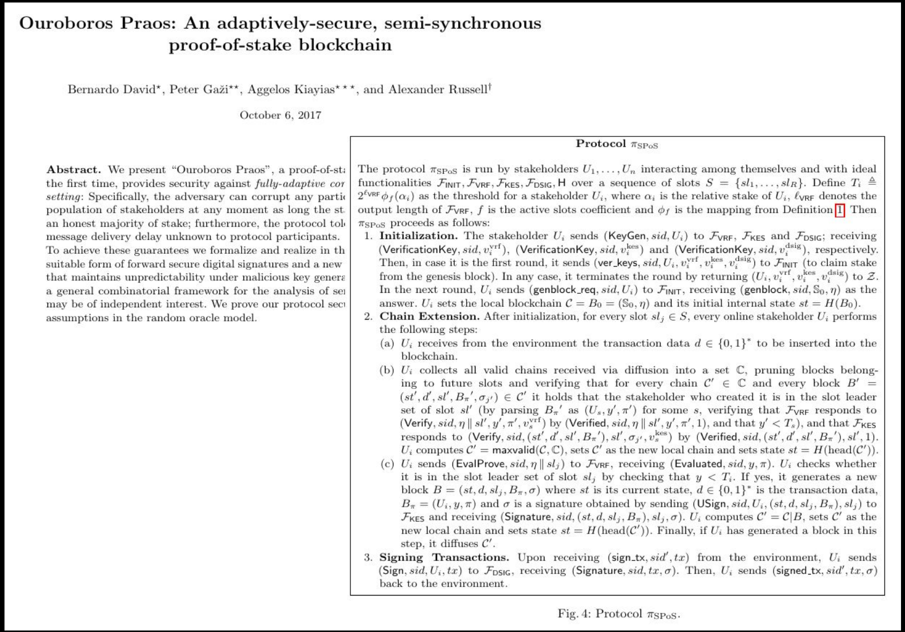
- Extension of Ouroboros to semi-synchronous setting.
- Deals gracefully with message delays.
- Currently being implemented for future versions of Cardano.

Ouroboros Genesis

- No checkpointing: New Players can safely join the protocol without any trusted advice.
- Security Proof in the UC-framework, making it easier to compare with Bitcoin (and other PoW systems).

Formal Methods

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 $B' = (st', d', sl', B_\pi{}', \sigma_{j'}) \in \mathcal{C}'$ it holds that the stakeholder who created it is in the slot leader set of slot sl' (by parsing $B_\pi{}'$ 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 $(\mathsf{Verify}, sid, (st', d', sl', B_\pi{}'), sl', \sigma_{j'}, v_s^{\mathrm{kes}})$ 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?

Why does it matter?

- We are very proud of the quality of our research branch. We want to ensure this quality translates into equal quality of our software.

- Literally billions of dollars are managed by our code. A single mistake can be extremely costly.

- We are interested in developing best practices that can be applied to a wide range of domains, pushing the envelope of what is possible and practicable.

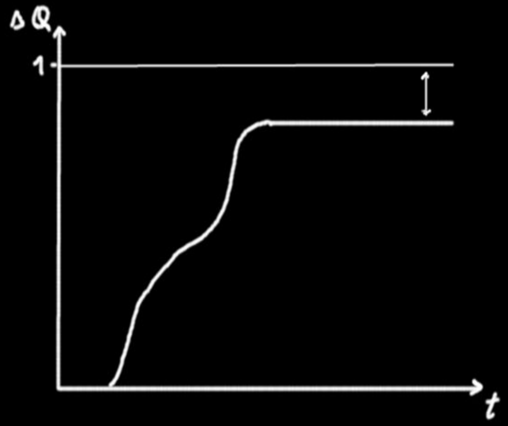
The Solution: Formal Methods

Scientific Paper → Highlevel Implementation → First Refinement → Second Refinement → Efficient Code

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

Chi Calculus

- Our version of the P(s)i Calculus (like Lambda Calculus, but for concurrent systems).
- Can be embedded in Haskell and then...
    - ...be executed.
    - ...be exported to a proof assistant (Isabelle).
    - ...be analyzed for performance (Delta Q).



Incentives

The people doing all the hard work...

- Prof. Aggelos Kiayias, University of Edinburgh (UK), Chief Scientist at IOHK

- Prof. Elias Koutsoupias, University of Oxford (UK), Senior Research Fellow at IOHK

- Aikaterini-Panagiota Stouka, University of Edinburgh (UK), Researcher at IOHK

What are Incentives?

- Incentives in the context of a cryptocurrency are ways of encouraging people to participate in the protocol and to follow it faithfully.
- In the case of Bitcoin, this means mining blocks and including as many valid transactions in those blocks as possible.
- In the case of Cardano, it means being online and creating a block when they have been elected slot leader and to participate in the election process.

(Non-)Monetary Incentives

- When the Bitcoin mining pool Ghash.io accumulated 42% of total mining power, people voluntarily started leaving the pool and brought it down to 38% in only two days.
(CoinDesk, 2014-01-09)

- The people who left Ghash.io did not receive any Bitcoin for leaving. Rather, they believed that concentrating too much mining power was bad and that leaving was ***the right thing*** to do.

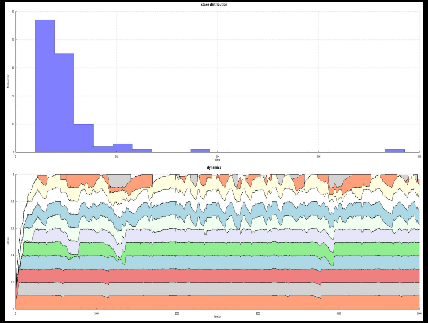
- **Ideally, monetary and moral incentives should align perfectly.**

Incentives in Cardano

- The above example shows that in Bitcoin, this ideal is not always achieved. Sometimes people have to choose between doing the morally right thing and pursuing their financial gain.

- In Cardano, we strive for perfect alignment of incentives.

- We use Game Theory and Simulations to develop and test our model.



Thank you!

- Please subscribe to the IOHK YouTube Channel.
- Visit our homepage at iohk.io.
- Follow us on Twitter: InputOutputHK.
- **IOHK is hiring: iohk.io/careers.**

- Questions?
- Comments?