Algorand: (Another) Better Bitcoin?

Based on:
Algorand: Scaling Byzantine Agreements for Cryptocurrencies, by Yossi Gilad et. al.

Presented by:
Guozhen Li
ECS 265 Distributed Database Systems, Fall 2018, UC Davis
Nov 27, 2018
What’s bad about Bitcoin

- **Wastes** electricity
- **Not really distributed**: computing power, thus decision power, (eventually) controlled by a few (~5) big mining companies
- **Vulnerable**: the big miners are known to the world & they have low profit margins → easy to corrupt
- **Scalability** is questionable
- **Ambiguity**: forks can form
- **Slow**: transaction takes ~1hr to confirm
## Algorand vs. Bitcoin

<table>
<thead>
<tr>
<th></th>
<th>Bitcoin</th>
<th>Algorand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who decides what value to agree on</td>
<td>One node that solves a complex puzzle fastest</td>
<td>Majority vote from a randomly selected committee</td>
</tr>
<tr>
<td>Main assumption</td>
<td>Majority of computing power is honest</td>
<td>Majority of funds are held by honest users</td>
</tr>
<tr>
<td>Computation workload on a node</td>
<td>Heavy: find a needle in a haystack</td>
<td>Light: add, count, compare, sign, verify</td>
</tr>
<tr>
<td>True decentralization?</td>
<td>Not really. Faster nodes have more power.</td>
<td>Yes (kinda). Everyone has a chance to vote.</td>
</tr>
</tbody>
</table>
Adding a Block in Algrand (when all goes well)

1. A random group of users (e.g. 26 users) each proposes a block based on payments it has observed from gossips, then broadcast its proposal to all users via gossiping.

2. A random committee (e.g. 1000 users) each collects proposals from legit proposers, and broadcast that it votes to the one proposal it heard often enough.

3. A different random committee (e.g. 1000 users) each counts legit votes from the previous committee. For each of them, if one proposal is found to win majority (e.g. over $\frac{2}{3}$ of previous committee) votes, that committee member accepts that proposal, and gossip “I accept block X”.

4. For all users, when they hear enough legit committee members say “I accept block X”, they also accepts block X. Thus the network reaches consensus.
VRF: The Guarantee for Randomness and Legitimacy

VRF = verifiable random function

- Everyone runs a “lottery” on its own
- The lottery generates a “winning ticket” and a “proof”, if one wins a role (e.g. proposer, committee)
- Everyone signs the winning ticket with its private key, and gossips out the signed winning ticket with the proof

- Everyone can verify everyone else’s “signed winning ticket + proof” pair to determine legitimacy
- Everyone only takes into account votes from verifiable messages
VRF: The Guarantee for Randomness and Legitimacy

Msg
“I won committee membership lottery for round 74 step 2”

Private key
“2C17C6393771EE3048AE34D6B380C5EC”

Public key
“4C9184F37CFF01BCDC32DC486EC36961”

Hash h
“8C0D968DBE2C064C3478A08A3AF149EAE”

Proof Π
“028DCE7598C280BA3697045A8316CE2”

Verify

Legit?
Yes, this guy is truly a committee member/
No, this guy is lying.
Algorand in More Details

(Sections 5-7)

CRYPTOGRAPHIC SORTITION - committee election/lottery

BLOCK PROPOSAL

BA★
Algorand in More Details: BA★

Two phases in BA★:

1. Reduction()
   “Everyone choose one of {proposal#56346, proposal#12059, empty_block} to pass to BinaryBA★()”

2. BinaryBA★()
   “Everyone choose one of {proposal_from_reduction, empty_block} as your final choice”

After these two phases, everyone counts other users’ final choices from gossips.
If your proposal_from_reduction receives enough votes, you accept it as a final block.
If your proposal_from_reduction does not receive enough votes, you mark it as a tentative block.
Algorand in More Details: BA★::Reduction()

Reduction(ctx, round, hblock):

CommitteeVote(ctx, round, REDUCTION_ONE, τstep, hblock)

\[\text{hblock1} \leftarrow \text{CountVotes}(\text{ctx}, \text{round}, \text{REDUCTION\_ONE}, \tau_{\text{step}}, \lambda_{\text{block}} + \lambda_{\text{step}})\]

\[\text{empty\_hash} \leftarrow H(\text{Empty}(\text{round}, H(\text{ctx.last\_block})))\]

\text{if} \ hblock1 = \text{TIMEOUT} \ \text{then}

\text{CommitteeVote(ctx, round, REDUCTION\_TWO, \tau_{\text{step}}, \text{empty\_hash})}

\text{else}

\text{CommitteeVote(ctx, round, REDUCTION\_TWO, \tau_{\text{step}}, \text{hblock1})}

\text{hblock2} \leftarrow \text{CountVotes}(\text{ctx}, \text{round}, \text{REDUCTION\_TWO}, \tau_{\text{step}}, \lambda_{\text{step}})

\text{if} \ hblock2 = \text{TIMEOUT} \ \text{then return empty\_hash;}

\text{else return hblock2;}

I vote for proposal#12059 in poll REDUCTION\_ONE for round 74.

Which proposal is the most popular in poll REDUCTION\_ONE?

Prepare hash of an empty block, just in case things go wrong.

If (from what I heard) no proposal wins majority votes from committee

I vote for empty_block in poll REDUCTION\_TWO of round 74.

If (from what I heard) some proposal wins majority votes

I vote for that proposal in poll REDUCTION\_TWO of round 74.

Which proposal is the most popular in poll REDUCTION\_TWO?

If no proposal is popular enough, I pass empty_block to my BinaryBA★()

If some proposal is popular enough, I pass that to my BinaryBA★()
Algorand in More Details: BA★::BinaryBA★()

Keep doing 3 things:

- CommitteeVote(ctx, round, step, τ_{STEP}, r)
  \[r \leftarrow \text{CountVotes}(ctx, round, step, T_{STEP}, τ_{STEP}, λ_{STEP})\]
  \[\text{if } r = \text{TIMEOUT then} \quad r \leftarrow \text{block_hash}\]
  \[\text{else if } r \neq \text{empty_hash then} \quad \text{for } step < s' \leq step + 3 \text{ do} \quad \text{CommitteeVote}(ctx, round, s', τ_{STEP}, r)\]
  \[\text{if } step = 1 \text{ then} \quad \text{CommitteeVote}(ctx, round, \text{FINAL}, τ_{FINAL}, r)\]
  \[\text{return } r\]

- CommitteeVote(ctx, round, step, τ_{STEP}, r)
  \[r \leftarrow \text{CountVotes}(ctx, round, step, T_{STEP}, τ_{STEP}, λ_{STEP})\]
  \[\text{if } r = \text{TIMEOUT then} \quad r \leftarrow \text{empty_hash}\]
  \[\text{else if } r = \text{empty_hash then} \quad \text{for } step < s' \leq step + 3 \text{ do} \quad \text{CommitteeVote}(ctx, round, s', τ_{STEP}, r)\]
  \[\text{return } r\]

- CommitteeVote(ctx, round, step, τ_{STEP}, r)
  \[r \leftarrow \text{CountVotes}(ctx, round, step, T_{STEP}, τ_{STEP}, λ_{STEP})\]
  \[\text{if } r = \text{TIMEOUT then} \quad r \leftarrow \text{block_hash}\]
  \[\text{else if } \text{CommonCoin}(ctx, round, step, τ_{STEP}) = 0 \text{ then} \quad r \leftarrow \text{block_hash}\]
  \[\text{else} \quad r \leftarrow \text{empty_hash}\]
  \[\text{step}++\]
Gist of Algorand

- Resolve disagreements with many polls
- For each poll, a different random committee show up and “shout out” their choice
- Everyone keeps listening the “shout outs” in the gossips, and decide what to choose in next poll
- VRFs (along with verifier functions) provide:
  - Randomness of whose “shout outs” are counted.
    (If most people are honest, I make good decisions most of the time.)
  - Legitimacy of the messages in gossips.
    (I can verify whether what I hear is truly that person saying a true thing)
Some Critiques of Algorand

- Not tested in any real-world environment
- No source code or binary released to public yet
- No incentives for users to turn on their machines and participate in the consensus protocol
- In its early years, it is easy for an adversary to buy over $\frac{2}{3}$ of all funds in the network
References

- (Video) “CESC2017 - Silvio Micali - ALGORAND”, uploaded by Blockchain at Berkeley: [https://youtu.be/NbnZi9SlmYY](https://youtu.be/NbnZi9SlmYY)
- (Video) “What is Algorand?”, uploaded by Jackson Palmer: [https://youtu.be/pLCmL7681oU](https://youtu.be/pLCmL7681oU)