Truebit

---- A scalable **verification solution** for blockchains

Yijie Hong
11/16/2018

What problem to solve?

- Ethereum is designed for **world computer** you can't shut down. But
  - Huge amount of computation power, but not as powerful as a smartphone
  - Classic PoW requires over 95% CPU time for hashing ⇒ Gas limit
  - All miners need to verify each computation

- Verifier's Dilemma
  - There is no incentives for verifier, so miners will skip the validation
  - Fork!

Target of Truebit

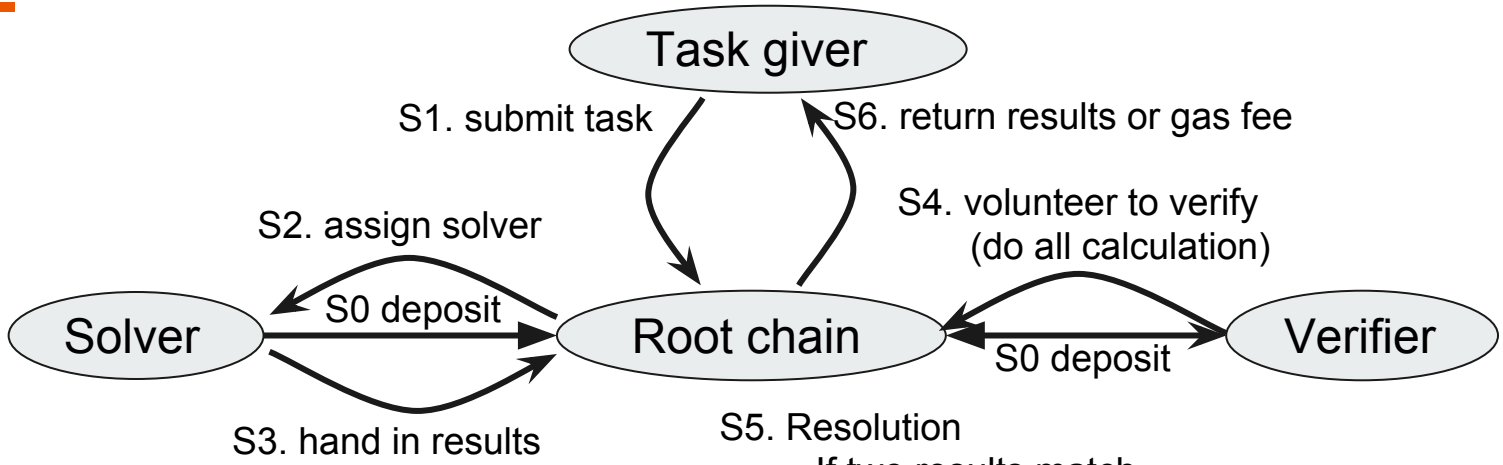
- Make secure blockchain computation affordable
  - Reduce redundant computation
  - Incentivize the verifier
  - Still as secure as Ethereum

Ideas

- Make secure blockchain computing affordable
  - Reduce redundant computation
    ⇒ Select solver and verifiers
  - Incentivize the verifier
    ⇒ Reward verifiers for detected errors
    ⇒ What if almost no error in all computation?
    ⇒ **Forced error** and **Jackpots!**
  - Still as secure as Ethereum
    ⇒ Using root chain as judge

Small innovations with thoughtful design are more promising.

Idea I (reduce redundant verification)

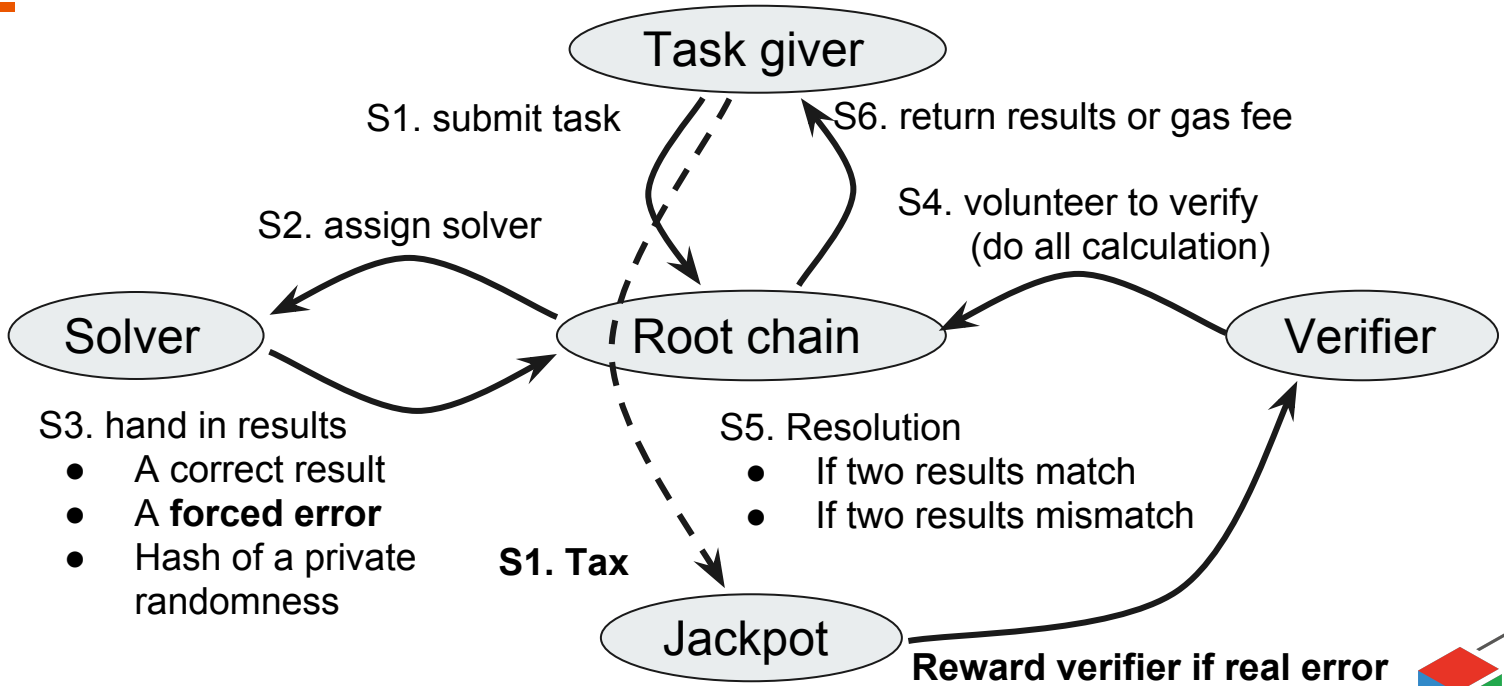


S5. Resolution

- If two results match
  - Solver take receive gas fee
- If two results mismatch
  - Submit to judge for arbitration
  - Punish solver or verifier

Pros: reduced verification work
Problems left: no incentives to verifier if the solver always do the calculation correctly

Idea II (incentivize verifiers more)



Pros: verifiers are incentivized to verify all computations.

Forced error

Forced error needs to be **unpredictable**

- Task giver shouldn’t know whether to use “forced error”
- Solver cannot choose “forced error”
- Verifier does not know if the solution is “forced error”

Forced error (cont'd)

1. Solver commits the private random bits "r" on chain before there is a task.
2. Solver commits the hash of correct and incorrect solutions on chain.
3. Based on **the hash of next mined block** and "r", solver designate whether to use forced erroneous solution.
4. If any verifier challenge the result form solver, the solver reveals "r". No punishment to solver if it is forced error.
5. If it is forced error, do verification based on "correct" solution

use forced error if hash("r" + hash-of-next-block) < difficulty

How solver and verifiers are selected?

- Solver
  a. Submit min-Deposit to become candidate
  b. Submit a Tx to show interest
  c. Select solver by include Tx in next mined block

- Verifiers
  - Volunteer to join
  - Split the rewards
  - Using incentive to control # of verifiers

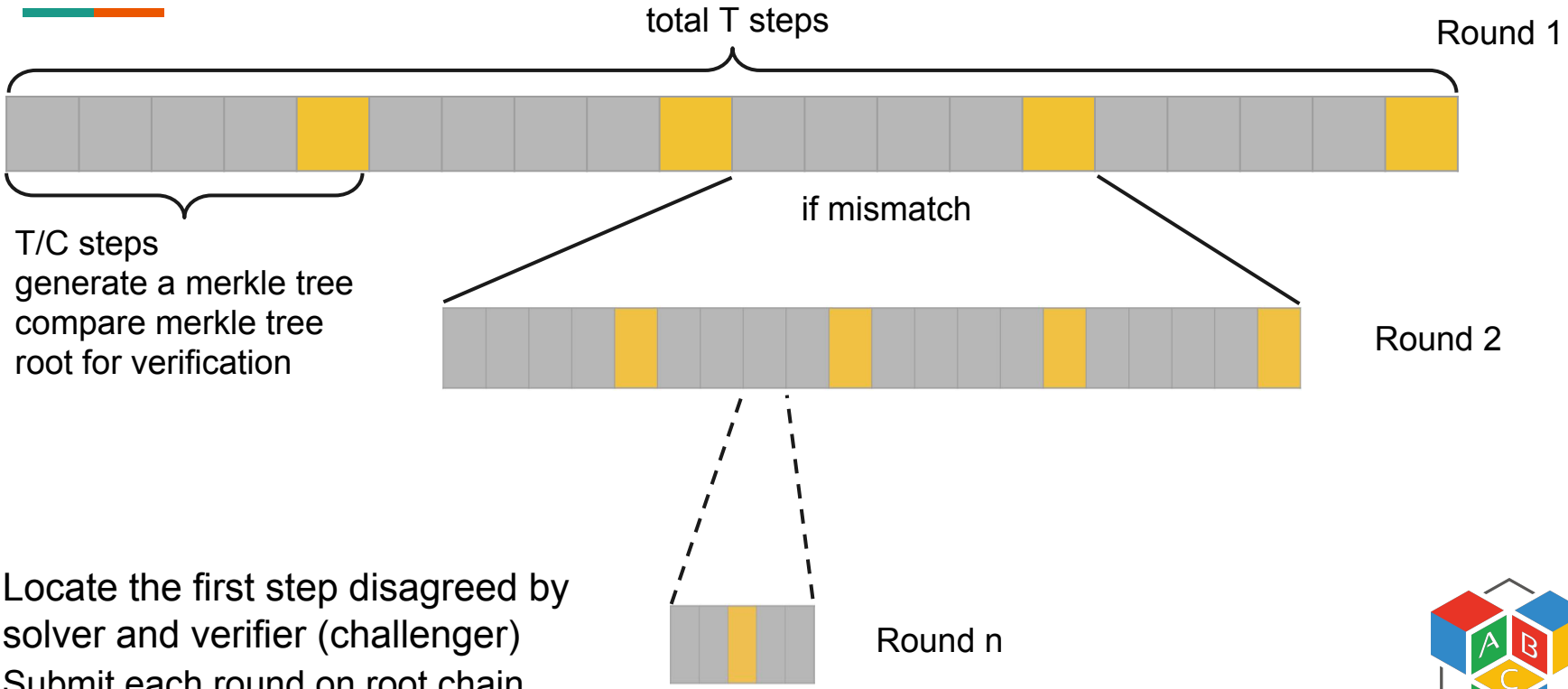
Key parameters

- Tax(to Jackpot): 500%~5000% of the cost to compute
- Forced error rate: 1/1000

Verification game

- Judge is needed, if the some verifiers challenge solver’s solution
- Root chain is the judge to ensure security
- Verify all steps?

Verification game



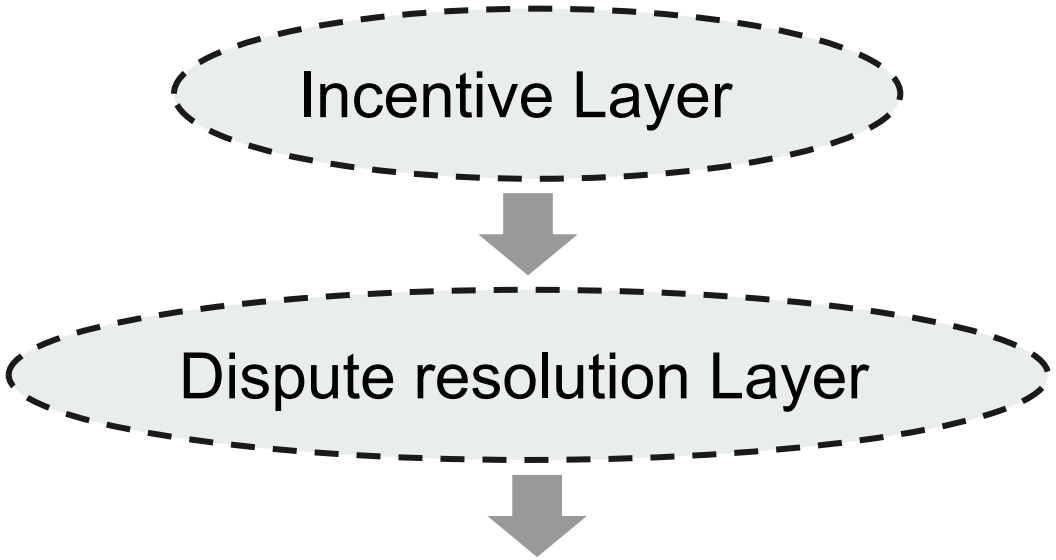
- T/C steps
- generate a merkle tree
- compare merkle tree root for verification

- Locate the first step disagreed by solver and verifier (challenger)
- Submit each round on root chain

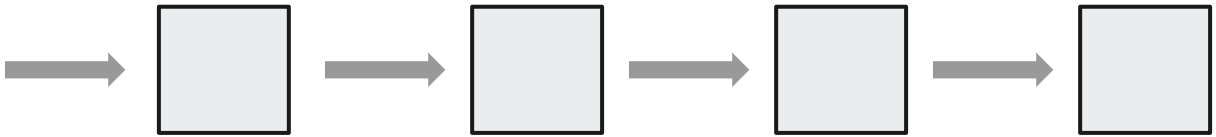
Verification game

- Locate the first step disagreed by solver and verifiers (challengers)
- Solver asks for arbitration from Judge (root chain) with providing
  - pointer to erroneous step “E”
  - state before “E”
  - State after “E”
  - A path from merkle root to “E”

Architecture



Root chain



Implementation

- Google Lanai interpreter
  - Simple code architecture
  - Supporting C, C++, or Rust through LLVM compiler
  - Truebit will be smart contract on Ethereum

Thank you!

Yijie Hong
11/16/2018