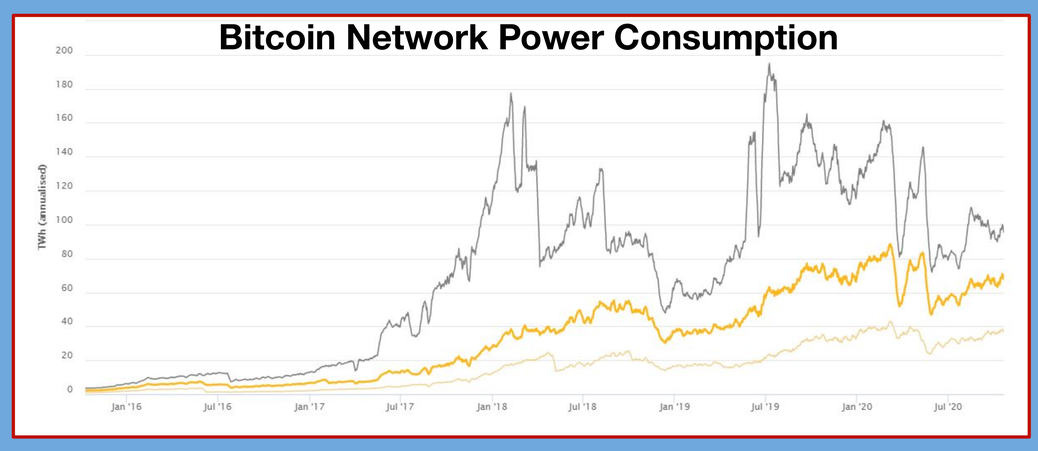
A Template for Useful Proof of Work

Riley Vaughn and Sajedul Talukder
Edinboro University

Objective: Create a template for useful Proof of Work protocols

Abstract

Many popular cryptocurrencies, such as Bitcoin, form consensus through a method known as Proof of Work. Problematically, current implementations of Proof of Work require immense amounts of energy consumption, where a majority of this energy is spent solely on securing consensus. Our focus is not to directly decrease energy consumption, but to allow for more useful and pragmatic computation to come from Proof of Work, such that energy is saved by not running these computational tasks separately. In this paper we create a template for Proof of Work protocols, such that if followed, can guarantee similar security assurances as the Proof of Work found in Bitcoin. Secondarily, we also develop "useful" prototypes based on this template.



Proof of Work Parameters

Random Oracle
- The Oracle--source of problems--must produce outputs randomly selected from the Oracle's range, otherwise the adversary could discover a pattern.

Interactivity
- An Oracle is interactive if problems are generated externally and non-interactive if generated locally. We default to non-interactive until shown otherwise.

Efficiency of Verifiability
- Without Efficient Verifiable, one would be unable to efficiently check transactions on the blockchain. The technology would be unusable.

Auditability and Trapdoors
- If a PoW had a trapdoor, it would definionally not be decentralized. Similarly, a PoW must defionitionally be publicly auditable.

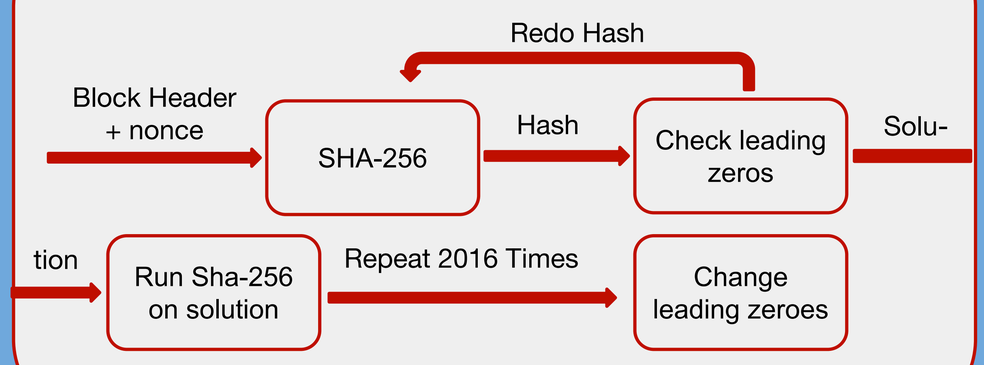
Bounds of Cost
- Having yet to show that the bounds of cost may be anything other, we default to the idea that they must have unbounded probabilistic cost.

Parallelizability
- The ability to parallelize the work does not add to the security of Bitcoin's PoW. As long as other parameters are met, parallelizability is not necessary.

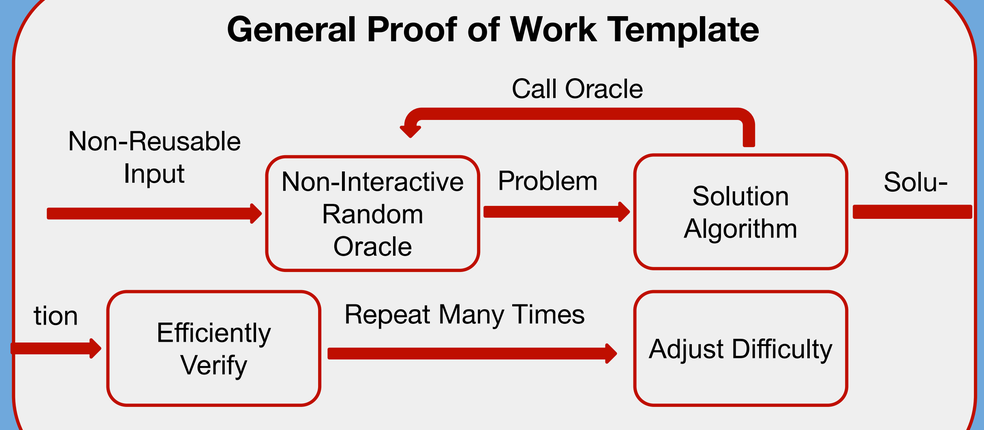
Adjustable difficulty
- In order for a PoW to be similarly as secure as Bitcoin's PoW, the Random Oracle must have linear adjustable difficulty.

Non-Reusability
- It is important that computational resources used to mine one block cannot be re-used to mine another.

Bitcoin's Proof of Work



General Proof of Work Template



List Sort Prototype

- The non-interactive Random Oracle receives a unique, non-reusable input and Produces a list of randomly ordered numbers.
- The actor sorts the list with any efficient sorting algorithm
- To verify, an actor checks if every item in the list is in order. This is efficient compared to PoW because verifying takes O(n) while sorting takes at minimum O(nlogn).
- Difficulty is adjustable by list size
- Is only parallelizable if using a sort that is not memory intensive.

Open Questions

- Do interactive Random Oracles exist? If so, are they as secure as the non-interactive counterparts? If the answer to these questions are yes, then truly useful Proofs of Work are possible.
- Must an Oracle have unbounded probabilistic cost? Can it be fixed cost or must it have a probabilistic cost? The answer to these questions determine the range of problems useful Proofs of Work will be able to help solve.

Interested in our research? Please visit our PENSLab:
http://penslab.cs.edinboro.edu/

PENSLAB Privacy ENhanced Security

EDINBORO UNIVERSITY