NCNY (no cool name yet)

TLDR

: A smart contract that contracts to validators the validation of a complex off-chain code, run only if contested by either side.

The problem

: Running complex code or using large data on the mainnet or even a shard is expensive. It is often easy for both/all parties to run the same code, and know the right answer, but not to enforce the other parties to accept the result.

Solution

: A smart contract that enables enforcing the correct execution of a deterministic code. If there is a dispute over the results, any party can call on the smart contract, that will send the hash functions of (i) the code (ii) input data and (iii) output to several validators. The validators download and run the code and decide if the execution was faithful. Based on the results from the validators, the smart contract will fine wrong parties, compensate right parties and change blockchain state data. Validators are only paid, by the losing party, when called upon.

Example

: Bob wrote a nice game. Jack and Alice play it online, not for money, just for bragging rights. The scores are kept on the Ethereum blockchain. Alice was about to win, when Jack made an impossible move, clearly by modifying the original code. To prevent such things, Bob placed the hash of the code in a smart account with all players putting a deposit to play for bragging rights. Alice can now send to the contract the hash of the illegal move by Jack (signed by Jack, as sent to her), as well as “links” to data she will make available, including Bob’s code, and the history of all moves if needed. Alice will also send a deposit, to be used to pay the validators if she is wrong. As Alice does not have access to Jack’s input (state of the game, plus his input, resulting in his signed output), making this data available falls on him (as specified by Bob to the contract). Within a given timeframe, he needs to provide the validators the input that will produce the signed output he sent Alice, or lose his deposit and score.

Jack fails to provide an input, comprised of a starting state (last move signed by Alice) and a move by himself, that results in the output he signed. His deposit is used to pay the validators and to update the blockchain for a win for Alice and loss of X points for cheating for Jack.

Note

: The whole idea is of course to make this all completely automatic, a software sees that one of the interactors misbehaves, calls in the smart contract that automatically fines the wrong party. There are no limits on the size or complexity of the software, only for it to be possible for a simple script to download and execute it on a given data and get a deterministic result. The cost is only ~3-50 extra executions of any questioned result, that should be kept to a minimum, because committing a crime will be rare when you have a 100% chance of getting caught and punished.

Potential problems and solutions:

Selecting validators

: Because this is not done in real time, this is less of a problem then for block validation. Validators can be selected from a given (PoS blacked) pool based on the hash value of future blocks. For example, if 3 validators are required, controlling all 3 will require controlling the next 3 blocks after the dispute is made. The validator pool can be shared for all codes, or a subset of validators can limit themselves based code complexity or required hardware for efficient execution.

Validator honesty

: Validators are paid if there is consensus among them. A small subset of validators not agreeing with the majority will be heavily fined, a large minority or a split will cause more validators to be called upon.

Data availability

: The contract specifies which party has to provide what. For example the disputer can make the full code and alleged false output available, while the generator of the output can provide his input. Failure to make available the data constitutes a loss. If only (x/n) validator claims no data access, x validator/s losses something. At some (m/n), more validators can be called upon. More than (y/n) validators cannot access data, responsible party loses.

Non-deterministic code:

If the code is non-deterministic for the given input, the validators return this as the output, and the code creator pays.

If anyone already did something similar that I just don’t know about, please let me know.