GoUncle: A Blockchain Of, By, For Modest Computers

Mao Wenbo and Wang Wenxiang
DaoliCloud Lab, Beijing, China
wenbo.mao@gmail.com, daolicloud.com

December 28, 2021

Abstract GoUncle is a blockchain for permissionless participation by modest computers. As in GHOST (Greedy Heaviest Observed SubTree, in successful implementation and use by the Ethereum blockchain's Proofs-of-Work version), GoUncle blocks also record the public-key IDs of some temporary forking blocks finders who are dearly called "uncles" (poorly named "orphans" in Bitcoin). While GHOST uncles are for saving PoW computations, GoUncle assigns jobs for its uncles to do. In a *payload distillation* job, uncles choose from block payloads only the logs which comply with the blockchain database (DB) policy to announce for to survive the blockchain gossip protocol. With uncles distillations, the blockchain address, aka height, for a no-longer-need-to-trust block, is *deterministic* right upon the block extending the blockchain. The deterministic blockchain addresses can index partition the distributed DB into small files to store in nowadays over provisioned external storage even for a low-cost computer. The index partitioned DB files can be fast operable for input, output, lookup, insert, update, manage, ..., etc., as a standard DB management system (DBMS) can. It is the fast operable property of the DBMS, even by a modest computer, that secures the blockchain DBMS by a *hop-by-hop firewall* among vast *semantics gossipers* who each looks up the local DBMS to judge either writing to the DB correct uncles distillations and forwarding them on, or discarding incorrect ones, both operations being quick. Since the hop-by-hop firewall works exactly as *correctness probability amplification* by repeated execution of a *randomized probabilistic* (RP) algorithm, the GoUncle work establishes:

$$\text{Blockchains} \subset \text{RP}.$$

Also to be manifested in the present work are more general blockchain consensus layer computations that uncles can and should execute and disseminate the execution output as No-Spam and No-Single-Point-of-Failure (No-SPOF) set of blockchain servers.

Key Words and Phrases: Easy Permissionless Blockchain. Blockchain Uncles as No-Spam and No-SPOF Servers. Blockchain Addressable Computers for Consensus Computations. Semantics Gossip as Hop-by-Hop Firewall for Blockchain. Blockchain as a Randomized Probabilistic (RP) Algorithm.

1 Introduction

Bitcoin [1] began the era of open source peer-to-peer money and inspired the blockchain area of study. The work of the GoUncle blockchain taps the potential of the blockchain technology to construct a secure, public-writing, scalable, low cost to use, and quick lookup DB for applications ranging from a public ledger to general purpose DB applications.

The presentation of this paper begins with discussions on a number of issues to serve analyses and identifications for how a very important security function of blockchain-gossip-diffusion has so-far been used in compromised ways in popular blockchains. These compromised usages can be referred to as "diffusion by syntax" for so-called Proofs-of-Work (PoW) blockchains (Section 2), and "diffusion by affluence" for so-called Proofs-of-Stake (PoS) blockchains (Section 3). The discussions lead to thoughts for improvement, which we shall name "diffusion by semantics" (Section 4). A new blockchain consensus algorithm is proposed with reasoning why and how it can make use of the semantics version of the gossip-diffusion function (Section 5). In reporting an implementation work for the GoUncle blockchain, a more general and useful fact for blockchains is evidently and constructively manifested (Section 6). Finally, this paper concludes the work presentation and asks a question to invite answers and/or explorations from interested readers (Section 7).

The semantics improvement in the gossip-diffusion function is due to an application of knowledge in the computational complexity theory. A randomized probabilistic (RP) algorithm needn't be sure about its output correctness for an execution instance. Repeating execution can quickly amplify the probability for its output correctness. Treating a blockchain as an RP algorithm, there is no need to be sure about the trustworthiness of a payload upon its recording block appending the blockchain. There is also no need to wait a long delay for the block to become trustworthy, as all known PoW blockchains slowly do. Where are the repeated probability amplification executors for a blockchain? We know of ways for a blockchain to record not only payloads but also addresses of computers which can even be permissionless participants in the blockchain network. Let these blockchain addressable computers play the needed function of probabilistic amplification.

There is a very nice benefit from the RP algorithm decoupling between the blockchain job for appending blocks to lengthen the chain, and that for checking correctness of the appended blocks. Since data written to an RP formulated blockchain's DB are diffusion-by-semantics distillations of some untrusted payloads, letting payloads remain untrusted is no longer a problem anymore. Then an untrusted payload gets *deterministic* address right upon appending its recording block. The distillations provide lookup connections between the deterministic address and the DB content position. With the connections, the DB can be the address indexed and partitioned into small files to support fast input, output, lookup, insert, update, manage and the like standard DBMS operations. A blockchain DB becoming a standard DBMS constitutes significant blockchain knowledge

and practice advances from the current status quo quality of DBs in PoW blockchains. The status quo quality of PoW blockchain DBs, e.g., that of the UTXO Set for Bitcoin, is that, with ever growing larger and larger size and without a trusted operator to manage the DB, such DBs are forever let occupy very expensive internal memory, e.g., Random Access Memory (RAM). Consequently, most of the participants in a PoW blockchain are prevented from validating the trustworthiness for the blockchain payloads, since the job requires a computing equipment cost they cannot afford. This poor quality of PoW blockchain DBs is probably the most relevant cause for PoW blockchains to have no scalability.

In addition to knowledge contribution, this paper also serves the technical basis for a blockchain project (`daolicloud.com`). The project is to construct an RP formulated blockchain with an easy participation design, in that most of its participants only need be modest computers, such as personal computers, smartphones, home WIFI routers, cloud containers, etc. Each of such modest devices is a fully functional semantics gossiper, and can take part in a set of No-Spam and No-Single-Point-of-Failure (No-SPOF) servers for consensus computations.

2 PoW Issue of Poor Gossip Quality

For Bitcoin to record users' transactions automatically, Nakamoto reinvented an ingenious use of a Proofs-of-Work (PoW) mechanism of Dwork and Naor for email spam prevention [2]. Nakamoto's use of PoW can enthuse permissionless participants, aka miners, to compete generating blocks for lengthening the Bitcoin blockchain lively. One miner will win in each round of competition, with its block composing users' transactions as "correct" entries to the Bitcoin ledger. By quoting "correct", we mean a low quality sense of correctness. A permissionless competition usually and easily goes into a vicious cycle of quality deterioration. It has been widely accepted that a blockchain using permissionless PoW competition cannot avoid limitations of inefficiency, slowness, and high cost of use. For making rationale behind these limitations, we notice a technical detail in two most popular PoW blockchains, Bitcoin and Ethereum (Ethereum 1.0, the PoW version). In specific, both stipulate the same specification of delaying trust as follows. Upon a block extending the chain, it remains untrusted until having been buried under 7 new blocks appending it.

Using the number 7 to instantiate n in the n-blocks-delaying-trust stipulation has a strong stopgap taste as an algorithm specification. This stopgap has fortunately been blessing a few PoW blockchains running properly, thanks to the fact that the PoW competition in these blockchains turned out to be extremely severe. So severe that finding several, e.g., 7, blocks in a winning streak by one miner is an extremely hard problem to be practically impossible. Suppose that this severity of PoW competition can be sustainable. The PoW delaying trust stipulation has some issues to discuss below.

With delaying trust, a PoW blockchain's DB is always written only by untrusted operators. There never exists a trusted DB operator to organize such a DB, e.g., into small

and index addressable files for fast input/output, as a standard database management system (DBMS) is organized. A delaying trust DB, after having grown to a large size, has to keep occupying the RAM, an expensive computing resource. To input/output such a DB (which data chunk?) between the RAM and external storage space would take too long time. Consequently, most of the participants and/or users in a PoW blockchain soon become unable to validate the trustworthiness for the untrusted payloads since they cannot afford having a huge RAM to do the validating job. The gossip protocol that the blockchain technology crucially employs for epidemic diffusion of influences, now for a PoW blockchain, is reduced to only disseminating syntax "correct" blocks, i.e., those having reached a "work" threshold. A so-called "51% Attack", and a so-called "Sybil Attack", that a PoW blockchain has to risk, can be mounted by a smaller fraction of the miners who are computationally very powerful and compose incorrect DB entries in syntax "correct" hence influential blocks, for the rest of the larger fraction of the network peers to have to gossip diffuse.

3 PoS Issue of SPOF Complications

In frustration of PoW blockchains' inefficiency, a so-called "Proofs-of-Stake" (PoS) blocking model emerged. Let a blockchain participant wishing to generate blocks deposit a sum of money, called security stake, in a PoS money lock-up mechanism. The more amount of money a participant stakes, the more chance for its turn to generate a block. The security stake is for punitive confiscation or destroy (in PoS programmers' jargon, "money slashing") when a block generator is found to be responsible for either an error ledger entry, or upon a timeout having no ledger writing action. A PoS version of "Ethereum 2.0" would be the most eminent PoW-to-PoS switch to take place in a near future of writing this paper.

An elegance revealed by Bitcoin is PoW's No-SPOF property. Miners competing for block generation renders the system being always alive. If a PoW won miner has composed an error entry for the DB, other error spotting miners will come to the blockchain point before the error block and PoW compete to fork the chain. These error correction miners' blocks will soon override the error block since these miners' combined PoW capability dominates the gossip influences in the network. However, the very PoS idea is for the system to always designate (elect, assign, appoint, or pick) a block generator for this single point to extend the chain. Namely, a PoS blockchain always enters a SPOF state. A money slashing PoS blockchain even suffers from worse SPOF complications, to discuss below.

Money slashing for deterring DB entry error is an easier-said-than-done idea. First, thinking a DB error entry to be something necessarily to do with a DB writer, this is already a misunderstanding on the unreliability of the gossip network. Secondly, thinking a network distributed identification for the nature of an error is a more absurd idea. We know of an academic conclusion, named "Fischer-Lynch-Paterson (FLP) Impossibility" [3], being the following statement. There never exists a network distributed

algorithm, inputting network gossip messages to output a consensus decision. Money slashing, if having been in practice in any PoS blockchain, necessarily is by some unspecified manual operation. Still a manual slasher would need to figure out an equitable exchange rate between an "error" and a money amount to slash.

The PoS idea should have little to do with programming, not to mention with the notion of consensus algorithm. It is more about a business arrangement based on a conventional belief in that the richer the more responsible. Therefore a PoS blockchain should mainly rely on some venture love participants from the business sector. A for-profit business participant will have to discover a comfortable profit margin over cost particularly including the risk of its staking money being mistakenly slashed. Such business participation costs will translate to service fees chargeable to the blockchain users. A number of PoS blockchains, e.g., Peercoin, Myriad, Blackcoin, VeriCoin, NXT, Algorand, have appeared. They serve pre-echos for new PoS blockchains to come, e.g., Ethereum 2.0, Solana, Cardano, for the blockchain industry to try-and-error learn how the PoS idea shall actually work.

4 Rethinking Blockchain Gossip

Public blockchains, whether PoW or PoS based, crucially use gossip relay network to achieve an "epidemic" spread of "certain" messages far and wide throughout the network for the messages to influence the network distributed DB. Here we quote the two words in order to examine their actually misused meanings for some blockchains. Our analyses so far on the causes of limitations with these blockchains shall ease our discussions below.

First, let us examine the modifier "certain" for gossiped messages in known public blockchains' consensus layer algorithms. With PoW, "certain" simply equals syntax "correct". Notice that it is possible for a PoW miner to compose incorrect payload contents in a syntax "correct" block. PoW blockchains stipulate a long delay for a truly correct block to get confirmation, or for a syntax "correct" but content incorrect block to be undone. PoW blockchains suffer from various low quality limitations, and are even possibly under attacks, e.g., a so-called "51% Attack". With PoS, "certain" even has some troublesome meanings in terms of botched system architecture design in that, the entire network of gossipers in a PoS blockchain always enter a SPOF state for listening to a block from a designated composer. In essence, a PoS blockchain always relies on a single entity for the blockchain liveness. There are still other SPOF complications such as the FLP-Impossibility to reach a money slashing consensus. Moreover, the PoS idea for securing blockchain has a questionable long term equitability due to interest difference between business stakeholders as designated block composers and blockchain users.

Secondly, the word "epidemic" for these blockchains does not really mean "a wide range diffusion". Known blockchains seemingly all have to use only a very small fraction of the network nodes for the role of gossipers, block generators, and/or validators. With PoW, they are a small fraction of the so-called full-nodes. As in need of being

computationally powerful, the full-nodes form a tiny minority of the entire participation nodes. With PoS, they are designated block extenders and/or qualified block validators, forming possibly even a smaller, in comparison with PoW, minority of the network nodes since in need of being monetarily affluent.

For a blockchain's gossip protocol, we desire "certain" to equal semantically correct, meaning that should only messages being consistent with the blockchain's DB writing policy survive gossiping. We also desire "epidemic" to mean for a larger, more desirable overwhelmingly larger, fraction of the blockchain participants being capable semantics gossipers, so that any semantically incorrect payload content will be discarded in *early* hops of semantics gossip.

Let us devise a permissionless blockchain running a semantics gossip protocol for modest computers, such as a client wallet, a personal computer, a smartphone, a home WIFI router, or a cloud container, etc., to be able to participate in as fully functional semantics gossipers, and can lookup the blockchain DB quickly.

5 A Blockchain Of, By, For Modest Computers

Our analyses of the PoW consensus mechanism in Section 2 revealed an issue of delaying trust in reaching a PoW consensus. We regard this issue to be responsible for a PoW blockchain's DB to keep occupying a huge and ever growing sized RAM. An in-RAM DB, after having grown to a large size, becomes unusable by a modest computer. That is why a Bitcoin wallet cannot know whether or not a transaction crediting it is valid and has to wait a long time for the full-nodes' to validate for it indirectly and implicitly. Obviously, a majority of the PoW blockchain participants, such as wallets, cannot function as semantics gossipers.

The evolution of digital computers once arrived at a point of enlightenment owing to von Neumann, that data for a computer to store and process can themselves be computers. This enlightenment has of course also happened to blockchain as a computer. Bitcoin's scripting system, and Ethereum's smart contracts, are in fact computers having von Neumann addresses stored in these blockchains. These blockchain addressed computers are very useful for processing users' transactions and/or third parties' contracts. We therefore regard that Bitcoin and Ethereum blockchains use von Neumann computers for solving the user-space problems.

The von Neumann enlightenment for blockchain can go just one small step further. Let blockchain addressed computers solve problems in not only the user space, but also the kernel space, e.g., blockchain consensus layer algorithms. In-RAM operating blockchain DB for PoW blockchains, SPOF complications for PoS blockchains, syntax gossipers incapable of making semantics judgment, are an example of blockchain kernel-space problems in need to solve.

GHOST (Greedy Heaviest Observed SubTree) [4] is a PoW algorithm to not only line up lucky blocks found by the PoW game won miners, but also store in the lucky blocks

public key identities of some less lucky blocks' finders. These less lucky blocks' finders are called *uncles* in Ethereum's (the PoW version) implementation of the GHOST algorithm. GHOST uncles are blockchain addressable computers, though they have never been so used. Both the GHOST research work and its implementation by Ethereum only use uncles for saving, otherwise wasted, PoW mining electricity.

We propose a novel use of GHOST uncles. Let them be a set of No-Spam and No-SPOF servers and execute some useful operations for the blockchain. By controlling the blocking traffic not to reach a spam level as GHOST doing, we can have a random and No-Spam volume set of uncles whose public-key identities have been exposed to the blockchain as addressable servers.

Let the blockchain's lucky blocks include at least three parts of data as follows.

- `Payload`: A set of logs which the users, and/or third parties, request to write to the DB. The current block address, aka height, is said to be the *logging address* for these logs.
- `Uncles`: Public-key identities of some less lucky block finders the lucky block records as GHOST-like uncles. The current block address is said to be the *residing address* for `Uncles`.
- `DB_Entries`: Semantics gossip surviving logs announced by uncles with earlier residing addresses.

The following "Uncle Algorithm" assigns the exclusive network broadcasting entitlement for a random set of No-Spam and No-SPOF blockchain addressable servers to announce some payload logs.

Uncle Algorithm An `uncle` announces a `log` if the logging address is earlier than its residing address and it judges that `log` is consistent with the blockchain's DB writing policy.

The following "Semantics Gossiper Algorithm" is executed by every blockchain participant.

Semantics Gossiper Algorithm Let a blockchain participant upon receiving a `log` announced by an `uncle`:

1. Return if `log` is of a third party application's; else
2. Discard `log` if its logging address is later than the residing address of `uncle`; else
3. Discard `log` if it has been already gossiped; else
4. Discard `log` if it is inconsistent with the DB writing policy; else
5. Write `log` to the local DB and forward `log` to the peer neighbors.

In this semantics gossiper algorithm, Step 1 states the responsibility of a third party's application; Step 2 prevents overzealous uncles from spamming the network; Step 3 guarantees that uncles' announcements will be quiet quickly; Step 4 specifies a hop-by-hop firewall securing the blockchain network distributed DB against any semantics attack even from a dishonest uncle; and finally Step 5 guarantees quick writing the DB.

Running these two algorithms, the blockchain has the following two properties.

1. Every DB-entry-policy consistent log will get some uncles' dissemination for semantics gossipers to diffuse far and widely, and enter the distributed DB, very quickly.
2. Any DB-entry-policy violating log announced by any uncle will be discarded by the semantics gossipers, also very quickly.

Property 1 holds simply because of the endless supply of future uncles. This also means that the uncles' service is of No-SPOF quality. Thus we have:

$$\text{Probability}(\text{ Log is written to DB} \mid \text{Log is correct }) = 1. \tag{1}$$

Let us reason Property 2, in particular its quickness. Since the two algorithms even tolerate a dishonest uncle, the blockchain payload logs can remain untrusted. Therefore a lucky block gets a *deterministic* address right upon it appending and extending the blockchain. The association between the deterministic block address and correct payload logs in the block becomes immediately usable by the global blockchain participants. Since correct logs will be written to the DB, a distributed DB managing algorithm can use this association to manage the DB into index-partitioned and index-searchable small files. So indexed DB files can be stored in external storage spaces (disk or solid state drives) which are nowadays low-cost over-provisioned to modest computers such as laptop computers, smartphones, home WIFI routers, or cloud containers, for fast DB operations such as input, output, lookup, insert, update, manage, etc., all being standard DBMS operations. Such a modest device can quickly lookup the blockchain DB and judge an entry correctness itself without any delay. Thus, we have:

$$\text{Probability}(\text{ Log is written to DB} \mid \text{Log is incorrect }) = 0. \tag{2}$$

Thus, repeated announcements from random uncles can indeed quickly "distill" blockchain payload logs into semantically consistent DB entries, exactly as a *randomized probabilistic* (RP) algorithm can amplify the correctness probability for its execution by repeating the algorithm. As contribution to knowledge, we establish

$$\text{Blockchains} \subset \text{RP(ZPP)}. \tag{3}$$

Here ZPP stands for Zero-sided-error Probabilistic Polynomial-time. The new blockchain has no completeness-side error, as stated by the conditional probability (1), and it also

has no soundness-side error, as stated by the conditional probability (2). This RP(ZPP-subclass) formulated blockchain is always correct and always fast, even executed by a modest device with low-cost over provisioned external storage space.

The data part `DB_Entrie` in a block suffices the blockchain's distributed DB to be diffusion-by-semantics (for terminology and its meaning, review Section 1) influenced by the semantics correct logs. Since a semantics gossiper writes a correct log to its local DB before forwarding it on (Step 5 in Semantics Gossiper Algorithm), the semantics correct logs are written to the distributed DB earlier than they show up explicitly on the blockchain. We can regard that the blockchain and the distributed DB operate on "disk". In other words, this RP(ZPP) formulated blockchain is a Disk Operating Blockchain (DOB).

With modest devices being capable semantics gossipers, this RP(ZPP) formulated blockchain can attract a large number of modest computers to participate in, to achieve that its hop-by-hop firewall is widely and vastly deployed by a majority of the blockchain participants in effective operation to secure the distributed DB. The new blockchain of GoUncle is of, by and for modest computers.

A common blockchain application of the public writing DB is a public ledger. The DB-entry policy for this application is that coins specified in a user transaction request can be looked up from the relevant DB small files as being currently locked to the transaction specified payer(s). The DBMS operations by the uncles and the semantics gossipers involve to lookup the relevant small DB files, and when the lookup returns YES, further involve to update the small DB file so that these coins become being locked to the transaction specified payee(s). To index partition the DB into small files, the blockchain's DBMS algorithm can use the blockchain addresses (block heights) to parameterize file names, so that all semantics gossipers know the names of the small files to create, write, lookup and update.

6 Uncles Disseminating Consensus Algorithm Executions

Quick and easy distilling payload logs into semantics consistent DB entries is only one way to use a No-Spam and No-SPOF set of blockchain uncles. These uncles can provide other useful services that a blockchain can and should use. Listed below are a number of blockchain computations in need of a NO-Spam set of blockchain servers to execute as No-SPOF services. An uncle can:

1. Announce a blockchain state progress, e.g., arrival of a network message, and/or occurrence of a time-out event. In these uses of a No-Spam set of uncles, the uncles' dissemination not only adds a No-SPOF reliability to the system, but also provides a probability sample space for the global participants to compute statistics formulations, e.g., median, mathematical expectation, variance and deviation. We shall describe a simple time-out formulation to be very useful for our version of the GHOST blocking algorithm.

2. (Having the public-key ID being exposed to the system) Provide the TLS public-key certification authority (CA) server functions for securing communications with peer neighbors, as originally proposed by the work of Cothority [5]. For uncles to play the role of TLS CAs is a very useful security service. With many blockchain uncles available as blockchain CAs and with peer communication encryption being mandatory, a new participant has to register a TLS public key with a plural number of earlier uncles. This anonymously registered TLS public key can be listed by the uncles in the blockchain DB as an entry for *Public-Key Defined Network* (PKDN) application. This novel PKDN application can let a mobile blockchain node be always route-able wherever the mobile device travels to and whatever its IP address has changed to.

3. Be a block generator upon the blockchain encountering an unknown liveness exception.

Let us see a use of uncles dissemination as a service in the fashion (1) above. The GHOST work [4] correctly reckons that the unavoidable imperfection of a blockchain network in varying message travelling times will inevitably partition the blockchain network to cause chain forks. It assumes a "delay diameter of the network" and crucially uses it however without a sure way to know it. This is where uncles dissemination can help. In specific, in the GHOST blocking time (Algorithm 1 in [4]), uncles can make echo announcements for a newly broadcast block. The plurality and varied distribution of the uncles can not only lower the probability for the network to split, but also explicitly tell an average value for the time delay diameter of the network since the uncles' echo announcements for a block must take place after the block broadcast.

The "GHOST" blocking for GoUncle uses a counter in place of a random nonce in the PoW blocking algorithm. Unlike a random nonce without a limiting range, the counter has the maximum value to deterministically stop valid PoW output. Thus, the counter actually serves the clock ticking function to be explicitly ticked by the uncles for the the global participants. Uncles will disseminate their counters reaching the maximum for the global gossipers to discard further blocking broadcasts. Thus, the blocking traffic will reach a global state of quietness for definitive calculation for the lucky block. Moreover, the deviation calculation for tie-break can be used in case when GHOST outputs encounter a tie.

In the GoUncle counter-replacing-nonce version of the GHOST algorithm, while the counter value is below the maximum setting, every participant has the same lucky probability for finding a valid block, whereas upon the counter reaching the maximum setting, no valid block can be found anymore. Both cases have nothing to do with the computational resource the participant has, however modest or powerful it may be. In particular, after the uncles collective disseminating that they have counted to the maximum setting of the counter, the blocking traffic becomes quiet for all. Therefore the GoUncle blockchain discourages participation using powerful computers.

To see a use of uncles dissemination as a service in the fashion (2) above, let the

blockchain require a newly joining participant to bind a wallet identity public key to its TLS key with the peer peer neighbors in that, the wallet key being different from the TLS key. Then the blockchain can prevent the participant, being a PKDN anonymous registrant, from launching a Sybil attack, by discarding blocks generated by an unregistered wallet key.

In a near future we shall report the implementation work for the GoUncle blockchain, where useful collective services from blockchain uncles will be described in detail.

7 Conclusion

The No-Spam and No-SPOF set of blockchain uncles provides a new methodology for permissionless, autonomously organized, redundant, replicated execution and output dissemination of blockchain consensus layer algorithms. The work of this paper has manifested the power of this methodology by: (1) A knowledge revelation that blockchains can be formulated in randomized probabilistic algorithms to run efficiently and reliably, and (2) A construction of a semantics gossiper algorithm running on a large number of modest computers to disperse a hop-by-hop firewall and strongly secure the blockchain's distributed public writing DBMS. The new blockchain has a robust reliability under distributed, redundant and independent control of a vast number of modest computers, is securely managed and maintained by them, and is for a modest computer, such as a client wallet, to lookup data quickly. Therefore the new blockchain of GoUncle is of, by, for modest computers.

Question: Let a majority of the participants in a blockchain be semantics gossipers. Can this blockchain accumulate a 51% Attacker?

Thought: Being the source of a semantically incorrect message seemingly equals being a non-participant since the message will be discarded in early semantics gossip hops. Hence the remainder of the blockchain's participants remains to be 100% of the network.

References

[1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system.* Tech. rep. Manubot, 2008.

[2] Cynthia Dwork and Moni Naor. "Pricing via processing or combatting junk mail". In: *Annual International Cryptology Conference.* Springer. 1992, pp. 139–147.

[3] M. J. Fischer, N. A. Lynch, and M. S. Paterson. "Impossibility of distributed consensus with one faulty process". In: *Journal of the ACM (JACM)* 32.2 (1985), pp. 374–382.

[4] Y. SOMPOLINSKY and A. ZOHAR. "Secure high-rate transaction processing in Bitcoin". In: *International Conference on Financial Cryptography and Data Security*. 2015, pp. 507–527.

[5] *Cothority*. `https://github.com/dedis/cothority`.