GoUncle: A Blockchain Of, By, For Modest Computers

Mao Wenbo and Wang Wenxiang
DaoliCloud Lab, Beijing, China
wenbo.mao@gmail.com, wenxiang.wang1204@gmail.com

December 28, 2021

Abstract GoUncle is a blockchain for permissionless participation by modest computers. As in GHOST (Greedy Heaviest Observed SubTree, successfully implemented by and used in the Ethereum blockchain's Proofs-of-Work version), GoUncle blocks also record public-key identities of some forking blocks' finders who are dearly called "uncles" (poorly named "orphans" in Bitcoin). While GHOST uses uncles only for saving PoW mining electricity, GoUncle assigns jobs for uncles to do. In a so-called *payload screening* job, uncles choose from earlier payloads only data items complying with the blockchain database (DB) policy to announce for the blockchain's gossip protocol to diffuse. Now that the blockchain can readily append blocks containing incorrect payloads, each block's height as a globally known address becomes *deterministic* right upon appending the block. The deterministic blockchain addresses can index partition the distributed blockchain DB into small files to store in nowadays low-cost over provisioned external storage, for fast input, output, lookup, insert, update, manage, ..., etc., exactly the same as a standard DB management system (DBMS) is operated. It is that the blockchain DB becomes a standard DBMS for fast operable even by a modest computer, that secures the DBMS by a *hop-by-hop firewall* among vast *semantics gossipers* who each, upon receipting a gossip of the uncles' screening, looks up its local DBMS and judges to either deposit it in and gossip it on, or discard it. This hop-by-hop firewall works exactly as *correctness probability amplification* by repeated execution of a *randomized probabilistic* (RP) algorithm, and thus the following becomes newly known:

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

Also to be manifested is a more general and methodological use of uncles 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. Permissionless Client-Server Architecture.

1 Introduction

In the Bitcoin paper [1], Nakamoto thought a "one-CPU-one-vote" way for the Bitcoin participating nodes to determine representation in majority decision making, and implemented a Proof-of-Work (PoW) hashing, aka mining, game for the participants to count the CPU votes. Since then Bitcoin has been working correctly for twelve plus years. However, because the PoW mining enthuses competition, the voting game has gradually and eventually turned to "one-X-one-vote" where X stands for massively parallelized CPUs and/or GPUs, specially designed hardware rigs, and even pooling of such into huge farms. What now remains being fortunately valid is that these X's, though no longer forming a majority of all Bitcoin participants, stay in control for Bitcoin to run correctly, thanks to the following two facts: (1) a majority of these X's are honest, and (2) the majority of all Bitcoin participants are not functional nodes for Bitcoin anymore, at least not in Nakamoto designed "one-CPU-one-vote" way. Thus, the participants in Bitcoin, who are functionally dependable for lining up blocks need to be computationally powerful, and increasingly so. Bitcoin has an increasingly lowering utilization of its participants.

Bitcoin's low utilization of its participants can also be clearly seen from its algorithmic method to diffuse messages. A public blockchain uses a hop-by-hop "gossip" way of communication to distribute data in blocks for the blockchain to log immutably in its public writing database (DB). The use of the word "gossip" as a technology descriptor, although sounding a little graphical, precisely describes how a blockchain DB is robustly and reliably secured by the distributed participants in the blockchain network. Each participant hop-by-hop receives and forwards a block, and deposits the block's payload data in its local duplicated copy of the DB. The gossip algorithm for Bitcoin can be referred to as "gossip-on-syntax" for most of the gossipers in that, a block qualifies gossiping if the easy hash validating returns YES for passing a *work* threshold. This easy syntax checking gossip function is serviceable by almost all participants. It instructs gossipers to assist only participants who are computationally powerful. Because a syntax gossip formulation cannot perceive the network topology structure of powerful miners, Bitcoin gossipers are unfortunately responsible helpers for a so-called "one-X-51%-of-the-votes" attacker.

A more resounding manifestation for Bitcoin's low utilization of its participants is its ledger lookup being notoriously slow. A newly found block extending the Bitcoin blockchain needs a long delay, in specific after having been buried under 7 new blocks extending the chain, for the block to gain confirmed trustworthiness. So a user looking up a transaction logged in this block must wait that long time (about 70 minutes). In need of a long delay for slowly confirming trustworthiness may be regarded as a nuisance only, however the necessary delay renders the Bitcoin ledger, i.e., the UTXO Set, to have never got a trusted operator to perform any DB management (DBMS) job. Such a no-trusted-operator-to-manage ledger DB, after having grown to a large size, has to keep occupying the RAM as a whole data chunk. To input/output the ledger (which data chunk?) between the RAM and external storage space would take

too long time. Consequently, most of the Bitcoin participants and/or users are unable to validate payload data correctness for Bitcoin as they cannot afford a huge RAM to hold the ledger. Nakamoto's "one-CPU-one-vote" ideal turned out to also necessarily mean "one-hugely-expensive-RAM-one-vote".

Let surrendering blockchain control to a small fraction of the blockchain participants be a have-to-accept way of life with public blockchain. A so-called Proof-of-Stake (PoS) model of block generation has appeared for improving blockchain efficiency and without wasting mining electricity. The PoS model bases blockchain security on a conventionally meaningful business belief that the richer the more responsible. Let a rich minority criterion qualified node deposit a sum of money, called security stake, in a PoS money staking mechanism. The more amount the participant, aka stakeholder, deposits, the more chance for its turn to generate a block. Thus, the gossip formulation for the PoS model can be referred to as "gossip-on-affluence", meaning a block qualifies gossiping as long as it origins from a rich stakeholder. The security stake is for punitive confiscation or destroy (in PoS programmers' jargon, "money slashing") when a block generator is found responsible for either having composed an erroneous block, or upon a timeout having not conducted its turn of blocking duty. A PoS version of "Ethereum2" [2] would be the most eminent PoW-to-PoS switch to take place in a near future of writing this paper.

The job of PoS block generation is easily done by signing a digital signature using a cryptographic (private) key. That of PoS block validation needs also producing a digital signature for proving a permissioned stakeholder's entitlement. Without these PoS proofs, the communication traffic for PoS blocking and/or validating would spam the network. Long term protection for, and frequent use of, a signature private key form single-point-of-failure (SPOF) challenges for each of the stakeholders. These SPOF points, although being distributed to the stakeholders, combine to centralized SPOF states with an amplified occurring frequency that a PoS blockchain is always in. If a private key can be worth very expensively for a stakeholder who very likely is a business organization investor in a PoS blockchain (venture loving organizations are the targeted participants for a PoS blockchain to recruit), the key has no value for an (in-organization) attacker who maybe managed to access the private key only for to cause disruption to the PoS blockchain. These frequency amplified and centralized SPOF breakdown scenarios for PoS blockchains are at best nothing-at-stake ones, a little bit of irony for PoS' naming.

There are other SPOF complications for a PoS blockchain. E.g., money slashing can hardly be done by a consensus algorithm due to a well-known academic conclusion, named "Fischer-Lynch-Paterson (FLP) Impossibility" [3], being the following statement. There can never exist a network distributed algorithm for its network distributed executors to input messages and output a consensus decision. If money slashing has ever been in practice by any PoS blockchain, it necessarily is by some unspecified manual operation. Still manual slashers would need to figure out an equitable exchange rate between an "error" and a money amount to slash. The quoted "error" here may well be an alleged one for the money slashers to take pains to discern.

1.1 Summary for Blockchain Consensus Layer Algorithms Quality

To this end a summary can be stated about the status quo qualities of the consensus layer algorithms for known public blockchains. These blocks lineup algorithms have so far only used small fractions of the blockchain participants as functional and dependable nodes. The remainder larger fractions of the blockchain participants have so far only been brutally syntax instructed or affluence stipulated by these algorithms to pass on diffusing messages which were supposed not understandable by the larger fractions of the participants as if they were not up to an intellectual capability. Low utilization of the blockchain participants has unfortunately wasted the collectively very strong capabilities that a majority of the blockchain participants can aggregate.

1.2 No Brutal Instruction for Modest Blockchain Participants

A blockchain should have the following property: A majority set of its participants should not be told, mandated, instructed, stipulated, manipulated, controlled, or even implicitly influenced, by the remainder set, i.e., a minority, of the participants to enter into a collective behavior. This blockchain property motivates intelligent uses of a majority of the blockchain participants. It requires careful algorithmic design. In fact, the PoW formulation of gossip-on-syntax, and the PoS stipulation of gossip-on-affluence, are two examples of brutal mandating a majority of the blockchain participants entering into collective behaviors. The PoW unresourceful nodes are algorithm designed to gossip syntax "correct" blocks, whereas the PoS proletarian nodes are rule stipulated to gossip stakeholders composed blocks. Both suffer undesirable consequences as a result of their unintelligent using a majority of the blockchain participants.

The present paper will take a "gossip-on-semantics" approach to using modest participants in public blockchains. Each modest blockchain participant upon receipting a gossip message will perform a calculation using its local information state and reach a semantically consistent decision on whether to update the local information state and forward the gossip message on, or to discard the gossip message.

Without further ado for now, it should be already self-evident that, due to the use of the local information state in the calculation of a semantics consistency decision, a modest blockchain participant can resist influence, manipulation, control, etc. from or by computationally powerful and/or monetarily affluent participants. As a matter of fact, the widely distributed and the vast number of semantics gossipers constitute a hop-by-hop firewall to guard against attacks mounted on a blockchain.

1.3 Organization of the Work Presentation

The remainder of this paper is organized as follows. In Section 2, a novel notion of blockchain as a randomized probabilistic algorithm is exposed. It can achieve a massive scale of utilization of the blockchain modest computers as semantics gossipers for truly honest majority control. In Section 3, a new randomized probabilistic algorithm

formulated blockchain is constructed with reasoning why, how and what it can benefit from utilizing the semantics gossipers. In Section 4, a system architecture level of generalization for blockchains is provided with a number of concrete ways for construction. Finally, this paper presentation concludes in Section 5 where a couple of questions regarding well-known blockchain attacks are asked for inviting answers and/or exploration discussions from interested readers.

2 Blockchain as a Randomized Probabilistic Algorithm

A well-known knowledge in the computational complexity theory can be described as follows. A randomized probabilistic (RP) algorithm needn't be sure about its output correctness in one instance of executing the algorithm. The algorithm's output correctness probability can be amplified quickly by repeating the execution.

Treating a blockchain as an RP algorithm, there is no need to be sure about trustworthiness of the payload data recorded in a block upon the block extending the blockchain. There is also no need to wait a long delay for that block to become trustworthy or be undone due to containing erroneous payload, as all known PoW blockchains have to slowly do. A blockchain no need of an uncertainly long wait for possible undoing an erroneous block can deterministically extend blocks even if some of them may have composed incorrect payloads. This blockchain determinism from the RP algorithm formulation can decouple a blockchain job, to be described in detail as follows.

The decoupling of the blockchain job is between extending the blockchain with newly found blocks, and amplifying the correctness probability for the payload logs in the already appended blocks. With the job decoupling, let only the logs in the untrusted payloads, which have survived semantics gossip be written to the blockchain DB (Algorithm to realize a semantics gossiper will be described in Section 3). Thus, any untrusted payload gets *deterministic* blockchain address, aka block height, right upon the payload composing block extending the blockchain.

Screening an untrusted payload and distilling semantics consistent logs to write to the blockchain DB provide lookup connections between the deterministic blockchain addresses and the logs' positions in the DB. With the deterministic connections, the DB content can be blockchain address indexed, and the DB can be address index partitioned into small files for fast input, output, lookup, insert, update, manage and the like standard DBMS operations. It is the very fact that a blockchain DB becomes a standard DBMS for fast operable even by modest computers that constitutes not only new knowledge advance, but also quality improvement for the public writing DBs of permissionless blockchains.

There is an obvious need of some blockchain nodes to initiate the above mentioned tasks: such as blockchain job decoupling, payload screening and distilling, correctness probability amplification, DB indexing, and DB index partitioning. So one question remains: Who and where are the blockchain deterministic addresses accessible computers?

2.1 Who and Where Are the Blockchain Servers?

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 networked computer. The Bitcoin scripting codes, and the Ethereum 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. Therefore Bitcoin and Ethereum blockchains can be regarded as von Neumann computers as servers. However, so far such blockchain addressable servers are only for solving user space problems, ones in the application layers of these blockchains. The blockchain issues having been analyzed in Section 1 are ones causing poor qualities with the kernel space, i.e., the consensus layer, of these popular blockchains. These kernel space problems are in urgent need of solutions.

GHOST (Greedy Heaviest Observed SubTree) [4] is a blockchain consensus layer algorithm to not only line up lucky blocks found by the PoW game won miners, but also let a lucky block record public key identities of some less lucky blocks' finders. These less lucky blocks' finders are dearly called *uncles* in Ethereum (the PoW version has implemented the GHOST algorithm). GHOST uncles can be blockchain addressable computers, though they have never been so used. Both the GHOST research work and its implementation by Ethereum have only used uncles for saving, otherwise wasted, PoW mining electricity.

A novel use of GHOST-like uncles is to let them be a set of No-Spam and No-SPOF servers and execute some useful operations for a blockchain. By controlling the blocking traffic not to reach a spam level as GHOST doing, the uncles for a random and No-Spam volume set of servers whose public-key identities have been exposed to the blockchain as global addresses.

In addition to knowledge contribution, this paper also serves the technical basis for a project to improve blockchain consensus layer algorithms. The project is to construct an RP formulated blockchain to attract easy permissionless participation, 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 computers is a fully functional semantics gossiper, an operator for a hop-by-hop semantics firewall, a capable manager for its local duplication copy of the blockchain distributed DBMS, and can take part in a set of No-Spam and No-SPOF servers for consensus computations.

2.2 Blockchain Having No-SPAM and No-SPOF Servers

In addition to initiating dissemination for blockchain payload screening and distilling, the following more generally useful properties are available from the blockchain addressable computers. They can form a set of No-Spam and No-SPOF servers to provide useful services for a blockchain.

The No-Spam property of such servers is easily achievable using a rarity algorithm, e.g., an easy and hence forky PoW hashing. The No-SPOF property of such servers can be achieved by duplicating them with sufficient redundancy, and assigning exclusive dissemination initiation entitlement to them. Let a blockchain record the addresses of such redundant servers in blockchain lineup sets, for each set to contain with redundancy a plural number of the servers. Let them speak orderly with the order being the global knowledge following the deterministic addresses of the blockchain.

Since the GHOST algorithm is a permissionless one, the GHOST uncles form permissionless, No-Spam, No-SPOF, servers providing useful services for and from blockchains. Thus, a permissionless blockchain enables a novel permissionless client-server architecture. Section 4 shall expose some useful applications for this architecture to enable.

2.3 Summary: RP Algorithm Formulation for Blockchain

Now for a summary on why, what and how the RP formulation for blockchain can achieve:

1. It is the insight that a blockchain can have an RP formulation, that lets the blockchain have deterministic addresses in any of its block extending state, even one having extended an erroneous block without a need to undo it.

2. It is the fact that a blockchain has deterministic addresses that enables job decoupling for the blockchain in that, appending blocks to the blockchain and writing data to the blockchain DB needn't be processed in the same time.

3. It is the blockchain job decoupling, permitting to write the blockchain DB with only logs screened and distilled from no-need-to-trust payloads in the already appended blocks, that can index partition the blockchain DB into a low-cost to construct and fast operable DBMS, even for modest computers.

4. It is the fact that a majority of the participants in a blockchain with each of them possessing a local copy of the blockchain's low-cost to construct and fast operable DBMS, that enables each of these blockchain participants being capable of making the local DBMS lookup supported semantics consistent decision, i.e., enables each of these blockchain participants being a capable semantics gossiper.

5. It is the fact that each of the overwhelmingly large fraction of the blockchain participants is a capable semantics gossiper, that they collectively constitute a pervasively distributed hop-by-hop firewall to filter out erroneous and/or malicious attacking logs in the no-need-to-trust payloads.

6. A permissionless blockchain can enable a novel notion of permissionless client-server architecture in which blockchain applications can have permissionless, No-Spam, No-SPOF servers to provide clients with useful services.

3 GoUncle: A Blockchain Of, By, For Modest Computers

Let the new blockchain of GoUncle use a version of the GHOST algorithm in its consensus layer. Let a GHOST lucky block compose inclusively 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 blockchain DB. The current block address, aka height, is said to be the *logging address* for these logs.
- `UNCLE`: Public-key identities of some less lucky block finders for whom the lucky block records as GHOST-like uncles. The current block address is said to be the *residing address* for `UNCLE`.
- `DB_ENTRIES`: Semantics gossip surviving logs announced by the uncles having 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 1 An `UNCLE` announces a `LOG` if the logging address of `LOG` is earlier than the residing address of `UNCLE`, and `UNCLE` judges that `LOG` is consistent with the blockchain's DB writing policy.

Here, for the exposition clarity, the description "`LOG` is consistent with the blockchain's DB writing policy" is purposely left unspecific. It suffices to understand that the judgment need to be made on the global range of the DB writing, beginning from the earliest history and up to the latest block composing `LOG` in its payload. Considering the case that the DB writing policy disallows duplication of `LOG` (e.g., the no-double-spending policy for the ledger application of the DB), then an implementation of the no-duplication policy can discard a duplicated `LOG` which has a later global address and/or has a later sequenced in-payload index.

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

Semantics Gossiper Algorithm 1 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 wide, 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 the GoUncle blockchain has following conditional probability:

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

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 the GoUncle blockchain also has following conditional probability:

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

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, the GoUncle blockchain establishes:

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

Here ZPP stands for Zero-sided-error Probabilistic Polynomial-time. The GoUncle 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_ENTRIES` in a block suffices the blockchain distributed DB to be influenced by the semantics consistent 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 consistent logs are written to the distributed DB earlier than they show up explicitly on the blockchain. Hence the blockchain and the distributed DB can be regarded as to 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.

4 A Permissionless Client-Server Architecture

Quick and easy screening and distilling payload logs into semantics consistent DBMS 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. Also as being a Software-as-a-Service (SaaS) system, a permissionless blockchain can use permissionless uncles to serve the need of some very useful applications. Thus, a novel notion of permissionless client-server architecture emerges.

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.

2. (Having the public-key ID being exposed to the global nodes in the system) Provide a TLS public-key certification authority (CA) server function for securing communications with peer neighbors, as originally proposed by the work of Cothority [5] (which impractically uses Bitcoin to mine its lineup blocks).

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 algorithm 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. Uncles have the exclusively assigned entitlement to announce this global clock's ticking state, in No-Spam and No-SPOF way, for the global participants to observe, e.g., a median of all uncles, and obey. Thus, the permissionless blocking traffic will reach a global state of quietness for definitive calculation of the lucky block.

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 a participant 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 another use of the uncles dissemination as a service in the fashion (2) above. Let the blockchain require a newly joining participant to bind its wallet identity public key to its TLS key with the peer-to-peer neighbors (the wallet key is different from the TLS communication 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. Also because the anonymously pre-registered TLS

key with the uncle servers can be used as "car registration number plates", algorithmic traffic calm scheme can be implemented and applied in case of the permissionless blocking traffic become too noisy.

Application layer uses of the uncles as No-Spam and No-SPOF servers can be as follows.

- The anonymous registration of the TLS keys with the uncles in the blockchain history letting these uncles play the role of TLS CAs for a binding wallet key. With a sufficient number of blockchain uncles available for having adequate servicing bandwidth, a binding wallet key obtains a very low-cost CA certification service. CA certificate and encryption protected peer-to-peer communications between modest computers without a middle man become a reality.
- 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 securely route-able for confidential and authentication peer-to-peer communications wherever the mobile device travels to and whatever its IP address has changed to.

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.

5 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, independent, and hence honest majority 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 indeed of, by, for modest computers. A novel permissionless client-server architecture is described to have very useful applications scenarios, with an exemplification of a Public-Key Defined Network application scenario.

Question 1: Let a majority of the participants in a blockchain be semantics gossipers. Can this blockchain accumulate a 51% Attacker?

Thought 1: 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.

Question 2: Can such a blockchain be attacked by Sybils?

Thought 2: With the semantics gossipers' hop-to-hop firewall widely dispersed and distributed in the network, concentration of Sybil nodes in one network location is seemingly not effective. On the other hand, if a Sybil attacker distributes Sybil nodes over the network, then is this attacker really a Sybil one?

References

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

[2] *THE ETH2 UPGRADES.* `https://ethereum.org/en/eth2/`.

[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`.