Geeq

John P. Conley
Vanderbilt University
May 2019

The Geeq Project Technical Paper

Version 2.0

Proof of Honesty: Coalition-Proof Blockchain Validation without Proof of Work or Stake

Proof of Honesty: Coalition-Proof Blockchain Validation without Proof of Work or Stake[1]

John P. Conley[2]
Vanderbilt University
May 2019

Version 2.0

Abstract

Blockchains are distributed, immutable, append only, ledgers designed to make trustless interactions between anonymous agents feasible and safe. The ledgers are maintained by networks of independent nodes who process transactions and come to a consensus view of which are valid and how these affect the ledger state. The integrity of blockchain ledgers therefore depends on the incentives contained in the consensus protocols that are designed to make the validating nodes behave honestly. In this paper, we argue that existing protocols based on Proof of Work, Proof of Authority, Proof of Stake, and so on, fundamentally offer relatively weak security guarantees. We propose a new blockchain validation protocol called Proof of Honesty (PoH). We show that PoH produces a blockchain that is 99% Byzantine Fault Tolerant, implements honest node behavior in coalition-proof equilibrium, and provides users with edge security which allows them to prove both the honesty and canonicalness of any fork or chain they see. In addition, PoH includes a fallback recovery procedure that ensures that an honest and canonical ledger will always exist and be accessible to users with at most a short delay even if all validating nodes on the network are dishonest.

---

1 This technical paper contains descriptions of proofs of the proprietary methods, patent pending. Proof of Honesty, PoH, Strategically Provable Security, SPS, Catastrophic Dissent Mechanism, CDM, Edge Security Protocol, ESP, GeeqCoin, GeeqChain, and Geeqosystem are all registered trademarks of The Geeq Corporation.

2 j.p.conley@vanderbilt.edu. Proof of Honesty is an element of a more general protocol developed for Geeq, a new infrastructure blockchain project in which the author is a participant. Part of this research was done while the author was on sabbatical at Microsoft Research over the 2016-2017 academic year. The author would like to thank Ric Asselstine, Darryl Patterson, Yorke Rhodes, Keshav Srinivasan, Stephanie So, Serge Sverdlov, Simon Wilkie, and Lun Shin Yuen for useful discussions about blockchain, applications, and ICT. The author takes sole responsibility for the contents of this paper.

1. Introduction

Blockchains are electronic ledgers that group transactions together into "blocks" and append them sequentially to an existing "chain." From a functional standpoint, blockchains are direct descendants of the paper ledgers using double-entry bookkeeping invented in the fifteenth century by the Milanese Franciscan monk, Luca Pacioli. Ledgers have a particularly simple data structure recording only account numbers/owners and current balances. They lack the meta and semantic web data of XML, the keys and relational table structures of SQL type databases, and even the columns and rows of spreadsheets.

Conventional databases are typically kept on a central server under the control of a single party. This party has the power to determine who has access to various parts of the database and as well as the ability to alter or delete records. This means the data is only as trustworthy as the party in control. In addition, since the data is kept in one place (or at least by a single entity), such databases have a central point of failure. As a result, state actors, courts, criminals, and hackers may be able to censor, alter, or even deny access to such data.

In contrast, blockchain ledgers are updated and stored by widely distributed sets of agents sometimes called nodes or miners. Transactions are sent by users to these nodes who communicate them to one another through peer-to-peer networks. Each node examines candidate transitions and decides if they are valid. The network of nodes as a whole comes to a consensus agreement and then uses the set of valid transactions to update the current ledger state. In computer science, blockchains are described as transition-state machines that use distributed ledger technology.

Although blockchains dramatically reduce the richness and utility of data, they offer several advantages. In particular, well-constructed blockchains allow users to interact and exchange value with each other in a trustless way and without requiring permission from any central authority. Blockchains can create a transaction record that is immutable in the sense that it would be computationally impractical to change historical data, as well as allowing users to make transactions while maintaining their anonymity.

All of the advantages that blockchains offer depend on honest transaction verification and block writing by the network of nodes. Blockchains use a number of different validation protocols to ensure this. Protocols lay out the set of rules that users and miners are supposed to follow and include incentive structures to make sure that they do.

In this paper we argue that existing protocols based on Proof of Work (PoW), Proof of Authority (PoA), Proof of Stake (PoS), and so on, fundamentally offer relatively weak security guarantees. We propose a new blockchain validation protocol called Proof of Honesty (PoH). We show PoH produces a blockchain that is 99% Byzantine Fault Tolerant (BFT)[3], implements honest node behavior

3 BFT comes from Lamport, Shostak, and Pease (1982) in which they describe "The Byzantine Generals Problem". To understand the idea, suppose that ten generals of the Byzantine empire are surrounding a city, some of whom have been bribed not to attack. Suppose the participation of at least seven is required in order to conquer the city. Otherwise, the attack fails. Such a battle plan is said to be 30% BFT. Most blockchain protocols have a BFT of 50% or less. More generally, BFT is a measure of how tolerant a system is to faulty components. Unfortunately, characterizing robustness in this way tends to make protocol designers think of nodes as parts of a system that either work as

in coalition-proof equilibrium, and provides users with edge security which allows them to prove both the honesty and canonicalness of any fork or chain they see. In addition, PoH includes a fall-back recovery procedure that ensures that an honest and canonical ledger will always exist and be accessible to users with at most a short delay even if all validating nodes on the network are dishonest.

2. Distributed Systems and Impossibility Theorems

Distributed data systems have traditionally tried to achieve three things:

Consistency: Every read receives the most recent write or an error.

Availability: Every request receives a (non-error) response, though not necessarily the most recent write.

Partition tolerance: The system continues to operate in the presence of network failure(s) that result in any number of messages being delayed or lost.

Unfortunately the CAP Theorem (Gilbert and Nancy 2002) which formalized Brewer's conjecture (Brewer 2000) tells us that a distributed data store can at best guarantee two out of three of these at one time.

It would also be desirable if a distributed data system could satisfy:

Termination: Every component will eventually decide on a value.

Safety: Different components will never decide on different values.

Unfortunately, the FLP Theorem (Fischer, Lynch, and Paterson 1985) tells us that both termination and safety cannot be satisfied in an asynchronous distributed system within a bounded time that is robust to the existence of at least one faulty component.

These requirements make sense in many contexts. For example, Visa-MasterCard needs each of its data servers to know the correct current state of the ledger. If two servers had different views of the current ledger because of partitions that prevented users from sharing a consistent, up-to-date view, or because of a safety failure that caused two nodes to decide on different values, double spending, incorrectly declined charges, and many other sorts of errors could arise. If a process failed to terminate or if coming to consensus across all servers took too long, approving or declining transactions correctly might be impossible or take an unreasonably long time to do with sufficient confidence.

---

expected or fail instead of as rational agents with preferences who are capable of doing either depending upon the circumstances.

We argue in this paper that these properties are not actually essential for distributed ledgers kept on blockchains to perform as they are intended. Having 10,000 Bitcoin nodes maintain consistent, correct, and available copies of a blockchains and ledgers is only a means to an end. What we really need from a blockchain is the availability of a ledger upon which users know that they can make correct, immutable, finalized transactions. There are three basic elements that ensure this:

Accessibility: There exists a correct copy of the ledger upon which non-partitioned users can transact.

Provable Honesty: Users with access to a ledger and its supporting transactions can prove whether it is honest or correct in the sense that it has always followed all protocols.

Provable Canonicalness: Users with access to a ledger and its supporting transactions can prove whether it is canonical in the sense that it is authoritative and no other version of the ledger and supporting transactions will ever supersede its authority.

Accessibility in this context does not mean that every node or process is able to provide a non-error response to any information request. It only requires that there exist at least one process, node, or ledger available to a non-partitioned user that gives any data request an up-to-date, non-error response.

Provability of honesty means that a user can distinguish between correct (by protocol) and incorrect responses. It does not matter if some nodes or processes fail safety or consistency as long as the user can tell that such responses are incorrect and can therefore be ignored.

Provability of canonicalness means that users who receive a response from a node or process can tell whether it is authoritative, immutable, and final. It does not matter if some processes or nodes are inconsistent, do not terminate, or fail to satisfy safety as long as users are able to distinguish the canonical ledger from those that are mutable or non-authoritative.

Note that honesty and canonicalness are completely different logical concepts. For example, a fork of a blockchain that follows all the rules of the protocol might be said to be honest. However, if the fork ends up being orphaned, then the transactions it contained were not canonical. Such transactions are effectively reversed and lose their validity despite being honest by protocol. On the other hand, stakeholders might vote unanimously to accept a block to be added to a chain. If this could not be reversed, then the transactions it contained would be canonical. However, the block might contain provably incorrect transactions despite the unanimous vote of the stakeholders. Thus, dishonest stakeholders may be able to confer canonicalness, (or at least authority) on incorrect transactions.

The key thing to notice here is that neither the CAP nor FLP Theorems imply that it is impossible to produce a distributed data store that has the three properties of accessibility, provable honesty, and provable canonicalness.

In this paper, we propose a new blockchain protocol called Proof of Honesty and show that it achieves all three. At a more detailed level, we provide a protocol that is:

- 99% Byzantine Fault Tolerant.
- Implements honest node behavior in coalition-proof equilibrium.
- Provides users with edge security which allows them to prove both the honesty and canonicalness of any fork.
- Includes a fallback recovery procedure that ensures that an honest and canonical ledger will always exist and be accessible to users with at most a short delay even under the worst of circumstances.

3. Honesty, Canonicalness, and Consensus

In this section, we show that existing blockchain protocols offer their users extremely weak security guarantees and that this is due to fundamental limits imposed by their consensus mechanisms.

We begin with some definitions.

Blockchain: We use this term to refer collectively to all blocks and the current ledger state up to some block height B.

Fork: A version or view of a blockchain at some block height B. One or more forks may exist.

Internal Honesty: A fork is internally honest if all of its blocks and the current ledger state follow all the rules laid down in the blockchain's protocol.

Global Honesty: A fork is globally honest if it is internally honest and there does not exist any other fork with content that implies the first fork is not honest under the blockchain's protocol.

Validity: A fork is valid if the blocks and current ledger state meet whatever criteria the protocol's consensus mechanism requires for blocks to be committed and ledgers to be updated.[4]

Canonicalness: A fork is canonical at some block height B if it is valid and any alternative fork of the same block height that may exist concurrently or in the future is invalid.

Determinism: A blockchain protocol is deterministic if given a ledger state at any block height B and a set of candidate transactions for inclusion in the next block, there is one and only one correct block and ledger state at block height B+1 under the protocol.

Provability: The honesty, validity, and canonicalness of a blockchain or fork of a blockchain are provable if an observer having full access to all information in the fork, but nothing more, can determine that it satisfies the conditions of honesty, validity, or canonicalness, respectively, as defined by the protocol.

Most implementations of PoW, PoA, PoA protocols share the same notion of internal honesty: a fork must follow all protocol rules. For the other three notions, protocols generally use some combination of the following elements:

4 Note that validity need not require that a fork be honest, in general, and honestly by itself does not necessarily imply that a fork is valid.

Correct Nonce (CN): Miners in PoW protocols compete to solve a cryptographic puzzle by brute force. The first to find the solution, called a "nonce," wins the right to propose a block. A block without a correct nonce is invalid.

Correct Hash Tree (CH): In almost every protocol, the hash of the previous block must be included in the current block. This recursive hashing creates a chain that makes tampering provable.

Longest Chain Rule (LC): In PoW protocols, the longest fork (that is, the fork with the largest block height) is canonical. More properly, any valid fork that is shorter than another is not canonical.

Correct Consensus Rule (CC): In PoA and PoS protocols, the correct approval of new blocks to be appended to a chain requires that various consensus rules to be satisfied.[5]

Correct Voting (CV): In PoA and PoS protocols, various conditions determine whether votes are valid and how much weight they carry. In particular, most protocols require that nodes vote for at most one proposed block at any given block height.

Unique Validity (UV): A fork satisfies unique validity if it is the only fork that currently does, or ever will, satisfy the conditions for validity at a given block height.

In basic PoA protocols, a simple majority or a two thirds majority of a fixed and known set of nodes cryptographically sign votes to signal that they agree that a proposed block should be accepted. Double voting is generally considered dishonest behavior. Examples include VeChain and Hyperledger Fabric and have roots in Castro and Liskov's 1999 work on Practical Byzantine Fault Tolerance (pBFT).[6]

In simple PoS protocols, users have voting power in proportion to the total share of the coinbase they own. In others, users gain voting power only by escrowing tokens in time-locked security accounts. This is to reduce the effectiveness of so called "nothing at stake" attacks. In some more complex PoS protocols voting is limited to the 100 or so nodes who have staked or hold the largest number of tokens. In others, stakeholders can delegate their tokens to nodes that they trust and have them vote on their behalf (this is called Delegated Proof of Stake or dPoS (Larimer 2014). In some versions of dPoS, delegators are paid a share of any transactions fees or other rewards received by the voter to whom they have delegated their tokens. In still other cases, voting power may be affected by proofs of liveness, activity levels, reputation, longevity and so on. The commonality

5 In computer science, consensus is a protocol independent notion defined as the combination of termination, integrity, and agreement (see https://en.wikipedia.org/wiki/Consensus_(computer_science), for example). Correct Consensus means something different here. Specifically, it refers to protocol specific requirements contained in a block or fork that indicate consent or agreement by the required number of nodes and/or other agents (and possibly the satisfaction of even more exotic rules). My thanks to Keshav Srinivasan for pointing this out.

6 Note that pBFT consensus systems generally select a leader or block proposer in a round robin or random way who can be replaced by a majority or supermajority vote on the part of a set of nodes. In this sense, all nodes have an equally weighted vote on whether a new block is valid and should be committed to the chain. Many PoS and dPoS consensus systems are similar in that the probability of being chosen as the block proposer is proportional to the fraction of the coinbase owned, delegated to, or actively staked by a user for the purposes of validating blocks.

in all these protocols is that correct voting and knowing what the correct majority is depends on knowing the set of eligible voters and how much voting weight they have. Examples include EOS, NEM, Dash, and Ethereum's forthcoming Casper protocol (Buterin and Griffith. 2017).

The table below shows how protocols typically define these properties.

Type	Internal Honesty	Global Honesty	Validity	Canonicalness
PoW	Protocol	IH	IH + CN + LC	UV
PoA PoS dPoS	Protocol	IH or IH + CC or IH + CC + CV	CH + CC or CH + CC + CV or IH + CH + CC + CV	UV

IH = Internal Honesty, CC = Correct Consensus, CV = Correct Voting
CN = Correct Nonce, LC = Longest Chain Rule, CH = Correct Hash Tree, UV = Unique Validity

Table 1

Observation 1**:** Assume that a protocol is deterministic. Then internal honesty is provable.

All well-constructed blockchain protocols are deterministic. This means that if one has access to the protocol rules or code and all the information in a given fork, it is possible to verify that its blocks and ledgers are consistent and correctly derived. Of course, several forks could be honest but mutually inconsistent if they have correctly built blocks with different sets of transactions.

Observation 2: PoW protocols are provably globally honest if this is defined as internal honesty alone.

Observation 3: If a protocol is deterministic and global honesty is defined as internal honesty plus correct consensus, then global honesty is provable only if a fork allows an observer to prove that consensus was correctly established.

For example, if all blocks in a fork contain the signatures of a sufficient majority of nodes then correct consensus is satisfied. Unfortunately, there may be several such forks if voters signed multiple blocks at a given height. This means that correct consensus is not a very useful definition of global honesty.

Observation 4: It is not possible to prove correct voting in most PoS, and PoA protocols, and therefore, global honesty is also not provable.[7]

7 The form of this statement is not precise in that it uses the word “most”. This is because it is not immediately clear how one could usefully define the whole class of PoW and PoS protocols in a way that would permit one to say “all” instead. For example, one could imagine a voting system that somehow took advantage of quantum entanglement so that if a stakeholder voted yes on one block, the entangled quanta would always record a no vote on all other forks. While no such technology is available now and I am unaware of protocols that somehow solve this and similar problems in a way that would allow one to prove canonicalness, it is not logically impossible that such approaches will be invented in the future.

Double voting is an example of an action that usually requires access to data on more than one fork in order to be provable. Correct majority may also depend on the behavior of nodes or voters on other forks. For example, a voter might violate protocol and have its stake or voting privileges removed on one fork which would render it ineligible to participate in a vote on any fork by protocol.

Observation 5: Correct consensus is provable in most PoS protocols

If validity depends only on counting or weighing the number of votes for each block in a fork, then it may be provable under some PoS and PoA protocols. However, many forks might qualify as valid in this case.

Observation 6: Correct voting is not provable in most PoS and PoA protocols.

As pointed out above, double voting and slashing on other forks may disqualify voters or reduce their weight. In addition, unless a protocol is carefully designed, proving which nodes are allowed to vote, who is allowed to propose the next block, which users have staked tokens and how much, may be complex and error prone.

Observation 7: Most PoW protocols are not provably canonical.

The longest fork rule is fundamentally relativistic. Proving a fork is longest requires proving that one is comparing the length of all valid forks that exist. There is no way of which we are aware to prove that all forks are visible to an observer. Even if an observer could be aware of all existing forks, it would still be impossible to prove that no other fork will ever become longer than the one that is currently in the lead.

Observation 8: Most PoS and PoA protocols are not provably canonical.

In every PoS and PoA protocol of which we are aware:

- Internal honesty and correct consensus can be satisfied by more than one fork at a given block height.
- Correct voting cannot be proven unless one at least knows how nodes or stakeholders voted on all other forks.
- It can never be proven by looking at the data contained in one fork of a given block height that other forks of the same height satisfying correct consensus and correct voting do not and will never exist in the future.

One might object that the arguments above simply say that if 51% of miners or 67% of stakeholders are dishonest, then security of the ledger is lost. Although there are many attacks that do not require this degree of dishonesty (see Eyal and Sirer 2014 and Houy 2014, for example), majority attacks are, in fact, a significant problem.[8]

In the case of PoW, a controlling majority of the hashing power can be created for a short time with proportionally small costs. Many actors have access to enough hardware directly (the NSA, Russia, Visa-MasterCard) or indirectly (renting VMs from cloud service providers or building bot-

8 Bonneau (2018) estimates that it would cost $1M per hour to rent enough capacity on EC2 to mount a 51% attack on Ethereum, and $1.5B to purchase enough capacity to mount such an attack on Bitcoin. Given that Bitcoin's token cap is approximately $140B at this writing, this number does not seem very high. Other authors have placed the cost of rental attack on smaller PoW blockchains such as EthereumClassic, Monero, and Dash at less than $10k per hour (see https://www.crypto51.app/). A number of such attacks have actually occurred on such chains.

nets) to take control of a chain for a few days, after which the damage is done (see Bonneau et al. 2015, for example). Bitcoin and Ethereum are both vulnerable, and smaller PoW chains even more so.[9] This aside, three or fewer mining pools already control a majority of Bitcoin's hashing power. It therefore only takes the agreement of (or a government mandate imposed on) three agents to mount a 51% attack.

In the case of PoS and dPoS, majority attacks may be even less costly. For example, suppose that only a fraction of the tokens on the platform are staked for the purposes of voting while the rest stay in circulation, or that users delegate their tokens to small subset of voting nodes. It might be that only 10% of the tokens were locked up for voting, or that the top 100 nodes who ended up as voters only held 10% of the coinbase in total. In these cases, 6.7% of the coinbase completely controls all the other tokens on the ledger as well as any smart contracts or tokenized assets on the platform. If the dishonest voters can convince others to delegate their stakes by making above-market profit sharing arrangements, they might acquire the needed majority with even less cost and risk.

The point is that it is impossible under existing protocols for users or other observers to prove the canonicalness of a fork. The protocols simply do not allow it. In PoS and PoA protocols, users cannot even prove the global honesty of a fork. In all such protocols, proof relies on something that is unknowable: the current or future existence of other valid forks with a superior claim.

Users must therefore rely on the honesty of the central network of validating nodes, miners, and stakeholders. Users can never know if the fork they are seeing is canonical and have no way to prevent rollbacks of history. Even if users could prove that the majority of miners and validators were dishonest, they must rely on the network to police and punish itself.

4. Preliminaries

The Geeq Project develops an approach to DLT that produces a blockchain that is provably globally honest and canonical using a consensus protocol called Proof of Honesty (PoH) based on four elements:

Geeq Workflow Protocol (GWP): This includes standard data item formats for transactions, blocks, etc., communications protocols, and rules that govern the validity of transactions, smart contracts, ledger updates and so on. The workflow protocol is deterministic in the sense defined above. We show below that the GWP is 99% Byzantine Fault Tolerant (BFT).

9 Even though a dishonest chain is invalid by protocol, there are reasonable equilibria in which both users and miners ignore this fact and accept the dishonest fork as authoritative. For example, suppose that a block was dishonest in that it gave a mining reward greater than allowed by protocol. Suppose that each miner believed that the other miners would accept such blocks. Then rejecting the block would mean that the miner would have to append any of his own blocks to a short chain that would likely be orphaned. What would users do? They could walk away from their tokens or hope that other users will accept transactions as valid on a short but honest chain that has blocks added very slowly. Alternatively, users may see the writing of greater rewards as a "misdemeanor" a type of dishonesty that does them very little material harm and so continue to transact on the dishonest chain. In effect, miners have simply demanded higher wages, but are otherwise honest. Users may decide to accept this despite protocol and token value may not suffer.

Catastrophic Dissent Mechanism (CDM): The CDM is a game theoretic mechanism that implements honesty on the part of validating nodes in coalition-proof equilibrium through a system of audits. This gives PoH Strategically Provable Security (SPS) in the sense that honest validation and following protocol gives nodes a higher payoff than any alternative strategy available to any coalition of validators including the grand coalition.

Catastrophic Recovery Mechanism (CRM): The CRM is a fall back mechanism used in the event that 100% of the validating nodes are dishonest despite the CDM, to reestablish an honest, canonical blockchain and continue block writing.

Edge Security Protocol (ESP): The ESP makes users rather than nodes the ultimate arbiters of the correctness of transactions, the blockchain, and the ledger state. We show below this allows users to avoid creating and accepting transactions on dishonest or noncanonical forks, prevents the theft of tokens regardless of the level of dishonesty on the part of validators and enables recovery even when 100% of the validating network is dishonest.

The Geeq Project builds an ecosystem of federated, interoperable, blockchains. Each instance of a GeeqChain has a validation layer with its own blockchain, and a separate application layer which may be customized to the requirements of the use case. Each instance also has its own independent set of validating nodes. We will not develop this architecture further in the current paper. Instead we will focus on the consensus protocol and its properties on a single isolated GeeqChain instance. The following subsections describe the actors, give definitions, and outline the GWP in detail.

4.1. Actors

The Geeq Workflow Protocol (GWP) is supported by four major categories of actors:

Users: Users are simply account holders on the various instances of GeeqChain. Each separate account is distinguished by a private key. Of course, real world agents may have many such accounts, but they are logically separate users to the protocol.

Nodes: Nodes are actors who have registered to serve as validators for a given instance of a GeeqChain. Different instances may also support applications and have slightly different validation rules. Nodes upload the application and validation layer code from the Genesis Block (GB) of the instance they wish to validate and add themselves to the Active Node List (ANL), both defined below.

Physically, nodes are computers or virtual machines attached to the internet that communicate with users and other nodes. Of course, behind these nodes are real people who pay for the machines and resources required, and who may wish to dishonestly manipulate the chain to steal tokens from users or for some other purpose. It is possible that many nodes may be owned by a single person. Such nodes are called “Sybils” since they can be coordinated to act jointly in the interests of their owner. This is not desirable, but there is no practical means of detecting Sybils and so the consensus algorithm must be robust to their presence. Each instance of GeeqChain is validated by N nodes (N may vary between instances) that are identified by public keys and IP addresses.

Hubs: Hubs are nodes that are tasked with coordinating the building of blocks on the chain. Nodes in the ANL take on the role of hub in a predetermined order.

Relays: Relays are similar to nodes in that they are machines set up by real people to help run Geeq's protocol. They have a very limited role, however. Relays simply certify that the correct hub is coordinating the building of a given block. In the unlikely event that all nodes are dishonest and prevent new transactions from entering the blockchain, relays initiate the Catastrophic Recovery Mechanism (CRM).

Relays do not validate transactions, propose or build blocks, or maintain the current ledger state. In this outline, we will consider the case with a single relay. In more advanced implementations, many relays will rotate checking various instances of GeeqChain. Relays are also identified by public keys and IP address.[10]

4.2. Definitions

This subsection defines the basic elements needed to define the workflow. We will not describe the format or other details of any data items in this paper.

Active Node List (ANL): A list maintained in the ledger of each chain with a record of the public key account, IP address, and current status of each node that participates in validating a given chain. The ANL also contains the public key account and IP address of the relay. Anonymous agents join, leave and suspend their membership in the ANL by submitting the appropriate transaction to an existing node. Nodes may also be placed on suspension or audited out of the ANL for certain kinds of bad behavior.

Good Behavior Bond (GBB): To join the ANL, an agent must have an account on the chain and must transfer a certain number of $GEEQs or gqs (gq, pronounced "geek", is the name of the token used in the Geeq Project's ecosystem) to a system account in the form of a GBB. If the node behaves badly or dishonestly, some or all of the GBB is confiscated (details are discussed in sections below). If a node wishes to leave the ANL, then the GBB is returned to his account after a certain delay.

Network Size: Each instance of GeeqChain has a parametric target number of nodes in its validation network denoted **N.** If the ANL includes a larger number of active nodes, a selection is made for each epoch, defined below. If the ANL has fewer than N members, then all nodes participate in validation each epoch.

10 Note that this means that relays are not trusted participants with ability to affect the content of blocks or the validity of transactions. The Geeq protocol uses a kind of "separation of powers" approach in which nodes individually offer users their view of a correct blockchain and ledger. The relay provides an endorsement to make a single fork canonical (but not necessarily correct), and each user ultimately determines for himself if the fork he is looking at is both correct and canonical. We discuss these elements at more length in subsequent sections.

Epoch: Each instance of a GeeqChain attempts to build and commit a block at every block time interval, ten seconds by default. Each block B is numbered successively as it is committed. Blocks are also grouped in epochs of size N, after which nodes may be rotated or changed. Thus, for block 1 to block N, one set of nodes is in charge of validation, for blocks N+1 to 2N, another set, and so on.

Hub Block Order (HBO): In a given epoch, a selection and ordering of nodes from the ANL is created. This will determine which node is responsible for proposing candidate transactions for each block in the epoch. If the ANL is too small, nodes may serve as hub more than once in an epoch. What is key is that the hub order is deterministic and can be derived from information in a given instance of a blockchain. All nodes in the ANL are required to stake GBBs and keep copies of all blocks and the CLS, defined below, even if they are not in the HBO for the current epoch.

Unverified User Transactions (UUT): Users submit transactions requests to the node of their choosing. These may be standard token transfer requests, requests to move tokens to another instance of a GeeqChain, requests to join or leave the ANL, requests for transactions on the application layer, and so on.

Verified User Transactions (VUT): Each node separately verifies the validity of each UUT and creates a corresponding VUT for inclusion in the next block if it passes.

Verified Node Transactions (VNT): Nodes also generate a number of transactions internally that are automatically valid in the eyes of their creators. These include fee payments from users, fee distributions to nodes, audit transactions, etc.

Sanity: Each UUT received by a node is checked to make sure it is plausibly valid before it is passed to other nodes. For a UUT to pass, it must be correctly formatted, the paying account must exist on the ledger, and it must contain enough of a balance to cover the transaction and associated fees.

Node Transaction Bundle (NTB): For each block, each node in the HBO collects UUTs from users, checks them for sanity, bundles them together, and sends them to the current hub as an NTB.

Hub Transaction Bundle (HTB): For each block, the current hub collects NTBs from the rest of the active nodes in the HBO and then bundles them together. The hub does not include any NTBs that do not have correct signatures, but does include an NTB of its own. The hub then signs the HTB and sends it to the relay.

Relay Transaction Bundle (RTB): Each block has a uniquely determined hub and relay pair that are allowed to propose an HTB. (In this version of the protocol, there is only one relay.) The hub constructs and signs the HTB and sends it to the relay. The relay checks to make sure the correct hub has signed and then may add some items (described later). The relay then signs this HTB transforming it to an RTB. Finally, the relay sends the RTB to every node in the ANL.

Note this includes nodes that are not in the current HBO since such nodes must be in sync if and when they join the HBO in future epochs.

Block: Upon receiving a correctly signed RTB from the relay, each node checks for the validity of each transaction it contains. Each node also checks that all the other nodes, the hub, and the relay are behaving honestly and according to protocol. Given its evaluation of the RTB, each node builds a block with whatever valid user and node transactions it thinks appropriate and then commits it to its own version of the chain. Given that synced nodes start with the same CLS and process the same RTB using the same deterministic protocol code, all honest nodes must commit the same block and update the ledger in the same way.

Current Ledger State (CLS): Each node maintains a ledger containing account records and other data. The transactions validated by each node and committed to each block are used to update the ledger state.

Empty Block (EB): It may be that the current hub is off-line, partitioned, or refuses to send an HTB to the relay. In this event, the relay sends each node a signed empty block to be committed instead of an RTB derived from a signed HTB.

Genesis Block (GB): Block 0 of any instance of a GeeqChain is the genesis block and has a number of special features. To begin with, it contains an initial ANL and initial ledger state (usually empty). More importantly, it includes a copy of the exact code that nodes are supposed to use to validate transactions and run any application that is built on the instance. This code is signed with the known private key of The Geeq Corporation. This allows users and others to verify whether any fork of the chain they can see is correct and honest in every respect.

Synced: A node is synced at block B if it has correctly signed blocks committed to its version of the chain up to block B-1.

Stalled: A node is stalled at block B if it is not synced. Under protocol, a stalled node is not allowed to add any blocks to its chain unless and until it can resync.

4.3. The Geeq Workflow Protocol - Overview

The Geeq Workflow Protocol (GWP) requires nodes, hubs, and relays to communicate with each other. Failure to do so breaks protocol, but is not considered dishonest. This is because such failures may be due to nodes dropping offline, network failures or latency, network partitioning, as well as strategic decisions by nodes not to communicate. Since the cause cannot be determined, such behavior cannot be proven to be the result of dishonesty. Any failure to act or communicate at a specific time or within a specific time interval are similarly ignored from the standpoint of honesty. Below we give a high level outline of the GWP.

Overview of the Geeq Workflow Protocol

1. Nodes resync if they are out-of-sync.
2. Hub for block B waits 5 seconds after it commits block B-1 as UUTs and NTBs arrive.
3. Hub creates its own NTB.
4. Hub creates an HTB from the NTBs above.
5. Hub sends the HTB to the relay for endorsement.
6. Relay sends a signed RTB or a signed EB back to each node.
7. Nodes try to resync if the RTB does not arrive.
8. Nodes that receive a correct RTB/EB check the RTB, HTB, and NTB headers for honesty.
9. Nodes verify the UUTs in the HTB.
10. Nodes update the CLS.
11. Nodes create and commit block B.

Blocks 1 through 100 are bootstrapping blocks. For block 1, all nodes wait 10 seconds for UUTs to arrive and then send them to the hub (that is, nodes start at step 11). The hub starts at step 1 which means it waits 15 seconds for all the NTBs to arrive. For the first hundred blocks, no fees are charged. The only transactions that are allowed are incoming $GEEQ transfers from other chains.[11] This is to fund GBBs and also to let users set up accounts to pay transaction fees. The size of incoming payments may be limited, and no outgoing payments are allowed. Any nodes that fail to fund their GBBs are audited out of the ANL in block 101.

5. The Geeq Workflow Protocol

In this section, we describe the GWP in detail and show that it gives PoH 99% BFT.

Detailed View of the Geeq Workflow Protocol with One Relay

1. Nodes resync if they are out of sync. All nodes know that blocks are written approximately every 10 seconds. If 15 seconds by a node's own clock have passed since the last RTB or EB has arrived from the relay, the node assumes it is out-of-sync. It tries to resync as follows:

   a. First, the node sends an out-of-sync message to the relay. It may be that the block is late due to audits, other out-of-sync nodes, etc. In this case, the relay tells the node to wait some length of time. Otherwise, the relay sends the node the missing RTBs or EBs.

11 We will not describe the workings of interchain transfers in this paper.

b. Second, if the node does not receive any response from the relay, the node sends messages to other nodes in succession attempting to obtain the missing RTBs. If it gets them, it is resynced.

c. Third, if both of these efforts fail, the node is out-of-sync. The node may be partitioned or the rest of the network may be ignoring it. The node continues to attempt the previous two steps every 10 seconds.

2. Hub for block B waits 5 seconds after it commits block B-1. All hubs know the HBO and so a node will be aware whether it is the hub for block B. Once it commits block B-1, the proper node takes on the hub role and waits for 5 seconds by its own clock to allow UUTs from users to include in its own NTB, and NTBs from other nodes to arrive.

3. Hub creates its own NTB. After 5 seconds, the hub puts any UUTs that have passed the sanity test in its own NTB (the hub acts as both hub and node).

4. Hub creates an HTB from the NTBs above. The hub puts any correctly signed NTBs that have arrived from other nodes in the HBO for the current epoch into the HTB along with its own. If two or more correctly signed NTBs arrive from a single node for block B, they are all included in the HTB since this proves that the node is dishonest. The hub does not check any NTBs for correctness or honesty otherwise. Those checks are done separately by each of the nodes starting at step 8.

5. Hub sends the HTB to the relay for endorsement. The hub signs the HTB and then sends it to the relay. The HTB might or might not arrive due to network failures, and could be ignored by the relay if the relay is dishonest.

6. Relay sends a signed RTB or a signed EB back to each node. The relay knows the last block it endorsed and so knows which is the correct hub for the next block.

   a. The relay waits 10 seconds for a correctly signed HTB to arrive from the correct hub for the next block.

   b. During this time, the relay also accepts audit, ANL, and ordinary transactions from users, though at a higher fee than charged by nodes. The relay does not validate transactions itself, but this allows users an additional avenue to get transactions included in the RTB in the event that nodes attempt to censor certain users.

   c. If the correct HTB does arrive, the relay validates any audit transactions it contains as well as audit transactions it received directly from users. The relay uses this to update its own view of the ANL and the HBO for the rest of the epoch. The relay does not validate any ANL or ordinary (non-audit) user transactions, and does not update the ledger state or keep a copy of the blockchain. Those tasks are reserved exclusively for the nodes.

   d. The relay then creates an NTB containing all the transactions it has received directly from users, attaches it to the end of the signed RTB (not within the HTB), and then signs the whole

thing. This allows nodes receiving the signed bundle to verify that it is signed by the correct hub and relay and also prevents them from selectively pretending they never received, (and thus pretend they could not process) transactions in the relay's NTB.

e. The relay then sends the signed RTB out to each node listed in its view of the ANL.

f. If the HTB fails to arrive from the correct hub after the 10 second delay in step (6a), the relay sends a message to each node in the HBO telling them to wait another 10 seconds. If the correct HTB still has not arrived at the end of an additional 5 seconds, however, the relay assumes that the hub is disconnected or is choosing to be silent. It then sends out a signed EB. This means that there is one and only one correct RTB or EB for the construction of block B, and it must be included in every canonical blockchain.

7. Nodes try to resync if the RTB does not arrive. Nodes either receive the correctly signed RTB/EB from the relay and move to step 8, or become out-of-sync and stalled, in which case they attempt to resync as in step 1.

8. Nodes that receive a correct RTB/EB check the RTB, HTB, and NTB headers for honesty. This is described separately. If dishonest nodes are discovered, an audit transaction is written. The NTBs for dishonest nodes are ignored when the CLS for block B is written.

9. Nodes verify the UUTs in the RTB. The UUTs in the correctly signed NTBs are evaluated for individual and joint validity under protocol and invalid transactions are discarded.

10. Nodes update the CLS. Valid UUTs are reformatted to VUTs and any necessary VNTs are created. These are collectively used to update the CLS under protocol by each node.

11. Nodes create and commit block B. The VUTs created above are placed in the correct order, the hash of block B-1, the hash of the newly derived Block B CLS, other header elements are created, and the block is constructed and committed to the chain maintained by each node.

Provided that the relay is honest, the GWP prevents the blockchain from stalling. In a technical sense, if the relay is honest then each block writing round terminates. If the relay does not get a correctly signed HTB within a certain time interval, it issues a signed EB. Of course, this is not very useful since no transactions are added to committed blocks in the case of an EB. In Section 6, we define the Catastrophic Recovery Protocol (CRP) and show that it guarantees the transaction processing will not stall even if all the nodes in the ANL are dishonest or non-responsive.

Claim 1: *If the relay is honest and Geeq's public key is known, then GWP produces at most one provably canonical and provably globally honest blockchain.*

Proof: Suppose that a fork is honest at block B. Since the Geeq protocol is deterministic, an observer can derive which node must serve as hub for block B+1. The RTB in Block B+1 must therefore be signed by both the proper hub and relay or else the fork is provably dishonest and noncanonical. If the RTB is correctly signed, an observer can use the protocol code to back out the CLS for block B by subtracting the valid transactions from the CLS for block B+1. The hash of the block B CLS can then be taken and compared to the hash of the block B CLS con-

tained in block B. If it matches, then the CLS was correctly updated with the correct set of valid transactions in block B+1. In this case, the fork is valid. In all other cases the block is invalid and therefore also not canonical. Thus, if an observer knows that a fork is honest and canonical at block B, he can prove whether it is honest and canonical at block B+1.

Block 0 is the genesis block and is honest and canonical if and only if it is signed with Geeq's private key. This is provable for any observer who knows Geeq's public key. Then by induction, any observer who knows Geeq's public key can prove whether the fork he observes is honest and canonical. Since only one fork can ever be canonical by definition, either one or no canonical and honest fork for any given block height B can exist. ■

Intuitively, if the relay is honest, it produces either a signed RTB which is also correctly signed by the correct hub, or a signed EB for block B. No other HTB or EB block exists that is signed by the relay. Thus, if a chain is honest at block B, then there is a unique set of provably honest blocks that make up the chain, and therefore only one honest ledger state.

The next claim says that if the relay is honest, then only one honest node is needed for the GWP to work.

Claim 2: *Assume that the relay is honest and at least one node is honest and able to communicate with the relay. Then one and only one honest, canonical fork will exist and transaction writing will not stall.*

Proof: Hub order is deterministic and known to the relay. Each hub, honest or not, must either send a correctly signed HTB to the relay which the relay signs and sends out to each node, or be silent, in which case the relay sends out a signed EB. In turn, all nodes in the HBO must use the signed RTB/EB to build each successive block, use something else and thereby create a provably dishonest and noncanonical fork instead, or use nothing at all and be stalled

Dishonest nodes may ignore users and not forward, as they should, UUTs in their NTBs, and dishonest hubs may ignore honest nodes and not include their NTBs in the HTBs sent to the relay. Nevertheless, the relay always sends either a signed RTB or EB to all nodes. Honest nodes use it to build an honest block and are able to stay synced.

It is impossible for nodes to write a false audit against an honest node. Proving a node is dishonest relies on the contents of the NTB signed by the node. Since the node is the only agent who knows its private key, dishonest hubs and nodes are not able to alter any node's NTB and then sign it correctly. Thus, provided that a node is honest, the relay will be able to confirm this fact using the correctly signed NTB and reject the audit as false. This prevents dishonest hubs and nodes from fraudulently removing honest nodes for the ANL or HBO.

Putting this together, a series of dishonest hubs in the HBO have only two dishonest options open to them. First, they can create a fork using RTBs or EBs that are processed dishonestly or incorrectly signed. Second, they can be silent or claim to be stalled and thereby not process any UUTs. The only other alternative is to be honest.

If dishonest hubs do the first, then users will be able to prove that their forks are dishonest. Dishonest hubs can follow the second strategy, but only until it is the honest node's turn to be the

hub and build the HTB. The honest node then provides a pathway for users to get UUTs into the HTB. Thus, honest blocks with committed transactions will continue to be written, although with a delay.

We conclude that if there is at least one honest node in communication with the relay, it will construct a canonical and honest fork and will allow users to add transactions to committed blocks. ■

99% Byzantine Fault Tolerance: A blockchain protocol is 99% BFT if it produces a provably globally honest and canonical fork and builds non-trivial blocks provided that there is at least one honest node that follows protocol and is able to communicate[12]

Theorem 1: *Assume that the relay is honest and at least one node is honest and in communication Then the GWP protocol is 99% BFT.*

Proof: Immediate from Claim 2. ■

The reader may notice that we have not explicitly discussed the partition resistance of the GWP protocol. Suppose that a network partition occurs. The relay will be a member of one of these partitions, and for nodes and users in the same partition, block building and transactions processing proceeds essentially as it normally does. The only difference is that when a node not within the partition is supposed to take the role of hub, the relay will be forced to issue an EB.

Nodes, in partitions that do not contain the relay will not be able to contact the relay and will immediately become out-of-sync. Users in partitions that do not include the relay will not be able find a canonical fork of the blockchain (that is able to process transactions) until the partition is resolved.

Not being able to execute transactions is an inconvenience, but the benefit is that it prevents partitioned users from inadvertently accepting token payments that may be double spends and which would become invalid in other protocols. For example, partitioned Bitcoin users might see their transaction buried six blocks deep in the longest chain of which they are aware only to find that there is a longer chain being build by nodes in the other partition.

If blockchain is to realize its potential as a technology that can be trusted to manage land titles, stock ownership, identity, sensitive or important records, and critical infrastructure, it must be absolutely secure. A protocol that can be brought down by an adversary with a 51% attack is simply not good enough. While in practice, partitions happen, they are not extremely common, wide-spread, or long in duration. Thus, while the decision to focus of security may result in temporary costs, we argue that the inconvenience experienced by some during partitions seems modest compared to the benefits.

12 We call this property 99% BFT for convenience. A more accurate name would be $(N-1)/N$ BFT where N is the number of nodes in the validating network.

6. Catastrophic Dissent Mechanism

Byzantine Fault Tolerance is the standard metric of blockchain security, but is ultimately not a good way of thinking about how nodes behave. Nodes on a validation network are not automata that innately work correctly or incorrectly, honestly or dishonestly. Nodes are the agents of the human beings who program and run them. Thus, any meaningful security measure must encompass the motivations and optimal responses of these humans given whatever they might believe about the behavior of other users and validators on the system. In other words, to be meaningful, a security measure should have a game theoretic foundation.

We begin with a formal definition of the three elements of a game:

Agents: $n \in \{1, \ldots, N\} \equiv \mathcal{N}$

Strategies: $v_n \in \mathbb{R}_+^1 \quad \forall\, n \in \mathcal{N}$

Payoff Function: $F_n : \mathbb{R}_+^N \rightarrow \mathbb{R}^1 \quad \forall\, n \in \mathcal{N}$

Thus, when the set of agents $\mathcal{N}$ playing the game choose a *strategy profile* $v \equiv \{v_1, \ldots, v_N\} \in \mathbb{R}_+^N$, any given agent $n \in \mathcal{N}$ receives a payoff of $F_n(v) \in \mathbb{R}^1$.

To define our equilibrium notion we will need some additional definitions and notation:

Coalition: $c \subseteq \mathcal{N}$

Deviation Strategy Profile: Consider a strategy profile $v \in \mathbb{R}^N$ for the grand coalition $\mathcal{N}$ and any coalition $c \subseteq \mathcal{N}$. Then, $\hat{v} \in \mathbb{R}_+^N$ is a deviation strategy profile from v for the coalition c if for all $n \notin c$, $\hat{v}_n = v_n$.

Credible Deviation: Let $\hat{v} \in \mathbb{R}_+^N$ be any deviation strategy profile from v for the coalition c. Then $\hat{v} \in \mathbb{R}_+^N$ is a credible deviation if for all subcoalitions $c' \subseteq c$ of the deviating coalition, there does not exist a deviation strategy profile $\tilde{v} \in \mathbb{R}^N$ for coalition c' from $\hat{v} \in \mathbb{R}^N$ such that for all $n \in c'$, $F_n(\tilde{v}) > F_n(\hat{v})$.

This allows us to define the following:

Coalition-Proof Equilibrium (CPE): A strategy profile $v \in \mathbb{R}^N$ is a coalition-proof equilibrium if for all coalitions $c \subseteq \mathcal{N}$ there does not exist a credible deviation $\hat{v} \in \mathbb{R}^N$ such that $F_n(\hat{v}) \geq F_n(v)$ for every $n \in c$ and there exists at least one $n \in c$ such that $F_n(\hat{v}) > F_n(v)$.

Recall that a Nash equilibrium is proof against unilateral deviations, that is, deviations where one agent chooses a new strategy assuming the remaining agents keep their original one. This would create a strategy profile which differed only for the one deviating agent. Here, the idea is extended to coalitional deviations, that is, deviations where a coalition of agents chooses a new set of strate-

gies while the remaining agents keep their original strategies. CPE requires an equilibrium strategy profile to be proof against all possible coalitional deviations, including deviations by the grand coalition and single agents (See Bernheim, Peleg, and Whinston 1987 and the discussion in Conley and Konishi 2002). CPE as defined here adds the condition that such deviations be credible. Technically, this makes it a refinement of CPE.

An *economic mechanism* is a game designed to achieve some outcome or to get agents to act in some desirable way. The objective of blockchain consensus protocols are a bit different. Protocols are constructed to ensure the correct termination of a process assuming enough of the agents follow the rules and act correctly. Most protocols also have the property that "honesty" on the part of all nodes in the validation network is a Nash equilibrium if they were to be expressed as games. The problem is that most protocol games have a great many other Nash equilibria as well. For example, if all nodes in a PoS blockchain were dishonest, no single node would benefit from deviating and acting honestly instead.

For this reason, mechanisms are designed such that not only is the desirable outcome an equilibrium, but is in fact the only equilibrium of a certain type (Nash, dominant strategy, or coalition-proof, for example). In such a case, the mechanism is said to *implemen*t the outcome in a certain type of equilibrium.

The value of implementing honest behavior of nodes in a blockchain protocol is obvious. Nash equilibrium and even dominant strategy equilibrium, however, are far too weak to guarantee honest consensus. It is well understood that nodes can form coalitions and collude through mining pools or Sybiling. Thus, for an outcome to be stable in the context of blockchain validation networks, it absolutely must be coalition-proof. Putting this together:

Strategically Provable Security (SPS): A blockchain protocol has SPS if honesty and faithful execution of the protocol by all the validating nodes is the *only* CPE for any reasonable belief structure.

We will need some additional notation to formally define the Catastrophic Dissent Mechanism and show it gives a blockchain SPS.

V – The total value that a coalition of nodes thinks it can gain through dishonesty.

N – The number of validating nodes.

A – The number of independent agents running validating nodes.

H – The number of nodes who decide to honestly report any bad behavior.

D – The number of nodes that follow the conspiracy and behave dishonestly. $(D+H=N)$

G – The amount held in the GBB of each node.

T – The expected present value of transactions processing fees to nodes who behave honestly.

Belief structures of nodes are what determine V, the total value that a coalition of dishonest nodes think they can gain through any sort of out-of-protocol behavior. This value may come from stealing tokens or assets represented by tokens, harming users of the blockchain, or simply that the dishonest coalition values any chaos or confusion that their actions may create. We will focus on token theft in the discussion below for simplicity.

Catastrophic Dissent Mechanism

We interpret v_n as the amount that agent n plans to steal and move off-chain for his personal gain. If $v_n=0$, then node n is behaving honestly and also reports any thefts by other nodes.[13] If $\sum_n v_n > V$, then we interpret the strategy profile as coordination failure which results in all dishonest nodes being unsuccessful in their attempts to steal tokens. This implies the following payoff function:

$$F_n(v_1,\ldots,v_n,\ldots v_N) =$$

T	if $v_n=0 \ \forall n\in\mathcal{N}$	(all nodes honest)
$T+\frac{GD}{H}$	if $v_n=0$ and $\exists m\in\mathcal{N}$ such that $v_m>0$	(honest with dishonest nodes)
$-G$	if $v_n>0$ and $\exists \ m\in\mathcal{N}$ such that $v_m=0$	(dishonest with honest nodes)
$-G$	if $v_n>0 \ \forall n\in\mathcal{N}$ and $\sum_n v_n>V$	(dishonest coordination failure)
v_n	if $v_n>0 \ \forall n\in\mathcal{N}$ and $\sum_n v_n\leq V$	(all nodes dishonest)

The reader might notice that this is a one-shot game while blockchain validation is really a repeated one-shot game. Our reason for taking this approach is that it is more straight-forward to truncate the infinite sequential continuation game into a single period. For example, nodes may think that it will take many periods for all agents to figure out that the chain has been compromised and plan complex strategies of obfuscation to move V off-chain over a number of periods. The real strategic choice, however, is whether to validate the chain honestly in all periods or follow the profit-maximizing, multi-period path of optimally collusive dishonest behavior. Thus, we can truncate this decision to a one-shot game.

As we outline above, we will not assume that nodes are honest or dishonest. This is an *ad hoc* behavioral assumption without foundation. Instead we will assume that all nodes are self-interested and maximize payoffs. We also will assume that nodes can coordinate actions and make binding agreements to share ill-gotten gains with one another. Finally, we will not assume that we can identify the self-interested agents behind the nodes. In real life, we cannot tell which nodes are Sybils, and which are independent. To sum up, we consider an environment where:

- All nodes are self-interested pay-off maximizers.

13 It is possible that a node might steal zero but be dishonest in other ways, or steal zero and not report dishonest nodes. We ignore these possibilities as they are dominated strategies and including them would needlessly complicate the description of the game.

- Coalitions of nodes can coordinate and make binding agreements to share any joint gains.
- Many nodes may be Sybils and work in the interests of a single agent.

Claim 3: *Suppose that all the nodes join a conspiracy to share the largest gain that dishonest strategies allow. That is, the grand coalition chooses a strategy that satisfies the following:*

$$v=(v_1,\ldots,v_N) \text{ where } \forall n\in\{1\ldots N\},\ v_n>0, \text{ and } \sum_n v_n=V.$$

Then if $G>\frac{V}{N(A-1)}$ *and* $A\geq 2$, v *is not a CPE of the CDM.*

Proof: Suppose that there are $A\geq 2$, independent agents running a total of N nodes in the ANL. Then the average payoff that dishonest colluding agents receive is $\frac{V}{A}$. If any agent gets more than this, another agent must get less. Suppose first that all agents run the same number of nodes: $\frac{N}{A}$. Then if one agent deviates by defecting from the conspiracy, he collects the GBBs from all other nodes.[14] Since $\frac{N}{A}$ of these are nodes owned by the defecting agent, the value of the GBBs collected from the nodes belonging to the other agents is $\frac{GN(A-1)}{A}$. Thus, if $\frac{GN(A-1)}{A}>\frac{V}{A}$, the agent is better off defecting. This is true if and only if $G>\frac{V}{N(A-1)}$. Obviously choosing honesty is a credible deviation since the deviating coalition consists of a single node and every other strategy possible at best gives a payoff of $\frac{V}{A}$.

Now suppose that agents run different numbers of nodes. Then some agent must run fewer than $\frac{N}{A}$ nodes. If one of these smaller than average agents (in terms of the number of Sybils he creates on the network) defects, he collects GBBs from more than $\frac{N(A-1)}{A}$ nodes. In other words, the total reward to defecting is even greater for agents running smaller numbers of nodes and so the agent with the smallest number of nodes will certainly defect if $G>\frac{V}{N(A-1)}$. The argument that honesty on the part of a single agent is a credible deviation follows the one in the previous paragraph.

Suppose that the dishonest conspiracy tried to head this off by giving these smaller agents a proportionally greater share of V. In particular, if agent n runs N_n nodes $(\sum_n N_n=N)$, then agent n gets an audit payoff of $G(N-N_n)$. Suppose that the dishonest coalition of the whole tries to give each agent this payoff for participating in the conspiracy to steal V. In total, these payoffs would equal:

14 If one agent defects from a conspiracy, it does not matter whether one, some, or all of the nodes he runs call for the audit. To see this, suppose only one of his nodes calls for an audit. The agents collects the GBB from the "dishonest" nodes he runs, but is is his own money. The net audit reward is therefore the GBBs of the nodes of the other agents if they decide to behave dishonestly.

$$\sum_n G(N-N_n)=G(AN-\sum_n N_n)=GN(A-1).$$

Thus, if the conspiracy were able the share V in this way, all agents would be better off participating and not calling for audits. Unfortunately, this would require that $GN(A-1)\geq V$. This in turn would require that $G\leq\frac{V}{N(A-1)}$, which contradicts the hypothesis of the claim. Again, the credibility of the deviation follows the same logic as above.

Therefore, regardless of the pattern of node ownership and the sharing of the proceeds of the theft, there is at least one agent can credibly defect from v, behave honestly, and report the conspiracy if $G>\frac{V}{N(A-1)}$. ■

Claim 4: *Suppose that all the nodes join a conspiracy to share something less than largest gain, V, that dishonest strategies allow. That is, the grand coalition chooses a strategy that satisfies the following:*

$$v=(v_1,\ldots,v_N) \text{ where } \forall n\in\{1\ldots N\},\ v_n>0 \text{ and } \sum_n v_n<V.$$

Then if $G>\frac{V}{N(A-1)}$ and $A\geq 2$, v is not a CPE of the CDM.

Proof: Immediate using the same argument as Claim 3.■

Claim 5: *Suppose that a strict subcoalition $c\subseteq\mathcal{N}$ of the nodes join a conspiracy to behave dishonestly while the remaining nodes behave honestly. Call this strategy profile v. Then if $G>\frac{V}{N(A-1)}$ and $A\geq 2$, v is not a CPE of the CDM.*

Proof: We need to show that a credible deviation from this strategy profile exists. Note first that the payoff to each of the nodes in the coalition c is $-G$ since the honest nodes not in the conspiracy report their bad behavior. Consider a strategy in which all agents in c were honest instead. Then if agent n runs N_n nodes, his payoff would be $N_nT>-GN_n$. To see that this deviation is credible, note that any deviation from honesty for any subcoalition $c'\subseteq c$ would necessarily require that at least one node of one agent be dishonest, in which case his payoff would be $(N_n-1)T-G<N_nT$. This deviation to honesty is therefore credible and so v is not a CPE of the CDM. ■

Claims 3, 4, and 5 collectively imply there is no strategy profile that involves dishonest behavior on the part of the grand coalition or any subcoalition that is a CPE of the CDM. It remains to show that honesty is a CPE.

Claim 6: *If $G>\frac{V}{N(A-1)}$ and $A\geq 2$, then $v=(0,\ldots,0)$ is a CPE of the CDM.*

Proof: If all agents are honest, they get a payoff of $N_nT>0$. By Claim 4, if any coalition smaller than the grand coalition chooses dishonesty, they get strictly smaller payoffs and deviating back to honesty is credible. On the other hand, if the grand coalition defects to dishonesty, it would appear that all agents get a higher payoff, that is, $\frac{V}{A}>\frac{NT}{A}$, assuming all agents run the same

number of nodes and divide any gains equally. Note that this is not a credible deviation since any single agent could decide to be honest and receive $\frac{GN(A-1)}{A}$ and by hypothesis, $G>\frac{V}{N(A-1)}$, which implies $\frac{GN(A-1)}{A}>\frac{V}{A}$. The argument for the case where agents run different numbers of nodes or divide any gains unequally follows the same structure is it does in Claim 3. We conclude that $v=(0,\ldots,0)$ is a CPE of the CDM. ■

Theorem 2: *If* $G>\frac{V}{N(A-1)}$ *and* $A\geq 2$, *then the CDM gives a blockchain SPS.*

Proof: Claim 6 shows that honesty is a CPE. Claims 3, 4 and 5 show that no other strategy is a CPE. Thus, the CDM implements honesty in CPE. ■

For Theorem 2 to be useful, V must be relatively small. The larger the V, the larger the GBB needs to be. If the GBBs are too large, potentially honest validators may find it is not worthwhile to participate in the validation network.

To get a sense of what this means, suppose the nodes believed that they could gain $V=\$10,000$ through a dishonest collusive strategy by the coalition of the whole. Suppose that there were 100 nodes and all were independent $(A=N=100)$. Then a good behavior bond of at least $\$10,000/100(99)$, would be enough to prevent the conspiracy. Suppose that the $GBB=\$1.02$, for example, and consider the incentives of a single node. If the node stays in the conspiracy, it gets an equal share of the proceeds, $(\$10{,}000/100=\$100)$. If he reports the conspiracy, it gets to collect the GBBs of the other 99 agents who followed the conspiracy $(\$1.02x99=\$100.98)$. Thus, a GBB of approximately $\$1$ is enough to deter actions worth $\$10,000$ to a dishonest coalition. The table below gives some other examples.

V	A	N	GBB
$10,000	100	100	$1.01
$1,000,000	100	100	$101.01
$100,000	25	10	$416.67
$10,000	10	5	$222.22

Table 2: Calculation of the Threshold Good Behavior Bond.

Sybiling is of no direct benefit in this case. In fact, it makes it even easier to prevent dishonesty. Consider the network above $(A=N=100)$. Suppose that one of the agents decides to create 900 extra nodes so that there are still 100 independent agents, 99 controlling one node each, and one controlling the other 901 nodes. The using the formula above, a GBB of anything over a payoff of or $0.101 would be enough to prevent dishonesty.

If instead the number of independent agents was reduced but the total number of nodes kept the same, a larger GBB would be required, as shown in the last two rows of Table 2 above. For example, if there were only 3 independent agents running, say, 50 Sybiled nodes each, then the GBB would need to be at least $10,000/150(2)$ or about $33.34. On the other hand, if these agents got rid of their Sybils and each agent ran only one node instead, the necessary GBB would go up to $10,000/3(2) = \$1667$.

The protocol therefore benefits if it can minimize the size of the GBB in order to incentive the participation of as many independent nodes as possible. This not withstanding, the protocol's security is immune to the creation of Sybils for any given number of independent nodes.

7. Catastrophic Recovery Procedure

If all nodes are dishonest despite the Catastrophic Dissent Mechanism (CDM), they have two dishonest paths open to them under the Geeq Workflow Protocol (GWP). First, the nodes could produce one or more provably dishonest and noncanonical forks, possibly rewriting history. Second, they could be silent, effectively stalling transaction writing (although block writing terminates trivially in empty blocks).

There are two potential gains from such actions. First, dishonest nodes might hope that there is some direct economic gain to be made through stealing tokens, assets, causing smart contracts to give incorrect outputs, etc. Second, it may be that nation states, hackers with botnets, competitors, disgruntled but wealthy individuals, and so on, value destroying or undermining confidence in a blockchain independently of any direct gain they might see.

Realizing direct economic gains through the creation of false ledgers depends on the ledger somehow inducing agents to do something of value in the real world. For example, the dishonest coalition could profit if they could convince a gullible agent to buy tokens on the dishonest chain and transfer dollars to the bank accounts of dishonest agents, or to transfer the title of a car, ownership of stock, etc., to dishonest agents in exchange for tokens transferred to an account on a dishonest fork. If no agent outside the dishonest coalition is willing to accept transfers of tokens or other actions on the dishonest fork as meaningful, however, then the dishonest ledger is simply a piece of fiction with no real world impact.

Why would any user accept a transaction on a ledger that is provably dishonest? One possibility is that the agent does not care or does not insist on proof that the chain was honest. There is not much to be done in this case. If gullible users are willing to accept an obviously forged check or a promise of payment based purely on trust, a dishonest blockchain fork is only one of many ways to rob them. In this case, the dishonest nodes do not even need to have ever posted GBBs on the original chain. They could simply create a fork by falsely adding themselves in the ANL and make any transactions they wish.

A second possibility is that users might see no alternative to transacting on the dishonest fork besides abandoning their accounts. Perhaps all the dishonest nodes did was roll back a transaction

they claimed was fraudulent in the real world sense.[15] Forgiving this and hoping that the dishonest nodes behave in the future might be seen as the best choice if the dishonest fork is the only one that will ever exist.

Halting transactions processing or offering only dishonest ledgers has the potential to undermine confidence in the blockchain and cause damage or confusion at the application level. Even in this case, if an honest fork can be quickly reestablished, then all such attacks accomplish is a temporary delay in transaction processing. This is an inconvenience to users, but strongly limits the payoff to this kind of griefing even for an adversary only interested in burning down the world.

With this in mind, we offer the Catastrophic Recovery Procedure (CRP). In the unlikely event that all nodes are dishonest and transactions are not being written to the chain, the following procedure is invoked:

Catastrophic Recovery Procedure

1. Detecting a stall: If three epochs pass with only EBs authorized by the relay, the relay issues a signed block declaring a stall. This block includes "suspend from ANL" transactions signed by the relay for all the nodes in HBO for those three epochs. The suspension does not confiscate the GBBs of the silent nodes, but takes them out of the hub-block rotation until they send an "unsuspend from ANL" transaction.

2. Opening the ANL: The relay pauses for 10 block writing intervals and attempts to reconstruct the ANL. The relay accepts "intention to join ANL" transactions from users and, at the end of the period, writes a block with these transactions approved. This generates a new HBO which is chosen from newly joined nodes and any nodes in the existing ANL that were not suspended.

3. Funding GBBs: The next 100 blocks are used for bootstrapping. Only incoming token transactions to the accounts of the newly joined nodes are allowed. These are validated through a HBO based on the new ANL. Note that some of these nodes will not yet have posted a GBB. By the end of the 100 blocks, all newly joined nodes must have succeeded in moving tokens onto the chain and then getting a "join ANL" transaction in a committed block that stakes the proper number of tokens.

4. Relaunch: At the end of these 100 rounds, normal operations resume with all transactions being accepted and validated by rotating through the new set of nodes in the ANL. Note that the blockchain will therefore include at least 3N EBs, a relay stall declaration block, 10 ANL reconstruction blocks, and then 100 blocks validated in the ordinary way but containing only transactions designed to complete the reconstruction of the ANL.

5. Relaunch Failure: If the 100 rounds of the attempted relaunch result in only EBs being added to the chain, then the new ANL also contains only unresponsive nodes. In this case, newly joined nodes are removed from the provisional ANL (since they have not funded their GBBs) and any nodes taken from the original ANL who have funded their GBBs but did not respond when

15 Consider, for example, the famous June 17, 2016 attack on the DAO which resulted in the theft of $70M and led Ethereum to execute a chain reorganization via a hard fork.

placed in the new hub-block rotation are suspended. The relay then goes back to step (1) and attempts to relaunch again.

Claim 7: *Assume that the relay is honest. Then if there are any users willing to report dishonest behavior directly to the relay, and any honest agents willing to join the ANL in response to the relay declaring that a chain is stalled, transaction writing will eventually resume even if 100% of the current ANL is dishonest.*

Proof: If the ANL contains only dishonest nodes, the validation network can either create a provably dishonest fork, or be completely silent and build a fork with only empty blocks. If it creates a provably dishonest fork, a user can submit an audit that will be affirmed by the honest relay. Any nodes participating in the dishonest fork will be audited out of the ANL by the relay and not included in any future HBO. This leaves only dishonest nodes who choose to be silent. But then, the chain will eventually include enough empty blocks for the relay to invoke the CRM.

The relay is honest and so it will accept the "intention to join ANL" transaction from the honest agent. Many dishonest agents may also send "intention to join ANL" to the relay, and it may be that all the nodes in the current ANL are dishonest as well. Nodes are randomly placed in the HBO during the 100 block bootstrapping period and so there is a positive probability that an honest node will be included. When the honest node's turn to be hub arrives, it can add the transaction to fund its GBB to the HTB it sends to the relay, which will then be signed and become the only canonical block. This will also mean that the relaunch is successful. No matter how many silent or dishonest nodes make their way into the ANL, the honest node will appear periodically in the HBO and will create a non-trivial HTB. ■

This claim implies honest agents, if they exist anywhere in the world, can force their way into the ANLs and HBOs of stalled chains despite the efforts of any number of dishonest agents or nodes. Users of the blockchain and The Geeq Corporation, at least, have strong incentives to prevent stalling and to restart stalled chains. Validating GeeqChains honestly is also profitable, so even otherwise disinterested agents have an incentive to join the ANL and un-stall the chain. In contract, in all other protocols of which we are aware, if one third or one half of the nodes are dishonest, the blockchains they support either stall or lose their security guarantees. The only mechanism to escape this situation is human intervention through governance or a hard fork.[16]

8. Hardening

The practical effectiveness of the CDM depends on two things. First, minimizing V, the potential reward that nodes might gain from dishonest behavior, and second, maximizing the proportion of independent, and hopefully honest, nodes in the ANL.

If dishonest agents can fill the ANL and flood the relay with "intention to join ANL" requests. the fraction of honest provisional nodes could be small enough that the relay could be forced to in-

16 Note that at best, governance is 50% BFT. A great deal of economic research in voting theory and public choice proves that it is essentially impossible to create non-manipulable mechanisms that are in any way fair, equitable, or efficient. The most famous are the Arrow Impossibility Theorem (Arrow 1950) and the Gibbard-Satterthwaite Theorem (Gibbard 1973 and Satterthwaite 1975).

voke the CRP frequently. A large fraction of silent nodes in the HBO also delays the addition of transactions to the chain since they force the node to issue EBs.

In this section, we describe two additions to the GWP that significantly reduce this potential for griefing. We do this in a separate section because neither of these additions have any provable impact on the security of the protocol. These measures all fall into a class of "weak incentives", that is, they give agents who are not bent on damaging the blockchain for strong external reasons enough incentive to be good citizens and help the system run as smoothly as possible. Transactions fees and mining rewards (on PoW chains) also fall into this category. On the other hand "strong incentives" prevent even externally motivated agents from causing harm. The CDM, CRP, and the ESP, discussed in the next section, fall into the class of strong incentives and are the basis of the security guarantee.

8.1. Adding Special Precautionary Nodes

To begin with, we define three categories of nodes.

Anchor nodes: Anchor nodes have their public keys listed in the genesis block. These nodes are equal to all other nodes with one exception: they are always in the HBO for every epoch, provided they are in the ANL. Like all nodes, anchor nodes can be audited out of the ANL, in which case they lose their GBBs and are subject to all the other rules that incentivize good behavior. The Geeq Corporation or the enterprise or organization that sponsored or developed the application living on a specific instance of a GeeqChain are examples of likely anchor nodes.

Certified nodes: Certified nodes are vetted by The Geeq Corporation or other designated agents. Certified nodes are free agents and can join or leave the ANL of any chain in the Geeq ecosystem. They will have received a signed certificate which they submit with their "join ANL" transaction to prove their status. Certified nodes are similar to anchor nodes except that they are not listed in any genesis block. Accounting firms, government regulatory agencies, and public interest non-profits are example of likely certified nodes.

Anonymous nodes: Anonymous nodes were described in Section 3.1. These are nodes that freely join and leave the ANL of any instance they wish, providing only IP and token account addresses.

Each category of node has advantages and disadvantages.

Anchor nodes have the strongest incentives to behave honestly since they are most invested in the integrity of any given instance of a GeeqChain. Including anchor nodes in every HBO makes it very unlikely that an HBO will include only dishonest nodes. As long as at least one honest node exists, dishonest nodes are audited out within one epoch, a provably honest and canonical chain exists, transaction writing does not stall, and the CRP is never invoked.

Certified nodes are backed by real world agents with reputations for honesty. Including them in the HBO is likely to increase the fraction of honest nodes. It also has the effect of increasing the

number of honest agents in the HBO in the sense of of Theorem 2. That is, certified nodes are not likely to be Sybils which increases the effectiveness of the CDM and lowers the required GBBs.

The big downside of the anchor and certified nodes is that they are not anonymous. Since they can be identified, they can be threatened by courts, governments, or criminals, and be compelled to behave dishonestly.

The major advantage of anonymous nodes, on the other hand, is that they cannot be identified or threatened. The disadvantage, or course, is that it is not possible to prevent Sybiling of dishonest nodes. Thus, the presence of an anonymous node in the HBO only really guarantees the addition of a single independent node.

Regardless of type, no node has a veto, is able to unilaterally prevent transactions from being processed, or shut down an instance of a GeeqChain. This is essential because if an anchor or certified node could shut down or dishonestly alter an instance, the agents running the nodes could be compelled to do so or otherwise act against protocol or the interests of the users. It is therefore important that the HBO have a mixture of anchor and certified nodes in order to ensure the presence of honest nodes run by several different agents, and anonymous nodes that cannot be compelled to behave dishonestly.

The exact method of choosing this mixed HBO is not important provided it is deterministic. One example of how to do this is the following:

- All anchor nodes are included in every HBO.
- A selection of certified nodes is chosen such that anchor and certified nodes together make up no more than half of the HBO.
- The other half of HBO is selected from anonymous nodes.

Appropriate adjustments are made if there are not enough each type of node in the ANL to meet these quotas.

For epoch E which runs from block $B+1$ to $(B+1) + N$, consider the nodes in the ANL of block B. A selection function maps the hash of the CLS to the ANL of block B. This function makes a uniformly distributed choice from the set of certified and anonymous nodes and then randomly creates a HBO with these nodes and the anchor nodes for the epoch. Of course, this is only pseudo-random. It is possible that a clever coalition could gain control of the hash of the CLS by adding the carefully chosen last transaction. This would not remove anchor and certified nodes from the HBO, however, and it is difficult to see what benefit it would confer.

Claim 8: *Assume that the relay is honest, S special type honest nodes that are always included in the HBO, and that the rest of the ANL consists of H honest and D dishonest anonymous nodes. Then*

$$\frac{S}{N}+\frac{(N-S)H}{(H+D)N}$$

is the probability that an honest node will be the hub for any given block.

Proof: Immediate. ■

These modifications to node type and selection make it very unlikely that griefing could ever create long delays in transaction writing. Even if the ANL was flooded with dishonest nodes who paid GBBs, transactions would be written in every epoch since at least S honest nodes are in every HBO. In the unlikely event that the special nodes turn to the dark side and are audited out for dishonesty, Claim 6 proves that honest nodes can still join the ANL, and when they end up in the HBO, can write an audit that removes all of the dishonest nodes and collects their GBBs. Thus, nontrivial transaction blocks will continue to be written after a delay that is proportional to the number of dishonest nodes. However, the cost of generating such delays to dishonest nodes increases in the same proportion.

8.2. Keeping the ANL Live

A dedicated adversary could still cause frequent short delays through failure to communicate. Honest nodes may also drop off the network unexpectedly or may drop-off without bothering to send a "suspend from ANL" transaction first and therefore remain in the HBO.

Punishing nodes for failures to communicate is generally undesirable since latency, partitioning, or network failures may be responsible. Nevertheless, if a node is unable to perform its duties efficiently for whatever reason, it should not be part of the validation network. The following are measures that balance these two considerations to improve the smooth operation of the protocol.

1. If a node fails to send an HTB to the relay when acting as hub 5 times in a row, the relay creates a "suspend from ANL" transaction which is valid by protocol. The node is suspended for at least 100N blocks after which it must send an "unsuspend from ANL" transaction if it wishes to rejoin the HBO selection process.

2. If a node fails to send an HTB to the relay when acting as hub 10 times in any string of 20, the relay creates a "suspend from ANL" transaction which is valid by protocol. The node is suspended for at least 1000N blocks after which it must send an "unsuspend from ANL" transaction if it wishes to rejoin the HBO selection process.

3. The fifth time a node is suspended by the relay for either type of silence, 10% of its GBB is taken in addition. The node must top this off as well as send an "unsuspend from ANL" transaction if it wishes to rejoin the HBO selection process. This goes up by 10% for each new suspension. Thus, if the node is suspended 14 times. 100% of the GBB is taken.

4. Nodes cannot withdraw their GBBs until their suspensions expire.

The penalties in cases 1 and 2 are mild. The node has tokens locked up in its GBB for a period and is not able to earn transaction fees. It also pushes nodes who may simply have shut down and forgotten to inform the network to take an active step to rejoin or leave the ANL. Case 3 is to disincentivize nodes from simply rejoining after each suspension without addressing the underlying problem. Dishonest agents running nodes can easily avoid these penalties by leaving the ANL after the fourth suspension and rejoining as a "new" node, but it is likely to push at least some honest agents to take corrective action.

9. The Edge Security Protocol

In this section, we describe a simple mechanism called the Edge Security Protocol (ESP) that leverages the provability of dishonest behavior to produce a new approach to blockchain verification. The ESP gives individual users immediate recourse to ensure the integrity of a ledger even if all of the validating nodes are dishonest.

PoW, PoS, Direct Acyclic Graph (DAG), and all other consensus protocols of which we are aware are "node-centric" in the sense that nodes, through some mechanism, are the ultimate source of truth and authority. Since nodes are the agents who potentially benefit from writing false transactions that steal tokens, all such protocols embed conflicts of interest between nodes and users that create intrinsic threats to global honesty and canonicalness.

In contrast, the ESP empowers individual users, who hold tokens on the chain which are at risk of being stolen by dishonest nodes, to protect themselves from dishonest validation networks or being fooled into accepting fraudulent token transactions on any chain. This is inherently incentive compatible.

Geeq's ESP is designed with mass adoption in mind and so does not require users to be technically proficient enough to understand the protocol or be willing and able to check a blockchain thoroughly for dishonesty through individual inspection. Instead, Geeq provides a web-based user client that both facilitates interactions and transactions with the chain, and automatically checks any given fork's properties. The ESP forecloses all avenues dishonest nodes might have to fool any user who follows the advice of his user client.

The Edge Security Protocol works as follows:

Edge Security Protocol

1. Chain Discovery: Users discover a given blockchain as well as any forks that might exist. In practice, users might be directed to a node that validates an application that a user wishes to use or find a node through a web search or by consulting a forum. Once he discovers any node, however, he can read the ANL and thereby find the IP address of all other nodes that are validating and keeping copies of the blockchain.

2. Honesty Checking: Users (through their clients) inspect the chain and its forks, if any, to whatever degree they wish in order to determine the honesty of the nodes and the validity of the chain or forks.

3. Transaction Creation: Users choose a node and send their transaction(s).

Thus, nodes must prove their honesty to users before users send them transactions. This allows users to identify dishonest nodes and simply ignore any transactions or ledgers that dishonest nodes maintain. It also produces a blockchain that is 99% BFT, since only a single honest node is required to allow users to protect themselves and transact securely on a globally honest and provably canonical chain. More formally, ESP gives the PoH protocol edge security.

Edge Security: A protocol has edge security if any observer having full access to all information in the fork, but nothing more, is able to initiate and accept transactions on forks that are globally honest and provably canonical, if they exist, and avoid doing so on forks that are not.

Edge security is a decentralized security guarantee in two senses. First, it makes users at the edge of the network, rather than nodes at the center, the ultimate arbiters of transaction and ledger validity. Second, it gives this power to each user individually. Edge security does not allow a consensus of users or validating nodes to impose a particular view of the ledger on anyone. Instead, each user is able to make his own determination and act accordingly.

Claim 9: *If the relay is honest and Geeq's public key is known, then GWP verified by ESP produces a blockchain with edge security.*

Proof: From Claim 3, we know that the GWP produces at most one provably canonical and globally honest blockchain under these assumptions. But then any user is able to initiate and accept transactions on a fork only if it is provably canonical and globally honest. ■

If nodes refuse to prove their honesty to users, then users can simply refuse to accept their ledgers as valid. Users can also initiate audits that will be accepted by the honest relay if any or all nodes present them with forks that are provably dishonest or non-canonical. This not only protects users from accepting tokens, stolen or otherwise, on an invalid ledger, it also gives users a way to proactively remove dishonest nodes, and even restart a blockchain from its last honest block if the entire network is dishonest.

Most importantly, tokens on an honest fork are completely safe from theft. Regardless of what dishonest transactions might exist on dishonest forks, if a globally honest and provably canonical fork exists, all token transactions are supported by transaction requests signed by the correct private key from accounts that have correct and sufficient balances.

10. Conclusions

Combined, the Catastrophic Recovery Protocol (CRP) and Edge Security Protocol (ESP) protect users from any level of dishonesty by the validating network, and ensures that nontrivial, honest and canonical block writing can at worst be delayed for a small number of block writing intervals. Such griefing has a high cost to dishonest nodes, but does not confer much benefit. Tokens can never be stolen, so any direct economic value of doing so would rely on the existence of users who act against their own interests by accepting provably invalid token transfers. Even if the goal of dishonest nodes is purely to cause chaos, delaying block writing for a few minutes is unlikely to be very satisfying.

Thus, the CRP plus the ESP imply that V, the maximum value that any coalition of nodes expect to extract from any type of dishonest behavior is small. In turn, this implies that the Catastrophic Dissent Mechanism (CDM) provides strong incentives for nodes to behave honestly even if the Good Behavior Bonds (GBBs) are relatively small.

The Geeq Workflow Protocol (GWP) by itself implies that Geeq's Proof of Honesty (PoH) protocol provides it with 99% BFT, and the addition of special nodes make it very unlikely that any validation network will ever be without at least one honest node.

There are three major elements of the Geeq project's architecture and protocol which are not discussed in this paper. First, the multilayered security guarantee outlined above relies on the presence of an honest relay. Thus, a mechanism is needed, both to ensure that relays are honest, and to give users and honest nodes effective recourse if they are not. Second, The Geeq Project envisions an ecosystem of many interoperable chains. This means that instances will need to exchange $GEEQs, native tokens, information, and other items. In general, secure interchain token transfers are difficult. Geeq's high level of security and ESP provide unique pathways to approach this problem. Third, transaction bundles, blocks, ledgers, and so on, need to be structured so that nodes can prove their honesty to users and one another through Merkle trees, references to provable ledger state checkpoints etc., as parsimoniously and efficiently as possible. These elements will be discussed in future versions of Geeq's technical papers.

References

Arrow, Kenneth J. (1950). "A Difficulty in the Concept of Social Welfare", *Journal of Political Economy*. Vol. 58 pp. 328–346, doi:10.1086/256963.

Bernheim, D., Peleg, B., and Whinston M. (1987) "Coalition-proof Nash Equilibria. I. Concepts", *Journal of Economic Theory*, Vol. 42 pp. 1-12. doi: 10.1016/0022-0531(87)90099-8

Bonneau, Joseph (2018) "Hostile Blockchain Takeovers (Short Paper)", Financial Cryptography Workshops 2018, doi:10.1007/978-3-662-58820-8_7

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., and Felten, E.W. (2015) "Research perspectives and challenges for bitcoin and cryptocurrencies", *2015 IEEE Symposium on Security and Privacy.*

Brewer, Eric (2000) Towards robust distributed systems (keynote). *19th ACM Symposium on Principles of Distributed Computing (PODC).*

Buterin, Vitalik, and Griffith, Virgil (2017) "Casper the Friendly Finality Gadget", *arXiv preprint*, arXiv:1710.09437.

Castro, M and Liskov, B (2002) “Practical byzantine fault tolerance and proactive recovery”, *ACM Trans. Comput. Syst.*, Vol 20, pp. 398–461.

Conley, John P.; Konishi, Hideo (2002) “Migration-Proof Tiebout Equilibrium: Existence and Asymptotic Efficiency”, *Journal of Public Economics*, Vol. 86, pp. 243–62.

Eyal, I and Sirer, E. G. (2014) “Majority is not enough: Bitcoin mining is vulnerable” *Financial Cryptography and Data Security 18th International Conference, FC 2014*, pp. 436–454.

Fischer, M., Lynch, N., and Paterson, M. (1985) “Impossibility of distributed consensus with one faulty process” *Journal of the ACM*, Vol. 32 pp. 374–382, doi:10.1145/3149.214121.

Gibbard, A. (1973) “Manipulation of voting schemes: a general result”. *Econometrica* Vol. 41, pp. 587–60.

Gilbert, Seth and Lynch, Nancy (2002) “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services”, *ACM SIGACT News*, Vol. 33 pp. 51–59, doi:10.1145/564585.564601.

Houy, Nicolas (2014) “It will cost you nothing to "kill" a proof-of-stake crypto-currency” *Economics Bulletin*, Vol. 34, pp. 1038-1044.

Lamport, L., Shostak, R., and Pease, M. (1982) “The Byzantine Generals Problem”. *ACM Transactions on Programming Languages and Systems*, Vol. 4, pp. 387–389.

Larimer, Daniel (2014) “Delegated Proof-of-Stake (DPOS)”, *Technical Report*, http://v3.bitshares.org/security/delegated-proof-of-stake.php.

Satterthwaite, M. A. (1975) “Strategy-proofness and Arrow’s conditions: existence and correspondence theorems for voting procedures and social welfare functions”, *Journal of Economic Theory*, Vol. 10, pp. 187–217.