ClariNet: Generalized Reputation Assessment for Witnessed Data Exchange

***Abstract*—Audit logging, a necessity for post-incident analysis, can break down when two participants—such as nodes in a distributed system communicating over a network—have different records of what occurred and neither is able to definitively prove its claim. Introducing a witness that records the message before forwarding it can provide a tie-breaking vote, but requires the witness be honest about the data it observed. In this paper, we present ClariNet, a reputation assessment scheme that allows participants in a distributed system to identify activity indicative of audit log forgery, penalize such malicious activity even when the exact source is not known, and ensures that the true origin of the malicious activity is in aggregate penalized more harshly than cooperative participants. It does this by verifying claims against a known point of reference, providing these claims, when verifiable, to other interested participants, and introducing the notion of different penalty degrees. We demonstrate through formal proofs and simulation that ClariNet allows cooperative participants to gain meaningful insight into peer behavior even when over half the network is malicious, when malicious peers act maliciously relatively infrequently (as low as 10%), and in relatively short intervals (as low as 100 actions for a given participant). With ClariNet, participants in distributed systems can gain some level of confidence in their ability to identify a cooperative witness and protect themselves from false claims by a malicious peer.**

I. Introduction

In distributed systems, participants must often send data to one another. These participants typically keep audit logs of the data they sent or received to facilitate post-incident analysis. Investigators can use these audit logs to determine data flow and identify corresponding actions, allowing them to identify where expectation diverges from reality. Identifying these divergences can greatly narrow the problem space and expedite root causes identification which is becoming ever more important as systems grow in scale and complexity.

However, parties must ensure that audit logs are responsibly recorded and maintained. Missing logs provide no benefit and inaccurate logs can result in false leads and wasted time. Malicious participants can exacerbate these problems by intentionally recording data that attempts to shift suspicion onto another party. While the other party may likely keep its own logs, unless the behaving participant has some means of supporting its claim, these conflicting logs can lead to further wasted time arguing over which account is correct. While cryptographic signatures can assist in identifying the origin of a message, a malicious party could provide an invalid signature undermining this proof of origin.

Consider a security system composed of IoT devices, such as cameras, and hubs that are capable of contacting a security service. The cameras and hubs are manufactured by different companies and equipped with onboard AI models to facilitate capabilities such as image cleanup and recognition. Unnecessarily notifying the security service is not ideal, but is significantly less harmful than failing to notify of a real security event, so the hubs are permitted to make decisions even when a signature is invalid. A malicious participant could exploit this, such as if it lacks faith in the accuracy of its cleanup. In one scenario, the camera cleans an image, supplies the cleaned image and an invalid signature while logging the uncleaned image and a valid signature. In the other, the hub receives an uncleaned image and a valid signature, but records its own cleaned image and an invalid signature as if these were what the camera transmitted. Despite different bad actors, both scenarios result in the camera having a log entry containing an uncleaned image and valid signature and the hub having a log entry of a cleaned image and an invalid signature.

Having an intermediary, such as another smart device, witness the data in transit can point investigators in a particular direction by supporting one participant's account. This requires the witness be impartial and responsibly record data. In a system with no agreed-upon implicitly trusted party, participants must rely on some means of identifying trustworthy witnesses, such as a reputation scheme.

To this end, we propose ClariNet, a set of criteria participants can use to identify and penalize malicious actions that facilitate log forgery. In addition to identification, ClariNet resists a malicious intermediary poisoning the communication between two behaving participants. Clarinet accomplishes these goals by providing three reputation actions—rewards, weak penalties, and strong penalties—rules about when to apply each, correctness criteria, and rules for safely sharing auditing information with peers. ClariNet uses a combination of a known point of reference and obvious signs of malicious activity—in this case invalid signatures—to identify malicious action and applies rewards when peers behave, weak penalties when the malicious party is unclear, and strong penalties when the malicious party is certain.

We evaluate these rules using formal proof and simulation using PeerSim [32] to demonstrate the following:

1) Participants always end up with either irrefutable proof a peer sent, witnessed, or received a message via signatures, or are able to penalize bad actors.
2) For a given message, a participant never penalizes behaving peers more harshly than malicious peers.
3) For a given message, the aggregate penalties for malicious peer(s) always outweigh the incorrect penalties for behaving peers.

4) When applied in general, malicious participants' average reputations decrease more quickly than cooperative participants'.

Through the simulations, we demonstrate *(4)* holds under a variety of network conditions such as frequency of malicious action and proportion of malicious peers. The simulation also shows that cooperative participants can gain meaningful insight relatively quickly.

Altogether, we make the following contributions:

1) ClariNet: a novel reputation assessment system that can differentiate known origins of malicious action from suspected origins and resists malicious exploitation.
2) Formal proofs demonstrating ClariNet's soundness.
3) Simulations demonstrating that ClariNet's theoretical claims hold up under situations approximating real-world networks.

We begin by discussing the requirements for such a system, then discuss the threat model, including adversary capabilities and goals. Next, we provide ClariNet's behaviors. With the protocol laid out, we move on to the formal proofs and discussion of interactions, followed by the simulation setup and results. We close with a discussion of related work and conclusion.

II. Problem Statement

Audit logging is a necessity for post-incident analysis, particularly when two systems have communicated and auditors must determine a responsible party. In the absence of centralized, trusted logging, auditors must rely on participant logs which can be forged by a malicious participant. When each participant claims a different log record, it can degenerate into one party's word against the other.

A. Abstract Example

P_A and P_B are participants in a distributed system, D and D' represent data with the requirement that $D \neq D'$, S represents a valid signature, and I represents an invalid signature.

Scenario 1.

1) P_A sends (D, S).
2) P_A logs (D, S).
3) P_B receives (D, S) but wishes to deny it.
4) P_B invents D' and acts on this.
5) P_B logs (D', I) because it cannot generate a valid signature for P_A.

Scenario 2.

1) P_A has some data D' it should send to P_B based on some agreement, but wishes to deny it sent D'.
2) P_A sends (D', I).
3) P_A invents data D that it wishes to claim.
4) P_A logs (D, S).
5) P_B receives (D', I).
6) P_B takes appropriate action based on (D', I).
7) P_B logs (D', I)

In both scenarios, P_A claims it sent (D, S) while P_B claims it received (D', I). In scenario 1, P_B is the malicious actor, while in scenario 2, P_A is the malicious actor.

III. ClariNet: Witnesses and Reputation

In ClariNet, a witness records the message in transit and serve as a tie-breaking vote, but this requires the witness be impartial. The sender and receiver would like some insight into a given witness's trustworthiness, which they can achieve through a reputation, but they need some criteria by which to form that reputation. These criteria must also be resistant to a malicious witness that simply wishes to poison the communication between behaving senders and receivers.

To accomplish this, participants need correctness criteria to determine when a message is suspicious, rules about how to adjust reputation based on the correctness criteria, rules about how to exchange information, and a secure means of authenticating their peers during these exchanges. In designing ClariNet, we attempt to address each of these in the face of intelligent malicious adversaries who are aware of the ClariNet protocol.

IV. Threat Model

Adversaries are any participants that wish to avoid culpability for data they sent or received. To accomplish this, they need to achieve three goals:

G1 If sending, provide the messages in a deniable fashion.
G2 Provide proof of messages that support their forged claim.
G3 Win the consensus protocol.

G1 and *G2* are reasonably simple. Providing an invalid signature on the initial send suffices to ensure that the receiver does not have definite proof the adversary sent the message. Similarly, recording the desired message accomplishes *G2*.

G3 is trickier since the adversary must find a peer willing to support its false claim. While we do not directly address the question of witness agreement between sender and receiver, such agreements would incorporate peers' reputation scores. This means that adversaries want to degrade non-colluding peers' reputations of each other while keeping their and their colluders' reputations high.

A. Assumptions

With goals defined, we turn to adversary capabilities and assumptions.

AA1 Adversaries are capable of intercepting and rewriting messages, but do not wish to entirely halt communication between cooperative participants.
AA2 Adversaries recognize their colluders and can communicate out of band up to and including sharing private keys when beneficial.
AA3 Adversaries are in complete control of what they report to other participants.
AA4 Adversaries make an honest effort to deliver data, just in a manipulated fashion when they deem appropriate.

The qualification to *AA1* is necessary to make the problem tractable. Without it, adversaries could simply halt all communication between non-colluding peers other than the small subset they deem acceptable. While traditional Byzantine solutions mitigate this with deadlines and default actions, we feel this weakening assumption is acceptable because adversaries still wish to gain the benefits of the distributed system. Halting all communication would reduce the capability of the distributed system and, for systems where membership is optional, cause many participants to leave, weakening the benefits. Even for systems where membership is mandatory, halting all communication between cooperative participants would require the adversaries to take on more work, which still undermines the potential benefits of a distributed system. Instead, adversaries only interfere when they have strong reason to suspect it would benefit them.

AA2 simply acknowledges that out-of-band communication is possible and subsequently that cooperative participants cannot rely on intercepting coordination between adversaries.

Similar to many other distributed systems, participants cannot know the internal state of their peers and must rely on those peers' reporting. *AA3* acknowledges that these reports are at the discretion of the peer and can potentially be deceptive.

AA4 means that should an adversary have message M that based on an agreement it should deliver to a recipient, it does deliver M, though M may be manipulated. In conjunction with *AA1*, this means that adversaries do not prevent a cooperative witness from delivering a message to the receiver. It may attempt to interfere in other ways, but only if it can be certain that it does not prevent the final delivery of the message. This is a simplifying assumption and a potential area for future work.

V. ClariNet Specification

A. Participant Identity and Authentication

We begin with identity and authentication as without these participants have no way of verifying the peer with whom they are communicating. Participants can accomplish this by using public key cryptography and tying identity to each participant's public key, similar to the approach used by libp2p [33]. By combining this with authenticated key exchange (AKE) [34], participants can exchange public keys ad hoc, identify the peer by hashing the public key, and then confirm that the peer is who it claims to be by using the public key as part of an AKE handshake with an additional ephemeral key to prevent replay attacks and maintain forward secrecy. The handshake can only succeed if the peer possesses the private key corresponding to the shared public key that itself corresponds to the peer's claimed identity, which we refer to as ID_P. From this point on, participants can be sure that messages encrypted with this session key come from the expected peer. These guarantees hold as long as participants use strong cryptographic keys and ensure their private keys are not compromised. Nodes in libp2p use the Noise protocol framework [35] [36] to implement this approach for real-world systems such as the Interplanetary File System [37].

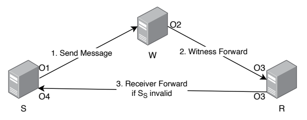

Fig. 1. Send Workflow Messages

While this approach ensures that peers cannot impersonate each other, it does not solve the issues of participants changing identity or allow participants to authenticate peers on a higher level. These are outside the scope of this paper and future work could incorporate mitigations.

B. Messaging & Correctness Criteria

We combine these sections because the design of each influences the other. Before moving on to details, we give an outline of each so readers have some context.

Correctness Criteria. The goals of correctness are to detect when an adversary may be attempting to deny a message or attempting to harm a peer's reputation. To detect these we use two criteria:

C1 Invalid signatures
C2 Known points of reference

Messaging. While we do not place restrictions on the transport protocol implementations can use, we do require that it be a reliable protocol such as TCP or QUIC [38]. This ensures that both sender and recipient are confident the message is delivered successfully and without error. There are two messaging categories, sending data and auditing, composed of the following operations:

O1 Sending
O2 Witness Sending
O3 Receiving
O4 Receiver Forward Receiving
O5 Querying
O6 Query Answering
O7 Query Forward Receiving

O1-3 make up sending. *O4-7* make up auditing. While *O4* is an audit step, it occurs as part of the send workflow. We require signatures on all messages and assume all participants log all messages. Fig. 1 shows message direction for the send workflow. Fig. 2 shows message direction for the query workflow.

Connections. There is also necessary connection setup. We use this approach as witness selection may be expensive and participants would like to have a stable witness for multiple messages. A connection is composed of a sender, receiver, and witness and uniquely identified with some separate key, such as UUID, that all participants agree upon. We refer to this key as the connection ID or ID_C. We place three requirements on a connection: 1) ID_C is unique, 2) $S \neq W \land R \neq W$, and 3) S, W, and R are stable for the life of the connection. This means that any participant changes necessitate a new connection, and that simultaneous connections may exist containing the same participants in the same positions, but with a different ID_C.

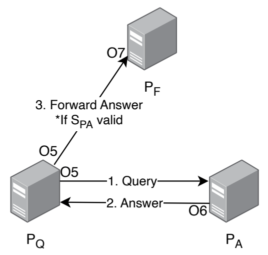

Fig. 2. Query Workflow Messages

Connections only apply to send operations. Audit operations utilize knowledge from connections, but do not require an active connection. Sent messages within a connection are uniquely identified with a sequence number and we denote the combination of ID_C and sequence number as ID_M.

We leave the process of connection initiation and termination intentionally flexible to prevent premature calcification.

1) Sending: With this defined, we lay out the general flow of a send for a sender S, witness W, and receiver R. This encompasses *O1-4*. D defines some data. All hash operations use the same hashing algorithm as the signing algorithm S, W, and R agreed on. We provide more formal state machine specifications in VI.

O1: S constructs ID_M. It then signs (ID_M, D) as S_S and delivers (ID_M, D, S_S) to W.

O2: Upon receiving a message, W checks if it has an open connection corresponding to the message's ID_C. If it does, it verifies S_S, signs (ID_M, D, S_S) as S_W, and delivers (ID_M, D, S_S, S_W) to R.

O3: Upon receiving a message, R checks if it has an open connection corresponding to the message's ID_C. If it does, it verifies S_W. If S_W is valid, it verifies S_S. If S_S is invalid, R hashes (ID_M, D, S_S) as H. It then signs (ID_M, H, S_W) as S_R and delivers (ID_M, H, S_W, S_R) to S.

O4: Upon receiving a message forward from R, S verifies S_R. If S_R is valid, it then verifies S_W. If S_W is valid, S finds its record of the message matching ID_M and constructs a hash of (ID_M, D, S_S) as V using these recorded values. It then compares V to H.

From these, *C1*'s usage is clear. *O4* also demonstrates how *C2* allows S to detect malicious behavior from W. We elide reputation operations for the moment since they have not been discussed. We address them in V-C and specify the full protocols in VI.

Note that these rules are not exhaustive, particularly with regards to conditional branches. This is to prevent protocol calcification, and we stipulate only that actions taken do not violate protocol semantics.

2) Auditing: While sending provides insight for incoming communications, it does not provide any for outgoing communications. This is why audit operations are necessary. Any participant may initiate an audit query at any time for any message. We discuss only querying for messages from connections participants were members of, but we do not forbid querying for other messages even potentially non-existent messages or messages participants should not know about. While this does not compromise the current design of ClariNet, further investigation would be necessary to determine the safety of decisions based around such queries.

In the following, P_Q is the participant initiating a query, P_A is the queried peer, P_F is the third participant in the communication, and S_S represents the sender's signature. All hashing algorithms use the same algorithm as in V-B1.

O5: P_Q constructs a query message composed of an ID_M and its signature of ID_M as S_{P_Q1}. It delivers (ID_M, S_{P_Q1}) to P_A. Upon receiving P_A's answer A, P_Q verifies S_{P_A}. If S_{P_A} is valid, P_Q hashes its recorded (ID_M, D, S_S) as V, and compares V to the hash P_A provided. Additionally, if S_{P_A} is valid, P_Q signs A as S_{P_Q2} and delivers (A, ID_{P_A}, S_{P_Q2}) to P_F where ID_{P_A} is P_A's ID_P.

O6: Upon receiving a query, P_A finds its record of the message identified by ID_M. It hashes (ID_M, D, S_S) as H. It then signs (ID_M, H) as S_{P_A} and delivers (ID_M, H, S_{P_A}) to P_Q.

O7: Upon receiving a forwarded answer, P_F verifies S_{P_Q}. If S_{P_Q} is valid, it verifies S_{P_A}. If S_{P_A} is valid, P_F finds its record of the message identified by ID_M and hashes the recorded (ID_M, D, S_S) as V. It compares V to the hash provided by P_A.

These audit steps leverage both *C1* and *C2* to allow S and W greater insight into outgoing communications. R is also free to query, but likely gains less due to its better visibility during send. The forwarding operation is necessary to guard against a participant that attempts to claim different messages to different parties. For example, a malicious W might receive M from S but send M' to R. It may then claim M when S queries it and M' when R queries it. While this discrepancy would cast suspicion on W during a subsequent investigation, we would like to allow S and R to discover W's malicious interference earlier and update reputations accordingly.

As in V-B1, we elide reputation operations for now and provide full specifications in VI.

C. Reputation Operations

There are three reputation operations defined below. X represents the affected peer.

1) Reward or $rew(X)$.
2) Weak penalty or $pen_W(X)$
3) Strong penalty or $pen_S(X)$

Rewards are given when a participant observes correct behavior from a peer. We place a slight caveat in that S and W do not give rewards during initial send. There is no particular need for this other than simplifying send logic and can be investigated in future work. Rewards must increase X's reputation, i.e. $rep(rew(X)) > rep(X)$.

Weak penalties are given when a participant observes malicious action, but is unsure of the origin. They are always given to all possible known sources of the malicious action. For example, R might weakly penalize both S and W if it receives an M where S_W is valid, but S_S is invalid. Weak penalties must decrease X's reputation, i.e. $rep(pen_W(X)) < rep(X)$.

Strong penalties are given when a participant observes malicious action and is sure of the origin. These are only given to the origin. Strong penalties must decrease X's reputation by a greater amount than weak penalties, i.e. $rep(pen_S(X)) < rep(pen_W(X))$.

1) Double Counting: It is important we prevent double counting, otherwise situations might arise where a participant penalizes a malicious peer less harshly than a cooperative peer. For example, S and R are cooperative and a malicious W alters the message before forwarding it to R. If S queries W, W reports M so it rewards W. Then S queries R and receives M', so it weakly penalizes both R and W resulting in a final state of $(rep(W) = rew(W) + pen_W(W)) > (rep(R) = pen_W(R))$.

To prevent this, participants must ensure that for every unique (ID_M, ID_P) there exists only one reputation operation. Additionally, the reputation operation tied to each (ID_M, ID_P) is always the harshest that has been observed, with $Pen_S > Pen_W > Rew$. We refer to these as assessments.

VI. Formal State Machines

In the below specifications, we use the same terminology for participants as in V-B1 and V-B2. Additionally, we define the following terms:

- $sign_X(Y)$: X signing Y using the agreed upon signing algorithm.
- $hash(X)$: hashing X using the hashing algorithm used in the agreed upon signing algorithm.
- $verify(X, Y)$: verifying signature Y for data X.
- $verifyHash(X, Y)$: verifying signature Y for pre-hashed X: $verify(X, Y) = verifyHash(hash(X), Y)$.

Additionally, we define direct communication to mean that within a connection, the two participants had no intermediary. As such, S and W directly communicate and W and R directly communicate, but S and R do not directly communicate.

A. O1: Sending

```
1) Construct $ID_M = (ID_C, SeqNo)$
2) Construct $S_S = sign_S(ID_M, D)$
3) Deliver $(ID_M, D, S_S)$ to $W$
```

B. O2: Witness Sending

```
1) Receive $(ID_M, D, S_S)$ from $P$
2) Check $P = S$
     F  $pen_S(P)$ and halt
3) Check corresponding open connection
     T  Continue
4) $verify((ID_M, D), S_S)$
     F  $pen_S(S)$
5) Sign $S_W = sign_W(ID_M, D, S_S)$
6) Deliver $(ID_M, D, S_S, S_W)$ to $R$
```

C. O3: Receiving

```
1) Receive $(ID_M, D, S_S, S_W)$ from $P$
2) Check $P = W$
     F  $pen_S(P)$ and halt
3) Check corresponding open connection
     T  Continue
4) $verify((ID_M, D, S_S), S_W)$
     F  $pen_S(W)$ and halt
5) $verify((ID_M, D), S_S)$
     T  $rew(S)$, $rew(W)$, and halt
     F  $pen_W(S)$, $pen_W(W)$, and continue
6) Construct $H = hash(ID_M, D, S_S)$
7) Construct $S_R = sign_R(ID_M, H, S_W)$
8) Deliver $(ID_M, H, S_W, S_R)$ to $S$
```

D. O4: Receiver Forward Receiving

```
1) Receive $(ID_M, H, S_W, S_R)$ from $P$
2) Check $P = R$
     F  $pen_S(P)$ and halt
3) $verify((ID_M, H, S_W), S_R)$
     F  $pen_S(R)$ and halt
4) $verifyHash(H, S_W)$
     F  $pen_S(R)$ and halt
5) Construct $V = hash(ID_M, D, S_S)$ from $S$'s record
6) Check $V = H$
     F  $pen_S(W)$
```

E. O5: Querying

```
1) Select some message $M$ for which to query
2) Construct $S_{P_Q 1} = sign_{P_Q}(M.ID_M)$
3) Deliver $(M.ID_M, S_{P_Q 1})$ to $P_A$
4) Receive $A = (ID_M, H, S_A)$ from $P_A$
5) $verify((A.ID_M, A.H), A.S_A)$
     F  $pen_S(P_A)$ and halt
6) Construct $V = hash(M.ID_M, M.D, M.S_S)$
7) Check $V = M.H$
     T  $rew(P_A)$
     F  Check direct communication between $P_Q$ and $P_A$
          T  $pen_S(P_A)$
          F  $pen_W(P_A)$, $pen_W(P_F)$
8) Optionally halt
9) Construct $S_{P_Q 2} = sign_{P_Q}(A)$
10) Construct $F = (A, ID_{P_A}, S_{P_Q 2})$ where $ID_{P_A}$ is the
    $ID_P$ of $P_A$
11) Deliver $F$ to $P_F$
```

F. O6: Query Answering

```
1) Receive $Q = (ID_M, S_{P_Q})$
2) Check for record of message $M$ corresponding to $ID_M$
     T  Construct $H = (M.ID_M, M.D, M.S_S)$
     F  Construct $H = null$
3) Sign $S_{P_A} = sign_{P_A}(ID_M, H)$
4) Deliver $(ID_M, H, S_{P_A})$ to $P_Q$
```

G. O7: Query Forward Receiving

```
1) Receive $F = (A, ID_{P_A}, S_{P_Q})$
2) $verify((A, ID_{P_A}), S_{P_Q})$
      T  $pen_S(P_Q)$ and halt
3) $verify((ID_M, H), S_{P_A})$
      F  $pen_S(P_Q)$ and halt
4) Find record of message $M$ corresponding to $ID_M$
5) Construct $V = hash(M.ID_M, M.D, M.S_S)$
6) Check $V = H$
      T  $rew(P_A)$
      F  Check direct communication between $P_F$ and $P_A$
            T  $pen_S(P_A)$
            F  $pen_W(P_A)$, $pen_W(P_Q)$
```

VII. Analysis

We focus only on cooperative participants because ClariNet's goal is to allow cooperative participants insight into which peers may be malicious. Malicious participants' actions only matter in how they affect cooperative participants.

We first define the potential outcomes of an interaction for each connection participant. This includes the messages they may see, the reputation action taken based on those messages, and an explanation of why this is desirable. We then define the different potential combinations of participants in a given connection and use the outcomes discussed to demonstrate how the protocol invariants hold.

A. Sending

1) S: S's only means of gaining insight into its peers' behavior during a send is if R forwards a message. First we define some ground rules to limit the number of possible values for the forward's fields.

Lemma 1. H has only two states: correct and incorrect.

Proof.
1) Because S has a record of the components of H, i.e. ID_M, D, and S_S, it can generate a baseline for comparison, V.
2) Since comparison of H and V is simple boolean equality, only two possibilities exist: $true$ and $false$.
3) Because correctness is a direct mapping from the boolean outcomes, $true \rightarrow correct$ and $false \rightarrow incorrect$, only these two possible outcomes exist for verification of H.

□

Lemma 2. H suffices for S to know the information R claims to have received was modified.

Proof.
1) By definition, $H = hash(ID_M, D, S_S)$.
2) Provided the hashing algorithm is robust, $H = H'$ *iff* the components used to generate H' are identical to those used to generate H.
3) By *lemma 1*, S can know what value H should have and use this to verify H.
4) By (2) and (3), if H is incorrect, S knows at least one of ID_M, D, or S_S does not match what it sent.

□

While H does not include S_W, it has no bearing on what S sent and no bearing on whether R received the message unmodified. Therefore, there is no need for S to verify S_W. This is beneficial because S cannot have a baseline for S_W without additional networking overhead.

In addition to these, S_W and S_R have only two states each due to being cryptographic signatures. Because signatures are well-studied, we omit proof for conciseness.

With these defined, the possibilities become a simple permutation list, yielding $2 \times 2 \times 2 = 8$ possibilities. We detail these in table I.

Lemma 3. Given a message forward containing an invalid H and a valid S_W, S can be sure that W is the source of discrepancy.

Proof.
1) Because we assume that participants do not leak their private keys, R cannot generate a valid S_W since it does not have W's private key.
2) By protocol definition, the same hash algorithm is used to generate both S_W and H.
3) By protocol definition, R generates H from the message it received.
4) By (1), R cannot generate a valid S_W should it attempt to generate H using any different data.
5) Therefore, if S_W is valid, S can safely assume that R generated H from exactly the data W sent.
6) Therefore, S can safely assume that W held modified data while generating S_W.
7) By protocol definition, S and W directly communicate.
8) Therefore, no intermediary could have modified the data between S and W.
9) Therefore, W must be the one who modified the data resulting in H and S_W.

□

2) W: W can only make assessments based on the signature in the message it receives from S. We detail these in table II. In *WS1* note that W does not reward S despite having proof of what S sent. We took this approach to simplify W's behavior during send and since W likely needs to perform some audit operations anyway, but should future work wish to add a reward here, the protocol and invariants as discussed should still hold.

3) R: R can make assessments based on both signatures in the message it receives from W. This yields 4 possible outcomes which we detail in table III. In **RS2** and **RS4**, R does not alter S's reputation. We do this for protocol simplicity—an invalid S_W means R always halts assessment and can make assessments during a later query. Additionally, it introduces some conservativism in penalties because R cannot forward in **RS4**.

B. Auditing

All participants follow the same auditing behavior; only penalties differ based on direct communication. Because of

TABLE I
S SEND INTERACTIONS

Outcome ID	H	S_W	S_R	**Action**	**Action Reason**	**Malicious Participant Reason**
SS1	Correct	Valid	Valid	No action	S has proof R receives M correctly	Wouldn't occur because no benefit
SS2	Correct	Valid	Invalid	$pen_S(R)$	S knows R violated protocol	Wouldn't occur because no benefit
SS3	Correct	Invalid	Valid	$pen_S(R)$	S knows R violated protocol	Wouldn't occur because no benefit
SS4	Correct	Invalid	Invalid	$pen_S(R)$	S knows R violated protocol	Wouldn't occur because no benefit
SS5	Incorrect	Valid	Valid	$pen_S(W)$	S knows W altered M by *lemma 3*	Malicious W is attempting to turn S and R against each other
SS6	Incorrect	Valid	Invalid	$pen_S(R)$	S knows R violated protocol	Wouldn't occur because no benefit
SS7	Incorrect	Invalid	Valid	$pen_S(R)$	S knows R violated protocol	Wouldn't occur because no benefit
SS8	Incorrect	Invalid	Invalid	$pen_S(R)$	S knows R violated protocol	Wouldn't occur because no benefit

TABLE II
W SEND INTERACTIONS

Outcome ID	S_S	**Action**	**Action Reason**	**Malicious Participant Reason**
WS1	Valid	No action	W has proof S sent M	Malicious S knows it won't want to later deny the message
WS2	Invalid	$pen_S(S)$	W knows S violated protocol	Malicious S wishes to later deny having sent M

TABLE III
R SEND INTERACTIONS

Outcome ID	S_S	S_W	**Action**	**Action Reason**	**Malicious Participant Reason**
RS1	Valid	Valid	rew(S), rew(W)	R has proof S sent M and W witnessed M	Malicious S and/or M know they won't want to deny the message
RS2	Valid	Invalid	$pen_S(W)$	R knows W violated the protocol	Wouldn't occur because no benefit
RS3	Invalid	Valid	$pen_W(S)$, $pen_W(W)$	R knows malicious action occurred, but cannot be sure if S or W is source	Malicious S wants to later deny M or malicious W wants to turn S and R against each other
RS4	Invalid	Invalid	$pen_S(W)$	R knows W violated the protocol	S_M and W_M want to deny sending and witnessing M or W_M wants to prevent R from Fwd_M

this similarity, we define the overall possibilities and separate some of the reputation operations. We outline the query outcomes in table IV, the forward interactions in table V, and the additional penalty operations in table VI. In VI, O represents the other participant in the communication.

C. Scenarios

A connection has three participants which can each be cooperative or malicious. Permuting these, we arrive at eight possible configurations shown in table VII.

The malicious participants may either be colluding or acting independently. For this analysis, we assume all malicious participants are colluding because malicious participants benefit from following protocol behaviors to discover non-colluding malicious peers and assisting all other peers in removing them.

Lemma 4. Malicious participants benefit from eliminating non-colluding malicious peers from other peers' witness candidate pools.

Proof. A malicious participant P_M wishes to exploit the consensus protocol to win a claim over the other participant in the communication. To do this, it must have its own record and a W supporting its claim. While any two form the necessary consensus, a dispute would not occur if S and R agree because W only logs the message without taking action.

P_M also wishes to ensure that, should it false report, W's false report matches. If W reports correctly, P_M loses the dispute via the consensus protocol and if W false reports differently, either P_M still loses the dispute if W agrees with the other side or all three participants have different answers. In the case where all three participants have different answers, no one wins the dispute and all three are likely to be placed under additional scrutiny, which malicious participants wish to avoid as it may reveal their malicious activities.

Because ClariNet does not require a specific witness agreement algorithm, P_M may not be able to force a colluding W. It thus benefits from making its colluders appear more trustworthy to the other side by causing the other side to penalize malicious non-colluders. □

Lemma 5. Malicious participants benefit from retaining their own record of peers' reputations for non-colluding peers.

Proof. Following from *lemma 4*, since malicious participants wish to assist peers in removing non-colluding malicious participants, it is trivial for them to follow the protocol where it benefits them and track non-colluding peers who may be

TABLE IV
QUERY INTERACTIONS

Outcome ID	H	S_T	**Action**	**Action Reason**	**Response Reason**
Q1	Correct	Valid	$rew(P_A)$	P_Q has proof P_A sent, witnessed, or received M	P_A is cooperative or malicious P_A does not wish to deny M
Q2	Correct	Invalid	$pen_S(P_A)$	P_Q knows P_A violated the protocol	Malicious P_A wishes to prevent forwarding
Q3	Incorrect	Valid	See table VI	Q See table VI	Malicious P_A wishes to claim alternate message or cooperative P_A received modified message
Q4	Incorrect	Invalid	$pen_S(P_A)$	P_Q knows P_A violated protocol	Malicious P_A wishes to claim an alternate message while preventing forwarding

TABLE V
QUERY FORWARD INTERACTIONS

Outcome ID	H	S_T	S_Q	**Action**	**Action Reason**	**Response/Forward Reason**
F1	Correct	Valid	Valid	$rew(P_A)$	P_F has proof P_A sent, witnessed, or received M	P_A is cooperative or malicious P_A does not wish to deny M
F2	Correct	Valid	Invalid	$pen_S(P_Q)$	P_F knows P_Q violated the protocol	Wouldn't occur because no benefit
F3	Correct	Invalid	Valid	$pen_S(P_Q)$	P_F knows P_Q violated the protocol	Wouldn't occur because no benefit
F4	Correct	Invalid	Invalid	$pen_S(P_Q)$	P_F knows P_Q violated the protocol	Wouldn't occur because no benefit
F5	Incorrect	Valid	Valid	See table VI	See table VI	Malicious P_A wishes to claim alternate message or cooperative P_A received modified message
F6	Incorrect	Valid	Invalid	$pen_S(P_Q)$	P_F knows P_Q violated the protocol	Wouldn't occur because no benefit
F7	Incorrect	Invalid	Valid	$pen_S(P_Q)$	P_F knows P_Q violated the protocol	Wouldn't occur because no benefit
F8	Incorrect	Invalid	Invalid	$pen_S(P_Q)$	P_F knows P_Q violated the protocol	Wouldn't occur because no benefit

TABLE VI
QUERY/FORWARD PENALTIES

Communication Type	**Action**	**Reason**
Direct	$pen_S(P_A)$	P_Q or P_F can be sure P_A is the source of malicious action
Indirect	$pen_W(P_A)$, $pen_W(O)$	P_Q or P_F knows malicious action occurred, but cannot be sure if P_A or O is source.

TABLE VII
CONNECTION PARTICIPANT POSSIBILITIES

Scenario No.	**Sender**	**Witness**	**Receiver**
(1)	Cooperative	Cooperative	Cooperative
(2)	Cooperative	Cooperative	Malicious
(3)	Cooperative	Malicious	Cooperative
(4)	Cooperative	Malicious	Malicious
(5)	Malicious	Cooperative	Cooperative
(6)	Malicious	Cooperative	Malicious
(7)	Malicious	Malicious	Cooperative
(8)	Malicious	Malicious	Malicious

malicious. This allows them to influence whatever witness agreement algorithm toward cooperative peers in the event they cannot steer it toward one of their colluders.

Given these considerations, it is in their best interest to behave cooperatively with regards to non-colluding malicious peers, such as penalizing and forwarding appropriately. □

By lemmas 4 and 5, we can simplify the above scenarios so that when two malicious participants are present they are colluding since one that isn't colluding behaves cooperatively. In addition to this, the primary invariants ClariNet wishes to maintain are:

1) Malicious action is always penalized.
2) A cooperative participant in a connection always correctly penalizes a malicious peer at least as much as it incorrectly penalizes cooperative peers.
3) Within a connection, the total reputation for malicious participants decreases more quickly than cooperative participants.

Given these, we exclude the following scenarios:

(1) By definition, none of the participants are malicious so no malicious action occurs and there are no penalties to analyze.

(4) Because there are two malicious participants and only one cooperative participant, there is no risk of accidentally penalizing cooperative peers guaranteeing malicious peers are penalized more harshly.

(6) Since S and R are colluding, there is no reason for them to end up in a dispute.

(7) Same as (4).
(8) Same as (6).

This leaves 3 scenarios to analyze: (2), (3), and (5). It might seem that a malicious participant would not act maliciously when it knows W is not a colluder, but a particularly aggressive malicious participant may still behave maliciously and attempt to claim that W and the other participant are malicious colluders.

D. Scenario (2)

By table I, the malicious R never forwards a message, and because both S and W are cooperative, both always provide valid signatures. This means no penalties occur during send. When querying, S and W always receive the correct data from each other, so they initially reward each other, but when either queries R, R may wish to deny it received M.

By table IV, R's most likely choice is **Q3**. **Q1** undermines its ability to claim a different message and **Q2** and **Q4** mean it is strongly penalized by S. **Q3** allows R to maintain its false claim, only be weakly penalized by S, and even cause S to weakly penalize W.

However, because W and R directly communicated, all of **Q2**-**Q4** cause W to strongly penalize R. We collect the penalties in table VIII, with $rew = 1/1$, $pen_W = 0/1$, and $pen_S = 0/3$.

TABLE VIII
SCENARIO (2)

	S	W	R
S	-	0/1	0/1
W	1/1	-	0/3
total	1/1	0/1	0/4

E. Scenario (3)

This scenario proves the most problematic because of W's position of power. It can alter the message to turn R and S against each other while also being able to answer both with their expected messages. The forwarding operations mitigate this. For example, if W alters M during send, R can forward the message and because of the presence of the valid S_W, S can strongly penalize W. Similarly, when S queries W, W must choose among **Q2**-**Q4**. **Q2** means S can forward the answer to R who would then strongly penalize W. **Q3** and **Q4** mean S would strongly penalize W, but W would retain its reputation with R either by preventing forwarding or supplying the A R expects. Regardless, W suffers a strong penalty from S, R, or both and can only cause S and R to weakly penalize each other.

Table IX displays the potential outcomes once both S and R have finished querying and forwarding. We very briefly discuss the sequence of events that might lead to each.

TABLE IX
SCENARIO (3)

	W attempts to maintain both			W sacrifices rep with S			3. W sacrifices rep with R		
	S	W	R	S	W	R	S	W	R
S	-	0/3	0/1	-	0/1	0/1	-	0/3	0/1
R	0/1	0/3	-	0/1	0/3	-	0/1	0/1	-
total	0/1	0/6	0/1	0/1	0/4	0/1	0/1	0/4	0/1

1) W attempts to maintain both: W alters M and includes a valid S_W, which causes R to forward the message and S to $pen_S(W)$. Later, S queries R, sees the diverging message, and $pen_W(R)$ while W keeps its pen_S. S queries W sees what it expects, but W keeps its pen_S. S forwards this answer to R who sees W's valid signature with a different message and $pen_S(W)$.

2) W sacrifices rep with S: W alters M and includes a valid S_W, which causes R to forward the message and S to $pen_S(W)$. Later, S queries R, sees the diverging message, and $pen_W(R)$ while W keeps its pen_S. S queries W and gets an invalid signature so it can't forward to R.

When R queries W it sees what it expects, but when it queries S, it sees diverging data with S's valid signature and $pen_W(S)$ and $pen_W(W)$.

3) W sacrifices rep with R: W alters M and includes an invalid S_W, which causes R to $pen_S(W)$ but prevents it from forwarding the message to S. Later, S queries R, sees the diverging message, and $pen_W(W)$ and $pen_W(R)$. S queries W and sees what it expects.

When R queries W, W includes an invalid signature to prevent R from forwarding the answer and W keeps its pen_S. When R queries S, it sees diverging data with S's valid signature and $pen_W(S)$ and $pen_W(W)$.

F. Scenario (5)

If S wishes to maintain deniability, it must provide an invalid S_S which would cause W to $pen_S(S)$ and R to $pen_W(W)$ and $pen_W(S)$ since the cooperative W provides a valid S_W. R is unlikely to gain further insight during queries since S most likely uses **Q3** because it knows W already strongly penalized it and can supply an H matching its alternate data[1]. Similarly, W supplies **Q1** to R so R never does harsher than $pen_W(W)$. W however can $rew(R)$ because R provides **Q1**. We compile these in table X.

G. Malicious Interference

In *AA1* we specified that adversaries are capable of intercepting and modifying messages, but that they do not wish to halt all communication between non-colluding peers.

[1]In theory, S could provide an $A.H$ to R matching the initial message, including the invalid signature. Were this to happen, R could potentially $pen_S(S)$ and $rew(W)$ because it has proof directly from S that W behaved and S is the source of the invalid signature, but we omit this because it adds complexity to protocol logic and is unlikely to ever happen since S does not benefit.

TABLE X
SCENARIO (5)

	S	W	R
W	0/3	-	1/1
R	0/1	0/1	-
total	0/4	0/1	1/1

Additionally, all communication uses AKE with an ephemeral key to set up a session key and cooperative participants do not leak keys. This means that while malicious participants can intercept messages, they cannot successfully read or modify them. If they attempt to modify the cyphertext, the recipient's decryption will fail which is effectively a corrupted message and would trigger a retransmission of the underlying reliable transport protocol.

Because malicious participants cannot read or successfully modify intercepted messages, they must rely on heuristics such as timing and messaging patterns if they wish to block potentially harmful messages. ClariNet is by design an eventually consistent protocol which helps to mitigates this. Message and answer forwards need only be delivered at some point in the future; their senders are free to attempt this delivery immediately, batch them, or queue them for when the sender is otherwise inactive. Additionally, since communication uses reliable transport protocols, participants can be sure that the recipient received the message correctly. Should delivery fail at one time, they can try again later. Because of the variability in these communications, it means malicious participants must take on significant additional state and heuristics may be unreliable, even potentially blocking beneficial messages. For example, a malicious participant may drop something it believes to be an answer forward. This might actually be connection setup, teardown, or a data message. Even if it is correct, the answer forward may be for a different, non-colluding malicious peer which, as discussed, the participant would like to be penalized.

1) Key Sharing: *AA2* does allow malicious participants to share their private keys and even session keys. ClariNet cannot detect this directly, but it does still detect any malicious action that these peers might perform. While it would result in penalizing the wrong peer, only malicious participants share their keys, so even though the participant that shared its key is penalized instead of the malicious actor, a malicious participant is still correctly penalized. Additionally, if a malicious participant is willing to share its key with a peer, it almost certainly means the two are colluding and a member of the collusion group is still correctly penalized.

VIII. PROTOCOL OVERHEAD

While VII demonstrates that ClariNet's invariants hold, a protocol is not practical if it requires excessive overhead. In this section we explore ClariNet's overhead compared to a generic protocol that retains similar logging, just omitting the witness and audit operations. Overhead is divided into two categories: network and storage. Network overhead includes both latency and added network load from additional messages. Storage addresses the additional space necessary. We start with network.

A. Network

1) Latency: We only discuss latency with regards to the operations necessary to transmit an initial data message from S to R. Audit operations can be performed asynchronously and do not have a strict deadline. The send meanwhile presumably has some desired deadline or latency requirements.

This varies significantly based on participant location and network size, but can be generalized to an average of the peers a participant communicates with. We define this average to be L. In the baseline, the latency is just L because S delivers the message directly to R.

If we assume a fully connected network with no particular preference for communication partners[2], this average applies to both the channel between S and W and the channel between W and R. Because the send involves both channels, send latency is $2L$. While this is not ideal, it should be acceptable in many situations.

2) Load: Both the hypothetical baseline and ClariNet transmit the same data during send. We measure load in terms of bytes transmitted on the network, so this does not change despite both S and W needing to transmit the data. As such, the data transmitted cancels out and we can focus solely on the additional fields and messages ClariNet requires. Table XI defines some reasonable component implementations and their associated sizes in bytes. We use 64-bit integers because it would allow sending over 9 quintillion ($9e18$) message in a single connection. It would take sending 28 billion messages per second for 10 years to exhaust this.

TABLE XI
PROTOCOL COMPONENT SIZES

Component	Implementation	Size (bytes)
ID_C	UUID	16
Sequence Number	64-bit integer	8
Hash	SHA-256	32
Signature	Encrypted hash	32

Sending requires, in addition to the data, an $ID_M = (ID_C, SeqNo)$, S_S, and S_W. The base case also requires S_S, so this cancels. It also likely includes a correlation ID [39] or trace ID [40] which may be something similar to a UUID. The ID_M can serve as a correlation ID within a connection, but an additional trace ID would be necessary for more extensive workflows, so we still count ID_M as overhead. This gives a total overhead of 56 bytes. While S_W is not included in the message S delivers to W we retain it in full to provide an upper bound for the most expensive portion.

[2]We acknowledge this is unrealistic for real-world networks, but for abstract analysis this should suffice.

In addition to the initial send, there is receiver forwarding. This consists of an ID_M, a hash, S_W, and S_R for a total of 120 bytes. In total, sending entails an overhead of 176 bytes.

Auditing a message consists of an initial query, an answer, and an answer forward per participant per peer involved in the connection. Since there are always 3 participants in a connection, it means that each participant would query once per peer for a total of two queries. While participants could query more than once per peer, they would not gain any benefit since it is unlikely a peer would change its reporting. This yields a max of 6 total queries (2 queries per each of 3 participants). This may be lower, for example if a participant does not need to query a peer because it received a fully informative answer forward, but we consider it in full to provide an upper bound.

A query consists of ID_M and a signature totaling 56 bytes. An answer consists of ID_M, a hash, and a signature totaling 88 bytes. An answer forward consists of the 88-byte answer, P_A's ID_P, and the forwarder's signature. ID_P is the hash of the public key and is therefore also 32 bytes, so the answer forward totals 152 bytes. In total a query entails 296 bytes which yields a total overhead of $1,776$ bytes across all 6 queries.

In total ClariNet requires $1,952$ bytes of overhead per message. We collect these numbers in table XII. While this is not completely negligible, we feel it is not excessive. For comparison, the SPDY whitepaper [41] found that HTTP headers range from 200 bytes to 2KB with typical cases being 700-800 bytes for more modern sites. While overhead must always be considered relative to data, these numbers demonstrate that ClariNet does not use excessively more overhead than commonly deployed tools. Additionally, much of this overhead is in messaging that does not have a strict schedule and can factor in conditions such as current network load.

TABLE XII
MESSAGE SIZES

Message	Components	Overhead (bytes)
Send	(ID_M, D, S_S, S_W)	56
Receiver Forward	(ID_M, H, S_W, S_R)	120
Query	(ID_M, S_{P_Q})	56
Answer	(ID_M, H, S_{P_A})	88
Answer Forward	(A, ID_C, S_{P_Q})	152

B. Storage

We assume both the baseline system and the ClariNet implementation log all messages. We additionally assume that the ClariNet implementation uses a queriable log to answer queries. Queriable logs do entail overhead in and of themselves, but are becoming more prevalent and vary by implementation, so we discuss only the ClariNet-specific overhead. We compile these components and their scale factor in table XIII. N_M is the messages a participant sends or receives, N_{WM} the messages a participant witnesses, N_P the number of known peers, and N_C the number of connections.

1) Messages: Since participants log all messages they send or receive, much of the storage overhead is the same as the network load overhead VIII-A2. In addition to this, participants need to store assessment records to ensure that the harshest reputation operation is always maintained. While the assessments could be stored as separate fields, because participants store the messages anyway, they can include the assessments as an additional field on the message record. An assessment has effectively 4 states, none and the 3 reputation operations, which can be represented as 2 bits, and a message always involves 3 participants so assessments can be stored in a single additional 8-bit integer field[3].

2) Additional: In addition to the assessment records, participants need to maintain records of connections and might keep an explicit reputation score to reduce calculations at the cost of some additional storage. Connection information needs to be retained so participants know the correct peers to reward or penalize and consists of the ID_C, and ID_Ps of the sender, witness, and receiver for a total of $112N_C$ bytes. The reputation score could be kept as a double precision floating point with one entry per known peer for a total of $8N_P$ bytes.

3) Witness: In the baseline system, witnesses are not involved, so all messages a participant witnesses entail storage overhead both for sending and auditing. Fortunately, it is only the witness's role to verify the claims of the sender or receiver when a dispute should arise. This means that witnesses do not have to store the entire message. Instead, they can store the ID_M, S_S, and a hash of the message, identical to the one they would need anyway for answering queries. This gives an additional storage overhead of $88N_{WM}$ bytes.

4) Total: The total storage for the baseline implementation is $N_M(D_{avg} + S_S)$ bytes. The total supplemental storage for ClariNet is $1953N_M + 112N_C + 8N_P + 2041N_{WM}$ bytes.

$N_C \leq N_M + N_{WM}$ almost certainly holds. The only way it would not is if a participant opens many connections without sending anything on them, and even in that case, peers can remove the connection records after they close because it has no messages to assess and therefore no need to remember the particular participants.

$N_P \leq N_M + N_{WM}$ can hold relatively easily. Participants can use a dynamic default reputation score for peers they have not interacted with in any fashion and participants need at least one message to assess.

Finally, because N_M is the same as in the baseline, the bytes from N_M are fixed overhead relative to the baseline.

This means that the primary source of storage overhead is witnessed messages and grows linearly. While this does somewhat disincentivize being a witness, it is offset by the greater visibility witnesses have into peers' behavior.

[3]While only 6 bits are completely necessary, many systems do not have much support for integer values less than 8 bits and the 2 additional bits are likely negligible.

TABLE XIII
STORAGE SIZES

Component	Implementation	Individual Overhead (bytes)	Scale Factor
Sent & Received Messages	Queriable log of table XII	1952	N_M
Witnessed Messages	Queriable log of table XII with hash of data	2040	N_{WM}
Assessments	8-bit integer attached to messages	1	$N_M + N_{WM}$
Reputation Scores	Double precisioin floating point	8	N_P
Connection Information	Queriable log	112	N_C

IX. Empirical Evaluation

In addition to theoretical analysis of a single message, it is important that ClariNet provide tangible benefit when applied at scale. We use PeerSim [32] to implement messaging and reputation largely as discussed. We make some slight modifications to ease implementation, like transmitting data directly rather than hashing because network bandwidth is not a concern, and implementing signatures as an enum with values “valid” and “invalid” rather than real cryptographic signatures. These do not compromise the protocol because all the same checks are present and potential values are appropriately represented.

We discuss unspecified behaviors, configuration, and participant actions, both cooperative and malicious, in the following subsections.

A. Reputation

1) Cooperative: Reputation is relatively simple consisting of a ratio of two counts, good and total. All participants begin with a count of 1 in each. Rewards increment both by 1, weak penalties increment total by 1, and strong penalties increment total by 3. For example a peer a participant has rewarded twice, weakly penalized 3 times, and strongly penalized once would have $1/1 + 2 \times (1/1) + 3 \times (0/1) + 0/3 = 3/9$.

Starting with a baseline of $1/1$ has the benefits of making unassessed peers not a special case, defaulting to trusting these unassessed peers, and allowing quick differentiation between weak and strong penalties. If the counts started at 0, then a weakly penalized peer $(0/1)$ and strongly penalized peer $(0/3)$ would both appear as 0 while 1 baseline would be $1/2$ and $1/4$.

2) Malicious: As discussed in VII-C, we assume all malicious participants are colluding and aware of each other. Because malicious participants do not need to determine non-colluding malicious peers, they do not need to track reputation. We still have them receive forwarded messages and generate their own queries, but they take no reputation action based on these.

B. Witness Agreement

1) Cooperative: We take a very basic approach to connection setup and witness agreement. Senders always initiate connections and select the witness. Cooperative senders calculate reputations for all their known peers and pass these reputations through a simple filter that removes peers the participant considers untrustworthy. The filter calculates the average reputation of all peers with at least one assessment, then considers a peer malicious if it has $reputation \leq 0.5 \vee reputation < (average - standardDeviation)$. The sender randomly selects from this list of trusted peers, and requests them to witness until one agrees. Participants only refuse to witness if they already have the maximum number of permitted open connections, which we set to an arbitrary value of 10. If no peer is able to witness, the sender terminates the connection.

2) Malicious: Malicious participants follow the same behavior of initiation and witness selection when they are the sender. They differ in their candidate selection. First they attempt to request all their known colluding peers to ensure they win the consensus protocol. This simulates their attempting to influence the witness selection agreement in favor of a colluder. If they are unable to find a colluder capable of witnessing, they then randomly select cooperative peers.

C. Configuration

Table XIV displays the configurable parameters with values we used. We permuted over all possible combinations, and also ran 5,000 and 10,000 participant scenarios using only 100 cycles for a total of 3,060 simulations. We briefly explain a few terms in more detail.

1) Number of cycles: PeerSim implements what it refers to as a cycle and runs the simulation for a fixed number of cycles. Cycles are rounds in which participants can initiate an action. A participant will never initiate more than one action per cycle, but may respond to actions initiated by a peer in the same cycle it acted. PeerSim does allow configuring the number of participants than can initiate an action per cycle. We allow all participants to initiate an action each cycle.

2) Malicious action threshold %: This simulates malicious participants that build up reputation prior to beginning their malicious actions. We do this with a simple counter that checks if it is past a certain threshold. The threshold is the number of cycles multiplied by the decimal form of the percentage. For example, in the 10,000 cycle scenario, if the threshold % is 40%, the counter would need to exceed 4,000.

The counter increments whenever a malicious participant performs an action that may be malicious. This includes sending data, witnessing a message, and responding to a query.

TABLE XIV
SIMULATION PARAMETERS

Parameter	Values	
Node count	The number of participants in the network	100, 250, 500, 750, 1,000
Number of cycles	The number of rounds the simulation runs	100, 1,000, 10,000
Malicious %	Percentage of participants that are malicious	10%, 20%, 30%, 50%, 90%
Malicious action threshold %	A threshold before which malicious always behave cooperatively	10%, 20%, 30%, 50%, 70%, 90%
Malicious action %	The probabilistic odds of a malicious participant acting maliciously after the threshold	10%, 20%, 30%, 50%, 70%, 90%

D. Participant Behavior

1) Cooperative: Cooperative participants randomly choose between one of 5 actions with equal probability.

1) Open a connection with a randomly selected participant in the network.
2) Send a message on a randomly selected open outgoing connection.
3) Query one peer for a randomly selected message.
4) Close a randomly selected outgoing connection.
5) Take no action.

They always behave properly, such as setting their corresponding signature value to the valid state and honestly responding to queries.

If a participant selects an action it cannot successfully complete it takes no action. This could occur for example if it attempts to send but has no open outgoing connections or if it has already queried both peers for all messages.

2) Malicious: Malicious participants use the same 5 actions with the same probabilities. The only difference is they probabilistically act maliciously based on the configurable parameters. This includes behaviors such as sending data with an invalid signature, altering witnessed data (including setting sender signature to invalid), and responding to a query with falsified data. As discussed in VII-C, when S and R are malicious, they do not act maliciously.

Malicious participants do not consider other actions when deciding to take malicious action. For example, they may supply a valid signature on a message and later attempt to claim falsified data. We feel this is a reasonable simplification since messages are not double counted and the simulation focuses on reputation trends. For example a send that later has a penalized query would in reality have taken malicious action during the send, but in both cases amounts to a single penalty for that message. Still, this dumb acting would increase the percentage of messages for which a participant behaves maliciously. This is not as dire as it may initially seem for a few reasons. First, the goal of the simulation is to observe general trends at scale rather than precise values. Second, it is somewhat offset by the threshold since that potentially doesn't increment for a given cycle. Third, it is additionally offset by the fact that only two of the five actions a participant can take provide any reputation insight.

X. RESULTS

We collected the following data that capture participants' views of the network:

1) Whether the participant was cooperative or malicious
2) Individual peer information such as:
    a) The peer's reputation with the participant
    b) Whether the peer was cooperative or malicious
    c) Whether the peer was trusted or not
3) Aggregate information for assessed peers:
    a) Average reputation
    b) Minimum reputation
    c) Maximum reputation
    d) Standard deviation
    e) Total
    f) Number trusted
    g) Number not trusted

One set of aggregate information exists for each of all assessed peers, cooperative assessed peers, and malicious assessed peers. We used PeerSim's global knowledge to facilitate these breakdowns, but cooperative participants were not aware of peer status during the simulation.

Due to the number of samples, we focused analysis on the simulations with 100 cycles. Initial analysis seemed to indicate that higher cycle counts showed similar overall trends and 100 cycles, being the minimum, demonstrates convergence speed. We consolidated this data by averaging the aggregate statistics of all the cooperative participants within a given simulation for their view of cooperative and malicious peers. We then averaged this over all simulations that had the same number of participants to get the final graphs.

Figures 3 and 4 plot the reputation against another parameter. Each plots the simulations broken down by the malicious action threshold percentage. We omitted 20% simply to reduce the number of figures and 90% because it did not display any interesting results. It appeared that 90% was so high that nearly no malicious participants misbehaved resulting in nearly all peers having the max reputation of 1.

Both show that as malicious participants wait longer to begin acting maliciously it becomes harder to detect them. This is expected because waiting longer means there are fewer malicious actions to penalize. Even 70%, despite seeming quite similar, is still an encouraging result. Because only two

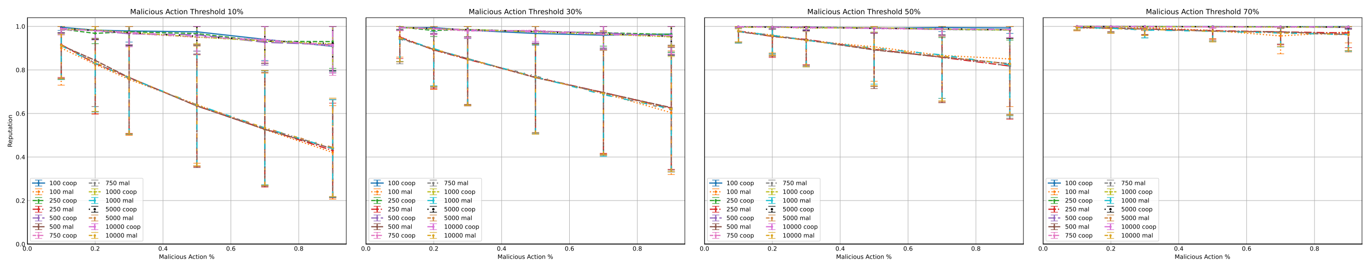

Fig. 3. Malicious Action % Reputations

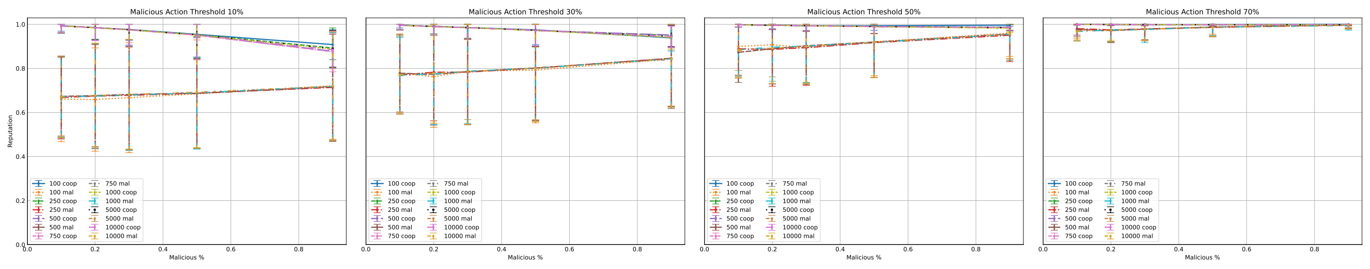

Fig. 4. Malicious % Reputations

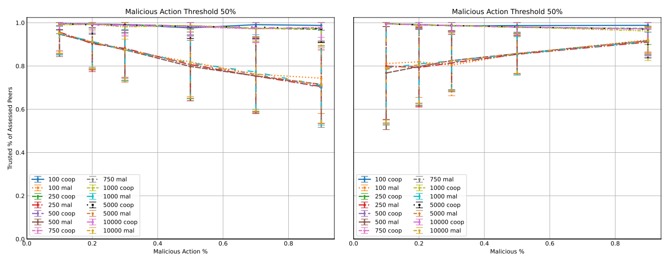

Fig. 5. Trust % of Assessed Nodes

out of the five actions a participant or its peers could take contribute to reaching this threshold, malicious participants do not necessarily being acting maliciously 70% of the way through the simulation. Some may reach the threshold earlier and others likely later. Given the odds of taking an action that does not contribute, we suspect that malicious participants more commonly reached the threshold over 70% of the way through the simulation. A more robust simulation might differentiate between malicious participants that actually acted malicious and those that never did, effectively being cooperative.

Figure 3 demonstrates that as peers act maliciously more often their reputation decreases more significantly. Cooperative peers suffer some decrease as well because of erroneous weak penalties, but are significantly more stable at a higher overall reputation, indicated by a smaller standard deviation. Malicious peers likely have a larger standard deviation due to the variance in whether they acted maliciously and whether the cooperative participants happened to assess them.

Figure 4 shows that as the percentage of malicious participants increases their overall reputation also increases while cooperative peers' reputations decrease. Malicious peers' reputation may increase because of their behavior of not acting maliciously when S and R are both malicious. Because more peers are malicious, it increases the odds of this configuration and decreases the overall malicious activity that cooperative participants witnessed. Additionally, because there are more malicious participants, cooperative participants may have ended up spreading penalties over a larger number of peers resulting in fewer penalties per malicious peer. Similarly, because there are fewer cooperative participants, erroneous false penalties may be more concentrated resulting in more penalties per peer. Even considering this, there is still typically a noticeable difference between cooperative and malicious peers.

Figure 5 shows the percentage of assessed peers a participant trusts corresponding to the third graph in each of figure 3 and 4. We include these to demonstrate that there is a significant correlation between the reputation and which peers a participant trusts.

XI. Limitations & Future Work

One of the largest limitations is that ClariNet is an entirely reactive protocol. It needs malicious activity to differentiate cooperative and malicious peers and subsequently can only guard against repeat offenders rather than proactive protection. In the worst case, the witness is a colluder, and the participant has no recourse should a dispute arise for that particular message. However, protection from repeat offenders is better than no protection and participants' ability to gain insight while serving as a witness helps them assess their peers without themselves being at risk.

The presence of only a single witness is also a limitation. Ideally, ClariNet would support multiple witnesses, but we limited analysis to one to constrain complexity. We feel this is reasonable for an initial exploration. Supporting multiple witnesses is a prime area for future work.

The current analysis does not investigate malicious participants blocking communication between peers. While ClariNet does have traits such as its eventual consistency nature, using AKE, and fully encrypted messages that guard against this, heuristics can still be effective. It would be interesting to investigate heuristics malicious participants might use, how effective they prove, and potential mitigations.

Similarly, the current analysis does not factor in malicious participants that claim to have sent or received messages that were never sent. This could be particularly problematic because participants cannot make assessments on messages they are not aware of. While they may be able to take certain steps like including messages they were queried about as query candidates, this would require additional analysis to ensure it is not exploitable and would still not guard against entirely fabricated messages.

As it stands, ClariNet requires a reasonably stable network. In highly dynamic networks where participants frequently join and leave or change identities, participants may not have enough time to conduct extensive audit operations to form a solid reputation for peers. The benefits from serving as a witness do encourage participants to stay active in the network while able, but this is not always practical, and on the whole means ClariNet may not be suitable for certain types of networks.

Additionally, the conditions for applying reputation operations are quite simplistic. There are likely more intelligent decisions participants could make based on combinations of messages to prevent erroneous penalties or better apply strong penalties. While we feel limiting these conditions for initial analysis was reasonable to reduce complexity, investigating and analyzing more intelligent conditions would be an excellent area for future work.

And lastly, empirical analysis was rather limited. We used only simulation with randomly generated traffic on an entirely stable network with minimally intelligent malicious participants. While randomization can serve as a reasonable approximation of large-scale systems, it would be better to test with communication patterns drawn from real-world distributed systems including participants joining and leaving at varying intervals. We would also like to investigate the system with more intelligent malicious participants and build a real, deployable version of ClariNet to test in a real distributed environment.

XII. Related Work

In this paper, we defined ClariNet, a protocol for evaluating peer reliability in three-party communications. While all the building blocks ClariNet uses have been extensively studied and even combined, to our knowledge no existing works combine them in an eventual-consistency model geared toward audit logging in the manner ClariNet does.

Some works [28] use cryptographic signatures to ensure that both sides of an exchange are able to prove that the other took part. However, these protocols do not maintain this accountability in the absence of valid signatures, such as those that are intentionally corrupted or altogether omitted. ClariNet uses a consensus protocol as a secondary measure and can even leverage invalid signatures to allow participants insight into peer reliability.

Other works [1], [17], [21] and surveys [11] address this issue by ensuring that neither side is able to gain an advantage during the data exchange. They often do this by conducting iterative, bit-by-bit transfer so the message either side wishes to receive is not usable until it is completely received. In doing so, they ensure that either both sides obtain proof of exchange or that neither side has anything. However, this exchange paradigm may not be a natural fit for some domains, like pub/sub or content delivery systems. Additionally, this bit-by-bit exchange must be performed during delivery which may introduce undesirable overhead. Clarinet allows senders to lazily obtain proof of receipt when they have capacity and allows for full transmission of data at once. It is flexible enough in its constraints that it could feasibly be adapted to augment such non-repudiation systems or even altered to implement such a system within its own messaging.

Another concern faced by such iterative non-repudiation systems is that if one side attempts to cheat, the other side may have difficulty proving the cheater caused the termination. ClariNet does not fully solve this problem, but its consensus protocol can provide support for the behaving party's claim. Approaches such as optimistic fair exchange [19] provide remuneration should one party attempt to cheat through the presence of an arbitrator, but are reliant on a trustworthy arbitrator. Some works [20] address malicious arbitrators by ensuring the arbitrator can be held accountable for its actions. ClariNet gives participants insight into peer trustworthiness when making an initial selection. Both approaches have value and could even potentially be combined to compliment each other.

More recent work on non-repudiation [29], consensus protocols, and consensus reputation [42]–[44] and surveys [6]–[8] leverage blockchain [13]. These protocols often use proof-of-work or proof-of-stake to guard against altering transaction history. Wile robust, this approach has raised environmental concerns [27] and the computational overhead and transmitting to all peers can introduce noticeable latency [23], [25]. As such, blockchain may not be desirable. ClariNet does not seek to supplant such systems; rather it can provide a less resource-intensive alternative. We also suspect that ClariNet could be adapted and combined with these more robust systems to gain the benefits of both. We acknowledge that such a combination would not be trivial, but we suspect that both protocols are flexible enough in their core requirements to not preclude it.

There does exist research on non-blockchain consensus protocols [24] and Byzantine agreement protocols [12], [18], but many of these topics are older works. While their core theory is sound, they often were designed with a different goals in mind, such as accommodating faulty hardware rather than active manipulation, or focus on upper-bounding agreement time. ClariNet is geared toward detecting malicious parties so they can be excluded from future committee construction.

Even with these differing goals, ClariNet could use findings of such research to augment its own consensus protocol.

There is a wealth of work on reputation systems. Some of it is geared toward more narrow domains [5], but much is generalized peer-to-peer networks. Some of these systems are focused on entirely different areas, like balancing reputation and privacy [22]. Others focus on finding reliable resource providers in systems like Gnutella [2], [3], [14]. Others aim to represent some generalized reputation score separate from specific systems [4], [15], [16], [26], [30]. All focus on securely sharing reputations to prevent manipulation; they do not discuss calculating initial reputations, likely to facilitate flexibility. ClariNet aims to provide a specific, generalizable, and tamper resistant calculation algorithm, making it complimentary to these systems. ClariNet can calculate the initial reputation and share it using an existing system. In this regard, it is similar to Guru [45], but Guru requires consensus rounds while ClariNet allows for eventual consistency and stronger insight at the cost of a more limited consensus committee.

Finally, while the topics are not directly related to ClariNet's core components we would like to address Sybil attacks and peer shuffling. Peer shuffling [10], aims to ensure that participants are more likely to end up with a collection of reliable peers than malicious peers. As with the majority of the related work, this is an orthogonal approach to ClariNet and could potentially be combined to provide benefits from both. Sybil attacks are a present concern in any distributed system and ClariNet is no exception; mitigation of Sybil attacks is outside the scope of ClariNet's aims, but we have attempted to design it in such a way that it would be reasonably easy to apply mitigations [9] to ClariNet.

XIII. Conclusion

In this paper we presented ClariNet, a novel reputation scheme that is highly general and can identify malicious peers even in the face of uncertainty. ClariNet aims to provide defense in depth by layering common security approaches such as cryptographic signatures, authenticated key exchange, a minimal consensus protocol, a reputation scheme, and log auditing to help participants defend themselves from false accusations by malicious peers. ClariNet allows participants to, on average, differentiate malicious and cooperative peers even when malicious peers misbehave infrequently or even dominate the network. Additionally, ClariNet does not place strict deadlines on many of its operations. This allows participants to separate data delivery from auditing, reducing latency in processing, and perform auditing at their discretion, such as deferring it until times when load is low. While ClariNet may not be universally applicable, it can provide assistance to participants in distributed systems where peers are not implicitly trusted.

References

[1] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest, "A fair protocol for signing contracts," IEEE Transactions on Information Theory, vol. 36, no. 1, pp. 40–46, Jan. 1990, doi: 10.1109/18.50372.

[2] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, in NOSSDAV '03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 144–152. doi: 10.1145/776322.776346.

[3] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in Proceedings of the 9th ACM conference on Computer and communications security, in CCS '02. New York, NY, USA: Association for Computing Machinery, Nov. 2002, pp. 207–216. doi: 10.1145/586110.586138.

[4] A. A. Selcuk, E. Uzun, and M. R. Pariente, "A reputation-based trust management system for P2P networks," in IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004., Apr. 2004, pp. 251–258. doi: 10.1109/CCGrid.2004.1336575.

[5] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer ecommerce communities [Extended Abstract]," in Proceedings of the 4th ACM conference on Electronic commerce, in EC '03. New York, NY, USA: Association for Computing Machinery, Jun. 2003, pp. 228–229. doi: 10.1145/779928.779972.

[6] A. Singh, G. Kumar, R. Saha, M. Conti, M. Alazab, and R. Thomas, "A survey and taxonomy of consensus protocols for blockchains," Journal of Systems Architecture, vol. 127, p. 102503, Jun. 2022, doi: 10.1016/j.sysarc.2022.102503.

[7] J. Xu, C. Wang, and X. Jia, "A Survey of Blockchain Consensus Protocols," ACM Comput. Surv., vol. 55, no. 13s, p. 278:1-278:35, Jul. 2023, doi: 10.1145/3579845.

[8] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1432–1465, 2020, doi: 10.1109/COMST.2020.2969706.

[9] B. N. Levine, C. Shields, and N. B. Margolin, "A Survey of Solutions to the Sybil Attack".

[10] M.-K. Yoon, "AccountNet: Accountable Data Propagation Using Verifiable Peer Shuffling," in 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, Hong Kong: IEEE, Jul. 2023, pp. 48–61. doi: 10.1109/ICDCS57875.2023.00050.

[11] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," Computer Communications, vol. 25, no. 17, pp. 1606–1621, Nov. 2002, doi: 10.1016/S0140-3664(02)00049-X.

[12] D. Dolev and H. R. Strong, "Authenticated Algorithms for Byzantine Agreement," SIAM J. Comput., vol. 12, no. 4, pp. 656–666, Nov. 1983, doi: 10.1137/0212045.

[13] "Bitcoin Whitepaper." Accessed: Oct. 03, 2024. [Online]. Available: https://portfolio-blog-starter.vercel.app/blog/bitcoin-whitepaper

[14] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a P2P network," in Proceedings of the 11th international conference on World Wide Web, in WWW '02. New York, NY, USA: Association for Computing Machinery, May 2002, pp. 376–386. doi: 10.1145/511446.511496.

[15] R. Zhou, K. Hwang, and M. Cai, "GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks," IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 9, pp. 1282–1295, Sep. 2008, doi: 10.1109/TKDE.2008.48.

[16] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in Proceedings of the tenth international conference on Information and knowledge management, in CIKM '01. New York, NY, USA: Association for Computing Machinery, Oct. 2001, pp. 310–317. doi: 10.1145/502585.502638.

[17] T. Coffey and P. Saidha, "Non-repudiation with mandatory proof of receipt," SIGCOMM Comput. Commun. Rev., vol. 26, no. 1, pp. 6–17, Jan. 1996, doi: 10.1145/232335.232338.

[18] P. Feldman and S. Micali, "Optimal algorithms for Byzantine agreement," in Proceedings of the twentieth annual ACM symposium on Theory of computing, in STOC '88. New York, NY, USA: Association for Computing Machinery, Jan. 1988, pp. 148–161. doi: 10.1145/62212.62225.

[19] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures".

[20] X. Huang, Y. Mu, W. Susilo, W. Wu, J. Zhou, and R. H. Deng, "Preserving Transparency and Accountability in Optimistic Fair Exchange of Digital Signatures," IEEE Transactions on Information Forensics and Security, vol. 6, no. 2, pp. 498–512, Jun. 2011, doi: 10.1109/TIFS.2011.2109952.

[21] O. Markowitch and Y. Roggeman, “Probabilistic Non-Repudiation without Trusted Third Party”.

[22] L. Lu et al., “Pseudo Trust: Zero-Knowledge Authentication in Anonymous P2Ps,” IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 10, pp. 1325–1337, Oct. 2008, doi: 10.1109/TPDS.2008.15.

[23] J. Chen and Y. Qin, “Reducing Block Propagation Delay in Blockchain Networks via Guarantee Verification,” in 2021 IEEE 29th International Conference on Network Protocols (ICNP), Nov. 2021, pp. 1–6. doi: 10.1109/ICNP52444.2021.9651926.

[24] G. Bracha and S. Toueg, “Resilient consensus protocols,” in Proceedings of the second annual ACM symposium on Principles of distributed computing, in PODC ’83. New York, NY, USA: Association for Computing Machinery, Aug. 1983, pp. 12–26. doi: 10.1145/800221.806706.

[25] X. Zhang et al., “The Block Propagation in Blockchain-Based Vehicular Networks,” IEEE Internet of Things Journal, vol. 9, no. 11, pp. 8001–8011, Jun. 2022, doi: 10.1109/JIOT.2021.3074924.

[26] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The Eigentrust algorithm for reputation management in P2P networks,” in Proceedings of the 12th international conference on World Wide Web, in WWW ’03. New York, NY, USA: Association for Computing Machinery, May 2003, pp. 640–651. doi: 10.1145/775152.775242.

[27] M. Wendl, M. H. Doan, and R. Sassen, “The environmental impact of cryptocurrencies using proof of work and proof of stake consensus algorithms: A systematic review,” Journal of Environmental Management, vol. 326, p. 116530, Jan. 2023, doi: 10.1016/j.jenvman.2022.116530.

[28] D. Boneh and C. Komlo, “Threshold Signatures with Private Accountability,” in Advances in Cryptology – CRYPTO 2022, Y. Dodis and T. Shrimpton, Eds., Cham: Springer Nature Switzerland, 2022, pp. 551–581. doi: 10.1007/978-3-031-15985-5_19.

[29] F. Chen, J. Wang, J. Li, Y. Xu, C. Zhang, and T. Xiang, “TrustBuilder: A non-repudiation scheme for IoT cloud applications,” Computers & Security, vol. 116, p. 102664, May 2022, doi: 10.1016/j.cose.2022.102664.

[30] T. Beth, M. Borcherding, and B. Klein, “Valuation of trust in open networks,” in Computer Security — ESORICS 94, D. Gollmann, Ed., Berlin, Heidelberg: Springer, 1994, pp. 1–18. doi: 10.1007/3-540-58618-0_53.

[31] C. Nist, “The digital signature standard,” Commun. ACM, vol. 35, no. 7, pp. 36–40, Jul. 1992, doi: 10.1145/129902.129904.

[32] A. Montresor and M. Jelasity, “PeerSim: A Scalable P2P Simulator,” in Proc. of the 9th Int. Conference on Peer-to-Peer (P2P’09), Seattle, WA, Sep. 2009, pp. 99–100.

[33] “Peers,” libp2p. Accessed: Nov. 14, 2024. [Online]. Available: https://docs.libp2p.io/concepts/fundamentals/peers/

[34] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, “Authentication and authenticated key exchanges,” Des Codes Crypt, vol. 2, no. 2, pp. 107–125, Jun. 1992, doi: 10.1007/BF00124891.

[35] “The Noise Protocol Framework.” Accessed: Nov. 14, 2024. [Online]. Available: https://noiseprotocol.org/noise.html

[36] T. Perrin, “The Noise Protocol Framework”.

[37] “An open system to manage data without a central server,” IPFS. Accessed: Nov. 14, 2024. [Online]. Available: https://ipfs.tech

[38] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” Internet Engineering Task Force, Request for Comments RFC 9000, May 2021. doi: 10.17487/RFC9000.

[39] “Correlation IDs - Engineering Fundamentals Playbook.” Accessed: Nov. 19, 2024. [Online]. Available: https://microsoft.github.io/code-with-engineering-playbook/observability/correlation-id/

[40] Datadog, “What is Distributed Tracing? How it Works & Use Cases,” Datadog. Accessed: Nov. 19, 2024. [Online]. Available: https://www.datadoghq.com/knowledge-center/distributed-tracing/

[41] “SPDY: An experimental protocol for a faster web.” Accessed: Nov. 19, 2024. [Online]. Available: https://www.chromium.org/spdy/spdy-whitepaper/

[42] W. Cai, W. Jiang, K. Xie, Y. Zhu, Y. Liu, and T. Shen, “Dynamic reputation–based consensus mechanism: Real-time transactions for energy blockchain,” International Journal of Distributed Sensor Networks, vol. 16, no. 3, p. 1550147720907335, Mar. 2020, doi: 10.1177/1550147720907335.

[43] [X. Wang and Y. Guan, “A Hierarchy Byzantine Fault Tolerance Consensus Protocol Based on Node Reputation,” Sensors, vol. 22, no. 15, Art. no. 15, Jan. 2022, doi: 10.3390/s22155887.

[44] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, “Proof of Reputation: A Reputation-based Consensus Protocol for Blockchain Based Systems,” in Proceedings of the 1st International Electronics Communication Conference, in IECC ’19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 131–138. doi: 10.1145/3343147.3343169.

[45] A. Biryukov, D. Feher, and D. Khovratovich, “Guru: Universal Reputation Module for Distributed Consensus Protocols”.