CLARINET: Generalized Reputation Framework for Witnessed Data in Distributed Systems

Andrew Robie and Man-Ki Yoon
Department of Computer Science
North Carolina State University
Raleigh, NC, USA

***Abstract*—Audit logging is essential for post-incident analysis, but it can fail when two participants—such as nodes in a distributed system communicating over a network—have conflicting records, and neither can definitively prove its claim. Introducing a witness that records data in transit can provide a tie-breaking vote, but this approach requires the witness to be honest. In this paper, we present CLARINET, a reputation assessment scheme that enables participants in a distributed system to detect behavior indicative of audit log forgery, penalize such malicious activity even when the exact source is unknown, and ensure that the true origin of the malicious activity is, in aggregate, penalized more harshly than cooperative participants. CLARINET achieves this by verifying claims against a known reference point, issuing verifiable claims to peers, and introducing the notion of different penalty degrees. Through formal proofs and simulations, we demonstrate that CLARINET enables cooperative participants to gain meaningful insights into peer behavior even when more than half the network is malicious and malicious actions occur relatively infrequently (as low as 10%). With CLARINET, participants in distributed systems can gain confidence in identifying cooperative witnesses and protecting themselves from false claims.**

I. INTRODUCTION

Audit logs are commonly used in distributed systems to facilitate post-incident analysis. They record participant actions, such as messages sent or received. Investigators can use these logs to trace data flow and correlate actions across the network. For example, if an audit log indicates that a message originated from another participant, the investigation can proceed by examining whether that participant contributed to the issue. Tracing data flow in this way can significantly facilitate root cause identification—an increasingly critical task as modern distributed systems grow in scale and complexity.

However, parties must ensure that audit logs are correctly recorded. Missing logs offer no utility, and inaccurate ones can result in false leads. Malicious participants may worsen the situation by intentionally recording misleading data to divert suspicion. Even if the other party keeps its own logs, if it lacks a reliable means to substantiate its version of events, conflicting records can lead to disputes over accuracy. Cryptographic signatures can help verify the origin of a message, but a malicious party might deliberately forge an invalid signature to deny having sent the message in the first place.

Consider a distributed analytics pipeline where multiple nodes process and forward data through several stages. Each node is operated by a different organization, and each applies its own data transformations, such as filtering, aggregation, or normalization, before passing results to the next node. The system relies on audit logs and metadata to track how data evolves, enabling end-to-end traceability for debugging and accountability. Now, imagine a node misbehaves, either due to a suboptimal algorithm, a bug, or even malicious intent. For instance, a node might apply an incorrect transformation, but log a sanitized or fabricated version of the data along with a misleading explanation to avoid potential liability. Alternatively, a downstream node might receive the faulty data, detect anomalies, and modify the input before logging it, all while claiming the upstream node sent it in that altered form. Despite different bad actors, both scenarios result in *disagreeing logs*. Without trusted checkpoints or verifiable signatures, it becomes challenging to determine the true origin of the fault and the actions taken by each node.

Having an intermediary, such as another node, *witness* the data in transit can point investigators in a particular direction by supporting one participant's account. However, this requires that the witness be impartial and accurately record what it observes. In systems without an agreed-upon trusted party, participants must rely on some means of identifying trustworthy witnesses, such as a *reputation* scheme.

To this end, we propose CLARINET, a protocol that enables participants to detect and penalize malicious actions aimed at forging audit logs. In addition to detection, CLARINET also mitigates the impact of a malicious intermediary attempting to poison communication between two honest participants. It does so by providing three reputation actions—*rewards* and *weak* and *strong penalties*—along with application rules, correctness criteria, and mechanisms for safely sharing audit information. CLARINET uses a combination of a known point of reference and obvious signs of malicious activity, such as invalid signatures, to identify malicious action: rewards are issued when participants behave correctly, weak penalties are applied when blame is uncertain, and strong penalties are enforced when the malicious party is certain. These reputations can then be shared using existing methods [10], [11], providing wider insight into the network. We evaluate these rules using formal proof and simulation to demonstrate the following:

- Participants always end up with either irrefutable proof that a peer sent, witnessed, or received a message via signatures, or they are able to penalize malicious actors.
- No honest participant is penalized more harshly than a malicious one for any given message.
- For any message, the total penalties imposed on malicious participants outweigh the total penalties incorrectly applied to honest participants.
- On average, malicious participants' reputations decline faster than those of cooperative ones.

Our simulations demonstrate that these properties hold under

a range of network conditions, including varying frequencies of malicious actions and different proportions of malicious participants.

Altogether, we make the following contributions:

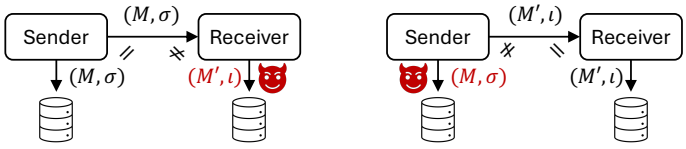
- CLARINET, a novel reputation assessment system that can differentiate known origins of malicious action from suspected origins and resists malicious exploitation.
- Formal proofs demonstrating CLARINET's soundness.
- Simulations demonstrating that CLARINET's theoretical claims hold at scale.

II. Problem Statement and Models

Audit logging is essential for post-incident analysis, particularly when two parties have communicated and auditors must determine the responsible party. Without a centralized, trusted logging mechanism, auditors must rely on logs provided by the participants themselves; however, malicious participants can forge these logs. When conflicting log records are presented, it can degenerate into one party's word against the other.

A. Abstract Example

Consider a scenario in which one participant sends a message to another, along with a signature for non-repudiation. Figure 1 illustrates two possible cases that show why the logs of the two parties alone are insufficient to determine the responsible party in the event of a dispute.



(a) Receiver is malicious (b) Sender is malicious

Fig. 1. Given conflicting logs of message-signature pairs, it is impossible to determine who is the malicious actor because (a) and (b) are indistinguishable from a third-party perspective.

Scenario 1. The sender transmits a pair consisting of message M and its signature σ, and logs this transmission. The receiver receives the pair but chooses not to log the receipt of M (e.g., to avoid liability). Instead, it fabricates a new message M' and acts upon it. Since the receiver cannot generate a valid signature on behalf of the sender, it fabricates an invalid signature ι and logs the pair (M', ι), falsely claiming that ι was also provided by the sender.

Scenario 2. The sender has message M' that it should send, but wishes to deny having sent it. Therefore, it sends (M', ι), where ι is an invalid signature, and logs (M, σ), where M is a different message that the sender wants to claim as having sent. The receiver receives (M', ι) and logs (M', ι) accordingly.

In both scenarios, the sender claims to have sent (M, σ), while the receiver claims to have received (M', ι). Notice that both scenarios result in the same set of conflicting logs, despite the malicious actor being different in each case. Consequently, based solely on the logs presented to a third-party auditor, it is impossible to identify the malicious actor. Therefore, additional evidence beyond the participant logs is required to determine who is telling the truth.

B. Adversary Model

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

G1. If sending, transmit the messages in a deniable fashion.
G2. Provide proof supporting the forged claim.
G3. Win the witness consensus protocol.

G1 and G2 are simple to achieve. Providing an invalid signature on the delivered message, as illustrated in the scenarios above, ensures that the receiver does not have definitive proof that the adversary sent the message. Logging the desired message satisfies G2. For G3, the adversary must find a peer willing to support its false claim. While we leave the mechanism for witness agreement flexible, such agreements would, by design, incorporate some form of reputation scoring. Consequently, adversaries are incentivized to degrade the reputations of non-colluding peers while maintaining high reputations for themselves and their colluders.

C. Assumptions

AS1. Adversaries are capable of intercepting and modifying messages, but they do not aim to completely halt communication between cooperative participants.
AS2. Adversaries can identify their colluders and communicate out-of-band, including sharing private keys.
AS3. Adversaries have full control over what information they report to their peers.
AS4. Adversaries make an honest effort to deliver data, albeit in a manipulated form, when it serves their goals.

The qualification in AS1 is necessary to keep the problem tractable. Without it, adversaries could block all communication between non-colluding peers, allowing only a small subset of interactions they deem acceptable. While traditional Byzantine fault-tolerant systems mitigate this using deadlines and default actions, we consider this weakened assumption reasonable: adversaries still wish to benefit from the distributed system. Completely halting communication would degrade the system's utility and impose a higher burden on the adversaries. Instead, adversaries only interfere when they strongly believe it will serve their interests.

AS2 simply acknowledges the feasibility of out-of-band communication between colluding adversaries, including the exchange of private keys. Cooperative participants cannot assume the ability to intercept or monitor such coordination.

AS3 reflects the fact that participants have no visibility into the internal state of their peers beyond what those peers report. These are entirely under the peers' control and may be false.

AS4 states that if an adversary holds a message intended for a recipient, it delivers the message, although it may be manipulated. In conjunction with AS1, this means adversaries do not prevent cooperative witnesses from delivering messages to their intended receivers.

III. CLARINET: Witnesses and Reputation

A. Overview

In CLARINET, any participant can serve as a *witness* for a communication between two other participants. A witness

records the message in transit and serves as a tie-breaking vote, but this requires the witness to be impartial. The sender and receiver would like some insight into a given witness's trustworthiness, which they can achieve through a *reputation*. They would then use this reputation during a witness agreement process to find a witness they deem acceptable. To do this, they need some criteria by which to form that reputation. These criteria must also be resistant to a malicious witness poisoning the communication between benign senders and receivers. For this, we propose two simple correctness criteria: invalid signatures and known points of reference. We discuss their usage in subsequent sections.

In addition to these correctness criteria, participants need reputation adjustment rules, information exchange rules, and a secure means of authenticating their peers. In designing CLARINET, we address each in the face of intelligent malicious adversaries who are aware of CLARINET.

B. Identity Specification

1) Participant Identity and Authentication: Without identity and authentication, participants cannot verify their communication peers. This can be addressed using public key cryptography, where identities are tied to public keys, as in libp2p [30]. By combining this with authenticated key exchange (AKE) [31], participants can ad hoc exchange public keys, identify peers via public key hashes referred to as ID_P, and confirm identities through an AKE handshake using an additional ephemeral key. The handshake succeeds only if the peer possesses the private key corresponding to its claimed identity. Once complete, the session key ensures message authenticity and confidentiality, assuming strong key usage and no key compromise. Libp2p implements this via the Noise Protocol Framework [32], [33], as used in real-world systems like Interplanetary File System (IPFS) [34].

While this prevents impersonation, it does not solve identity changes. This is equivalent to a Sybil attack, in which a single entity presents itself as multiple distinct logical participants. Such attacks are practically impossible to prevent in decentralized systems like those targeted by CLARINET that lack a centralized trusted authority [40]. However, some mitigations [10], [41] can be employed. For instance, witness agreement protocols might guard against newcomer status, reputations could be shared via systems like EigenTrust [10] or GossipTrust [11], or economic incentives and puzzles could raise entry barriers. By keeping these flexible, CLARINET remains broadly applicable, adaptable to future advances, and resilient to problematic legacy behaviors. This aligns with existing reputation schemes that focus on computing and sharing reputations rather than stipulating their use.

2) Connection and Message Identifiers: Participants need to know which peers were involved in a particular data delivery so they can reward and penalize the appropriate peers. Since witness agreement can be costly, participants may wish to reuse a witness across multiple messages. To support this, we define *connections*, each consisting of a sender S, witness W, and receiver R, identified by a shared connection ID ID_C. Messages are uniquely identified by a message ID ID_M, which comprises the connection ID ID_C and a sequence number.

C. Protocol Detail

CLARINET has two main components: data delivery and auditing. *Data delivery* is the process of sending a message from one participant to another via a witness, while *auditing* is the process of verifying that the message was sent and received correctly, from which participants can assess *reputations*. CLARINET can be supplemented with a reputation sharing framework like Eigentrust [10] to provide participants insight into a larger portion of the network.

1) Messaging: The CLARINET protocol can be implemented in any system that allows participants to communicate with each other. The only requirement is that the underlying transport be reliable, such as TCP or QUIC [35]. This ensures that both sides are confident the message is delivered without error. All communications include a signature of the payload.

The formal state machines for the CLARINET protocol are described in detail in Appendix A, in which the operations (Op1-Op7) are formally defined.

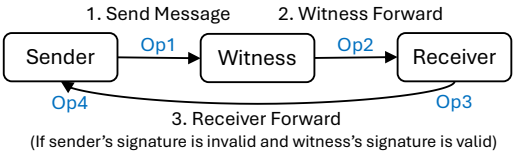


Fig. 2. Data delivery workflow in CLARINET. Message is sent from Sender to Receiver via Witness. Optionally, the Receiver forwards the (hash of the) message to the Sender if it received an invalid signature of the Sender (via the Witness) and a valid signature of the Witness.

Data delivery. Figure 2 shows the data delivery workflow. The sender S signs a message M and sends it to the witness W, who signs both the message and the sender's signature, then forwards everything to the receiver R. R can optionally forward a hash of the message and the sender's signature, along with the witness's signature and its own signature over the hash and witness's signature, back to the sender S—but only if the message contained an invalid sender signature and a valid witness signature. While the hash does not include the witness's signature, this is unnecessary for S to determine whether R received a message that differs from what S originally sent. Note that this last step is part of the audit process, even though it occurs during data delivery.

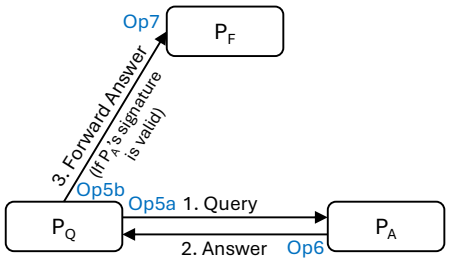


Fig. 3. Audit workflow in CLARINET. P_Q is the participant initiating a query, P_A the one being queried, and P_F the third participant in the connection for the queried message.

Auditing. While data delivery provides insight into incoming communications, it provides none for outgoing messages. This

gap necessitates auditing. Figure 3 illustrates the audit workflow. Any participant may initiate an audit, taking the role of P_Q,[1] at any time for any message by sending a query to another participant, P_A. Allowing P_Q to independently query both peers for the same message ensures that all participants have full visibility into what their peers claim to have sent or received during data delivery. Upon receiving the query, P_A returns a signed hash of the message and sender's signature to P_Q. Once P_Q receives the response, it verifies the hash and signature. If the signature is valid, P_Q may forward the answer to a third participant, P_F. If P_F receives a forwarded response, it also performs verification. Forwarding provides P_Q no direct benefit, but allows it to be a good neighbor and assist P_F in discovering malicious activity. This is especially beneficial when querying a peer who makes different claims when interacting with different participants.

2) Reputation Operations: Participants reward or penalize their peers based on their observation of them. Letting P be a peer, there are three reputation operations:

- **Reward**, or $\texttt{rew}(P)$, is given when a participant observes *correct* behavior from peer P. The sender and witness do not issue rewards during data delivery, solely to simplify the logic. Rewards increase P's reputation.
- **Weak penalty**, or $\texttt{pen}_\texttt{W}(P)$, is given when a participant observes malicious behavior that P *may* have caused. Weak penalties are always applied to all known potential sources and decrease the reputation.
- **Strong penalty**, or $\texttt{pen}_\texttt{S}(P)$, is given when a participant observes malicious behavior it is *certain* P caused. These are applied only to a single peer and reduce the reputation more than weak penalties.

At a high level, participants *strongly penalize* a peer when the peer provides an invalid signature, otherwise violates protocol behaviors, or the peer provided a mismatched hash during audit and the participant and peer directly communicated during data delivery (S and W or W and R). Receivers weakly penalize both the sender and witness if they receive a data delivery containing an invalid sender signature. Participants *weakly penalize* both peers during auditing if they receive an answer with a mismatched hash and communicated indirectly with the answering peer during data delivery. Participants *reward* peers when all signatures are valid and the provided hash matches the expected.

Detailed Reputation Operations. (a) Data Delivery: When S wishes to send a message M, it signs M as σ_S and delivers both to W. Upon receiving M and σ_S, W verifies σ_S; if it is invalid, W strongly penalizes S. Then, W signs the combination of M and σ_S as σ_W and delivers all three to R. Upon receipt, R first verifies σ_W. If σ_W is invalid, R strongly penalizes W and halts. Otherwise, R verifies σ_S. If σ_S is valid, R rewards both S and W. If σ_S is invalid, R weakly penalizes S and W, and forwards a hash $H = \texttt{hash}(M, \sigma_S)$ and σ_W to S. Upon receiving this forward, S reviews σ_W; if it is invalid, S strongly penalizes R. Otherwise, S compares its record of M to H, and if they do not match, S strongly penalizes W.

(b) Auditing: When a participant P_Q wishes to audit a message, it selects one of the peers, P_A, with which it communicated and sends an audit request including the message identifier ID_M to P_A. Upon receiving the query, P_A constructs a hash $H = \texttt{hash}(M, \sigma_S)$ based on its record of the message, signs this hash as σ_{P_A}, and delivers it to P_Q. When P_Q receives this answer, it verifies σ_{P_A}; if the signature is invalid, P_Q strongly penalizes P_A and halts. Otherwise, P_Q compares its own record to H. If H matches, P_Q rewards P_A; otherwise, P_Q strongly penalizes P_A if the two directly communicated during data delivery, or weakly penalizes both P_A and the other participant if their communication was indirect. Additionally, P_Q may sign and forward this response to the third participant in the data delivery, P_F. Upon receiving such a forwarded response, P_F verifies P_Q's signature, strongly penalizing and halting if it is invalid. If the signature is valid, P_F verifies σ_{P_A}; if it is invalid, P_F strongly penalizes P_Q. Otherwise, P_F compares its own record to H and applies penalties in the same manner as P_Q.

Example 1. *Consider a scenario in which both the sender and receiver are cooperative, while the witness is malicious, as illustrated in Figure 4.*

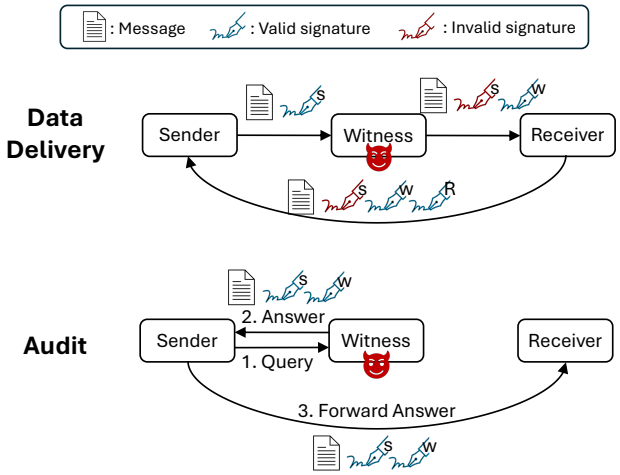


Fig. 4. Malicious witness is penalized for corrupting sender's signature and making different claims to sender and receiver.

During data delivery, the witness corrupts the sender's signature before forwarding the message to the receiver. Unable to determine whether the sender or the witness is responsible for the invalid signature, the receiver weakly *penalizes both. Since the witness's own signature is valid, the receiver forwards the message (specifically, the hash of the message) back to the sender. The sender, however, is sure that the witness tampered with its original signature and therefore* strongly *penalizes the witness. Later, the sender queries the witness, who returns an answer matching the sender's expectation. Despite this, the witness retains the strong penalty from the sender. The sender then forwards the witness's answer to the receiver. Upon comparing this with the version previously received from the*

[1]That is, P_Q can be any participant, including sender, witness, or receiver. This is important because it allows participants to audit their peers. However, auditing provides the sender and witness with greater insight into outgoing communications. While receiver is also free to query others, it likely gains less due to its better visibility during data delivery.

TABLE I
DATA DELIVERY INTERACTIONS: SENDER S

Outcome ID	Malicious Participant Reason	H	σ_W	σ_R	Action	Action Reason
SS1	Malicious W is attempting to turn S and R against each other	Incorrect	Valid	Valid	$\texttt{pen}_\texttt{S}(W)$	S knows W altered M by Lemma 2

witness, the receiver detects a discrepancy. Now confident that the witness provided conflicting information to it and the sender, the receiver concludes that the witness is misbehaving and strongly *penalizes it.*

Example 2. *Consider a case in which a malicious receiver falsely claims to have received a different message than it actually did, as illustrated in Figure 5. In this scenario, both the*

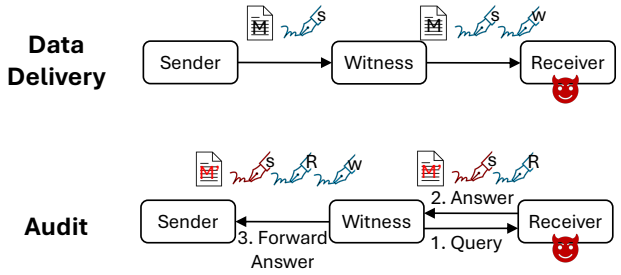


Fig. 5. Malicious receiver is penalized for claiming a different message.

sender and the witness behave correctly during data delivery. During the audit stage, when the witness queries the malicious receiver, the receiver provides a response consistent with the message it wishes to falsely claim. Since it cannot produce a valid sender signature, it must include an invalid one. Upon receiving this response, the witness can strongly *penalize the receiver because the answer does not match what the witness originally sent, and the two communicated directly during data delivery. The witness then forwards this response to the sender. When the sender reviews it, it recognizes that the message does not match what it originally sent during data delivery. However, it cannot determine whether the witness tampered with the message during data delivery or the receiver is attempting to falsify it. As a result, the sender* weakly *penalizes both parties. The sender could arrive at the same conclusion by querying the receiver directly, but forwarding the answer from the witness reduces network overhead by avoiding an additional query.*

Double-Counting. It is important to prevent double-counting; otherwise, situations may arise in which a malicious peer is penalized less harshly than a cooperative one. For example, suppose a malicious witness W alters the message M between a cooperative sender S and receiver R. If S queries W, and W reports M, then S rewards W. Later, when S queries R and receives M' (altered by W), it weakly penalizes both W and R. This results in a final state where the malicious W receives both a penalty and a reward, while R receives only a penalty. To prevent this, participants must ensure that for every unique (ID_M, ID_P), there is exactly one reputation operation, and it must be the harshest one observed, with $\texttt{pen}_\texttt{S} > \texttt{pen}_\texttt{W} > \texttt{rew}$.

IV. PROTOCOL ANALYSIS

In this section, we demonstrate three key properties of CLARINET: (1) malicious behavior cannot escape detection due to the design of messaging and auditing protocols; (2) participants always correctly penalize malicious peers at least as harshly as they erroneously penalize cooperative ones; and (3) the aggregate penalties for malicious peers always outweigh those for cooperative peers.

A. Data Delivery

We begin by discussing the possible outcomes of data delivery for a cooperative participant in each role.

1) Sender S: Its only insight during data delivery is whether the receiver R forwards a message hash H along with the witness's signature σ_W. By Lemmas 1 and 2 below, this allows S to strongly penalize misbehavior by W.

Since σ_W and the receiver's signature σ_R are cryptographic signatures, each can be in one of two states (valid or invalid). Together, the tuple (H, σ_W, σ_R) yields eight possible combinations. Table I lists the only scenario that is likely to occur. The remaining seven do not benefit either a malicious or cooperative receiver, making it unlikely that either would perform them, and W is unable to influence R's behavior.

Lemma 1. *Hash H from witness W suffices for sender S to know that the information receiver R claims to have received was modified.*

Proof. By protocol definition, $H = \texttt{hash}(ID_M, D, \sigma_S)$ where D is the message content and σ_S is S's signature. Because S retains a record of what it sent, it can construct a verification hash H'. Provided the hash algorithm is preimage resistant and collision resistant, we have $H = H'$ *iff* the components used to generate H' are identical to those used to generate H. If $H \neq H'$, then S knows that at least one of ID_M, D, or σ_S does not match what it sent. □

Lemma 2. *Given a message forward containing an invalid H and a valid σ_W, S can be sure W is the source of the discrepancy.*

Proof. Without access to W's private key, which W is assumed not to leak, R cannot generate a valid σ_W. By protocol definition, the same hash function is used to generate both σ_W and H. Specifically, R computes H from the received message. If R used any data other than that sent by W, it could not produce a valid σ_W. Therefore, a valid σ_W implies it must have been generated from exactly the data W delivered. Consequently, S can confidently conclude that W has access to and signed the modified data that resulted in H and σ_W. Furthermore, since S and W communicate directly, no intermediary could have tampered with the message in transit. Hence, W must be the one who modified the data resulting in H and σ_W. □

2) Witness W: W can only assess based on σ_S, shown in Table II. Since there is only one field with two possible

TABLE II
DATA DELIVERY INTERACTIONS: WITNESS W

Outcome ID	Malicious Participant Reason	σ_S	Action	Action Reason
WS1	Malicious S knows it won't want to later deny M	Valid	No action	W has proof S sent M
WS2	Malicious S wishes to later deny having sent M	Invalid	$\texttt{pen}_\texttt{S}(S)$	W knows S violated protocol

TABLE III
DATA DELIVERY INTERACTIONS: RECEIVER R

Outcome ID	Malicious Participant Reason	σ_S	σ_W	Action	Action Reason
RS1	Malicious S and/or W know they will not want to deny the message	Valid	Valid	rew(S), rew(W)	R has proof S sent M and W witnessed M
RS2	Wouldn't occur because no benefit	Valid	Invalid	$\texttt{pen}_\texttt{S}(W)$	R knows W violated the protocol
RS3	Malicious S wants to later deny M or malicious W wants to turn S and R against each other	Invalid	Valid	$\texttt{pen}_\texttt{W}(S)$, $\texttt{pen}_\texttt{W}(W)$	R knows malicious action occurred, but cannot be sure if S or W is source
RS4	Malicious S and W want to deny sending and witnessing M or malicious W wants to prevent R from forwarding M	Invalid	Invalid	$\texttt{pen}_\texttt{S}(W)$	R knows W violated the protocol

TABLE IV
QUERY INTERACTIONS

Outcome ID	Response Reason	H	σ_{P_A}	Action	Action Reason
Q1	P_A is cooperative or malicious P_A does not wish to deny M	Correct	Valid	$\texttt{rew}(P_A)$	P_Q has proof P_A sent, witnessed, or received M
Q2	Malicious P_A wishes to prevent forwarding	Correct	Invalid	$\texttt{pen}_\texttt{S}(P_A)$	P_Q knows P_A violated the protocol
Q3	Malicious P_A wishes to claim alternate message or cooperative P_A received modified message	Incorrect	Valid	See Table VI	See Table VI
Q4	Malicious P_A wishes to claim an alternate message while preventing forwarding	Incorrect	Invalid	$\texttt{pen}_\texttt{S}(P_A)$	P_Q knows P_A violated protocol

TABLE V
QUERY FORWARD INTERACTIONS

Outcome ID	Response/Forward Reason	H	σ_{P_A}	σ_Q	Action	Action Reason
F1	P_A is cooperative or malicious P_A does not wish to deny M	Correct	Valid	Valid	$\texttt{rew}(P_A)$	P_F has proof P_A sent, witnessed, or received M
F2	Malicious P_A wishes to claim alternate message or cooperative P_A received modified message	Incorrect	Valid	Valid	See Table VI	See Table VI

TABLE VI
QUERY/FORWARD PENALTIES

Communication	Action	Reason
Direct	$\texttt{pen}_\texttt{S}(P_A)$	P_Q or P_F can be sure P_A is the source of malicious action
Indirect	$\texttt{pen}_\texttt{W}(P_A)$, $\texttt{pen}_\texttt{W}(P_O)$	P_Q or P_F knows malicious action occurred, but cannot be sure if P_A or P_O is source.

values, there are only two possible states, and because S and W directly communicate, W can strongly penalize S should σ_S be invalid.

3) Receiver R: R assesses the four possible outcomes listed in Table III based on σ_S and σ_W. In cases RS2 and RS4, R does not alter S's reputation. This design choice simplifies the protocol—an invalid witness signature σ_W means receiver R always halts and can make further assessments during a later query. Additionally, it introduces conservatism in penalty assignment, since R cannot forward the message in RS4.

B. Auditing

Next, we examine the various combinations of cooperative and malicious participants within a connection and how each affects auditing. Participants audit their peers to obtain proof of receipt or detect misbehavior. Here, we discuss how auditing enables this and prove that there are no situations in which a malicious peer can evade detection or cause cooperative participants to more harshly penalize their cooperative peers.

All participants follow the same auditing behavior; only penalties differ based on direct communication. Because of this similarity, we 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 Table VI, P_O represents the other participant in the data delivery. In Table V, we omit scenarios that P_Q would never perform, as they would benefit neither P_Q nor P_F.

To establish the robustness of auditing, we need to address all combinations of cooperative and malicious participants. Each connection involves three participants—sender, witness, and receiver—each of whom can be either cooperative or malicious, resulting in eight possible configurations. Malicious participants may act independently or collude. However, Lemmas 3 and 4 below show that a non-colluding malicious peer is better off behaving cooperatively. Based on this, we can simplify the analysis by treating any scenario with two malicious participants as

TABLE VII
REPUTATION OPERATION APPLICATIONS

	Scenario (1) R is malicious			**Scenario (2)** W is malicious									**Scenario (3)** S is malicious		
				1) W attempts to maintain both			2) W sacrifices reputation with R			3) W sacrifices reputation with S					
	S	W	R	S	W	R	S	W	R	S	W	R	S	W	R
S	-	pen_W	pen_W	-	pen_S	pen_W	-	pen_W	pen_W	-	pen_S	pen_W	-	-	-
W	rew	-	pen_S	-	-	-	-	-	-	-	-	-	pen_S	-	rew
R	-	-	-	pen_W	pen_S	-	pen_W	pen_S	-	pen_W	pen_W	-	pen_W	pen_W	-

colluding. Moreover, if all pairs of malicious participants are colluding, they can be treated as a single collusion group.

Lemma 3. *Malicious participants gain an advantage by eliminating non-colluding malicious peers from the witness candidate pools of other peers.*

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 so, P_M must have its own record and a witness W supporting its claim. While consensus requires agreement from any two parties, no dispute occurs if S and R agree, since W only logs messages without taking action. P_M also wishes to ensure that W's false report matches; otherwise, P_M loses or neither participant wins if all three report differently. Such disagreement triggers scrutiny, which malicious participants wish to avoid as it may reveal their malicious activities. Because CLARINET does not enforce a specific witness agreement algorithm, P_M may not be able to force a colluding W. It thus benefits from making non-colluders, both cooperative and malicious, appear less trustworthy than colluders. □

Lemma 4. *Malicious participants benefit from retaining their own record of peers' reputations for non-colluding peers.*

Proof. Following from Lemma 3, since malicious participants wish to assist their peers in removing non-colluding malicious nodes, it is trivial for them to identify such peers. This allows them to influence witness agreement toward cooperative peers when they cannot steer it toward one of their colluders. Consequently, it is in their best interest to adhere to the protocol regarding non-colluding malicious peers. □

CLARINET maintains invariants that ensure malicious peers are penalized at least as harshly as any mistakenly penalized cooperative peers. Accordingly, we exclude from our analysis scenarios involving two or more malicious peers, as they pose no risk of erroneous penalization. We also exclude scenarios in which all participants are cooperative, since, by definition, no malicious action occurs. This leaves the three scenarios shown in Table VII. It presents the rewards and penalties for all scenarios once all cooperative participants have completed their audits.

With the possible scenarios constrained, we now examine the likely actions a malicious participant could take in an attempt to evade culpability or damage the reputations of cooperative peers. We show that, regardless of the action taken, CLARINET's invariants hold.

Scenario (1) - Malicious Receiver R. According to Section IV-A, the malicious receiver R never forwards a message. Since both S and W are cooperative, they always provide valid signatures, meaning no penalties occur during data delivery. During querying, S and W always receive the correct data from each other and initially reward one another. However, when either queries R, R may wish to deny having received M.

As shown in Table IV, R's most likely choice is Q3. Choosing Q1 would undermine its ability to claim a different message, while Q2 and Q4 would lead S to strongly penalize R. In contrast, Q3 allows R to maintain its false claim, receive only a weak penalty, and even cause S to weakly penalize W. Nevertheless, since W and R directly communicated, any of Q2–Q4 would result in W strongly penalizing R.

Scenario (2) - Malicious Witness W. During data delivery, R can present S with definitive proof of W's malicious behavior should σ_W be valid. When S queries W, W must choose among Q2–Q4. Selecting Q2 enables S to forward the answer to R who would strongly penalize W. Choosing Q3 or Q4 results in S strongly penalizing W, while W retains its reputation with R. Regardless of its choice, W suffers a strong penalty from S, R, or both, and can only cause S and R to weakly penalize each other.

Table VII shows three likely sub-scenarios. In the first, W attempts to tamper with the message and later provides S and R with responses they each expect during queries. In the second, W deliberately provides an invalid σ_W during data delivery, sacrificing its reputation with R to prevent the message from being forwarded; possibly to preserve its standing with S. In the third, W tampers with the message but subsequently answers S's queries with responses that align with R's expectations to maintain its reputation with R. In all cases, W is strongly penalized by S, R, or both.

Scenario (3) - Malicious Sender S. If S wishes to maintain deniability, it must provide an invalid σ_S, which would lead W to strongly penalize S, while R weakly penalizes both S and W, since the cooperative W provides a valid σ_W. R is unlikely to gain further insight during queries, as S will most likely choose Q3, knowing that it has already been strongly penalized by W and that it can present a hash H consistent with its alternate data. Meanwhile, W provides Q1 to R, ensuring that R never penalizes W more than weakly. Finally, W can

reward R, since R provides Q1.

C. Overhead

Since CLARINET is a protocol rather than a specific implementation, we discuss the performance characteristics of a hypothetical CLARINET implementation. We assume that a comparative system not using CLARINET still signs and logs all messages it sends or receives. Thus, the overhead introduced by CLARINET stems solely from the additional messages it imposes, i.e., audit messages and witnessed data delivery. It is unlikely that any participant would query a peer more than once for the same message because peers are unlikely to change their answer. This allows us to upper bound the number of audits to six per data delivery: three participants auditing two peers each.

TABLE VIII
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

TABLE IX
MESSAGE SIZES

Message	Components	Overhead (bytes)
Send	$(ID_M, M, \sigma_S, \sigma_W)$	56
Receiver Forward	$(ID_M, H, \sigma_W, \sigma_R)$	120
Query	(ID_M, σ_{P_Q})	56
Answer	(ID_M, H, σ_{P_A})	88
Answer Forward	(A, ID_C, σ_{P_Q})	152

Table VIII details reasonable implementations for each message component, and Table IX lists the corresponding overhead for each message type. The total overhead for all messages involving a participant as a sender or receiver is capped at 1,952 bytes. For comparison, HTTP headers typically range from 200 bytes to 2KB, with modern websites averaging around 700-800 bytes [42]. Since both requests and responses include headers, the combined overhead is roughly 1,400-1,600 bytes, which is comparable to that of CLARINET.

Witness overhead is the only component not fixed relative to the messages participants would exchange without CLARINET. Because only the hash is needed for queries and to support a party's claim, witnesses can store the hash of a message rather than the full data. This results in an overhead of 2,040 bytes per witnessed message. Although this overhead is noticeable, it scales linearly and is offset by the benefits witnesses gain into peer behavior, as well as the accountability that witnessing provides.

The other consideration is latency. Since the sender must transmit to the witness, who then forwards to the receiver, the average end-to-end latency is approximately doubled.

V. EVALUATION

In addition to the theoretical analysis of a single message, it is important that CLARINET provides tangible benefits at scale. We implement the messaging and reputation mechanisms largely as described using PeerSim [29]; any divergences are made solely for ease of implementation and do not alter the protocol semantics. Details regarding unspecified behaviors, configuration settings, and participant actions are provided in the following subsections.

A. Reputation

Reputation is implemented as the ratio of two scores, `good` and `total`, $\frac{\texttt{good}}{\texttt{total}}$. This ratio is capped at 1.0. The scores are updated as follows:

- **Rewards** increment both `good` and `total` by 1,
- **Weak penalties** increment `total` by 1, and
- **Strong penalties** increment `total` by 3.

Cooperative behavior increases the ratio or leaves it unchanged if it is already at its maximum. Malicious behavior decreases the ratio, as only the denominator, `total`, increases. All participants start with `good`=1 and `total`=1 to avoid division by zero, default to trusting, and allow for quick differentiation. If both values were initialized to 0, weak ($\frac{0}{1}$) and strong ($\frac{0}{3}$) penalties would both appear to have a reputation of 0, making them indistinguishable. In contrast, starting at 1 results in distinct reputations of $\frac{1}{2}$ and $\frac{1}{4}$, respectively.

Malicious participants engage in audit behaviors to simulate attempting to appear cooperative, but do not track reputation. Since they are all colluding, as described in Section IV-B, they have no incentive to penalize and eliminate non-colluding malicious peers.

B. Witness Agreement

We take a basic approach to connection setup and witness agreement. Senders always initiate and select the witness. They calculate reputations for all known peers, find the average μ and standard deviation σ of peers with at least one assessment, and then remove peers where $reputation \leq 0.5 \vee reputation < (\mu - \sigma)$. The sender then randomly requests witnesses from this trust list until one agrees. Peers only refuse if they already have the maximum number of permitted open connections, which we set to an arbitrary value of 10. If no peers are available, the sender terminates the connection.

Malicious participants differ only in witness selection. First, they request all colluding peers, simulating influencing witness agreement in favor of a colluder. If none can witness, they randomly select cooperative peers.

C. Parameters

Table X shows the configurable parameters with values used. We permuted over all possible combinations.

The *malicious action threshold percentage* models strategic adversaries who first build up their reputation before beginning to misbehave. A counter is incremented each time a malicious participant performs an action that could be malicious: sending, witnessing, or answering a query. Once this counter exceeds a specified threshold, the participant is allowed to start misbehaving. The threshold is calculated as the product of the total number of simulation rounds and the specified percentage.

TABLE X
SIMULATION PARAMETERS

Parameter	Description	Values
Node count	The number of participants in the network	100, 250, 500, 750, 1K, 5K, 10K
Malicious %	Percentage of participants that are malicious	10%, 20%, 30%, 50%, 90%
Malicious action threshold %	Before threshold malicious participants always behave	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

At each step, participants randomly select one of the following actions with equal probability:

- OPEN a connection with a randomly selected peer.
- SEND a message over a randomly selected open outgoing connection.
- QUERY a peer for a randomly selected message.
- CLOSE a randomly selected outgoing connection.
- Take no action.

If the selected action cannot be successfully completed, the participant takes no action.

Malicious participants differ from honest ones only in that they act maliciously with a configurable probability. These actions include sending messages with invalid signatures, altering witnessed data (e.g., modifying the sender's signature), and forging query responses. However, when both the sender and receiver are malicious, they do not behave maliciously toward each other, as colluding participants have no incentive to deceive one another.

E. Evaluation Results

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

- Whether the participant was cooperative or malicious.
- Information about individual peers, including:
  - The peer's reputation as recorded by the participant.
  - Whether the peer was cooperative or malicious.
- Aggregate statistics for all, cooperative, and malicious assessed peers: average, minimum, maximum reputation, and standard deviation.

We consolidate this data by first averaging the aggregate statistics reported by all cooperative participants within a given simulation for their cooperative and malicious peers. We then compute the overall average of these values across all simulations with the same number of participants.

Network size. CLARINET remains stable with respect to network size; it does not significantly impact the average reputation of cooperative or malicious participants. Therefore, in the following plots, we present results only for the two extremes: networks with 100 and 10,000 participants. This stability primarily stems from the fact that reputation is determined by peer interactions, which occur either when a participant initiates contact or is contacted by others. In a set amount of time, a participant is likely to reach out to a similar number of peers regardless of the network size. While a larger network increases the number of peers that could potentially contact a given participant, each peer has more other peers it could also potentially contact, which offsets the increased likelihood of any specific peer contacting a particular participant.

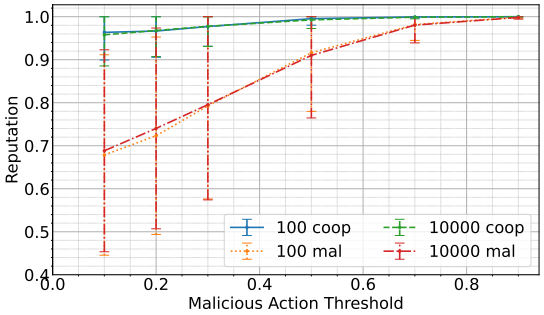


Fig. 6. Reputation as affected by how long malicious participants wait to begin acting maliciously. Error bars represent standard deviation.

Malicious action threshold percentage. Figure 6 shows that as malicious participants wait longer to begin acting maliciously, it becomes harder to detect them. This outcome is expected, as delaying malicious activity results in fewer malicious actions to penalize. Even at the 70% threshold, a small but noticeable gap in reputation is observed. However, because other factors influence misbehavior, a 70% threshold does not imply that all malicious peers begin misbehaving exactly 70% of the way through the simulation. At the 90% threshold, it appears almost no malicious participants misbehave, resulting in nearly all participants reaching the maximum reputation of 1.0.

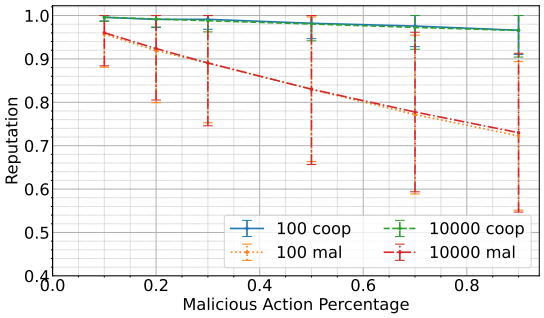


Fig. 7. Reputation as affected by how often malicious participants act maliciously. Error bars represent standard deviation.

Malicious action percentage. Figure 7 demonstrates that as peers act maliciously more often, their reputation decreases more significantly. Cooperative peers suffer a slight decrease as well due to erroneous weak penalties, but their reputations remain significantly more stable at a higher overall reputation, as reflected by a smaller standard deviation. The larger standard deviation observed among malicious peers likely results from variance in both their frequency of malicious actions and whether cooperative participants happened to assess them.

Malicious participants percentage. Figure 8 shows that as the percentage of malicious participants increases, their average reputations also increase, while the reputations of cooperative peers decrease. This increase in malicious peers' reputations occurs in part because they refrain from acting maliciously when both sender and receiver are malicious. As the number of

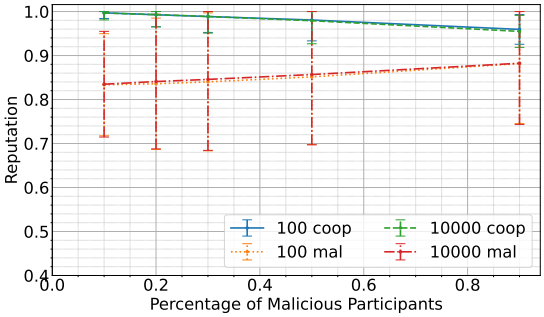


Fig. 8. Reputation as affected by how many participants are malicious. Error bars represent standard deviation.

malicious participants grows, the likelihood of such interactions increases, decreasing the overall malicious activity witnessed by cooperative peers. Furthermore, penalties may be distributed across a larger pool of malicious peers, resulting in fewer penalties per individual. Conversely, with fewer cooperative participants, erroneous weak penalties tend to concentrate, increasing the penalty per cooperative peer. Despite these effects, cooperative participants maintain noticeably higher and more stable reputations even at the most extreme levels of malicious presence.

F. Limitations and Future Work

CLARINET is entirely reactive, so it can only guard against repeat offenders. In the worst case, the witness is a colluder, and the participant has no recourse should a dispute arise. However, protection from repeat offenders is better than no protection, and participants' ability to gain insight as a witness helps them assess peers without being at risk. Additionally, CLARINET can be combined with reputation sharing methods like [10] to let participants learn from peer interactions.

Ideally, CLARINET would support multiple witnesses, but we limited the analysis to one to constrain complexity. We believe this is reasonable for an initial exploration. Supporting multiple witnesses is a prime area for future work.

The analysis does not factor in malicious participants who claim unsent messages. Participants cannot assess messages they are unaware of. While they may be able to take certain steps, like using received queries as query candidates, this requires additional analysis to ensure it is not exploitable and would still not guard against entirely fabricated messages.

CLARINET currently requires a reasonably stable network. In highly dynamic networks, participants may not have enough time to conduct extensive auditing to form solid peer reputations. The benefits from serving as a witness do encourage participants to stay active, but this is not always practical.

Participants could likely make more intelligent decisions based on combinations of messages to prevent erroneous penalties or better apply strong penalties. While limiting these conditions for initial analysis was reasonable to reduce complexity, investigating and analyzing more intelligent conditions would be an area for future work.

The empirical analysis was limited. While randomized traffic can approximate large-scale systems, drawing from real-world dynamic distributed systems would provide better data. We intend to investigate the system with more intelligent malicious participants and build a real, deployable version of CLARINET to test in a real distributed environment.

VI. RELATED WORK

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 the way CLARINET does. The majority of similar works are orthogonal and can potentially be combined with CLARINET to gain the benefits of both.

Several works [1]–[4] address non-repudiation 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 transfers so that the message that either side wishes to receive is not usable until it is completely received. This ensures that either both sides obtain proof of exchange or that neither side has anything. CLARINET allows senders to lazily obtain proof of receipt when they have capacity and allows for full transmission of data at once.

Optimistic fair exchange [21] provides remuneration should one party attempt to cheat through the presence of an arbitrator, but is reliant on a trustworthy arbitrator. Some works [22] address malicious arbitrators by ensuring the arbitrator can be held accountable. CLARINET gives participants insight into peer trustworthiness when making an initial selection.

More recent work on non-repudiation [27], consensus protocols, and consensus reputation [36]–[38] and surveys [14]–[16] leverage blockchain [20]. While robust, this approach has raised environmental concerns [26], and the overhead can introduce noticeable latency [24], [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.

Non-blockchain consensus and Byzantine agreement protocols [17]–[19] are typically older works often designed with different goals, such as accommodating faulty hardware rather than active manipulation or upper-bounding agreement time. CLARINET detects malicious parties so they can be excluded from future committee construction.

There is a wealth of work on reputation. Some focus on narrow domains [13], but much is generalized to peer-to-peer networks. Some address areas like balancing reputation and privacy [23]. Others on finding reliable resource providers [5]–[7]. Others aim to represent some generalized reputation score separate from specific systems [8]–[12]. The vast majority focus on securely sharing reputations rather than calculating initial reputations. CLARINET aims to provide a specific, generalizable, and tamper-resistant calculation algorithm. In this regard, it is similar to Guru [39], but Guru requires consensus rounds while CLARINET allows for eventual consistency and stronger insight at the cost of a more limited consensus committee.

VII. 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 against 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 the load is low. While CLARINET may not be universally applicable, it can assist participants in distributed systems where peers are not implicitly trusted.

ACKNOWLEDGMENT

This work is supported in part by the Google Cloud Research Credits program with the award GCP19980904. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of sponsors. We also thank Brad Reaves for his feedback.

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.

[2] O. Markowitch and Y. Roggeman, "Probabilistic Non-Repudiation without Trusted Third Party," in Proceedings of the Conference on Security in Communication Networks, 1999.

[3] 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.

[4] 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.

[5] 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 (NOSSDAV), Jun. 2003.

[6] 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 (CCS), Nov. 2002.

[7] 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 (WWW), May 2002.

[8] 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 (CCGrid), Apr. 2004.

[9] 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 (CIKM), Oct. 2001.

[10] 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 (WWW), May 2003.

[11] 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.

[12] T. Beth, M. Borcherding, and B. Klein, "Valuation of trust in open networks," in Computer Security (ESORICS), 1994.

[13] 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 (EC), Jun. 2003.

[14] 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.

[15] 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.

[16] 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.

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

[18] P. Feldman and S. Micali, "Optimal algorithms for Byzantine agreement," in Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC), Jan. 1988.

[19] G. Bracha and S. Toueg, "Resilient consensus protocols," in Proceedings of the second annual ACM symposium on Principles of distributed computing (PODC), Aug. 1983.

[20] "Bitcoin Whitepaper." [Online]. Available: https://www.zouantcha.com/blog/bitcoin-whitepaper

[21] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in Advances in Cryptology (EUROCRYPT), 1998.

[22] 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.

[23] 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.

[24] 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.

[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.

[26] 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.

[27] 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.

[28] C. Nist, "The digital signature standard," Commun. ACM, vol. 35, no. 7, pp. 36–40, Jul. 1992.

[29] A. Montresor and M. Jelasity, "PeerSim: A Scalable P2P Simulator," in Proceedings of the 9th International Conference on Peer-to-Peer (P2P), Sep. 2009.

[30] "Peers," libp2p. [Online]. Available: https://docs.libp2p.io/concepts/fundamentals/peers/

[31] 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.

[32] "The Noise Protocol Framework." [Online]. Available: https://noiseprotocol.org/noise.html

[33] T. Perrin, "The Noise Protocol Framework". [Online]. Available: https://noiseprotocol.org/noise.pdf

[34] "An open system to manage data without a central server," IPFS. [Online]. Available: https://ipfs.tech

[35] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Request for Comments RFC 9000, May 2021.

[36] 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.

[37] 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.

[38] 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 (IECC), Jul. 2019.

[39] A. Biryukov, D. Feher, and D. Khovratovich, "Guru: Universal Reputation Module for Distributed Consensus Protocols," Cryptology ePrint Archive, 2017.

[40] J. R. Douceur, "The Sybil Attack," in Peer-to-Peer Systems, P. Druschel, F. Kaashoek, and A. Rowstron, Eds., Berlin, Heidelberg: Springer, 2002, pp. 251–260.

[41] A. M. Bhise and S. D. Kamble, "Review on Detection and Mitigation of Sybil Attack in the Network," Procedia Computer Science, vol. 78, pp. 395–401, Jan. 2016.

[42] "SPDY: An experimental protocol for a faster web." [Online]. Available: https://www.chromium.org/spdy/spdy-whitepaper/

APPENDIX A
CLARINET PROTOCOL STATE MACHINES

The CLARINET protocol state machines are defined in terms of the following cryptographic primitives. We assume that all participants have a public/private key pair and that they have agreed on a hashing algorithm. The notation used is as follows:

- $\sigma_X \leftarrow \texttt{sign}(X, Y)$ signs message Y using X's private key.
- $H \leftarrow \texttt{hash}(Y)$ returns the hash of Y using the agreed upon hashing algorithm.
- $\texttt{True/False} \leftarrow \texttt{verify}(X, Y, Z)$ returns true if Z is a valid signature for data Y using X's public key.
- $\texttt{True/False} \leftarrow \texttt{verifyHash}(X, Y, Z)$ verifies signature Z for pre-hashed Y such that:

$$\texttt{verify}(X, Y, Z) = \texttt{verifyHash}(X, \texttt{hash}(Y), Z).$$

Additionally, we define direct communication to mean that within a connection, the two participants had no intermediary, i.e., S and R do not directly communicate, but all others do.

CLARINET has seven operations that fall into two categories: Data Delivery (Op1-Op3) and Auditing (Op4-Op7).

Op1: Sending Message

1. Construct $ID_M = (ID_C, SeqNo)$
2. Construct $\sigma_S = \texttt{sign}(S, (ID_M, D))$
3. Deliver (ID_M, D, σ_S) to witness W

Op2: Witness Forwarding Message

1. Receive (ID_M, D, σ_S) from a participant P
2. Check $P = S$
   If False, $\texttt{pen}_\texttt{S}(P)$ and halt
3. Check corresponding open connection
   If True, Continue
4. $\texttt{verify}(S, (ID_M, D), \sigma_S)$
   If False, $\texttt{pen}_\texttt{S}(S)$
5. Sign $\sigma_W = \texttt{sign}(W, (ID_M, D, \sigma_S))$
6. Deliver $(ID_M, D, \sigma_S, \sigma_W)$ to receiver R

Op3: Receiving Message

1. Receive $(ID_M, D, \sigma_S, \sigma_W)$ from P
2. Check $P = W$
   If False, $\texttt{pen}_\texttt{S}(P)$ and halt
3. Check corresponding open connection
   If True, Continue
4. $\texttt{verify}(W, (ID_M, D, \sigma_S), \sigma_W)$
   If False, $\texttt{pen}_\texttt{S}(W)$ and halt
5. $\texttt{verify}(S, (ID_M, D), \sigma_S)$
   If True, $\texttt{rew}(S)$, $\texttt{rew}(W)$, and halt
   If False, $\texttt{pen}_\texttt{W}(S)$, $\texttt{pen}_\texttt{W}(W)$, and continue
6. Construct $H = \texttt{hash}(ID_M, D, \sigma_S)$
7. Construct $\sigma_R = \texttt{sign}(R, (ID_M, H, \sigma_W))$
8. Deliver $(ID_M, H, \sigma_W, \sigma_R)$ to S

Op4: Receiver Forward Receiving

1. Receive $(ID_M, H, \sigma_W, \sigma_R)$ from P
2. Check $P = R$
   If False, $\texttt{pen}_\texttt{S}(P)$ and halt
3. $\texttt{verify}(R, (ID_M, H, \sigma_W), \sigma_R)$
   If False, $\texttt{pen}_\texttt{S}(R)$ and halt
4. $\texttt{verifyHash}(W, H, \sigma_W)$
   If False, $\texttt{pen}_\texttt{S}(R)$ and halt
5. Construct $V = \texttt{hash}(ID_M, D, \sigma_S)$ from S's record
6. Check $V = H$
   If False, $\texttt{pen}_\texttt{S}(W)$

Op5: Querying

For *Op5-7*, P_Q is the participant initiating a query, P_A the one being queried, and P_F the third participant in the connection for the queried message.

5a. Initial Query

1. Select some message M for which to query
2. Construct $\sigma_{P_Q 1} = \texttt{sign}(P_Q, M.ID_M)$
3. Deliver $(M.ID_M, \sigma_{P_Q 1})$ to P_A

5b. Review and Forward

1. Receive $A = (ID_M, H, \sigma_{P_A})$ from P_A
2. $\texttt{verify}(P_A, (A.ID_M, A.H), A.\sigma_{P_A})$
   If False, $\texttt{pen}_\texttt{S}(P_A)$ and halt
3. Construct $V = \texttt{hash}(M.ID_M, M.D, M.\sigma_S)$
4. Check $V = A.H$
   If True, $\texttt{rew}(P_A)$
   If False, Check direct communication between P_Q and P_A
   If True, $\texttt{pen}_\texttt{S}(P_A)$
   If False, $\texttt{pen}_\texttt{W}(P_A)$, $\texttt{pen}_\texttt{W}(P_F)$
5. Optionally halt
6. Construct $\sigma_{P_Q 2} = \texttt{sign}(P_Q, A)$
7. Construct $F = (A, ID_{P_A}, \sigma_{P_Q 2})$ where ID_{P_A} is the ID_P of P_A
8. Deliver F to P_F

Op6: Query Answering

1. Receive $Q = (ID_M, \sigma_{P_Q})$
2. Check for record of message M corresponding to $Q.ID_M$
   If True, Construct $H = (M.ID_M, M.D, M.\sigma_S)$
   If False, Construct $H = null$
3. Sign $\sigma_{P_A} = \texttt{sign}(P_A, (ID_M, H))$
4. Deliver (ID_M, H, σ_{P_A}) to P_Q

Op7: Query Forward Receiving

1. Receive $F = (A, ID_{P_A}, \sigma_{P_Q})$
2. $\texttt{verify}(P_Q, (A, ID_{P_A}), \sigma_{P_Q})$
   If False, $\texttt{pen}_\texttt{S}(P_Q)$ and halt
3. $\texttt{verify}(P_A, (A.ID_M, A.H), \sigma_{P_A})$
   If False, $\texttt{pen}_\texttt{S}(P_Q)$ and halt
4. Find record of message M corresponding to $A.ID_M$
5. Construct $V = \texttt{hash}(M.ID_M, M.D, M.\sigma_S)$
6. Check $V = A.H$
   If True, $\texttt{rew}(P_A)$
   If False, Check direct communication between P_F and P_A
   If True, $\texttt{pen}_\texttt{S}(P_A)$
   If False, $\texttt{pen}_\texttt{W}(P_A)$, $\texttt{pen}_\texttt{W}(P_Q)$