Hindawi
Security and Communication Networks
Volume 2022, Article ID 7760509, 20 pages
https://doi.org/10.1155/2022/7760509

Research Article

A Reputation-Based Approach Using Consortium Blockchain for Cyber Threat Intelligence Sharing

Xiaohui Zhang,[1] Xianghua Miao ,[1,2] and Mingying Xue [3]

[1]*Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China*
[2]*Computer Technology Application Key Laboratory of Yunnan Province, Kunming, China*
[3]*Faculty of Management and Economics, Kunming University of Science and Technology, Kunming, China*

Correspondence should be addressed to Xianghua Miao; xianghuamiao@126.com

Received 14 July 2021; Revised 19 September 2021; Accepted 22 June 2022; Published 10 August 2022

Academic Editor: Konstantinos Rantos

Copyright © 2022 Xiaohui Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The CTI (Cyber Threat Intelligence) sharing and exchange is an effective method to improve the responsiveness of the protection party. Blockchain technology enables sharing collaboration consortium to conduct a trusted CTI sharing and exchange without a centralized institution. However, the distributed connectivity of the blockchain-based CTI sharing model proposed before exposes the systems to byzantine attacks. The compromised members of partner organizations will further decrease the accuracy and trust level of CTI by generating false reporting. This paper proposes a new blockchain-based CTI model to address the unbalance issues of performance in speed, scalability, and security, which combines consortium blockchain and distributed reputation management systems to achieve automated analysis and response of tactical threat intelligence. In addition, the novel consensus algorithm of consortium blockchain that is fit for CTI sharing and exchange is introduced in this paper. The new consensus algorithm is called "Proof-of Reputation" (PoR) consensus, which meets the requirements of transaction rate and makes the consensus in a creditable network environment through constructing a reputation model. Finally, the effectiveness and security performance of the proposed model and consensus algorithm is verified by experiments.

1. Introduction

Organizations need to be supported by more effective and responsive defense methods to mitigate the danger of increasingly complex attack methods or threats such as advanced persistent threats (APTs) and zero-day vulnerabilities brought about by the development of information technology. As the proactive approach, CTI (Cyber Threat Intelligence) is a collection of information that can cause potential harm and direct harm to organizations and institutions [1]. The typical application of CTI is shown in Figure 1. CTI has become an essential weapon in the arsenal of cyber defenders to address the information asymmetry of issues that happened on offensive and defensive sides. Taking advantage of the value behind the CTI, such as evaluating and simulating malicious behavior in networks, is a critical measure to mitigate increasing cyberattacks.

The CTI sharing and exchange in a cooperative approach promises to be the most effective method to maximize the benefit of CTI through improving the issue of information islands, which means the CTI generated from partner organizations can aid cybersecurity policymakers in making decisions. To meet the needs of CTI sharing, the stakeholders have formulated a series of standards for the exchange of threat intelligence, such as STIX, IODEF, and OpenIoC [2]. The typical application structure of the CTI sharing system is shown in Figure 2. The core idea behind threat intelligence sharing is to create situation awareness among stakeholders by sharing information about the newest threats and vulnerabilities and swiftly implementing the remedies [3]. However, a survey conducted in 2014 shows that slow and manual sharing processes impede full CTI exchange participation [4]. For example, there have been large-scale WannaCry viruses in education, medical, and other industries [5]; if this threat intelligence can be

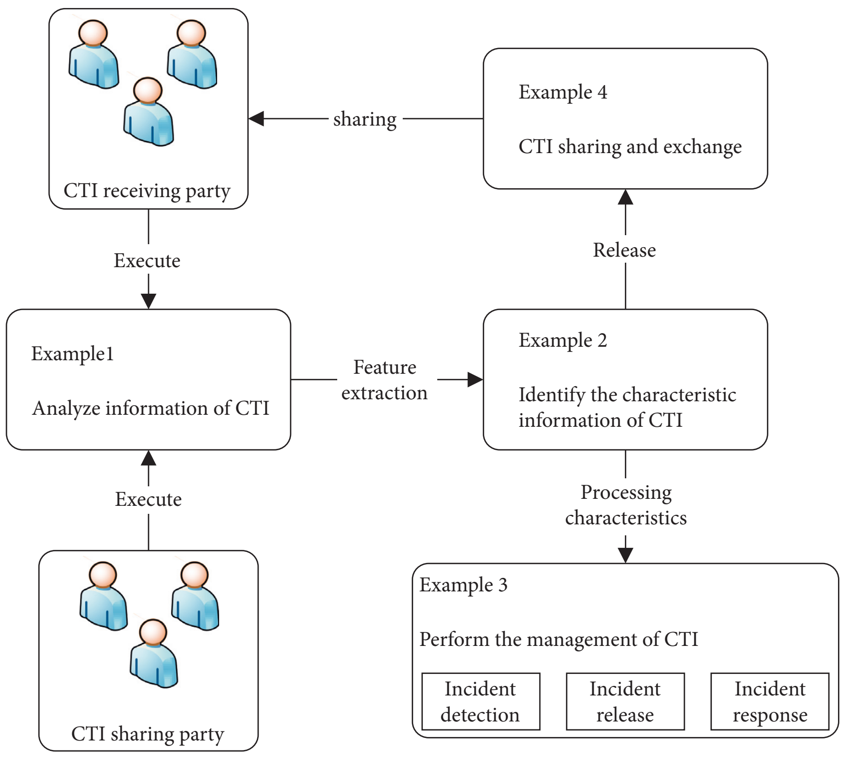

FIGURE 1: The application of cybersecurity threat intelligence.

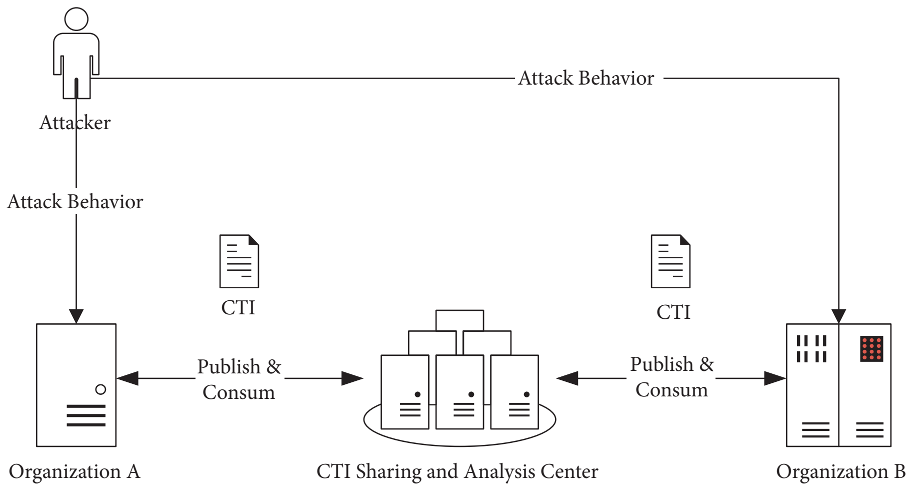

FIGURE 2: The typical structure of the CTI sharing system.

timely released, then most organizations will be able to avoid intrusion, which means that automating the sharing and exchange processes can extremely increase the effectiveness of CTI.

The different applications of CTI can be categorized as tactical threat intelligence, operational threat intelligence, strategic threat intelligence, and technical threat intelligence [6]. Incident responders consume tactical threat intelligence to ensure that their defenses and investigation are prepared for current tactics [7]. Consequently, the key to achieving CTI sharing automation is to be accurately received and processed tactical threat intelligence quickly.

The inappropriate CTI sharing may lead to the disclosure of critical and sensitive intelligence data included in CTI, which can affect the enthusiasm of enterprises to participate in CTI exchange [8]. Hence, there is still a contradiction between automated sharing and the privacy protection requirement in the CTI sharing platform. The blockchain-based CTI sharing model has brought hope to solving the above paradox [9]. As a novel framework, blockchain technology, which uses account anonymity, a tamper-free mechanism, and an encryption function, enables sharing participants to conduct a trusted CTI sharing and exchange without a centralized institution [10]. However, the distributed connectivity of the blockchain-based CTI sharing model exposes the systems to various challenges.

On the one hand, in a distributed environment, the CTI sharing platform is vulnerable to "false reporting" issues caused by federation members maliciously reporting cyber-attack intelligence [11]. Byzantine behaviors that happen in the blockchain system may decrease the trust of each other among the members of the CTI sharing collaboration consortium [12]. On the other hand, the high throughput is significant for achieving interoperability in CTI sharing and exchange [13]. Still, many studies implemented blockchain solutions through flawed consensus algorithms to exchange data. The performance and scalability limitations still exist in these consensus algorithms [14].

The CTI proposal needs to be shared with high transaction throughput, low latency of confirmation, and security measures simultaneously. Therefore, in response to the current problems in CTI sharing, a new model which combines consortium blockchain and distributed reputation management systems to achieve automated sharing of tactical threat intelligence is presented. The main contributions of our work are summarized as follows:

(1) The common feature of traditional CTI sharing platforms is that they require an authoritative third-party organization to review and manage all CTI proposals that are put by participators, which reduces the timeliness and leads to the potential risk of centralization, that is, once the trusted centralized institution fails, the entire CTI sharing platform will be completely ineffective. So, this paper proposes a decentralized CTI sharing approach based on a consortium blockchain. The participants operate under a governance model with a degree of trust, which provides a way to protect interactions between organizations that share common goals but may not fully trust each other. In addition, our approach can use a more efficient consensus protocol to meet the demand for CTI sharing in aspects of throughput and latency.

(2) The CTI sharing consortium blockchain is usually established by several companies or organizations that do not fully trust each other. It is an acceptable solution that selects a trusted accountant to package CTI proposals into blocks; the accountant needs to be generated according to their reputation level and cannot be monopolized. Because CTI proposal is a type of confidential data that is highly real-time, containing detailed descriptions of security vulnerabilities that can only be disclosed to trusted stakeholders, it can have a disastrous impact on the organization's security situation when the CTI data falls into the wrong hands. Thus, we have designed a new decentralized consensus, called Proof-of-Reputation (PoR) algorithm, to avoid monopolistic behavior in consortium blockchain. The consensus of CTI data relies on cooperation between all roles. At the same time, different roles can achieve conversion under certain conditions, and no one can permanently serve as the accountant in the PoR algorithm.

(3) Due to the differences in the network environment and security capabilities, it is difficult to resolve a dispute between a provider and a consumer about the validity of a certain CTI in a decentralized sharing platform. What is more, the diversity of intelligence sources further amplifies the issue of quality. Most stakeholders look to the trusted centralized institution for data governance in CTI sharing. However, no such authority exists in a decentralized peer to peer network environment. Regarding the issue above, this paper proposes a reputation computing model to address the problem of false or malicious reports that may be submitted by participants in a distributed environment, ensuring that only high-value and confident CTI proposals could be available for sharing without the trusted centralized institution. The reputation management model is together with the PoR algorithm to reduce the impact of the byzantine behaviors in the blockchain-based CTI sharing collaboration consortium.

2. Related Works

2.1. Consensus Algorithm in Blockchain. Blockchain is a distributed ledger behind bitcoin, founded by Nakamoto in 2008 [15]. As the foundation and core technologies of the blockchain system, the consensus algorithm is critical for the security and performance of the blockchain [16].

Public blockchain technology such as Bitcoin and Ethereum employs the methods that "mined" the cryptocurrency based on their computing powers or elect the accountant based on their stake to mitigate the absence of trust. PoW (Proof-of-Work), PoS (Proof of Stake), and DPoS (Delegated Proof of Stake) are classified as the public blockchain consensus protocol.

However, PoW has limitations in computing power consumption and small throughput. In addition, the PoW consensus may suffer the tailored attack behavior such as 51% attacks [17]. Although the PoS and DPoS solve the waste of resources in PoW, there are still problems such as low efficiency [18].

The consortium blockchain is more suitable for CIT sharing and exchange than the public blockchain due to its high transaction throughput performance and low latency of transaction confirmation. The consortium blockchain can use classic CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) to reach the consensus among entities due to the requirements such as participants must be identified, and permission is considered in a consortium blockchain. Table 1 presents a comparison between CFT and BFT of consensus algorithm in consortium blockchain.

In many use cases, high throughput of CTI exchange is a requirement. Consortium blockchain consensus algorithms such as Raft [19] can achieve high throughput, but they can only be suitable for nonbyzantine environments that only honest nodes in the network [20]. Therefore, many

Table 1: Comparisons between two types of consensus algorithm in blockchain.

Criteria	Crash fault tolerance	Byzantine fault tolerance
The basis of agreement	Mostly are voting-based	Mostly are proof-based
Decentralization	Low	Mostly high
The way of nodes management	Join network need to be authorized	Join network freely
Award	Mostly no	Yes
Security	Mostly lower	Mostly higher
Speed	Fast	Low

researchers want to use the Byzantine Fault Tolerance (BFT) mechanism to optimize the security performance of the consortium blockchain consensus algorithm.

The traffic complexity and scalability of the Practical Byzantine Fault Tolerance (PBFT) algorithm is the main reason to limit the application of which [21]. Chen et al. proposed a Raft blockchain consensus algorithm based on a credit model (Craft), which can be used in a byzantine network environment in 2018 [22]; experimental results show that the CRaft algorithm has better performance than PBFT. However, there still exists a 17.89% false-positive rate of byzantine nodes. The new consensus algorithm, Proof-of-Trust (PoT), suitable for crowdsourcing services, was proposed in 2019 [23]; the PoT can provide a feasible accountability method for applying online services using blockchain technology by selecting the validator of the transaction based on the trust value of the service participants. In 2020, Wang et al. developed the Beh-Raft algorithm [24], which combines the Proof-of-Behavior algorithm (PoB) and Raft algorithm, ensuring that only honest nodes can become the network leader to reduce the impact of byzantine nodes.

Many related algorithms cannot efficiently meet CTI sharing scenarios and require pre-designed malicious behavior models. However, in an untrust network environment, the imbalance in the number of normal and malicious nodes makes it challenging to construct an accurate classifier. So, it is necessary to develop a new consensus algorithm to achieve better performance trade-offs in efficiency and security.

2.2. Cyber Threat Intelligence. From a practical point of view, cyber threat intelligence describes existing or imminent threats or hazards to assets. It can help organizations identify and analyze current security situations and respond to them. “Security Threat Intelligence Services Market Guide” was published by Gartner in 2014, which states that threat intelligence is evidence-based knowledge that includes context, mechanisms, indicators, impact, and operational recommendations [25]. In 2015, Friedman and Bouchard further refined the definition of CTI in their publication “Authoritative Guide to Threat Intelligence”: A series of information that analyzes and disseminates about motivations, attempts, and methods of the adversary. This information also can be used in organizations to improve their protection capabilities for enterprise assets” [26]. In short, information that poses a risk or loss of benefit to an organization can be called cyber threat intelligence, which is also the default definition nowadays.

Sharing the CTI data is expected to be the most effective way to break the “information isolated” problem and maximize the value of CTI [8]. In terms of CTI sharing models, it is common to use a centralized sharing platform, where users from different organizations can upload and access the CTI data [27]. In addition, the threat intelligence sharing platform can be further subdivided into four types [28]: strategic partnerships, commercial cooperation, mutually beneficial exchange, and threat intelligence community:

Strategic partnerships: in a strategic partnership, security companies with technical advantages sell and transfer the CTI proposals, integrate them with the existing situation of partners, and form customized CTI products that meet their needs, helping them implement security capabilities.

Commercial cooperation: security companies in different industries form commercial cooperation to exchange more accurate and targeted proposals to fully leverage CTI data’s value.

Mutually beneficial exchange: organizations lead beneficial mutual exchange with the massive CTI data; these data result from what they have accumulated over the years. Employing store such intelligence data into big data platforms and open access to clients, thus building security threat situational awareness capabilities in the client environment.

Threat intelligence community: the threat intelligence community is maintained by an organization specializing in CTI services and opens low-level CTI data to public users.

Academia and industry have developed a series of unified CTI data standards to facilitate sharing and exchange, further promoting CTI sharing technology development. Structured Threat Information Expression (STIX) is a machine-readable format for exchanging CTI proposals that enable organizations to perform collaborative threat analysis, automated intelligence exchange, and detection response [25]. STIX can significantly reduce ambiguity and misunderstanding during the sharing and exchange process. Trusted Automated eXchange of Indicator Information (TAXII) is used to ensure threat intelligence security during transmission [29]. TAXII also supports the transmission of threat intelligence data in multiple formats for increased compatibility. CybOX defines a method for describing the machine objects and network dynamics and has a solid ability to represent various observable indicators [30]. So, the content of STIX also refers to the CybOX specification.

STIX and TAXII have been widely used as two major sharing standards [31]. The current primary approach to sharing CTI data is to use TAXII for data transmission, STIX for intelligence description, and CybOX as elements of STIX.

2.3. Blockchain-Based CTI Sharing Model. Blockchain technology can enable sharing partner organizations to conduct a trusted CTI sharing and exchange without a centralized institution. Many studies of the blockchain-based CTI sharing approach carried out by researchers provide a basis and reference for this paper.

A blockchain-based CTI sharing framework, iShare, was proposed in 2018 [32], where members participating in the framework can only share the experience of network security protection; ishare uses game theory to analyze malicious behaviors within the framework. Huang et al. published a blockchain-based CTI exchange model in 2019 [33], which uses the one-way encryption function to protect the privacy information of participating organizations and analyze the complete network attack chain. In response to the trust and privacy protection issues in CTI sharing, Homan et al. used the channel and membership manager technology in consortium blockchain to enable trusted participants to disseminate highly sensitive data privately [9].

Collaborative Intrusion Detection Systems (CIDN) [34] is one of the specific applications of tactical threat intelligence. To eliminate insider attacks such as random poisoning attacks and special on-off attacks, and improve the accuracy and effectiveness of threat intelligence in CIDN, a threat intelligence aggregation algorithm based on the Bayesian decision is proposed by Fung et al. [35], which reduces the risk cost of wrong decisions effectively. Li et al. use blockchain technology to enhance the robustness of the threat intelligence sharing system and protect against insider attacks during the intelligence aggregation process in CIDN [36]. Yanugunti and Yau published a new consensus algorithm based on the trust value of nodes [37] by using the IDS component of each node in the blockchain to verify the traffic log and evaluate the credibility of the threat intelligence received from others.

Table 2 shows a comparison of some related works using the blockchain to implement CTI sharing and exchange on the consensus algorithm and the contributions and shortcomings. The main idea of the current research is to combine the decentralization and tamper-free mechanism of the blockchain with the CTI exchange system to improve the performance of security and robustness.

However, we see that very few studies consider the following issues: On the one hand, the existing approaches suffer from problems that cannot determine whether the generated CTI has been tampered with due to malicious attacks. On the other hand, to realize the CTI sharing, these studies on blockchain failed to propose a consensus algorithm suitable for CTI exchange. The confidence level of CTI is few considered in many papers.

The defense actions are not trusted when the value of level in CTI is low, and it will bring new questions to the application of automated action using CTI. Our work on the CTI sharing model is motivated by the above results and incentivizing federation members via a distributed reputation management system [38].

3. The Proposed Architecture

According to the sources of CTI, threat intelligence can be divided into internal and external [39]. Internal threat intelligence is generally produced from security devices and system event logs. External threat intelligence includes commercial threat intelligence sold by the cybersecurity service provider and open-source threat intelligence shared on public network platforms.

The architecture of CTI sharing and exchange using consortium blockchain is shown in Figure 3. CTI partner organizations from external obtain the original CTI, and they can also be triggered when the internal cybersecurity system finds an abnormal state. Each CTI sharing collaboration consortium member comprises a proposal generation, consensus, and analysis component.

The proposal generation component work to generate proposals that are transformed from the original CTI for the CTI consortium network, which is used to protect the private information in CTI. Compared to the original CTI, the proposal only includes critical information such as attack characteristics. Proposal results will submit to the intelligence generation component for further processing.

The consensus component realizes the consensus and transmission of proposals among CTI consortium networks and stores the results in blockchain to ensure its immutability and reliability by an innovative consensus algorithm that is fit for the CTI sharing and exchange called "Proof-of-Reputation" (PoR). This algorithm makes the consensus of the proposal in a creditable network environment by constructing a reputation model. The content of the PoR consensus algorithm and reputation model will be elaborated on in Chapter 5.

The analysis component represents the cybersecurity policymakers, such as the security operations center and security analysts. The intelligence from the intelligence generation component will be processed further by the CTI sharing collaboration consortium member to make treatment decisions that provide information support or automated take response.

4. The Proof-of-Reputation Consensus Algorithm

4.1. Basic Definitions. This paper proposes a consensus algorithm based on the reputation model - "Proof-of Reputation" (PoR) to address the problem that the consensus algorithm of consortium blockchain cannot meet the transmission requirements of CTI sharing and exchange or only be used in the nonbyzantine environment.

Definition 1. The threat proposal in PoR. As shown in Figure 3, the proposal generation component generates a threat proposal used to exchange in the PoR consensus algorithm according to the alert fusion information that the threat object

Table 2: Comparisons between related works about CTI sharing.

Study	Consensus	Contributions and shortcomings
Huang et al. [33]	—	Use the blockchain to address the contradiction between the privacy protection requirements of CTI sharing and the need to build a complete attack chain, but not consider the transmission performance of the CTI sharing and exchange in this study.
Homan et al. [9]	Solo	Use the blockchain to allow trusted parties to disseminate highly sensitive data privately. But solo consensus can only be used in the test environment, which is not suitable for a realistic network.
Li et al. [36]	Proof-of-concept	Use blockchain to verify trust management and alert aggregation in a challenge-based trust mechanism. But the proof-of-concept chain is used in this approach to investigate the performance rather than the real blockchain.
Yanugunti and Yau [37]	PBFT	Use blockchain to improve the accuracy of intelligence by identifying compromised nodes in the CIDN. But this study employs the PBFT algorithm to reach a consensus that transmission performance will be affected.

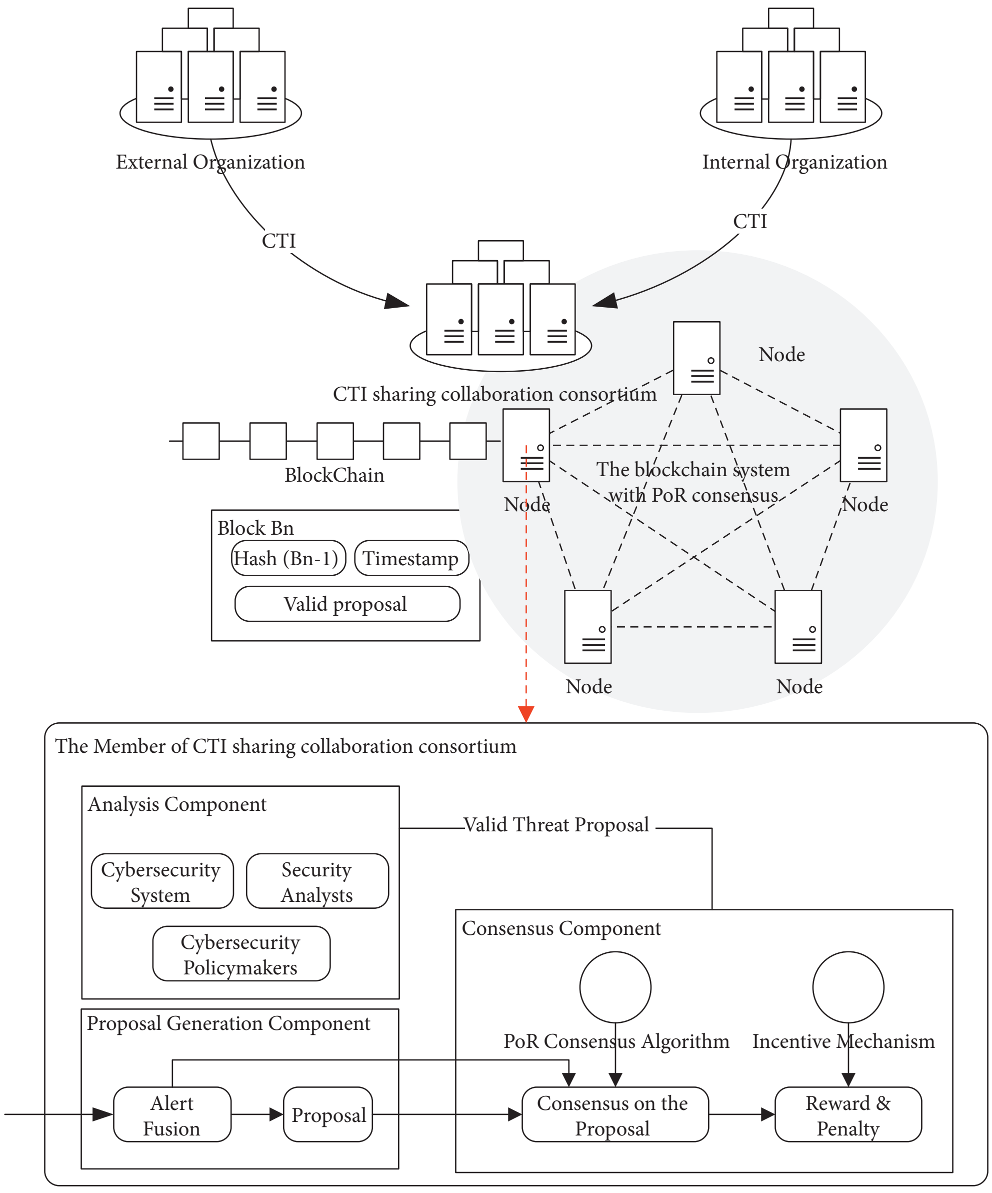

Figure 3: The architecture of our approach.

can be extracted. A threat proposal mainly includes IoC (Indicators of Compromise) information, and it can be modeled and expressed by extracting some features from classic STIX (structured threat information expression). The information of the threat proposal can be indicated in a triad group: $\langle tp, \text{IoC}, \text{level}\rangle$, where tp means the timestamp that the alert information happened. The IoCin the proposal can be expressed as a triad group: $\langle \text{type}, \text{value}, \text{name}, \text{payload} \rangle$, in which $\text{type} \in (\text{campaign}, \text{malware}, \text{threat} - \text{actor}, \text{attack} - \text{pattern}, \ldots)$ indicates the element type and value means the certain element value, name is the detailed type of malicious behavior, payload represents the cyber security attack payload matched in IoC. We use the level to indicate the potential use value of IoC based on the research of BIANCO [40] as shown in Figure 4.

A detailed description of level is shown as follows:

(1) *Hash Value*. The hash value represents a unique identification of a specific malware, but it is not worth analyzing in many cases because the hash value is easy to change. So, we set level = 1 to indicate it.

(2) *IP Addresses*. Certain network behaviors accompany most malicious software. The IP address information is involved in it definitely, but the IP address is straightforward to change when attackers use the technology of anonymous proxy or Tor (The Onion Router). So, we set level = 2 to indicates it.

(3) *Domain Names*. Domain names are usually more valuable than IP address in the sharing of CTI because it needs to be registered at a certain cost of time or economic. So, we set level = 3 to indicates it.

(4) *Artifacts*. Artifacts are divided into network artifacts and host artifacts. The malware requests the resource file of the specified path on C2(command and control) servers or uploads the file to the specified URL. As long as the instruction structure of malware remains unchanged, the network artifacts and host artifacts are difficult to change. So, we set level = 4 to indicates it.

(5) *Tools*. Attackers often spend a lot of time using, developing and customizing some special tools to achieve their purposes, such as Dealers Choice and Xagent in the APT attack. The attacker will abandon these special attack tools currently used if their features are accurately identified by the organization, which will undoubtedly increase the cost of attack behavior. So, we set level = 5 to indicates it.

(6) *TTPs*. TTPs describe the attacker's tactics, techniques, and procedures; TTPs are the most valuable IoC because the strategy and tactics of an attack are often difficult to change. The attacker must either give up the attack or develop a new tactic when the TTPs are recognized. So, we set level = 6 to indicates it.

We use the example shown in Figure 5 to demonstrate the threat proposal in the PoR according to the above definition. Figure 5 shows that a node detects the threat campaign of a web application attack because this alert fusion information from internal CTI sharing organization matched with the attack payload of SQL Injection and defined this threat proposal as level 4.

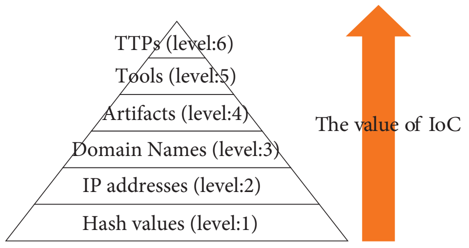

Figure 4: The level of potential value of IoC.

```
    {
      "tp": "20210830061020",
      "IOC":
          {
            "type": "campaign",
            "value": "web application attack",
            "name": "SQL Injection",
            "payload": "GET/search/city?airportCode=%28select%2Afrom%
28select%2Bsleep%282%29union%2F%2A%2A%2Fselect%2B1%29a%29"
          }
      "level": "4"
    }
```

Figure 5: Example threat proposal shared in the PoR consensus.

Definition 2. The state of nodes in PoR. The PoR consensus algorithm improves the algorithm in Ref [19] to solve the byzantine problem in the consortium blockchain network. These nodes of PoR are in one of the following four states: leader, candidate, follower, and supervisor. There is only one leader in the standard PoR cluster, and all of the other nodes are followers. The candidate is the intermediate state between the follower and leader. Handling client requests positively or not is the significant difference between the leader and follower. To be specific, the follower only responds to the request from the leader or candidate according to certain operations as described in Section 4.2.

On the contrary, accepting all CTI sharing requests and replicating them to other followers is the leader's responsibility. In other words, the leader node plays a crucial role in the consensus process. In addition, we introduce a novel node called supervisor that evaluates the reputation score of all nodes based on their dynamical behavior to address issues that classic RAFT consensus cannot prevent malicious nodes. The supervisor node is part-time by the follower to ensure the feature of decentralized in the blockchain.

Algorithms 1–3 cover the logic of cooperation between different states. The description of the four states is described in Table 3.

Nodes in different states can be converted under certain conditions, the conversion relationship of the four states is shown in Figure 6.

Definition 3. The type of nodes in PoR. The type of nodes in PoR are divided into two categories, faithful nodes, and unfaithful nodes. A faithful node indicates the node making the right decision on the proposal. An unfaithful node means

Table 3: The description of four states in POR.

State	Responsibilities	Remark
Leader	Handle all requests from the client. Regularly send heartbeat requests to the follower nodes in the cluster to prevent triggering a new round of elections when the election timer of the follower nodes is out.	Only one exists in network
Follower	Response the request from leader or candidate and redirect requests from the client to the leader node in the cluster.	—
Candidate	The intermediate state of follower and leader.	Not long-lived in the network.
Supervisor	Evaluates the accuracy of the threat proposal and decide the node as faithful node or unfaithful node based on reputation model.	Part-time by the follower

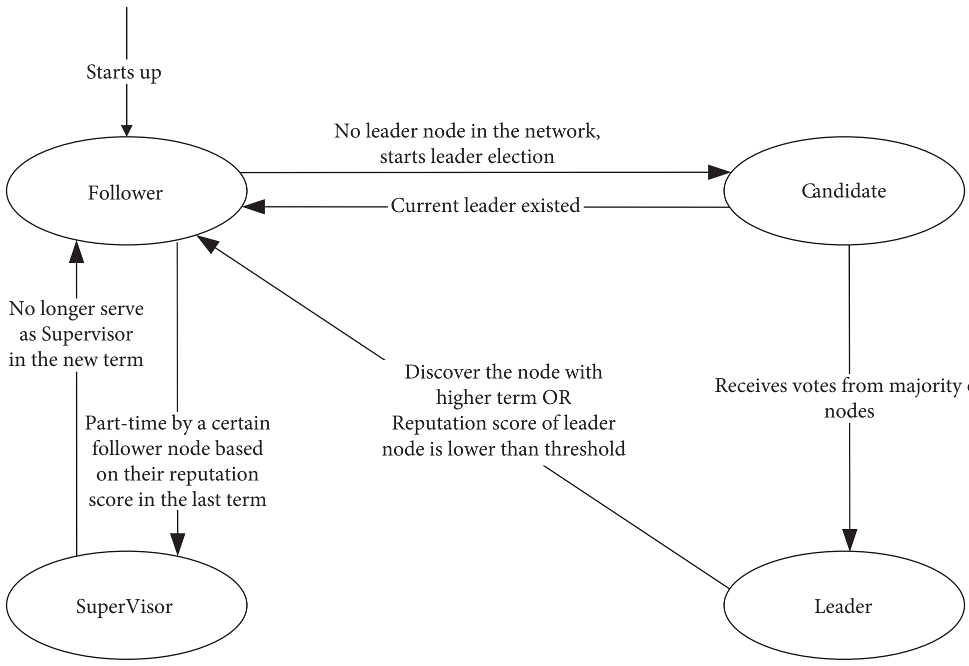
Starts up
Follower
No leader node in the network, starts leader election
Candidate
Current leader existed
No longer serve as Supervisor in the new term
Part-time by a certain follower node based on their reputation score in the last term
Discover the node with higher term OR Reputation score of leader node is lower than threshold
Receives votes from majority of nodes
SuperVisor
Leader

Figure 6: The conversion relationship in PoR.

a node that makes the wrong decision or provides low-value IoC on the proposal due to a lack of enough experience or generates false reporting to decrease the proposal's accuracy due to being under malicious control.

Definition 4. Reputation score**.** The reputation score of the node means the probability of peers providing reliable information, which is used to determine the type of nodes in PoR. Reputation score expressed by $R_i \in [1, 100]$. The initial reputation score R_{init} is the constant that indicates the trust level of the new node. A node's reputation score will be calculated according to the behavior and performance in the network. The node is not trusted anymore when the reputation score is below the threshold R_{thld}. The supervisor constructs UNL (Unfaithful Node List) based on the node's reputation score and uses UNL to control the process of leader election.

Definition 5. Term and Index. Considering the asynchronous feature in the distributed network, Term plays like a logical clock to divide the time into arbitrary lengths, which can avoid the consensus process being affected by timestamp errors. The Term is numbered using consecutive integers, the current term number stored in each node. Only one leader exists in the PoR network, and the Term is updated to a larger term number when a new leader is elected from the candidate. The Index is indispensable for the PoR consensus algorithm to realize highly available services. The Index is used to uniquely identify the log that the leader node replicates to follower nodes to ensure that the order of logs in all nodes is consistent with the leader node—the description of Term and Index as shown in Figure 7.

4.2. Process Description.

4.2. Process Description. The PoR algorithm is divided into three steps: the visor election phase, the reputation model computing phase, and the consensus phase. Nodes use Remote Procedure Call (RPC) to communicate in the network. The consensus process of PoR is shown in Figure 8.

4.2.1. Phase 1: Leader and Supervisor Election Phase. Phase 1 means that no leader node existed at the beginning of this phase; all nodes are in a follower state. The

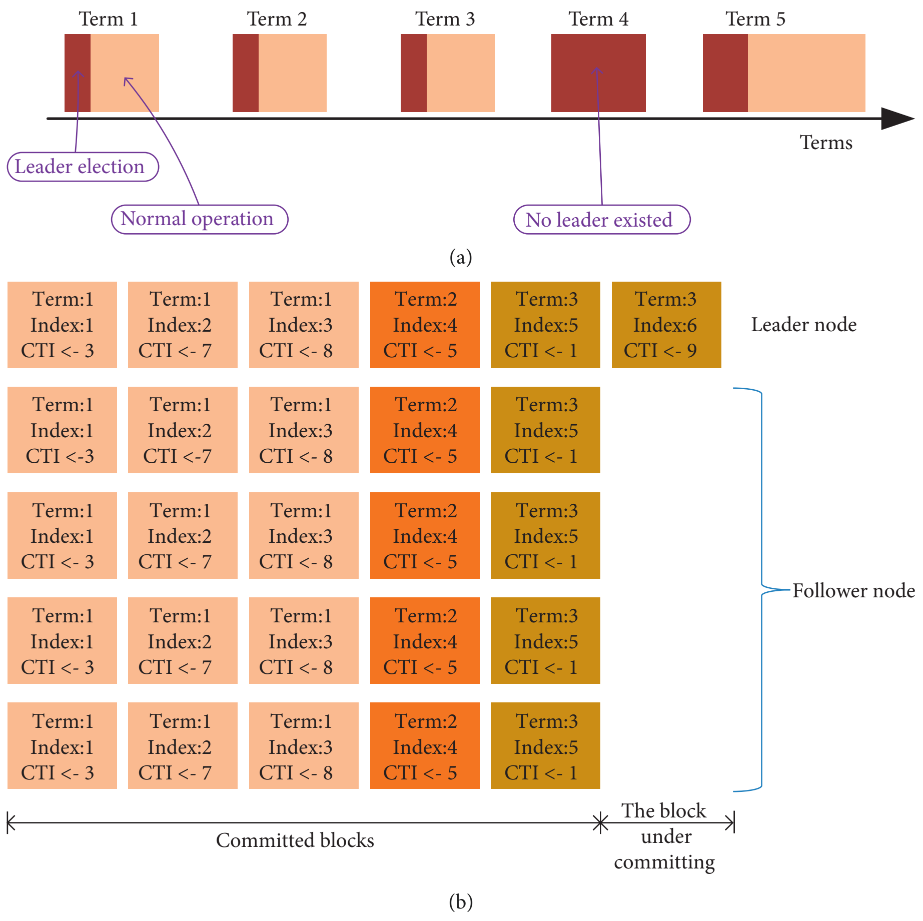

Figure 7: The description of term and index in the PoR. (a) After a successful election, a single leader manages the cluster until the end of the term. (b) Using Index to identify the submission situation of the block in all nodes, a block is considered committed when the block is to be applied to state machines.

follower node will become a candidate node and initiate leader election when the heartbeat from the leader is a timeout, or the term of the leader is less than the current term. The candidate node will try to become the leader by sending RequestVote RPC to the node i, which $i \in [1, \mathrm{num}]$ is the follower node id. If the candidate is decided as the faithful node not in the UNL generated by the supervisor, the node i will send a vote to the candidate when receiving the RequestVote RPC. The description of RequestVote RPC and ReputationValue RPC are shown in Tables 4 and 5.

The candidate will be elected as a leader when he receives the vote from most follower nodes. The function of the leader is described in Definition 1. The leader will send a heartbeat request to all nodes in the network regularly to extend the term. The supervisor node in the new term has been generated based on the reputation score of follower nodes. The details of implementation in phase 1 are given in Algorithm 1.

4.2.2. Reputation Model Computing Phase. The leader node transforms the alert information from sharing parties into threat proposals, as shown in Definition 1, then broadcasts the alert information and threat proposal to all follower nodes and supervisor nodes. The follower node and supervisor node decide on the threat proposal to be included in the next block. In order to prevent the false positives of the proposal being generated by the leader and to increase the accuracy of CTI, the supervisor uses the approach of probabilistic to determine the validity of the proposal and calculate the reputation score by communicating with all followers in ReputationCompute RPC after received a proposal from the leader, which called reputation model. The description of ReputationCompute RPC is shown in Table 6. The computation of the reputation model will be elaborated on in chapter 4.3.

We use Algorithm 2 to describe the key implementation logic of phase 2. The supervisor constructs and updates a list of unfaithful nodes based on the following criteria as shown in Algorithm 2:

(1) Unfaithful node indicates whose reputation score is less than a predefined threshold, and the reputation score is calculated based on a reputation model.

(2) The supervisor sends the unfaithful node list to the other nodes. The nodes maintain their own UNL data based on the received message of the unfaithful

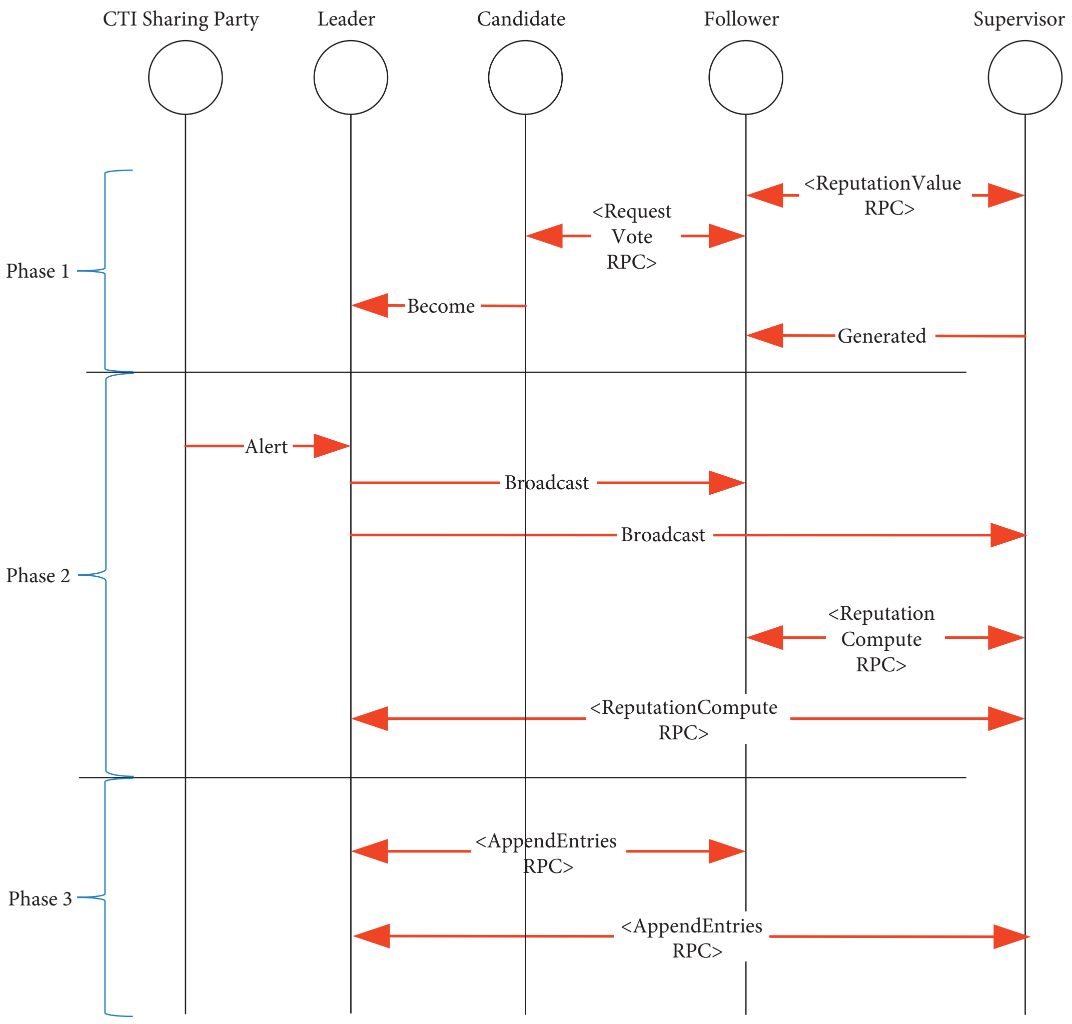

Figure 8: The consensus process of PoR.

Table 4: The description of requestvote RPC communication.

Parameter	Description
Term	The current term of the candidate node.
CandidateID	The id of the candidate node.
Return	Description
Term	The current term of this follower node.
VoteGranted	Set to true when the candidate won this vote.

Table 5: The description of reputationvalue RPC communication.

Parameter	Description
Term	The current term of this node.
NodeID	The id number of this node.
Return	Description
Term	The current term of this node.
UNL	The unfaithful node list based on reputation score.

```
Data: state: the state of a node, num: total number of nodes in the CTI sharing consortium network.
Result : void
(1)  begin
(2)    switch state do:
(3)      case "follower" do:
(4)        communicate with the current Supervisor node by ReputationValue RPC;
(5)        update the UNL data of local;
(6)        if receive the RequestVote RPC from candidate then:
(7)          if candidate is not in UNL then:/* Ensure that only the faithful node can serve as the leader */
(8)            vote to candidate;
(9)          else:
(10)           reject vote to candidate;
(11)       if not receive the heartbeat request from the leader in a period then:/* The timeout of the heartbeat request indicates
     that there is no leader node in the current network */
(12)         state = candidate;
(13)         break
(14)       if the term in heartbeat request from the leader is less than current term then:/* The leader's term must be greater than
     or equal to the term of current network */
(15)         state = candidate;
(16)         break
(17)       else if the node is not in UNL then:/* The faithful follower node can become the supervisor node in the new term */
(18)         become supervisor and step to phase 2;
(19)         break
(20)     case "candidate" do:
(21)       for $i = 1$, $i$++, $i$<=num do:
(22)         communicate with the node $i$ by using RequestVote RPC;
(23)         if received vote from most follower nodes then:
(24)           the number of term is increase;
(25)           become the leader and step to phase 2;
(26)           break
(27)         else:
(28)           state = follower;
(29)           break
(30)     case supervisor do:
(31)       If receive the ReputationValue RPC from follower node then:
(32)         send UNL data to node;
(33)       if receive the RequestVote RPC from candidate node then:
(34)         if candidate is not in UNL then:
(35)           vote to candidate;
(36)         else:
(37)           reject vote to candidate;
(38)       if the term in heartbeat request from the leader is more than current term then:
(39)         become the follower and step to phase 2; /* the supervisor node of last term no longer serves as supervisor in the new
     term */
(40)         break
(41) end
```

ALGORITHM 1: Leader and supervisor election phase.

node list. A new leader needs to be elected again when the leader is decided as the unfaithful node.

4.2.3. Consensus Phase. Store the valid proposal into a block is a permitted operation when most members in CTI sharing collaboration consortium members agree with it. When the supervisor decided the threat proposal was valid, the leader node broadcasted an AppendEntries RPC to all follower nodes. The description of AppendEntries RPC is shown in Table 7.

Each follower node that receives the AppendEntries RPC confirms the correctness of the proposal in that message to the leader when verification is passed, the standard of a correct proposal as shown in Table 8.

A consensus has been reached when the leader receives verification responses from the supervisor and more than 51% of followers. Then each follower node records the threat proposal along with the term and index number on their local blockchain. The details of implementation in phase 3 are given in Algorithm 3.

Table 6: The description of reputationcompute RPC communication.

Parameter	Description
Term	The current term of leader node.
NodeID	The id of the node.
PrevIndex	The index of consensus proposal immediately preceding new ones.
Entries[ ]	The threat proposal that was generated.
Return	Description
Term	The current term of leader node.
Success	Set to true when the threat proposal submitted is valid (details can be viewed in chapter 5.1).
Fail	Set to true when the threat proposal submitted is invalid ((details can be viewed in chapter 5.1).

```
Data: state: the state of a node, AC: the alert information from internal organization or the original CTI from external organization,
    threshold: threshold is the predefined constant that distinguishes the faithful node and unfaithful node, num: total number of nodes
    in the CTI sharing consortium network.
Result: void
 (1) begin
 (2)   switch state do:
 (3)     case "leader":
 (4)       generate proposal when received the AC from client;
 (5)       for i = 1, i++, i<=num do:
 (6)         send proposal and AC to node i;
 (7)       communicate with the Supervisor node by ReputationCompute RPC;
 (8)       if "success" in the return of ReputationCompute RPC then:
 (9)         step to phase 3; /* The leader provides a high-value threat proposal correctly, which needs to be stored in each node
    through phase 3 */
(10)         break
(11)       else:
(12)         step to phase 2 again to process the new alert information from client; /* The leader failed to provide the correct
    proposal of this alert */
(13)         break
(14)     case "follower":
(15)       generate threat proposal based on the the AC from leader;
(16)       communicate with the Supervisor node by ReputationCompute RPC;
(17)       receive the UNL data from the supervisor;
(18)       if the leader node is in UNL then:
(19)         term number +1; /* The leader node may provide too much false proposal due to malware control, so a new leader
    needs to be elected again in the phase 1 */
(20)         break
(21)       break
(22)     case "supervisor":
(23)       receive the ReputationCompute RPC from all nodes in the network;
(24)       compute the reputation score of nodes based on reputation model;
(25)       if threat proposal from node i decided as "success" then:
(26)         the reputation score of node i increase;
(27)         if the reputation score of node i >= threshold then:
(28)           remove node i from UNL;
(29)       else:
(30)         the reputation score of node i decrease;
(31)         if the reputation score of node i < threshold then:
(32)           add node i to UNL;
(33)       for i = 1, i++, i<=num do:
(34)         send UNL to the node i;
(35)       break
(36) end
```

Algorithm 2: Reputation model computing phase.

Table 7: The description of appendentries RPC communication.

Parameter	Description
Term	The current term of leader node.
LeaderID	The id of leader node.
PrevIndex	The index of consensus proposal immediately preceding new ones.
Entries[ ]	Proposal entries to store in each follower node (empty for heartbeat request).
LeaderCommit	The commitIndex of leader node.
Return	Description
Term	The current term of leader node.
Success	Set to true when verification of proposal that from leader is passed.

Table 8: The description of correct proposal in consensus phase.

Index	Criteria
Term	Leader's term ≥ follower's term.
PrevIndex	The prevIndex of this proposal's is more than the immediately preceding new ones.
Entries[ ]	This proposal's detailed information that from leader is the same as the responses of reputation model from supervisor.

```
Data: state: the state of a node, num: total number of nodes in the CTI sharing consortium network.
Result: void
 (1) begin
 (2)   switch state do:
 (3)     case "leader" do:
 (4)       for $i = 1$, $i$<=num, $i$++ do:
 (5)         send valid threat proposal to node $i$
 (6)         if number of ack message received <1/2num then:
 (7)           update the index;
 (8)           respond to client;
 (9)           break
(10)     default do:
(11)       received the valid threat proposal from leader;
(12)       if valid threat proposal from leader is correct then:
(13)         update the index;
(14)         send ack message to leader;
(15)         break
(16) end
```

Algorithm 3: Consensus phase.

5. Reputation Model

5.1. Model Scheme. Naive Bayes algorithms as an instance to demonstrate the Reputation model proposed in this paper. Let eigenvector $X = \{x_1, x_2, \ldots, x_k\}$ indicates to the IoC that generated by follower node $\{n_1, n_2, \ldots, n_k\}$, where k is number of follower nodes that provided threat proposals. We assume that there are N nodes in the network. The proportion of follower nodes that submit threat proposal to supervisor node by ReputationCompute RPC in phase 2 is $P(y) = k/N$. The probability of a proposal determined by follower nodes can be written as $P(y \mid X)$. Assume that the node provides information independently, then the equation can be further written as follows by using Bayes' theorem.

$$\begin{aligned} P(y \mid X) &= \frac{P(y) * P(X \mid y)}{P(X)} \\ &= \frac{P(y) * \prod_{i=1}^{i=|k|} P(x_i \mid y)}{P(X)}. \end{aligned} \tag{1}$$

The consensus is reached among the CTI sharing collaboration consortium by evaluating the credibility of proposal received from the leader node in the method that checks eigenvector X. Supervisor node calculates the reputation score of leader node and each follower node based on the credibility of IoC information eigenvector. We only pick the proposal eigenvector with $P(y|X) \geq P(T)$, which means valid threat proposal, where $P(T)$ is the threshold that set by situation among CTI sharing collaboration consortium. (2) is a calculation method of valid threat proposal:

$$\text{threat proposal} = \begin{cases} \text{valid}, & \text{if} P(y \mid X) \geq P(T), \\ \text{invalid}, & \text{if} P(y \mid X) < P(T). \end{cases} \tag{2}$$

As shown in Table 6, the supervisor node decides the abnormal state mainly from the return value of ReputationCompute RPC submitted by each follower node. The decision rule of ReputationCompute result is presented in (3). Here, x_i indicates an instance of threat proposal from a node's decision, X_{valid} represents the random vector of complete valid threat proposal from the all-nodes decision:

$$\text{the value of return} = \begin{cases} \text{success}, & \text{if} x_i \in X_{\text{valid}}, \\ \text{fail}, & \text{if} x_i \notin X_{\text{valid}}. \end{cases} \tag{3}$$

5.2. Model of Reputation Computing. We list the symbol glossary in Table 9 to facilitate expressing the reputation model formulation.

We use DFA (Deterministic Finite Automaton) to describe the unfaithful node that has the following behaviors: provide wrong decisions or low-value IoC in the reputation model computing phase. A DFA is a quintuple $X\langle S, \Sigma, \delta, S_0, F\rangle$, where S is a finite set of states, S_0 is the initial state, F is a set of acceptable states. δ is a finite set of alphabets. Σ is conversion function, Σ can be expressed as $S \times \sum \longrightarrow S$.

As shown in Figure 9, we define the Distinguishing Automaton for the behavior of a follower node in the reputation model computing phase, in which the initial state is 0 ($S = 0$), acceptable states are [4–6] ($F \in \{4, 5, 6\}$), $\sum$ is the action.

As shown in Figure 10, we define the Distinguishing Automaton for the behavior of a leader node in the reputation model computing phase, in which, the initial state is 0 ($S = 0$), acceptable states are [5, 6] ($F \in \{5, 6\}$), $\sum$ is the action.

Criterion 1. Unfaithful Behavior of Follower. Let <1, tr> denote the behavior of node 1; if the state of the node is follower, it will be considered that node 1 has unfaithful behavior in the reputation model computing phase of the PoR consensus when the following situations occur:

(1) Node 1 reaches state 3 indicates that the threat proposal generated by node 1 from the alert information is decided as a fail by the supervisor node

(2) Node 1 reaches state 4 indicates that node 1 does not respond to the IoC in time

Criterion 2. Unfaithful Behavior of Leader. Let <1, tr> denote the behavior of node 1. It will be considered that node 1 has unfaithful behavior in the reputation model computing phase of the PoR consensus when node 1 reaches state 3 because the IoC generated by the node 1 from the alert information is decided as fail by the supervisor node.

The node i that has not submitted a valid threat proposal on time will be decided as unfaithful behavior according to Criteria 1 and 2. The method for calculation of the reputation score in node i can be expressed as (4) and (5), where R_i is the reputation score of node i, $t_{0,i}$ means the time when current term start, $t_{\text{current},i}$ means the current time of valid proposal reach consensus, M indicates the reputation weight that used for further incentives or penalties.

$$R_i = R_i + M * \frac{\sum_{t_{0,i}}^{t_{\text{current},i}} \text{faithful behaviors}}{\sum_{t_{0,i}}^{t_{\text{current},i}} \text{alerts}}, \tag{4}$$

$$R_i = R_i + M * \frac{\sum_{t_{0,i}}^{t_{\text{current},i}} \text{Unfaithful behaviors}}{\sum_{t_{0,i}}^{t_{\text{current},i}} \text{alerts}}. \tag{5}$$

As defined in Definition 3, when a new node joins the CTI sharing system, its reputation score is R_{init}; if the node matins faithful behavior in proposal detected, its reputation score R_i will increase and have more opportunities to be leader node or supervisor node. The node is unfaithful whose reputation score R_i is lower than the threshold R_{thld}. If the leader is an unfaithful node, its qualifications will be terminated in this term. The reputation model proposed in this paper can reduce the impact of unfaithful behaviors under malicious attacks or wrong decisions.

6. Performance and Evaluations

6.1. Performance of the PoR Algorithm. The proposed PoR algorithm in the consortium blockchain CTI sharing model can achieve Byzantine fault tolerance and defense against blockchain attacks. Compared to the main consensus algorithm of consortium blockchain, we analyze the security and performance of the PoR algorithm and summarize them in Table 10.

Crash Fault Tolerance represented the fail-stop or crash failure in that no malicious behaviors happened in a blockchain system. Byzantine Fault Tolerance represents the byzantine behaviors in blockchain systems, such as tampering or submitting wrong information, and Crash Fault Tolerance is a particular type of Byzantine Fault Tolerance. The blockchain using PBFT must meet the conditions that collect $2f + 1$ messages in each node if it wants to reach a consensus in $3f + 1$ server nodes so that the PBFT consensus algorithm can tolerate at most 33% malicious nodes or crash nodes. RAFT consensus algorithm can only be used in the nonbyzantine network because it cannot tolerate malicious nodes, but it can tolerate up to 50% nodes of crash fault. As an improvement of the RAFT algorithm, the PoR algorithm can achieve better performance in Crash Fault Tolerance and Byzantine Fault Tolerance; simultaneously, we demonstrate the conclusion by verifying the following hypothesis.

Hypothesis 1. PoR consensus algorithm can achieve Byzantine fault tolerance. The CTI proposal can be shared correctly by the PoR algorithm when the number of byzantine nodes is less than 1/2 of all nodes.

Table 9: Symbol definition about the reputation model.

Symbol	Description
Action	The set of operations performed by the leader, follower, and supervisor node in the reputation model computing phase. Action = {generateProposal, broadcast, sendRPC, receive, success, fail, timeout}, where "sendRPC" indicates to communicate with the supervisor node by ReputationCompute RPC, "success," "fail" is the value of return in ReputationCompute RPC.
Trace	A sequence on the set action, such as {receive⟶generateProposal⟶sendRPC⟶success}.
Behavior	The behavior of node 1 can be denoted as <1, tr>, the identification of the node is 1, and tr is a trace.

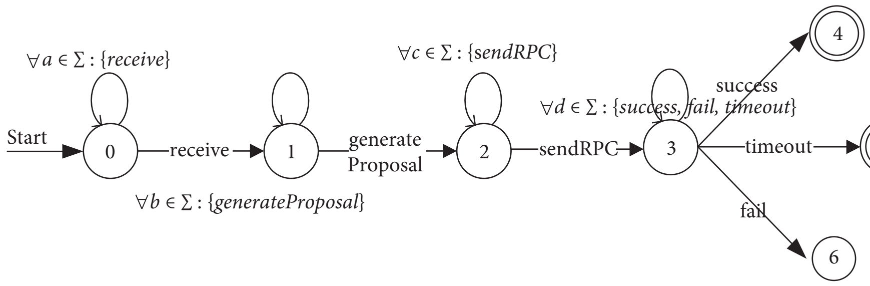
$\forall a \in \Sigma : \{receive\}$
$\forall b \in \Sigma : \{generateProposal\}$
$\forall c \in \Sigma : \{sendRPC\}$
$\forall d \in \Sigma : \{success, fail, timeout\}$
Start
0
receive
1
generate Proposal
2
sendRPC
3
success
timeout
fail
4
5
6

Figure 9: The state graph in reputation model computing phase.

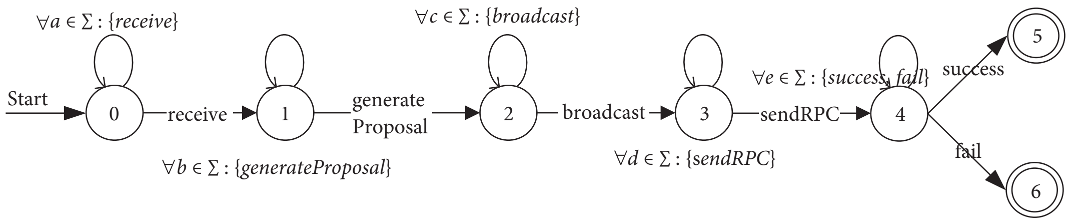
$\forall a \in \Sigma : \{receive\}$
$\forall b \in \Sigma : \{generateProposal\}$
$\forall c \in \Sigma : \{broadcast\}$
$\forall d \in \Sigma : \{sendRPC\}$
$\forall e \in \Sigma : \{success, fail\}$
Start
0
receive
1
generate Proposal
2
broadcast
3
sendRPC
4
success
fail
5
6

Figure 10: The state graph in reputation model computing phase.

Table 10: Performance in the consensus algorithm of consortium blockchain.

Algorithms	PBFT	Raft	PoR
Crash fault tolerance (%)	33	50	50
Byzantine fault tolerance (%)	33	N/A	50
Time complexity	$O(n^2)$	$O(n)$	$O(n)$
Security	Strong	Weak	Strong

Proof. The Byzantine fault tolerance of PoR depends on the reputation model in the consensus algorithm. Naive Bayes algorithms as an example of the reputation model in this paper. Assume that the number of byzantine nodes in the network is f, supervisor node in PoR can analyze correctly byzantine behaviors from eigenvector X composed of detailed information detected by all follower nodes when the total number of nodes in CTI sharing collaboration consortium network is more than $2f+1$. So PoR algorithm can tolerate 50% byzantine nodes or crash nodes.

The metrics of time complexity represented the communication cost and scalability of the consensus algorithm. Adding blocks to blockchain in PBFT needs verification by communicating in every two nodes and three-phase commit, the time complexity of PBFT is $O(n^2)$. Consensus processes in Raft and PoR only require the leader nodes to send messages to the follower nodes, and there is no need to communicate between the follower nodes. So, the time complexity of Raft and PoR is $O(n)$.

The security metrics represented the defense ability against consortium blockchain attacks such as bribery attacks. Bribery attacks mean the attacker deliberately bribed the node in the blockchain system to generate a block that is beneficial to the attacker. The bribery attack will occur in the PBFT consensus algorithm when the number of compromised nodes exceeds $2f+1$ in a blockchain system with a total number of nodes is $3f+1$. However, once the leader node is compromised, the blockchain system with the RAFT algorithm will reach a consensus beneficial to the attacker due to the lack of Byzantine Fault Tolerance. So, the security performance of RAFT is weaker than PBFT. The reputation value of nodes with malicious behaviors will decrease rapidly due to the reputation model's role in PoR consensus. The nodes with low reputations will not be able to become the leader nodes that dominate the consensus. In addition, the POR algorithm

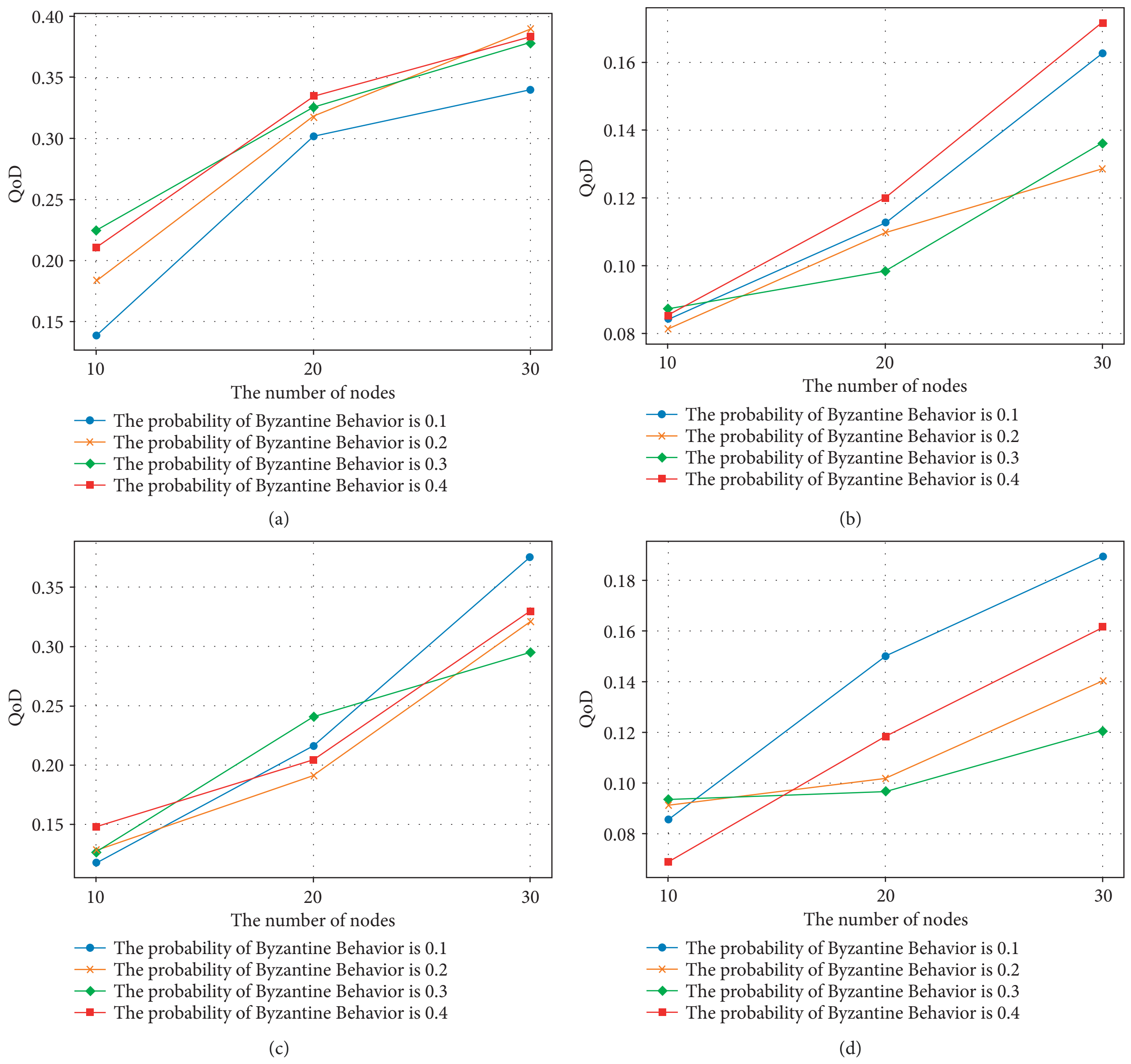

Figure 11: The security performance of PoR-based CTI sharing model. (a) $R_{\text{init}} = 50$, $R_{\text{thld}} = 10$, $M = 5$. (b) $R_{\text{init}} = 50$, $R_{\text{thld}} = 10$, $M = 15$. (c) $R_{\text{init}} = 50$, $R_{\text{thld}} = 20$, $M = 5$. (d) $R_{\text{init}} = 50$, $R_{\text{thld}} = 20$, $M = 15$.

uses the double confirmation mechanism to reach consensus; thus, it has a good defense against attacks.

6.2. Evaluations. We conducted the experiments using a computer with an Intel Core i5 and 16 GB RAM running macOS operating system. The construction of the reputation model is implemented using Python3.6. The PoR consensus algorithm uses Golang1.14.7. To further test the performance of the proposed approach in a cluster environment, we use the technology of container, thread, and virtual machine to represent the different network nodes, the technology of container, thread, and virtual machine are implemented with goreman0.3, docker18.09 and VMware fusion11.5.5. We created a test network in a simulation environment to confirm our approach can meet CTI sharing and exchange requirements.

We compare the PoR-based CTI sharing model with other consortium blockchain-based CTI sharing models that use different consensus algorithms discussed as follows:

(1) Byzantine Fault Tolerance Consensus Based Model. Store the proposal of CTI into a block is a permitted operation when confirmed by most members of the CTI sharing collaboration consortium. Every two nodes need to verify with each other to confirm CTI proposal to prevent Byzantine attacks in the network. A typical example of this model is Tendermint [41].

(2) Crash Fault Tolerance Consensus Based Model. Store the proposal of CTI into a block is a permitted operation when most members in CTI sharing collaboration consortium agree with it. This model

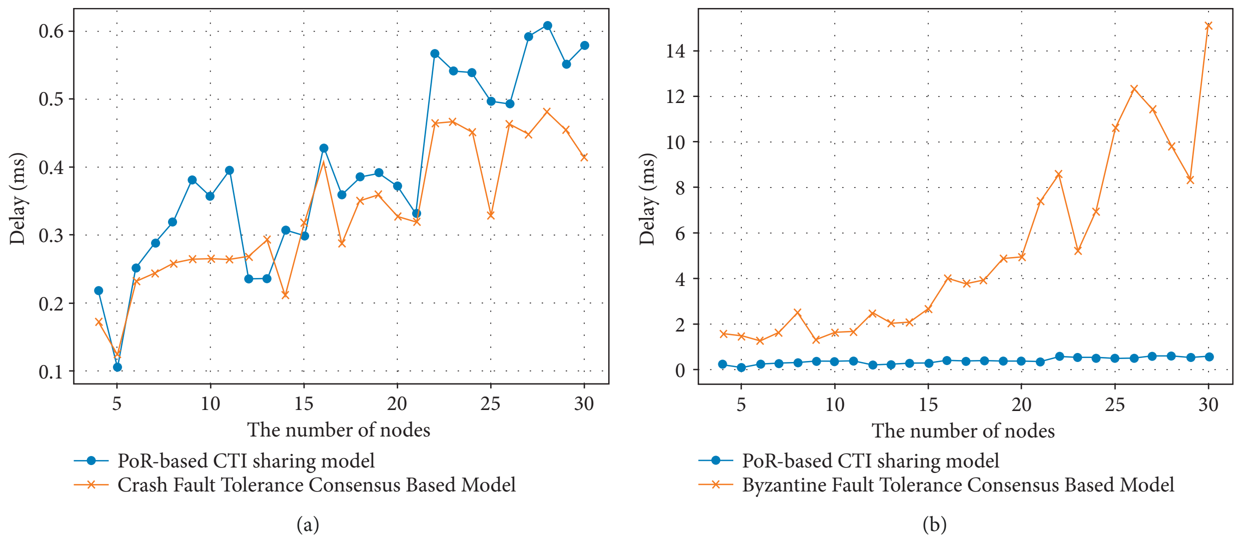

FIGURE 12: Latency to reach consensus with different nodes. (a) Compared with the CFT-based model. (b) Compared with the BFT-based model.

can achieve low latency and high throughput, but it only is used in a nonbyzantine environment. The typical example of this model is HyperledgerFabric [42] (v1.4 and above).

6.2.1. Experiment 1: The Security of the PoR-Based CTI Sharing Model. The security of the PoR-based CTI sharing model is measured by the cost of time in distinguish byzantine nodes in the network. We use the metrics of 'Quality of Detection in Byzantine Node (QoD)' to quantify the performance in security. The calculation method of QoD is described in (6), where $\sum$ Consensus means the time consumed of total threat proposal in reach consensus, $\sum$ ByzantineNode indicates the time consumed that all byzantine nodes were determined to be unfaithful node in the network.

$$\mathrm{QoD} = \frac{\sum_{t_{0,i}}^{t_{\mathrm{current},i}} \text{Byzantine Node}}{\sum_{t_{0,i}}^{t_{\mathrm{current},i}} \text{Consensus}}. \tag{6}$$

The experiment has been simulated under various conditions: Reputation weight, Reputation score threshold, Probability of byzantine behavior, The number of nodes. The experiment results in Figure 11 illustrate that the time increase as the proportion of the byzantine nodes and the scale of the sharing collaboration consortium varies.

6.2.2. The Efficiency Comparison of Different Sharing Models. The method of our evaluation is measuring the efficiency by latency and throughput. Latency refers to the time required for a single proposal of CTI to reach the consensus on the whole network, the process of a proposal update in the blockchain, including the reputation model computing phase and consensus phase. The experiment compares the latency between the PoR-based CTI sharing model and other blockchain CTI sharing models, as shown in Figure 12. Although the latency of our approach is worse than the CFT-based model by about 20% due to the confirmation mechanism of the reputation model in the PoR algorithm, it is still remarkably better than the CFT-based model.

Throughput is represented in the PoR consensus algorithm as the number of transactions of the CTI proposal that reach a consensus over time. We use ten client nodes to generate 1000 transactions of CTI proposal and calculate the corresponding throughput based on the time required to reach a consensus under different numbers of transactions. As shown in Figure 13, with the number of nodes being further increased, our proposed approach's throughput is better than the BFT-based model. In addition, there is a loss of about 30% in throughput compared with the CFT-based model because of byzantine fault tolerance supported by our approach.

6.2.3. Efficiency Performance Comparison under Massive CTI Data. Rapid information sharing is an essential attribute of CTI data, determined by the nature of cybersecurity attacks. For example, 60 percent of malicious domains have a survival time of one hour or less, which means that the value of some CTI data can be zeroed out in a very short period. In addition, the amount of threat intelligence data is hard to count in the CTI sharing system, massive CTI data can hinder the efficiency performance of blockchain network.

Therefore, in this experiment, we compare the PoR-based CTI sharing model with the PBFT-based CTI sharing model [37] for efficiency performance to prove that our approach can perform well in the real network environment of massive CTI data. In the simulation environment, we use 100-client nodes to generate a large number of CTI proposal transactions; the size of every CTI proposal generated is 256 KB. As shown in Figure 14, the efficiency performance of the PoR-based CTI sharing model is better than the PBFT-based CTI sharing model due to communicational complexity is greatly improved. As the number of CTI transactions

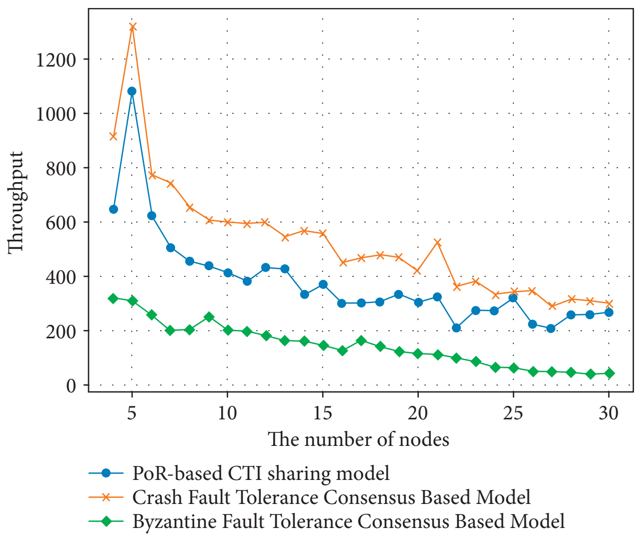

FIGURE 13: The performance of throughput with different nodes.

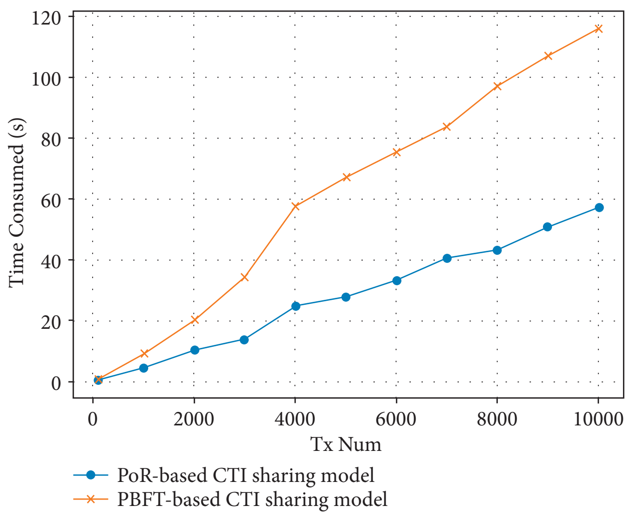

FIGURE 14: Efficiency performance comparison under massive CTI data.

increases, the PoR-based CTI sharing model takes less time to reach consensus than the PBFT-based CTI sharing model.

6.3. Summary. In the simulation environment, compared to the Crash Fault Tolerance Consensus Based Model, the PoR-based CTI sharing model requires an additional reputation computing process, so there is a loss in efficiency of consensus. However, our model still has advantages in latency and throughput performance compared to the Byzantine Fault Tolerance Consensus Based Model. Thus, our results show that the PoR-based CTI sharing model reaches a better performance balance in speed, scalability, security, and byzantine fault tolerance.

7. Conclusions and Future Works

This paper's contributions include a novel cyber threat intelligence (CTI) sharing approach using consortium blockchain that leverages advancements in consortium blockchain and distributed reputation management systems to automated process and defends against cyber-attack threats, as well as a consensus algorithm called PoR (Proof-of-Reputation)-based reputation model for meeting the effectiveness and security requirements. We devised three test scenarios in a simulation environment to evaluate the proposed approach. Our evaluation results from simulation results show that the proposed PoR-based CTI sharing model can achieve the needs of exchange of threat intelligence data in terms of performance of speed, scalability, and security. Thus, it can be applied to CTI sharing and exchange scenarios.

Although our approach can defend against blockchain attacks such as bribery attacks, it would be worthwhile to design and implement a defense mechanism for the tailored attacks in the future, such as nodes with high trust scores beginning to generate false high-level threat proposals maliciously. Tailored attacks in the example are essentially one of a poisoning attack. It aims to deliberately increase the error rate of CTI by inputting untruthful threat proposals and making the organization vulnerable to advanced attacks. So, the reinforcement learning method or adversarial network can be used to find the optimal defense mechanism, which also is our next research idea.

Data Availability

All data included in this study are available upon request to the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research work was supported by the Ministry of Education Industry-University Cooperation Collaborative Education Project.

References

[1] L. Yue, P. Liu, He. Wang, W. Wang, and Y. Zhang, "Overview of threat intelligence sharing and exchange in cybersecurity," *Journal of Computer Research and Development*, vol. 57, no. 10, pp. 2052–2065, 2020.

[2] O. Yurekten and M. Demirci, "Citadel: Cyber Threat Intelligence Assisted Defense System for Software-Defined Networks," *Computers & Security*, vol. 191, Article ID 108013, 2021.

[3] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber Threat Intelligence Sharing: Survey and Research Directions," *Computers & Security*, vol. 87, Article ID 101589, 2019.

[4] PI. Llc, *Exchanging Cyber Threat Intelligence: There Has to Be a Better Way Sponsored by IID Independently Conducted by*, Ponemon Institute LLC, Traverse City. Michigan, USA, 2014.

[5] G. Lu, Y. Liu, Y. Chen, C. Zhang, Y. Gao, and G. Zhong, "A comprehensive detection approach of wannacry: principles,

rules and experiments," in *Proceedings of the 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 41–49, Chongqing, China, October 2020.

[6] W. Tounsi and H. Rais, "A Survey on Technical Threat Intelligence in the Age of Sophisticated Cyber Attacks," *Computers & Security*, vol. 72, 2017.

[7] D. Chismon and M. Ruks, *Threat Intelligence: Collecting, Analysing, Evaluating*, MWR Infosecurity, UK Cert, United Kingdom, 2015.

[8] F. Skopik, G. Settanni, and R. Fiedler, "A problem shared is a problem halved: a survey on the dimensions of collective cyber defense through security information sharing," *Computers & Security*, vol. 60, pp. 154–176, 2016.

[9] D. Homan, I. Thorpe, and C. Thorpe, "A new network model for cyber threat intelligence sharing using blockchain technology," in *Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–6, Canary Islands, Spain, June 2019.

[10] B. Bhushan and A. SinhaSagayamJ, "Untangling blockchain technology: a survey on state of the art, security threats, privacy services, applications and future research directions," *Computers & Electrical Engineering*, vol. 90, Article ID 106897, 2021.

[11] A. Gruhler, B. Rodrigues, and B. Stiller, "A Reputation Scheme for a Blockchain-Based Network Cooperative Defense," in *Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 71–79, IFIP/IEEE IM, Washington DC, USA, April 2019.

[12] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: a state of the art survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 858–880, 2019.

[13] O. Cabana, M. Debbabi, B. Lebel, M. Kassouf, R. Atallah, and B. L. Agba, "Threat intelligence generation using network telescope data for industrial control systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3355–3370, 2021.

[14] Y. Yuan and F. Y. Wang, "Blockchain and cryptocurrencies: model, techniques, and applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1421–1428, 2018.

[15] S. Nakamoto, *Bitcoin: A Peer To Peer Electronic Cash System*, Consulted, 2008.

[16] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xianwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2567–2572, Banff, Canada, October 2017.

[17] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, vol. 32, 2015.

[18] X. Fu, H. Wang, and P. Shi, "A survey of Blockchain consensus algorithms: mechanism, design and applications," *Science China Information Sciences*, vol. 64, no. 2, Article ID 121101, 2021.

[19] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in *Proceedings of the 2014 USENIX Annual Technical Conference*, pp. 305–319, Philadelphia, United States, June 2014.

[20] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[21] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99)*, pp. 173–186, USENIX Association, Berkeley, California USA, February 1999.

[22] Y. Chen, P. Liu, and W. Zhang, "Raft consensus algorithm based on credit model in consortium blockchain," *Wuhan University Journal of Natural Sciences*, vol. 2, no. 8, 2020.

[23] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A Proof-of-Trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 429–445, 2019.

[24] L. e. Wang, Y. Bai, Q. Jiang, V. C. M. Leung, W. Cai, and X. Li, "Beh-raft-chain: a behavior-based fast blockchain protocol for complex networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1154–1166, 2021.

[25] "STIX 2.1 Specification," 2022, https://docs.oasis-open.org/cti/stix/v2.1/cs01/stixv2.1-cs01.html.

[26] J. Friedman and M. Bouchard, *Definitive Guide to Cyber Threat Intelligence: Using Knowledge about Adversaries to Win the War against Targeted Attacks*, CyberEdge Group, Annapolis Exchange, Annapolis, MD, USA, 2015.

[27] C. Sauerwein, C. Sillaber, M. Andrea, and B. Ruth, "Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives," in *Proceedings of the 13th International Conference on Wirtschaftsin for matik*, St. Gallen, Switzerland, February 2017.

[28] L. I. Jian-hua, "Overview of the technologies of threat intelligence sensing, sharing and analysis in cyber space," *Chinese Journal of Network and Information Security*, vol. 2, no. 2, pp. 16–29, 2016.

[29] a T. A. X. I. I. Hail, "Open Source Cyber Threat Intelligence Provider in STIX Format," 2022, http://hailataxii.com.

[30] Cybox, "Cyber Observable eXpression," 2022, https://cyboxproject.github.io/.

[31] S. Qamar, Z. Anwar, M. S. Rahman, E. Al-Shaer, and B. T. Chu, "Data-driven analytics for cyber-threat intelligence and information sharing," *Computers & Security*, vol. 67, pp. 35–58, 2017.

[32] D. B. Rawat, L. Njilla, K. Kwiat, and C. Kamhoua, "iShare: blockchain-based privacy-aware multi-agent information sharing games for cybersecurity," in *Proceedings of the 2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 425–431, Maui, HI, USA, March 2018.

[33] K. Huang, Y. Lian, F. Dengguo, H. Zhang, Y. Liu, and X. Ma, "Cyber security threat intelligence sharing model based on blockchain," *Journal of Computer Research and Development*, vol. 57, no. 4, pp. 836–846, 2020.

[34] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: a review," *IEEE Access*, vol. 6, Article ID 10188, 2018.

[35] C. J. Fung, Q. Zhu, R. Boutaba, and T. Başar, "Bayesian decision aggregation in collaborative intrusion detection networks," in *Proceedings of the 2010 IEEE Network Operations and Management Symposium - NOMS*, pp. 349–356, Osaka, Japan, April 2010.

[36] W. Li, Y. Wang, J. Li, and M. H. Au, "Toward a blockchain-based framework for challenge-based collaborative intrusion detection," *International Journal of Information Security*, vol. 20, no. 2, pp. 127–139, 2021.

[37] C. Yanugunti and S. S. Yau, "A blockchain approach to identifying compromised nodes in collaborative intrusion detection systems," in *Proceedings of the 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf*

on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 87–93, Calgary, AB, Canada, August 2020.

[38] E. Bellini, Y. Iraqi, and E. Damiani, “Blockchain-based distributed trust and reputation management systems: a survey,” *IEEE Access*, vol. 8, Article ID 21151, 2020.

[39] S. Qamar, Z. Anwar, M. A. Rahman, E. Al-Shaer, and B. T. Chu, “Data-driven analytics for cyber-threat intelligence and information sharing,” *Computers & Security*, vol. 67, pp. 35–58, 2017.

[40] D. Bianco, “The pyramid of pain,” 2013.

[41] E. Buchman, “Tendermint: Byzantine Fault Tolerance in the Age of Blockchains,” Dissertation for Ph.D. Degree, University of Guelph, Guelph, Ontario, Canada, 2016.

[42] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, and A. D. Caro, “Hyperledger fabric: a distributed operating system for permissioned blockchains,” *Proceedings of the thirteenth EuroSys conference*, Porto, Portugal, April 2018.