electronics

Article

Optimization Scheme of Collaborative Intrusion Detection System Based on Blockchain Technology

Jiachen Huang [1], Yuling Chen [1,*], Xuewei Wang [2], Zhi Ouyang [1] and Nisuo Du [1]

[1] State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China; 18085795537@163.com (J.H.); zouyang@gzu.edu.cn (Z.O.); nsdu@gzu.edu.cn (N.D.)
[2] Computer College, Weifang University of Science and Technology, Weifang 261000, China; wangxuewei@wfust.edu.cn
* Correspondence: ylchen3@gzu.edu.cn

Abstract: In light of the escalating complexity of the cyber threat environment, the role of Collaborative Intrusion Detection Systems (CIDSs) in reinforcing contemporary cybersecurity defenses is becoming ever more critical. This paper presents a Blockchain-based Collaborative Intrusion Detection Framework (BCIDF), an innovative methodology aimed at enhancing the efficacy of threat detection and information dissemination. To address the issue of alert collisions during data exchange, an Alternating Random Assignment Selection Mechanism (ARASM) is proposed. This mechanism aims to optimize the selection process of domain leader nodes, thereby partitioning traffic and reducing the size of conflict domains. Unlike conventional CIDS approaches that typically rely on independent node-level detection, our framework incorporates a Weighted Random Forest (WRF) ensemble learning algorithm, enabling collaborative detection among nodes and significantly boosting the system's overall detection capability. The viability of the BCIDF framework has been rigorously assessed through extensive experimentation utilizing the NSL-KDD dataset. The empirical findings indicate that BCIDF outperforms traditional intrusion detection systems in terms of detection precision, offering a robust and highly effective solution within the realm of cybersecurity.

Keywords: collaborative intrusion detection; ensemble learning; weighted random forest

check for updates

Academic Editor: Valeri Mladenov

Received: 7 December 2024
Revised: 27 December 2024
Accepted: 8 January 2025
Published: 10 January 2025

Citation: Huang, J.; Chen, Y.; Wang, X.; Ouyang, Z.; Du, N. Optimization Scheme of Collaborative Intrusion Detection System Based on Blockchain Technology. *Electronics* **2025**, *14*, 261. https://doi.org/10.3390/electronics14020261

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the digital age, the importance of cybersecurity is increasingly recognized, with intrusion detection systems (IDSs) serving as a cornerstone for detecting and defending against unauthorized access and malicious activities. IDSs traditionally employ misuse detection and anomaly detection strategies, leveraging rule-based monitoring and machine learning techniques to maintain the confidentiality, integrity, and availability of systems [1]. However, as network attacks evolve, traditional IDSs have shown limitations in detection accuracy, alert-sharing efficiency, and the ability to identify novel threats. These systems often operate independently, lacking real-time data exchange with other security components, which limits their global perspective on cybersecurity threats [2]. Distributed environments exacerbate these challenges by increasing the difficulty of identifying and mitigating malicious events, necessitating faster response times and higher detection precision [3]. CIDSs, as an extension of distributed intrusion detection systems (DIDSs), integrate multiple technologies and collaborative strategies to effectively reveal large-scale coordinated attacks. This approach enhances detection accuracy and ensures efficient deployment in complex

networks. However, this strategy also introduces additional system overhead, particularly in data sharing efficiency, which requires optimization and improvement.

In recent years, the extensive adoption and application of blockchain technology have assumed an important role across various domains, with particular significance in the realm of network intrusion detection. As the digital ecosystem becomes increasingly intricate and cybersecurity threats continue to evolve in sophistication and frequency, the intrinsic properties of blockchain are conducive to establishing a resilient framework that can significantly enhance the detection and mitigation of security breaches [4]. By integrating a distributed ledger to consolidate security-related data from disparate nodes and employing smart contracts to automate the execution of predefined security protocols, blockchain facilitates the rapid and accurate identification of, as well as responses to, anomalous activities [5].

As a decentralized ledger technology, blockchain's immutability and transparency make it well suited for the long-term recording and tracking of potential malicious activities, enhancing the detection sensitivity and comprehensiveness of network intrusions [6]. By applying blockchain, CIDSs can more accurately capture and deeply analyze anomalous behavior patterns, optimizing detection efficiency [7]. Additionally, blockchain ensures tamper-proof records of suspicious activities, crucial for identifying security threats and taking prompt action [8,9]. However, current intrusion detection models face several limitations. They often fail to adequately consider the impact of device performance and network conditions on detection efficiency. Furthermore, independently deployed models are vulnerable to single points of failure. Existing models also overlook the negative effects of growing data volumes on detection efficiency. Although many CIDS solutions focus on optimizing detection strategies, they do not address the efficiency issues caused by data growth.

To surmount these constraints, this paper introduces a novel system architecture that delineates the network into a central primary domain and an array of secondary node constellations, thereby optimizing the distribution and efficiency of intrusion detection processes. This design includes a mapping set of leader nodes from each secondary region in the core primary region, aiming to ensure efficient and secure data synchronization and sharing among sub-regions. The essence of our contributions are encapsulated in the following points:

- A Blockchain-Based Collaborative Intrusion Detection System (BCIDF): We introduce a hierarchical and distributed CIDS architecture, where nodes are logically allocated to different secondary regions based on geographical and functional criteria. A mapping set of leader nodes is established in the core primary region to facilitate data synchronization and sharing across sub-regions.
- An Alternating Random Assignment Selection Mechanism: We propose a mechanism that includes a Random Assignment Strategy and a periodic inspection function to select leader nodes in secondary regions. Nodes with higher importance are given favorable configurations to be elected as leaders in the next cycle, reducing the risk of single-point failures and improving alert-sharing efficiency.
- A Weighted Random Forest (WRF) Algorithm: Unlike other approaches, we improve forest prediction ability by introducing purity scores to measure node confidence. This algorithm adjusts the weight of individual decision trees based on their performance during training and validation, enhancing detection rates.

The subsequent sections of this manuscript are structured in the following manner: Section 2 provides an overview of pertinent literature and foundational concepts. The architecture of the BCIDF framework is meticulously delineated in Section 3. We conducted

a detailed experimental evaluation in Section 4. Concluding remarks and prospective directions for future research are articulated in Section 5.

2. Background and Related Work

This chapter will be divided into two parts for discussion: background knowledge and related work. In the background knowledge section, we will focus on the development history of CIDSs and delve into the characteristics of the Random Forest algorithm and its application advantages in this field. In the relevant work section, the focus is on summarizing the cutting-edge research progress of using blockchain technology to enhance intrusion detection systems, while also outlining the current application status and development prospects of intrusion detection technology in the Internet of Things (IoT) environment.

2.1. Background

This section provides foundational knowledge and concepts related to CIDSs and the Random Forest algorithm, which form the basis of our proposed methodology.

2.1.1. Collaborative Intrusion Detection Systems (CIDSs)

James P. Anderson pioneered the concept of intrusion detection in 1980 with his seminal technical report, "Computer Security Threat Monitoring and Surveillance", laying the groundwork for this pivotal cybersecurity domain. IDSs typically employ two main detection methodologies: misuse detection and anomaly detection. The first method relies on known attack signatures and system vulnerabilities. The information collected by this pattern is compared with known databases of network intrusion and system misuse patterns to identify issues that violate security policies. The second approach monitors system behavior and detects deviations from normal activity, which may indicate an intrusion or misuse.

IDSs are often deployed independently, making them susceptible to advanced evasion techniques such as fragmented attack paths, hidden attack features, and low-observability attacks [10]. These systems lack a comprehensive and deep understanding of the network environment due to their limited detection perspectives and data collection scopes. In response to the limitations of traditional IDSs, CIDSs have emerged. As shown in Figure 1, CIDSs can be divided into three types. Wu et al. [11] were the first to propose a CIDS that consists of three data collectors to gather data from diverse sources and a central data processor for analysis. CIDSs are a type of DIDS characterized by the following:

- Multiple independent IDS nodes that collaborate to achieve a common goal [12].
- Enhanced detection through data sharing and multi-dimensional analysis [13,14].
- Improved responsiveness and adaptability to new and unknown attack patterns [15,16].

Since the introduction of the CIDS model by Wu et al [11], its distinctive "Message Queue" mechanism has been widely adopted in subsequent research. Both the "Event Dispatcher" and the final "Combined Decision" processes necessitate the dissemination of a significant amount of information within the network. Furthermore, some studies have introduced node collaboration mechanisms, which further increase the burden on the network.

According to the "Message Queue" mechanism, when a node receives a message, it forwards this message to all reachable neighboring nodes, excluding the one from which it initially received the message. This process aims to update the case libraries held by each node or to disseminate relevant information. While this method ensures that every node in the network eventually receives the message, it also presents significant issues, particularly in large-scale or highly interconnected networks. Such a propagation method can lead to a proliferation of duplicate messages, not only increasing unnecessary network traffic but

also providing malicious actors with opportunities to exploit this through denial-of-service (DoS) attacks, where an overwhelming number of messages can be sent to flood the network and cause service disruptions. Therefore, for applications requiring efficient handling of large-scale data, the aforementioned message propagation method is not ideal.

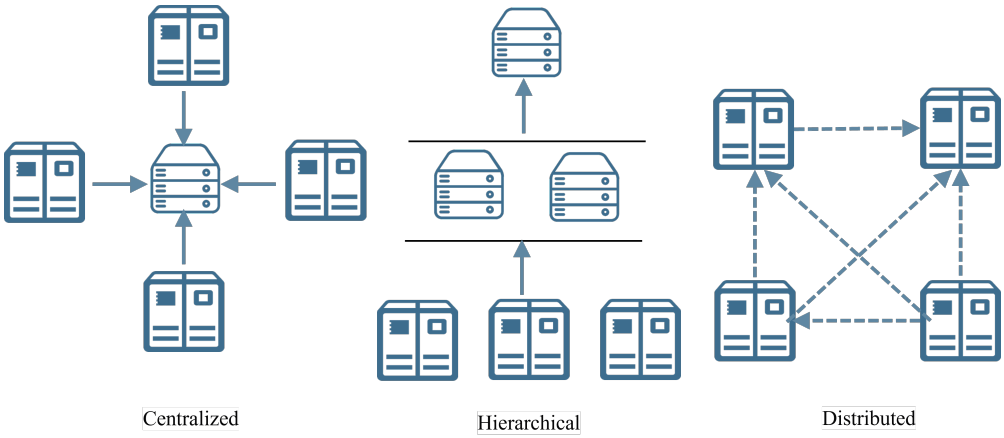


Figure 1. Categories of CIDSs.

Blockchain is a distributed ledger technology in which all participating nodes maintain a complete copy of the ledger. This means that once data are recorded on the blockchain, all nodes can access and verify these data without the need for multiple transmissions of the same data across the network. Furthermore, blockchain projects can be integrated with decentralized storage solutions (such as IPFS and Filecoin), allowing large files or datasets to be stored across multiple nodes in the network, with their locations recorded on the blockchain. Consequently, data need to be uploaded only once, and subsequent access can be achieved through references, thereby avoiding redundant transmissions. Therefore, leveraging blockchain technology to enhance the collaborative capabilities of CIDSs is a highly suitable choice.

2.1.2. Random Forest

The core mechanism of Random Forest (RF) lies in its enhancement of model performance through the introduction of randomness and ensemble learning. Specifically, during the construction of each decision tree, the dataset is resampled using the bootstrap method to form a new training set, and only a subset of randomly selected features is considered for splitting at each node. This process ensures diversity among the trees, thereby reducing the model's variance and enhancing its generalization capability.

Furthermore, each tree within RF is generated independently of the others, meaning that during the prediction phase, each decision tree makes an independent judgment about the data subspace it is responsible for. The final prediction is typically derived via majority voting (for classification tasks) or averaging (for regression tasks), which not only improves the accuracy of the model but also mitigates the influence of outliers.

Another notable advantage of RF is its ability to handle non-linear relationships and high-dimensional data while effectively assessing feature importance. This assessment is achieved by statistically evaluating the purity gain or error reduction contributed by each feature across all trees. Feature importance evaluation aids in subsequent feature selection, thereby optimizing model performance and simplifying its structure.

From an algorithmic implementation standpoint, Random Forest exhibits significant parallelization advantages. Given that the construction of each tree is independent, the algorithm can be executed in parallel on multicore processors or distributed computing environments, significantly accelerating the training process.

2.2. Related Work

2.2.1. Boosting CIDS Performance via Blockchain

In recent years, numerous academic efforts have been dedicated to enhancing the framework and efficacy of CIDSs through the application of blockchain technology. In 2017, Alexopoulos et al. [17] pioneered the research direction of constructing and optimizing CIDSs with blockchain, outlining the core design principles. Following this, Meng et al. [18] provided an in-depth analysis of the challenges faced by CIDSs, particularly focusing on the issues of efficient data sharing and trust mechanism establishment. They highlighted the innovative potential of blockchain technology in these areas, offering a fresh perspective on enhancing CIDS performance and guiding future research directions.

Within the context of designing IDSs for the Internet of Things (IoT) environment, the introduction of blockchain has demonstrated significant advantages. Li et al. [7] developed CBSigIDS, which effectively addressed the trust and security deficiencies in traditional CIDSs; however, the impact of large-scale data growth on detection efficiency was not fully considered.

The application of blockchain technology in new types of CIDSs has shown substantial progress, particularly in complex real-world scenarios. Gurung et al. [6] utilized the modularity, privacy protection, and membership management features of the Hyperledger Fabric platform to implement a blockchain-driven CIDS.

Hızal et al. [19] propose a blockchain-based IDS research framework aimed at enhancing the security of IoT networks and facilitating the global sharing of security solutions. This paper introduces a platform that integrates an IDS with blockchain technology, ensuring controlled service access and simplified network management through the definition of different node types. Research institutions can contribute their findings to the blockchain, allowing other entities to access IDS services via this platform.

Jiang et al. [20] introduce a blockchain-reinforced federated learning (FL) architecture specifically designed for the Industrial Internet of Things (IIoT). FL enables collaborative model training across distributed edge devices, ensuring data privacy and localized insights without requiring centralized data aggregation. However, the networked parameter-sharing mechanism in FL leaves it vulnerable to man-in-the-middle (MITM) attacks, potentially disrupting the model training process. To mitigate this threat, they propose a novel blockchain-reinforced FL architecture designed to enable cooperative intrusion detection.

2.2.2. Application of Intrusion Detection Systems in the Internet of Things (IoT)

The application of IDSs within IoT is paramount to safeguarding the security of both devices and networks. With the exponential proliferation of IoT devices, these interconnected nodes have emerged as potential sources of vulnerability, as they can be exploited by cybercriminals to launch attacks or pilfer sensitive information. An IDS is capable of monitoring network traffic and device behavior to identify and respond to potential threats, thereby shielding the IoT ecosystem from unauthorized access, malware, and other security risks.

Simultaneously, Hu et al. [21] applied blockchain technology in multi-microgrid (MMG) systems, proposing a collaborative strategy to enhance intrusion detection accuracy without reliance on a central server, marking one of the earliest practices of blockchain-enhanced Collaborative Intrusion Detection Systems. As research continues to evolve, blockchain-supported CIDSs have increasingly met the data security requirements across diverse sectors. Liang et al. [22] innovatively introduced the Micro-Blockchain Intrusion Detection (MBID) system in 2020, employing micro-blockchain to establish tamper-proof

vehicle intrusion detection mechanisms within a limited scope. However, they also highlighted the complexities involved in managing large-scale micro-blockchain networks.

Mirzaee et al. [23] sought to tackle emerging security threats through a two-tiered vehicle and edge Collaborative Intrusion Detection System, though they did not fully evaluate the impact of large-scale data processing on system efficiency. While the integration with established tools such as Snort enhanced system security, it also introduced additional complexities, including issues related to smart contract development and data format conversion.

He et al. [24] integrated Conditional Generative Adversarial Networks (CGANs) to construct a CIDS for drone networks, improving system security, privacy protection, and model training efficiency through blockchain-facilitated distributed federated learning. Furthermore, Alkhpor [25] designed an intelligent detection system that integrates a federated learning model, effectively identifying advanced persistent threats (APTs) and achieving high-precision, low-false-alarm detection objectives.

Zohourian et al. [26] introduced IoT-PRIDS, a novel framework designed for intrusion detection in IoT networks that utilizes 'packet representations' to establish a baseline profile for device behavior. This method focuses on understanding the communication patterns, services, and packet header values of IoT devices to provide a lightweight, non-machine-learning-based intrusion detection system. Kalaria et al. [27] introduced IoTPredictor, an advanced security framework designed to predict and detect malicious activities in IoT devices. IoTPredictor integrates an Anomaly Detection System (ADS) to proactively identify and thwart attacks within the complex IoT-fog computing landscape.

Malathi and Begum [28] introduced a cyber attack detection method for IoT networks that employs ensemble deep learning techniques. The proposed framework uses a combination of Graph Convolutional Neural Networks (GCNNs), RF, Random Space (RS), and Extreme Gradient Boosting (XGBoost) to enhance the reliability and accuracy of network traffic categorization.

Salim et al. [29] developed a cyber threat detection system for IoT networks that integrates digital twin technology and an optimized federated learning approach. Ruffo et al. [30] provided an empirical literature review on the state-of-the-art network intrusion detection systems (NIDSs) based on deep learning for defending software-defined networks (SDNs).

Alserhani [31] proposed a new intrusion detection system (IDS) design that combines Lightweight Deep Neural Networks (LDNNs) and Hybrid Genetic Simulated Annealing-Reinforced Opposition Algorithm (HGS-ROA) to offer an efficient and intelligent detection model for enhancing cybersecurity in dynamic environments.

However, current blockchain intrusion detection schemes in the IoT generally face challenges in balancing security and practicality. Faced with the massive and rapidly generated data characteristics of the IoT environment, detecting malicious intrusions in a timely and accurate manner has become particularly difficult. Therefore, this paper proposes a new research direction—the Blockchain-based Collaborative Intrusion Detection Framework—aiming to further explore the potential and solutions in this field. In Table 1, we present a comparative analysis between our study and prior research efforts, underscoring the distinct advantages of our proposed method.

Table 1. Comparison of different intrusion detection system schemes.

Schemes	Algorithm	Blockchain	Reliable Data Sharing	Collaborative IDS
Hızal et al. [19]	ML	✓	✗	✓
Jiang et al. [20]	FL	✓	✓	✗
Salim et al. [29]	FL	✗	✓	✗
Ruffo et al. [30]	DL	✗	✗	✗
Alserhani et al. [31]	DL	✗	✓	✗
Proposed system	WRF	✓	✓	✓

3. Blockchain-Based Collaborative Intrusion Detection System Framework

In response to the inherent constraints of conventional intrusion detection methodologies, we introduce a novel framework for a blockchain-empowered Collaborative Intrusion Detection System, termed BCIDF. In this section, we detail the components and operational mechanisms of the BCIDF. Specifically, we describe the system model in Section 3.1, the process for selecting regional leader nodes in Section 3.2, and the specifics of alert verification in Section 3.3.

3.1. System Model

We present a BCIDF framework with a hierarchical structure consisting of multiple sub-regions, as illustrated in Figure 2. The system comprises three core entities:

- System Manager (SM): Responsible for initiating and configuring the entire BCIDF system. Sets up the initial framework, divides the system into distinct functional areas, and ensures the system operates in a secure and stable environment.
- Regional Service Providers (RSPs): Receive potential intrusion data from Normal Nodes and verify these data. Convert verified data into alerts and upload them to the InterPlanetary File System (IPFS) for cross-validation by other RSPs. Act as full nodes on the blockchain, providing collaborative intrusion detection services to NNs.
- Normal Nodes (NNs): Collect and forward detection information to the appropriate RSPs. Serve as light nodes on the blockchain, utilizing the collaborative intrusion detection services provided by RSPs. May become regional leader nodes through a selection mechanism.

We construct the framework of our Collaborative Intrusion Detection System in four phases:

1. Sub-Region Partitioning: We adjust security configurations according to the specific threat landscape of each region, enabling better resource management and load balancing within localized areas.
2. Selection of Regional Security Providers: The System Manager employs an Alternating Random Assignment Selection Mechanism (ARASM) to select RSPs from each sub-region. These selected nodes not only act as leaders within their respective sub-regions but also serve as full nodes in the main region.
3. Alarm Verification: Regular nodes gather and forward various detection reports. Upon identifying potential attack indicators, these nodes broadcast requests for collaborative detection to the RSPs within their region, disseminating suspicious activity information to relevant nodes for further scrutiny. Each regular node independently matches this information against its internal database of attack signatures and integrates the feedback from other nodes to generate alarms that assist the RSP in making accurate judgments and decisions.

4. Alarm upload: RSPs will upload these alarms to the InterPlanetary File System (IPFS), and the RSP in the main area will download and analyze the alarm data. Through blockchain consensus, the decision results verified by different RSPs are recorded in the blockchain ledger using unique identifiers.

During the processing of suspicious alerts, ordinary nodes initially consult with the RSP nodes within their area. As full nodes in the blockchain network, RSP nodes maintain a comprehensive and validated index of alert rules. In cases where the RSP node lacks relevant entries, the system automatically invokes the anomaly intrusion detection algorithm, as detailed in Section 3.3, for a more thorough analysis of the alert. The findings from this algorithm are then uploaded to IPFS, whereupon the blockchain consensus mechanism evaluates whether to incorporate the newly identified alert characteristics into the existing rule repository. This dynamic process ensures that the system continuously learns and evolves, thereby enhancing its capability to address emerging threats efficiently.

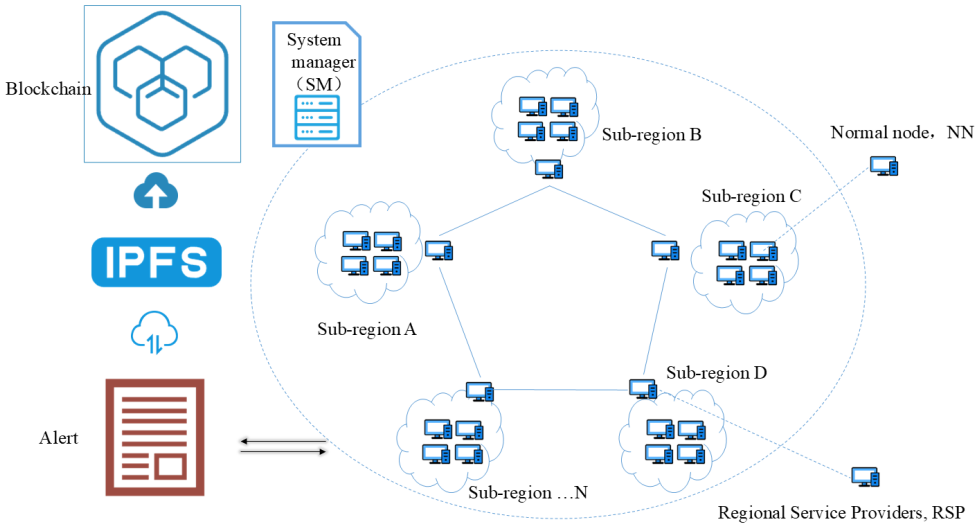


Figure 2. System framework.

Sub-Region Partitioning

In a distributed computing environment, detecting configuration differences between nodes in the detection area is a key design consideration. The importance of this principle lies in its ability to ensure that each component in a heterogeneous service environment receives optimized security monitoring based on its specific needs. Different types of detection mechanisms have been designed to identify various security threats, including but not limited to malicious attacks and abnormal traffic behavior. In addition, some applications have built-in specialized detection features, such as a rule set for preventing traffic fraud in VoIP (Voice over IP) services, which is an important component of the application itself [12].

From the perspective of SM, it is crucial to deploy various detectors reasonably to respond to different security incidents. This not only involves selecting appropriate tools to address specific security challenges, but also requires a deep understanding of the response characteristics of each detector. For example, Libsafe, as a detection tool focused on buffer overflow protection, should exhibit high sensitivity to relevant types of security events, while it may not have the same response strength to other forms of attacks such as flood attacks. In our plan, we divide CIDSs into smaller regions and need to carefully consider the types of most attacks within the region for secure configuration. This targeted deployment strategy helps improve the overall security of the system and can accelerate the learning and response process of specific security events.

3.2. Selection of Sub-Regional Leader Nodes

In the domain of computer networking, when two devices attempt to transmit data frames simultaneously over a shared physical medium, a collision may occur. Such collisions not only corrupt the data frames being transmitted but also necessitate retransmission to ensure the integrity of the information. To address this challenge, early researchers proposed the concept of segmenting the network into multiple subnets. Each subnet operates as an independent collision domain, which significantly reduces the likelihood of collision events and enhances the overall efficiency and reliability of the network architecture.

Inspired by this design philosophy, a similar approach has been adopted in the development of modern CIDSs. By meticulously dividing the CIDS architecture into a series of relatively independent operational sub-regions or detection units, it is possible to effectively mitigate potential analysis overload within any single region.

This section elaborates on the candidate random assignment selection mechanism, which is designed to mitigate the risk of regional dysfunction caused by leader node failures. This mechanism ensures the efficient execution of communication and control directives among devices within a sub-region. Candidates for leadership are equipped with the prerequisites for successful election in subsequent leadership processes.

To achieve this, a Random Assignment Strategy (RAS) is employed to allocate unique configurations to various nodes. Additionally, a periodic inspection function (PIF) dynamically rearranges and reallocates these configurations to adapt to changing network conditions. Nodes within a sub-region are categorized into three distinct roles: leader nodes, candidate nodes, and worker nodes, as depicted in Figure 3.

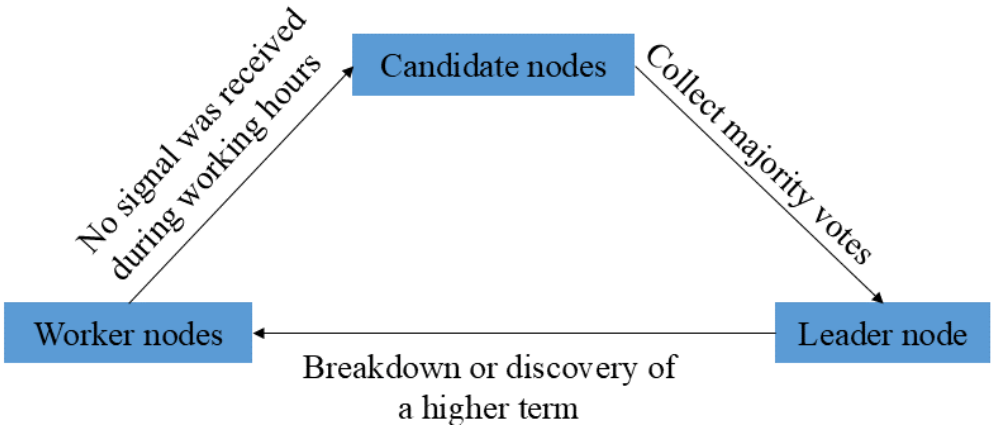


Figure 3. Node state transition.

Worker nodes are responsible for responding to requests from other nodes within the sub-region. Should a worker node fail to receive a collection signal from the leader node within a predetermined listening period, it transitions to a candidate node status and initiates a new leadership election process.

During this process, the candidate node that garners the majority of support from the other nodes through voting is officially recognized as the new leader node. Upon successful election, the incumbent leader node assumes the responsibility of supervising and coordinating the activities of the sub-regional cluster for the duration of its term. This ensures the stable operation and efficient execution of the system.

3.2.1. A Random Assignment Strategy

The Random Assignment Strategy (RAS) dynamically adjusts the term growth rate of nodes during the leader node election, thereby ensuring that candidate leader nodes are equipped with the conditions necessary for successful election in subsequent processes.

This mechanism is predicated on the assumption that nodes maintaining the most recent and consistent logs within the region are capable of serving as leader nodes. To facilitate efficient elections, the architecture adopts a differentiated configuration strategy, allowing the term of each candidate node to increase at varying rates. This design ensures the efficiency of the election process and the effective establishment of leader nodes. Each candidate node's configuration includes three key elements: (1) a unique priority identifier; (2) term duration; (3) term progression.

Upon initial integration into the system framework, each node is assigned a unique priority identifier that establishes the order of priority among nodes, denoted as P_i. This priority not only serves as a distinctive marker of the node's importance but also significantly influences the node's term growth rate and duration. At the same time, the structure of the node is denoted as $\pi^{(\mathcal{P}_i,k)}$.

The initial term duration for a node is defined by the following formula:

$$period_i = baseTime + k * (n - Pi), \tag{1}$$

where *baseTime* is a constant set significantly higher than the network latency, and k is a constant (in milliseconds) used to adjust the interval between terms; the higher the value of k, the greater the disparity in term durations among nodes. n represents the total number of nodes in the region.

In distributed systems, combining a high-priority node configuration with a shorter term duration is an effective strategy. This configuration allows high-priority nodes to swiftly detect leader node failures upon the expiration of their term. Due to the shorter term duration, communication and state synchronization among nodes are more frequent, thereby enhancing the speed of fault response, and Figure 4 illustrates a case of term change. This innovative mechanism significantly enhances the system's resilience by accelerating the detection and resolution of faults. It fortifies the system's integrity and dependability, ensuring swift recovery and maintaining operational continuity.

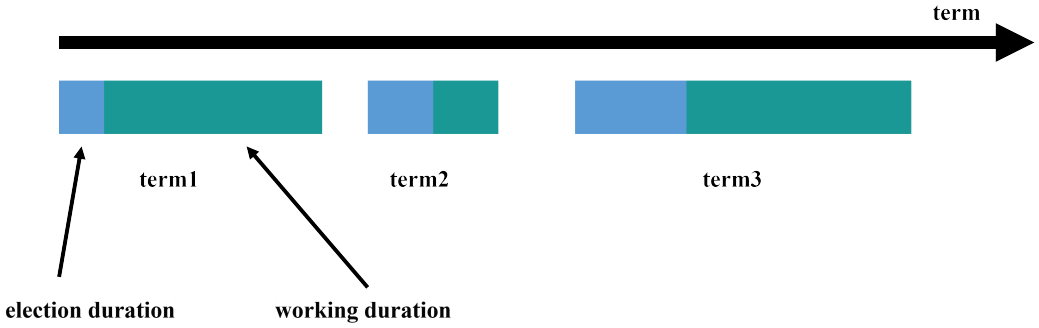


Figure 4. Term display.

The priority of a node dictates its term growth process within the CIDS. We denote the term of node S_i as T_{Si}. Should S_i initiate a new leadership campaign, T_{Si} is incremented according to Equation (2):

$$T_{si}^{(k+1)} \leftarrow T_{si}^{k} + p_i, \tag{2}$$

where $T_{Si}^{(k+1)}$ represents the current term and T_{Si}^{k} represents the previous term.

Furthermore, upon receiving a message with a higher term from another node, a node consistently updates its term:

$$T_{si}^{(k+1)} \leftarrow \max(T_{si}^{k}, T_{sj}^{(k+1)}), \tag{3}$$

where $T_{Sj}^{(k+1)}$ is the term received from another node and $i \neq j$. Regardless of other parameters in the received message, S_i always sets its term to the maximum value.

3.2.2. A Periodic Inspection Function

The leader node is determined through a voting process where a candidate node with the highest term gains the majority of votes from other nodes. This election process is theoretically independent of the specific configurations of nodes and the real-time status of log replication. However, if a candidate node with the highest term level does not possess the most up-to-date log information, its chances of election success will be significantly reduced. This is because such candidates do not have the qualifications to maintain the most up-to-date logs for other members of the system. Consequently, even if a candidate node enjoys the advantage of term growth due to its high-priority configuration, this advantage loses its effectiveness if it cannot ensure the recency of its logs. In such cases, the configurational advantage not only fails to take effect but may also contradict the principles of the RAS, thereby undermining the efficacy and integrity of the framework design. To solve this problem, we use the PIF. Such a function will help ensure that the leader node not only has a high priority but also maintains the consistency and integrity of the system state, thereby better aligning with the objectives of the RAS strategy.

As shown in Table 2, to mitigate the interference caused by obsolete configurations during the leader election process, ARASM introduces a hyperparameter known as the configuration clock (denoted as confClock). This parameter is a crucial logical indicator that signifies the temporal validity and freshness of the configuration, playing a pivotal role in the reordering of configurations. In the context of a node's configuration $\pi^{(\mathcal{P}_i,k)}$, the variable k denotes the configuration clock. In leadership elections, servers will never vote for candidates with outdated clock configurations. Similarly, in order to determine the new leader, the candidate's configuration clock should not be smaller than the voters' configuration clock. Heartbeat is a mechanism used to detect disconnection in a timely manner. By sending heartbeat data at regular intervals, it detects whether the other party is connected, which is part of the application protocol.

Table 2. Main parameters in ARASM.

	Parameter	Type
Node parameters	term	int 64
	leaderId	string
	prevlogIndex	int 64
	prevlogTerm	int 64
	leaderCommit	int 64
	timerPeriod	time.Duration
	priority	int 64
	confClock	int 64
Reply messages parameters	term	int 64
	success	bool
	logindex	int 64
	timerPeriod	time.Duration

The PIF is designed to reassign configurations that are likely to win future elections to nodes that possess the most up-to-date logs. Node configuration parameters can be allocated and adjusted according to system requirements and network dynamics, ensuring the efficacy and adaptability of the protocol. Initially, PIF periodically dispatches commands that track the log index, prompting nodes to respond with the status of their logs. Subsequently, the leader node collates these responses and reorganizes configurations for

each node. It broadcasts the new configurations in the subsequent process. Finally, if a node receives a different configuration, it updates to the newly assigned parameters. Since PIF rearranges configurations based on the state of log replication, nodes with the most recent logs that match the leader node's will be assigned high-priority configurations, even if they are not the highest-priority nodes.

In Figure 5a,b, due to the absence of the most recent log entries, high priority is provisionally assigned to nodes S_2 and S_3. Considering that S_3 originally holds a higher precedence, it is consequently designated as the node with the highest priority. In contrast, Figure 5c,d illustrate a scenario where nodes S_2 and S_4 fail and become unresponsive to the system. Upon S_2's recovery in the subsequent heartbeat cycle, its original high-priority status is instead transferred to node S_5, thereby maintaining system integrity and operational efficiency.

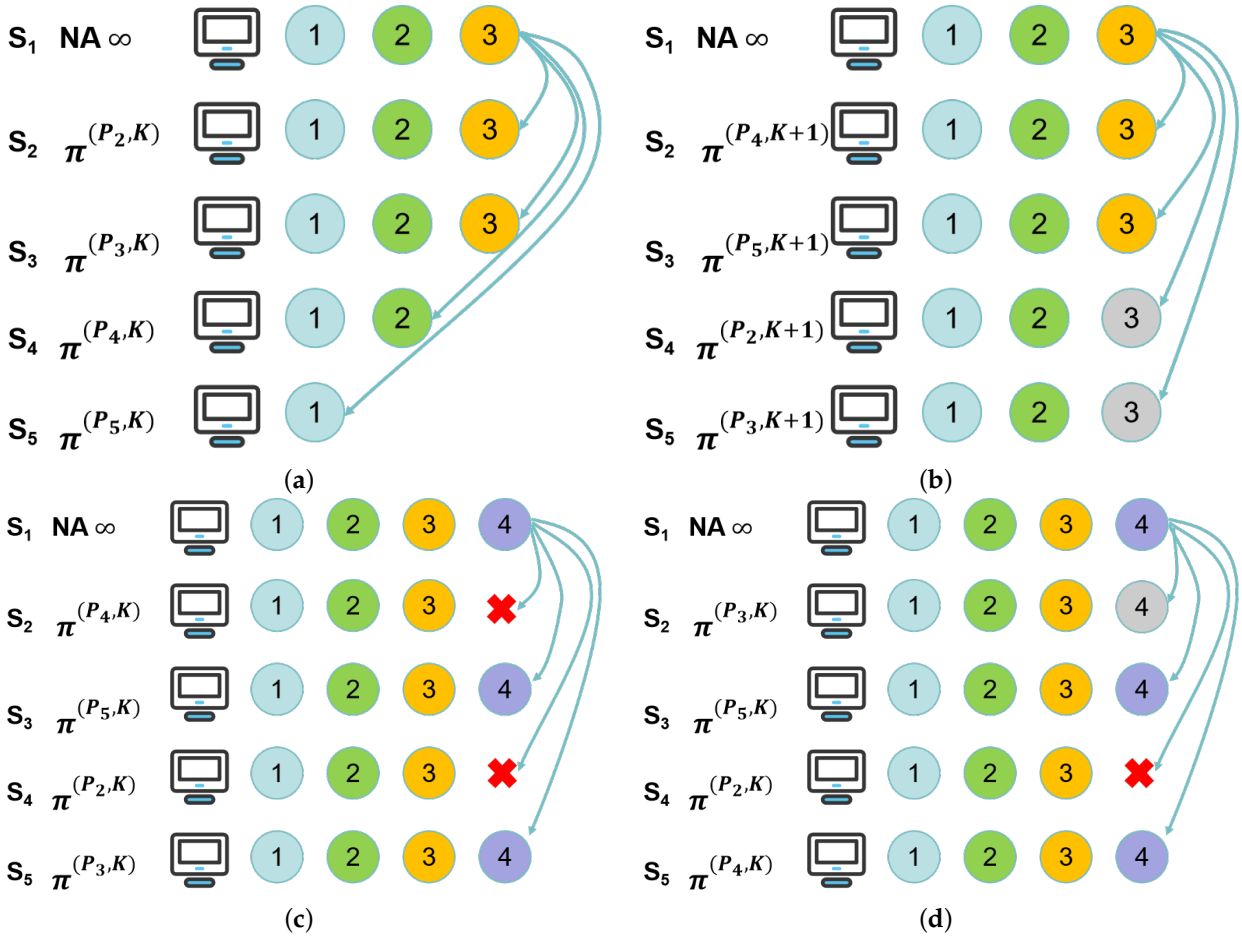


Figure 5. Examples for the PIF. (**a**) In the initial state, S_4 and S_5 have not been updated to the latest logs. (**b**) The status of system changes after the next heartbeat. (**c**) In the initial state, S_2 and S_4 have malfunctioned and are not responding to the leader node. (**d**) In the next heartbeat, S_4 still cannot respond to the leader.

The PIF achieves a resilient distribution of configurations and enhances the efficiency of leader node elections by avoiding the allocation of higher-priority configurations to candidate nodes that may fail. As a result, in the event of a leader node's failure, the node endowed with the most elevated priority is poised to promptly recognize the malfunction and instigate a new leadership election, preempting any other contenders. This process not only expedites the time to failure recovery but also improves the overall efficiency of the election process by prioritizing candidates most likely to succeed. This ensures that the system can continue to operate efficiently even in the face of complex and unpredictable environments.

Figure 6 illustrates a special scenario to elucidate the realization of three concurrent leader election activities. Assuming that the leader in term t, as depicted in Figure 5b, crashes, nodes S_2, S_3, and S_4 initiate their leader election processes at time points A_1, B_1, and C_1, respectively (prior to these points, they were in the working duration phase). When node S_2 sends out its candidate message for the leader election, it reaches nodes S_3, S_4, and S_5. Notably, node S_3, due to its higher priority, disregards this message, while node S_4 updates its term upon receiving the message at A_2 and subsequently casts its vote. Node S_5, still being in the working duration phase, discards the message. Similarly, when node S_2 broadcasts its candidacy for leader election, it garners a majority of votes, thereby becoming the new leader. It is crucial to highlight that during each process of electing a new leader, the system does not update the term of the nodes. Consequently, within concurrent elections, only one new leader is elected uniquely by the nodes. In summary, this figure demonstrates the robust mechanism of leader election under concurrent activities, ensuring that despite multiple nodes initiating elections almost simultaneously, the system maintains consistency and integrity by selecting a single leader during the election process.

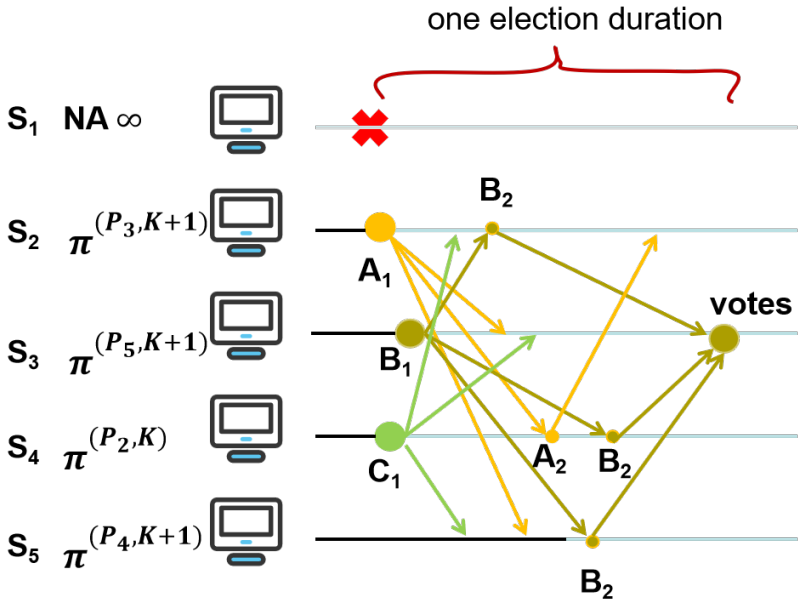


Figure 6. Selection of leader nodes.

The combination of RAS and PIF can achieve rapid leadership elections and avoid split voting. When candidates start new elections at the same time, the election with the longest term always defeats the other elections. When candidates synchronize to the highest item, they will reject requests that respond to lower items. Therefore, the voting for a given term is aggregated on a server, resulting in the termination of the election in a single campaign. Without these two functions RAS and PIF, it is highly likely that multiple election campaigns would occur within one election cycle, with candidates receiving the same number of votes and having to run again. Therefore, our method can avoid potential competition and promote the election process, thereby accelerating the transmission of information.

3.3. Alert Verification in BCIDF

In the Blockchain-based Collaborative Intrusion Detection Framework, ordinary nodes are tasked with collecting and forwarding critical detection information. These nodes are capable of identifying and responding to potential security threats. Upon detecting suspicious signs of an attack, an ordinary node immediately initiates a collaborative detection request to RSPs within its network region. This approach ensures that information about

suspicious attack behaviors is rapidly disseminated to all relevant nodes in the region for further analysis and verification.

Collaborative detection is a key method for efficiently analyzing and responding to anomaly detection information within the BCIDF. Through this detection method, nodes can share information and work together to address potential security threats, thereby enhancing the accuracy and speed of responses. This cross-node collaborative model not only strengthens the system's defensive capabilities but also effectively improves the identification and response to complex attack patterns through collective decision-making.

This paper constructs decision trees based on the description in RF [32]. Each node acts as a classifier. Throughout the training regimen, we ascertain the confidence levels of each leaf node across the ensemble of trees, subsequently harnessing these metrics to inform our decision-making processes during the subsequent testing phase. Mean impurity is an intuitive method for assessing feature importance. In the construction of a Random Forest, each feature contributes to a reduction in impurity when the decision tree nodes are split. By calculating the average reduction in impurity for each feature across all decision trees, we can evaluate the role of the feature in the model's performance. The magnitude of this metric is directly proportional to its influence on the model's predictive outcomes. An elevated value signifies a more substantial contribution to the forecasted results.

3.3.1. Confidence Calculation

After receiving anomaly detection information, ordinary nodes independently analyze it and cross-validate it against the built-in attack feature database. Due to potential differences in computational power, datasets, and algorithms among nodes, their predictive accuracy and reliability may vary. To quantify this variation, the concept of confidence is introduced to represent the level of certainty a node has in its predictive results.

In the Random Forest's training phase, we systematically determine the confidence of the leaf nodes for each constituent tree. This confidence is derived by assessing the impurity of nodes along the path from root to leaf, employing measures such as entropy $IE = -\sum_{i=1}^{k} p_i \log_2(p_i)$ or the Gini index $IG = 1 - \sum_{i=1}^{k} p_i^2$. A lower impurity value indicates higher node purity, a criterion also applied in the formation of the traditional Random Forest. Assuming the path's depth is denoted by q, the nodes $I_1, I_2, \ldots, I_q$ represent the impurity value of the node on this path. The purity of these nodes is ascertained through a defined metric of purity.

$$P_l = (1 + I_l)^{-1}, \ \forall \ l \in \{1, 2, \ldots q\} \tag{4}$$

where "l" represents the depth level of a node in the tree. The purity values obtained are $P_1, P_2, \ldots, P_q$. Purity Score Calculation The purity score S is used to calculate the node purity values:

$$S_l = lP_l, \ \forall \ l \in \{1, 2, \ldots q\} \tag{5}$$

Based on the characteristics of the data, certain transformations (such as logarithmic or square root transformations) are first performed. If a linear model is obtained after the transformation, linear regression is conducted. The regression equation for a linear model is

$$\check{S}_l = \alpha l + c \ , \tag{6}$$

where α denotes the gradient of the linear model, and conf represents the intercept on the y-axis. Subsequently, the overall confidence attributed to the tree is ascertained through the following computation:

$$conf = \frac{1}{1 + e^{-\alpha}}, \tag{7}$$

where delineates a direct relationship between the magnitude of α and the confidence level. A more pronounced positive α correlates with enhanced confidence. Consequently, leaf nodes boasting greater conf values are endowed with a higher degree of confidence. Algorithm 1 demonstrates the model construction process.

Algorithm 1: Generation of Weighted Random Forest.

Input: Training Set, D and number of trees to construct, n;
Output: A set of n trees $\{tree_1, \ldots tree_n\}$ with $con.f$ value for each leaf node of the trees

for $\forall t \in \{1, \ldots n\}$ **do**
  Draw bootstrap sample D_t from D
  $tree_t \leftarrow$ BuildTree(D_t, ϕ)
end
Function *BuildTree(D, purityList)*:
  Calculate $impuritgy$ from D using *Gini* or *Entrop*
  Calculate $puritgy$ from $impurity$ using (4)
  lnsert $purit$ to $puritList$
  if *satisfy the termination criteria* **then**
    $leaf \leftarrow$ createLeaf()
    Calculate $leaf.prob$ from D
    Calculate $leaf.conf$ from $purityList$ using(5), (6), (7)
    return leaf
  else
    node $\leftarrow$ createNode()
    Find best feature threshold, f_t from random feature subset
    Split D into D_{left}, D_{right} based on f_t
    node.left$\leftarrow$BuildTree(D_{left}, $puritList$)
    node.right$\leftarrow$BuildTree(D_{right}, $puritList$)
    return node
  end
end
return

3.3.2. Prediction

During the forest's evaluation phase, a test instance traverses the ensemble of trees, culminating at the respective leaf nodes. From this traversal, we derive a set of class probabilities $p_1, p_2, \ldots, p_n$ and their corresponding confidence metrics $conf_1, conf_2, \ldots, conf_n$. Optimal performance is achieved with trees of robust predictive power and minimal inter-tree correlation. These probabilities and confidences are amalgamated to yield a composite set of weighted probabilities for k potential classes. For a given test instance x, the weighted probability for each class c_k, denoted as $p(c_k|x)$, is determined by the following formula:

$$p(c_k|x) = \frac{\sum_{t=1}^{n} p_t(c_k|x) * conf_t}{n}, \ \forall \ c_i \in \{c_1, c_2, \ldots c_k\} \tag{8}$$

where $p_t(c_k|x)$ is the probability of class c_k occurring in the $tree_t$, and $conf_t$ represents the confidence of the corresponding $tree_t$. Algorithm 2 demonstrates the testing of optimizing random forests.

Algorithm 2: Testing of Weighted Random Forest.

Input: Test sample X:{x_1, x_2,.. x_f}
Output: Predicted class label L

for $\forall t \in \{1, \ldots n\}$ **do**
  $[prob(t), conf(t)] \leftarrow$ TraverseTree($tree_t$,X)
end
for $\forall t \in \{1, \ldots n\}$ **do**
  Calculate $p(c_k|x)$ using (8)
end
L$\leftarrow$ label(max{$p(c_1|x), p(c_2|x) \ldots p(c_k|x)$})
Function *TraverseTree* ($tree_t$, X)**:**
  $node \leftarrow root\{tree\}$;
  while $node \neq leaf$ **do**
    if $X_i \leq f_t$ **then**
      node $\leftarrow$ node.left
    else
      node $\leftarrow$ node.right
    end
  end
  return $node.prob, node.con.f$
end
return

4. Experiments and Analysis

This chapter will be structured into four parts for discussion. The first part will provide a detailed introduction to the construction and configuration of the experimental environment. The second part will supplement the relevant preparation work before the experiment, including but not limited to the introduction of the experimental dataset and evaluation indicators. The third and fourth parts will elaborate on the specific implementation details of the experiment.

4.1. Experimental Setup

In this study, we leveraged the CloudLab platform to deploy a series of RSPs, designed to emulate a distributed computing environment. Each virtual machine (VM) was configured with Ubuntu 20.04 LTS 64-bit Standard Edition as the default operating system to ensure uniformity across the experimental setup. To establish a blockchain peer-to-peer network, we installed the Ethereum client Geth (version 1.9.16-stable) and the Solidity programming language (version 0.5.16) on every RSP node. Additionally, the Truffle framework (version 5.1.34) was integrated to facilitate the development and deployment processes of smart contracts within the Ethereum environment formed by these nodes.

Performance evaluation was conducted on a standalone workstation equipped with an Intel Core i7-10700K CPU (featuring 8 cores and 16 threads, base clocked at 3.8 GHz), 32 GB DDR4 RAM, and 1TB NVMe SSD storage, running Ubuntu 20.04 LTS. The ManageEngine software (version 3.2.45) was utilized on this platform for the acquisition and analysis of performance metrics.

4.2. Implementation Details

4.2.1. Experimental Datasets

In the tree-based experiment, the dataset employed in the experiments is the NSL-KDD, a processed and optimized version of the KDD Cup 1999 dataset. It encompasses a variety of network intrusion behaviors as well as normal traffic patterns. The dataset comprises 43 feature variables and a label column, totaling approximately 494,021 records. Each record in the dataset comprises 43 features. Of these, 41 features pertain to the characteristics of the input traffic itself, capturing various attributes and metrics that describe the nature and behavior of the network traffic. The final two features serve as labels: one indicating whether the traffic is normal or represents an attack, and the other providing a score that reflects the severity level of the input traffic. These records are divided into standard training and testing sets; we set the ratio of training set to testing set to 7:3. Prior to the commencement of the experiments, the NSL-KDD dataset underwent preprocessing, which primarily included feature normalization to mitigate scale effects and binary encoding of the multi-class attack labels for a binary classification task.

The dataset encompasses four distinct categories of cyber-attacks: denial of service (DoS), probing, user-to-root (U2R) privilege escalation, and remote-to-local (R2L) access. Below is a detailed explanation of each type:

- Denial of Service: This attack aims to disrupt the normal traffic of a targeted system by overwhelming it with an excessive volume of requests that the system cannot handle. The intention is to exhaust the resources or bandwidth of the target network or service, thereby preventing legitimate users from accessing it. For instance, during a major sales event, an online retailer might be subjected to an overwhelming number of order requests that its network infrastructure cannot support, leading to a situation where paying customers are unable to complete their purchases. DoS attacks represent the most frequent type of attack in this dataset.
- Probing: Also referred to as reconnaissance, these attacks involve gathering information about a target network or system without actively compromising it. The attacker's objective is to stealthily collect valuable data, such as personal details of customers or financial information, which could later facilitate more invasive attacks. Probes are a critical first step in many sophisticated cyber campaigns, allowing attackers to identify potential vulnerabilities and defenses within the target environment.
- User-to-Root Privilege Escalation: U2R attacks start from a compromised user account and attempt to escalate privileges to achieve root or administrative access over the system or network. By exploiting vulnerabilities within the operating system or applications, attackers can gain unauthorized control, enabling them to perform actions that are typically restricted to system administrators. This type of attack poses a significant threat because it can lead to full compromise of the affected systems.
- Remote-to-Local Access: An R2L attack seeks to obtain local access to a machine from a remote location. In this scenario, the attacker does not have initial local access to the target system but attempts to breach the network perimeter and establish a foothold inside the network. This often involves exploiting software vulnerabilities or misconfigurations that allow external entities to execute commands on the target machine. Once inside, the attacker may further explore the network to expand their control.

4.2.2. Evaluation Parameters

In this study, the entire NSL-KDD dataset is utilized for testing purposes. The proposed model performs binary classification, where all attack categories present in the dataset are considered anomalies and labeled as 1, while normal behavior is labeled as 0. To better

describe the performance of the classifier on the test dataset, it is important to define the following metrics:

- True Positives (TPs): These are instances where the classifier correctly identifies an attack (anomaly) as such.
- False Negatives (FNs): These occur when the classifier incorrectly labels an actual attack as normal behavior, failing to detect the anomaly.
- True Negatives (TNs): These are cases where the classifier accurately identifies normal behavior as not being an attack.
- False Positives (FPs): These arise when the classifier incorrectly flags normal behavior as an attack.

The classifier's performance can be quantified using these four fundamental metrics, which form the basis for calculating various performance measures such as accuracy, precision, false positive rate, and F1-score. Understanding these metrics is crucial for evaluating the effectiveness of the binary classifier in distinguishing between attacks and normal activities within the dataset.

Accuracy is defined as the ratio of correctly classified instances to the total number of test instances, encompassing both attacks and normal behavior. The formula is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

Precision represents the proportion of correctly identified attacks out of all instances flagged as attacks by the classifier. The formula is as follows.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

The F1-score measures the balance between precision and recall, focusing specifically on the detection of positive classes (attacks) while excluding true negatives (actual normal behavior). Unlike the balanced accuracy metric, the F1-score does not account for negative classifications (TN). This score provides a useful measure for evaluating a model's ability to detect positives, making it particularly suitable for assessing anomaly or outlier detection models. The formula is as follows.

$$F1\text{-}score = \frac{2 * TP}{2 * TP + FP + FN} \tag{11}$$

The false positive rate (FPR) is the ratio of false positives to the total number of actual normal behavior observations. The formula is as follows:

$$FPR = \frac{FP}{FP + FN} \tag{12}$$

4.3. Comparative Analysis in Tree-Based Models

To assess the proposed approach's efficacy and performance metrics, we conducted an empirical study using the NSL-KDD dataset, a benchmark resource extensively recognized within the domain of network intrusion detection. This dataset offers a refined version of the original KDD Cup 99 dataset, mitigating certain limitations such as redundant records, which makes it more suitable for evaluating intrusion detection systems.

Prior to presenting the detailed experimental outcomes, it is imperative to outline the various parameters configured for the experiments. The configurations were meticulously selected to ensure that they reflect real-world scenarios while also providing a rigorous test environment for our method. The specifics of these configurations are summarized in the table below.

In Table 3, the "Maxi Tree Depth" parameter controls the maximum number of levels that any single decision tree can grow, while the "Number of Trees" parameter specifies the total number of trees within the ensemble model. These settings have a direct impact on the complexity and generalization ability of the model. The "Max Features" parameter determines the maximum number of features to consider when searching for the best split at each node; setting this parameter to "sqrt" means that the square root of the total number of features is considered.

Table 3. General model parameters.

Parameter	Value
Max Tree Depth	10
Number of Trees	100
Max Features	sqrt
Min Samples Leaf	5
Criterion	Gini, Entrop
Predictor categories	normal (0), anomaly (1)

Taking into account the characteristics of the specific dataset used in this study and drawing upon best practices from the relevant literature, we have decided to set the maximum tree depth to 10 and the number of trees to 100. Additionally, to enhance the stability of the model and minimize variance, we have set the max features to the square root of the total feature count, in line with recommendations found in the literature [32–34]. Criteria represent the criteria for dividing nodes. Gini represents the use of the Gini index, while Entropy represents the factual information gain standard. Predictor categories represent the types of predicted results.

In this part, the detection performance of the Blockchain-based Collaborative Intrusion Detection Framework (BCIDF) was evaluated through simulation experiments, with three other classic classifiers used for comparative analysis: Random Forest (RF) [32], Extra-Trees (ET) [33], and Global Refined Random Forest (GRRF) [34]. It was also compared with the current cutting-edge algorithms using tree types [35,36]. However, the special parameters in these two articles still need to be introduced.

- Extra-Trees (ET): The extreme random tree algorithm proposed by Pierre Geurts et al. further enhances randomness. In the process of node splitting, not only is the selection of features random, but the cutting points are also completely randomly determined without relying on output values. This method can generate a completely randomized tree with a structure independent of the output values of the training samples. In this study, three parameters are crucial, with the parameter max features representing the maximum number of features considered when constructing the optimal Extra-Trees model. The second parameter of the Extra-Trees method is the required number of samples (Min Samples Leaf) for segmenting nodes. The larger the value of Min Samples Leaf, the smaller the tree, the greater the deviation, and the smaller the variance. Therefore, its optimal value generally depends on the level of output noise in the dataset. The parameter number of trees represents the number of trees in the collection, and specific parameter settings can be obtained from the literature [33].
- Optimizing tree (OT): In this study, researchers used a genetic algorithm to optimize the RF model, and we configured the following special parameters when using this algorithm: Iteration times: We set the termination condition of the genetic algorithm to 100 generations. This setting ensures that the population has sufficient iterations to fully evolve towards convergence, avoiding premature convergence due to insufficient generations. Population size: 20. This size strikes a balance between maintaining

population diversity and controlling computational complexity, helping to prevent early convergence to local optima. Selection mechanism: We adopt the roulette wheel selection method, which assigns selection probabilities proportional to individual fitness to ensure the effective transmission of excellent genes to the next generation. Mutation rate: 0.1. An appropriate mutation rate can maintain population diversity while preventing the rapid loss of beneficial genes that may occur when the mutation rate is too low, thereby ensuring a stable and effective search process. The configuration of relevant parameters can refer to the literature [35,37].

In the construction of Random Forest models, while it is conventional for individual trees to be grown to their full depth, similar to independent decision trees, this practice can significantly escalate computational costs. To optimize efficiency without compromising model performance, our study imposed a limitation on tree depth, tailored to the dataset's scale and complexity. For feature selection at each node, we adopted the square root of the total feature count to determine the size of the feature subset, aligning with standard methodology. Our ensemble comprised 100 trees, with this parameter held constant unless otherwise noted. When constructing decision tree models, to prevent overfitting and ensure that the model generalizes well to unseen data, we have established a criterion that each leaf node must contain a minimum of 5 samples ($MinSamplesLeaf = 5$) for further partitioning to occur. In assessing node purity, we employed both entropy and the Gini index as splitting criteria, thereby enabling a comparative analysis of different research protocols impacts on model outcomes.

The results shown in Table 4 emphasize that BCIDF outperforms other tree-based algorithms in terms of accuracy and false positive rate in responding to intrusion. This is attributed to BCIDF's ability to more effectively identify malicious attacks and more effectively protect underlying data structures and relationships. In addition, BCIDF outperforms traditional RF and other decision tree algorithms in terms of false positive rate, which is a key indicator of intrusion detection systems that ensures fewer healthy instances are misclassified as anomalies. From the results, it can be seen that the accuracy of RRF is slightly lower than other algorithms because this algorithm focuses on generating a large number of shallow trees, while standard Random Forests tend to select a small number of deep trees. In the process of node separation, the selection of eigenvalues in ET is random, which significantly improves the generalization ability of the algorithm. Using a genetic algorithm to optimize the Random Forest model in OT effectively increases the diversity of the population, resulting in better and more accurate prediction performance.

Table 4. Comparison results of various algorithms.

Algorithm	Accuracy	FPR	F1-Score	Precision
RRF	64.5	18.4	82.3	79.8
ET	73.8	19.2	80.2	82.5
RF	74.1	20.1	81.4	84.3
BCIDF	78.4	17.3	83.5	86.8
EDT	75.6	17.6	81.5	79.9
OT	73.2	18.3	80.2	80.1

In the field of machine learning, particularly for ensemble algorithms based on decision trees such as Random Forests and Gradient Boosting Machines, the accuracy of the model is closely related to the number of decision trees that constitute these models and the maximum depth of each tree. Generally speaking, increasing the number of trees n enhances the model's generalization ability, thereby improving prediction accuracy. A greater number of trees means a stronger capacity for learning from data and a reduction in model variance.

Regarding the maximum depth of the trees, it determines the complexity of individual trees: deeper trees can capture more nuanced features within the data; however, excessively increasing depth may lead to overfitting.

To investigate how these two parameters influence the accuracy of models trained with different impurity metrics, we conducted a detailed experimental analysis, the results of which are presented in Figure 7. These findings underscore the importance of appropriately tuning hyperparameters in practical applications to ensure that the model has good generalization capabilities without overly fitting to the training data.

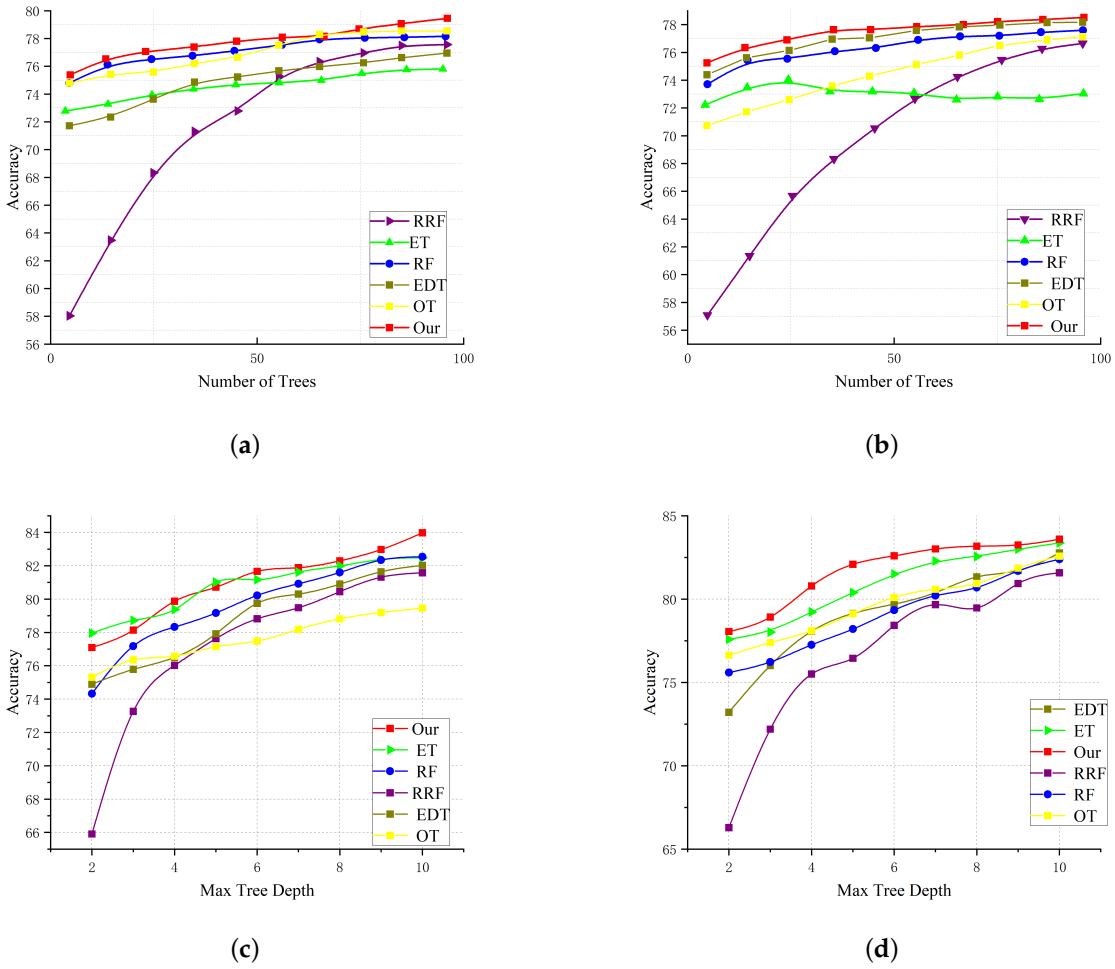


Figure 7. Influence of number of trees and maximum tree depth on accuracy, (**a**) I_G as impurity metric; (**b**) I_E as impurity metric; (**c**) I_G as impurity metric; (**d**) I_E as impurity metric.

The present study highlights the delicate balance between leveraging the power of ensemble methods to enhance predictive performance and maintaining control over model complexity to avoid overfitting.

In machine learning practice, the number of trees and the maximum depth of each tree in ensemble models based on decision trees are typically set to predefined finite values. This study evaluated five distinct detection algorithms on a target dataset. While these algorithms exhibited their unique strengths in detection accuracy, none surpassed the performance of the improved method proposed in this research.

The experimental results revealed that as the number of decision trees and their maximum depth increased, the recognition accuracy of these algorithms approached a fixed performance ceiling, demonstrating a clear trend toward convergence. This phenomenon clearly underscores the significant relationship between system performance and the

parameters of the number of trees and their maximum depth. It also suggests that beyond a certain level of complexity, increasing model complexity further does not necessarily yield substantial improvements in performance.

4.4. Spend Time Comparison Test

In this study, we conducted a systematic comparative analysis of the time efficiency in alert rule distribution, contrasting a traditional centralized server model with our proposed innovative approach. Our evaluation encompassed all nodes within the network. The traditional model employs a thread pool strategy to efficiently manage client connection requests, with clients querying the server for new alert rules at one-second intervals. Upon the release of new alert rules by the server, these rules are promptly disseminated to the querying nodes.

Our research introduces a candidate leader-based model that diverges from the conventional centralized feedback mechanism. When new alert rules emerge, our approach ensures active participation of all leader nodes in the distributed system's verification process, while node selection for appending blocks on the blockchain is coordinated by a consensus-driven algorithm. This contrasts sharply with the traditional centralized model, which relies on specific servers for rule dissemination, highlighting the advantages of decentralized processing.

Figure 8 illustrates the comparative study of alert distribution times under the two models: the traditional time series tracks the synchronization of nodes responding to newly issued rules from the server over time; in contrast, the time series of our proposed method reflects the cascade of rule updates initiated by management nodes through ARASM after introducing new rules. Our findings demonstrate that our methodology significantly outperforms the traditional centralized paradigm in terms of timeliness and efficiency, underscoring the importance of optimizing rule distribution processes in distributed environments.

By using RAS and PIF mechanisms, ARASM can quickly and effectively elect a new round of leader nodes, avoiding the collision of messages in traditional models and greatly accelerating the distribution of alerts. This makes our system more efficient in distributing alerts, which is an advantage that traditional models cannot match. Not only that, by dividing regions and reducing collision domains, nodes in each region will respond more quickly to leader nodes compared to traditional models, completing the distribution of alerts.

By default, each node randomly selects a fixed number of its neighbors—specifically ten—to propagate new rules. To validate the efficacy of our scheme in disseminating intrusion detection rules, we conducted a series of comparative experiments by varying the number of neighboring nodes involved in the propagation process. Each experimental configuration was repeated ten times to ensure reliability, with results averaged to provide a comprehensive evaluation.

In this experimental setup, we configured the number of forwarding nodes to be 5, 10, 15, 20, and 25, to assess their impact on forwarding time. Figure 8 illustrates a clear trend: as the number of forwarding nodes increases, the time required for network-wide rule dissemination decreases significantly. This phenomenon can be attributed to the fact that an increased number of forwarding nodes leads to a higher volume of nodes being updated per cycle, thereby reducing the need for intermediate transmissions and markedly enhancing overall forwarding efficiency. The specific results are shown in Figure 9.

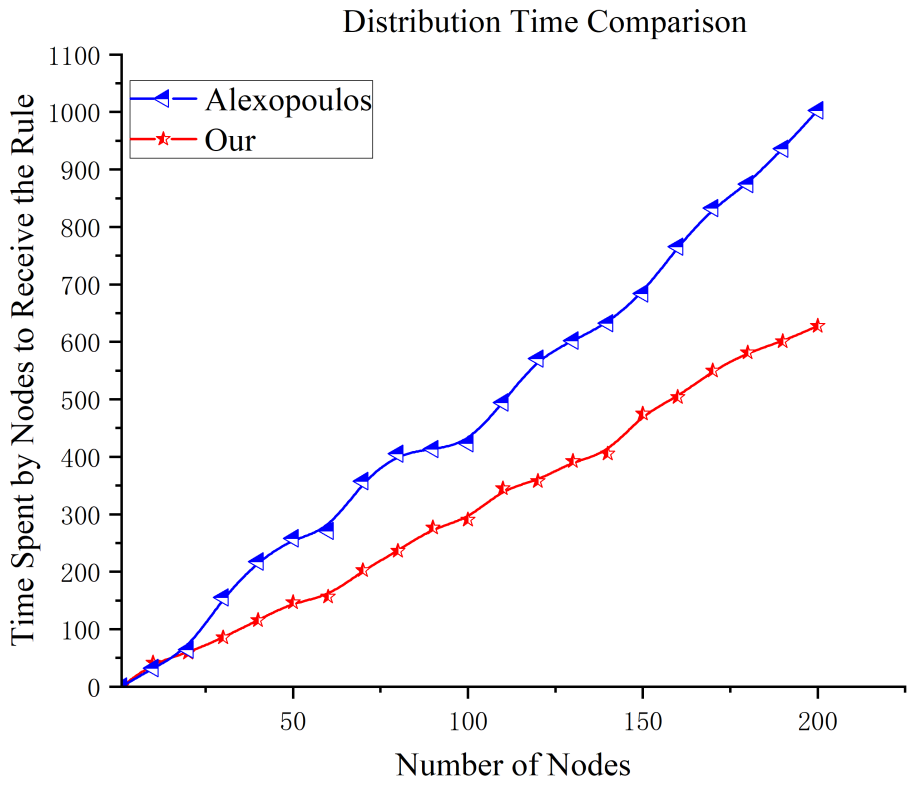


Figure 8. Comparison of distribution time among different schemes.

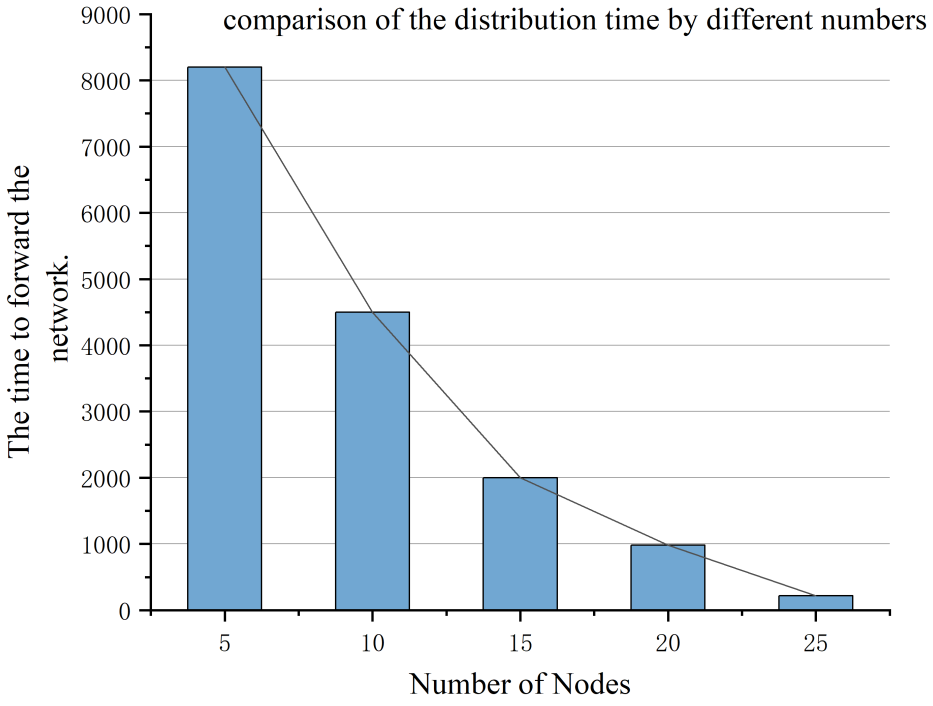


Figure 9. Comparison of the distribution time by different numbers.

5. Conclusions

This study proposes an innovative collaborative intrusion detection scheme that cleverly combines blockchain technology and improved Random Forest algorithm, aiming to improve the efficiency and reliability of intrusion detection in distributed network environments. By introducing a leadership node election mechanism, we have successfully solved

the time and resource cost issues associated with re-electing new leaders when leadership nodes fail. This mechanism ensures that each region has a stable leader node responsible for coordinating activities within the area, greatly reducing the time for alarm distribution within the system and improving the system's response speed and overall performance.

In the decision-making process, we first introduced "trustworthiness" as the weight of each leaf node in the Random Forest algorithm. The introduction of this concept is not accidental, but based on careful calculation of the impurity values of each node on the path. It not only makes the model more intuitive, but also significantly improves the efficiency of intrusion detection frameworks. Compared with traditional intrusion detection methods, our approach has demonstrated superior performance, achieving satisfactory results in detection accuracy, response time, and computational resource utilization.

For future research directions, we have set two main goals: firstly, to design an incentive mechanism to reward nodes that have made high contributions to network security, in order to encourage positive security behaviors. The second is to develop an internal attack detection model to further enhance intrusion detection capabilities. By dynamically adjusting the node weights in the Weighted Random Forest algorithm, we expect to achieve more accurate and effective intrusion detection, and provide new ideas and technical means for solving internal threats. In addition, this study has laid the foundation for future work and opened up avenues for exploring more possibilities.

Author Contributions: Conceptualization, J.H. and Z.O.; methodology, J.H.; validation, J.H., Y.C. and Z.O.; formal analysis, Z.O.; investigation, Y.C.; resources, Y.C.; data curation, Y.C.; writing—original draft preparation, Y.C.; writing—review and editing, J.H.; supervision, X.W. and N.D.; project administration, J.H.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Foundation of National Natural Science Foundation of China Joint Fund "Research and Demonstration Application of Key Technologies for Trusted Security of Data Circulation and Trading" (U24A20241); National Natural Science Foundation of China, "Research on the Trusted Theory and Key Technologies of Data Security Transaction Based on Blockchain" (62202118); Major Scientific and Technological Special Project of Guizhou Province ([2024]014; Scientific and Technological Research Projects from Guizhou Education Department (Qian jiao ji [2023]003); Guizhou Provincial Department of Science and Technology 100-level Innovative Talent Project (Guizhou Science and Technology Platform Talents-GCC[2023]018); Guizhou Provincial Major Project "Research and Application of Key Technologies of Trusted Large Models for Public Big Data" (Qiankehe Major Special Project [2024]003) and Foundation of Chongqing Key Laboratory of Public Big Data Security Technology (CQKL-QJ202300001); The Foundation of National Natural Science Foundation of China (72261004).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yost, J.R. The march of ides: Early history of intrusion-detection expert systems. *IEEE Ann. Hist. Comput.* **2015**, *38*, 42–54.
2. Kemmerer, R.A.; Vigna, G. Intrusion detection: A brief history and overview. *Computer* **2002**, *35*, 27–30. [CrossRef]
3. Garcia-Teodoro, P.; Diaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* **2009**, *28*, 18–28. [CrossRef]
4. Alserhani, F. Hasbullah I. Intrusion Detection Systems Using Blockchain Technology: A Review, Issues and Challenges. *Appl. Artif. Intell.* **2024**, *38*, 2381882. [CrossRef]
5. Alserhani, F.; Adele, G.; Borah, A.; Paranjothi, A.; Khan, M.S.; Poulkov, V.K. A Comprehensive Systematic Review of Blockchain-based Intrusion Detection Systems. In Proceedings of the 2024 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 29–31 May 2024 ; IEEE: Piscataway, NJ, USA, 2024; pp. 605–611. *Appl. Artif. Intell.* **2024**, *38*, 2381882. [CrossRef]

6. Gurung, G.; Bendiab, G.; Shiaele, M.; Shiaeles, S. Cids: Collaborative intrusion detection system using blockchain technology. In Proceedings of the 2022 IEEE International Conference on Cyber Security and Resilience (CSR), Virtual, 27–29 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 125–130.
7. Li, W.; Tug, S.; Meng, W.; Wang, Y. Designing collaborative blockchained signature-based intrusion detection in IoT environments. *Future Gener. Comput. Syst.* **2019**, *96*, 481–489. [CrossRef]
8. Laufenberg, D.; Li, L.; Shahriar, H.; Han, M. An architecture for blockchain-enabled collaborative signature-based intrusion detection system. In Proceedings of the 20th Annual SIG Conference on Information Technology Education, Tacoma, WA, USA, 3–5 October 2019; p. 169.
9. Ujjan, R.M.A.; Pervez, Z.; Dahal, K. Snort based collaborative intrusion detection system using blockchain in SDN. In Proceedings of the 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Island of Ulkulhas, Maldives, 26–28 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–8.
10. Xuan, C.D.; Nguyen, T.T. A novel approach for APT attack detection based on an advanced computing. *Sci. Rep.* **2024**, *14*, 22223. [CrossRef]
11. Wu, Y.S.; Foo, B.; Mei, Y.; Bagchi, S. Collaborative intrusion detection system (CIDS): A framework for accurate and efficient IDS. In Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, NV, USA, 8–12 December 2003; Proceedings; IEEE: Piscataway, NJ, USA, 2003; pp. 234–244.
12. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
13. Ahmed, M.; Mahmood, A.N.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [CrossRef]
14. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 557–564.
15. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
16. Al-E'mari, S.; Anbar, M.; Sanjalawe, Y.; Manickam, S.; Hasbullah, I. Intrusion detection systems using blockchain technology: A review, issues and challenges. *Comput. Syst. Sci. Eng.* **2022**, *40*, 87–112. [CrossRef]
17. Alexopoulos, N.; Vasilomanolakis, E.; Ivánkó, N.R.; Mühlhäuser, M. Towards blockchain-based collaborative intrusion detection systems. In Proceedings of the Critical Information Infrastructures Security: 12th International Conference, CRITIS 2017, Lucca, Italy, 8–13 October 2017; Revised Selected Papers 12; Springer: Berlin/Heidelberg, Germany, 2018; pp. 107–118.
18. Meng, W.; Tischhauser, E.W.; Wang, Q.; Wang, Y.; Han, J. When intrusion detection meets blockchain technology: A review. *IEEE Access* **2018**, *6*, 10179–10188. [CrossRef]
19. Hızal, S.; Akhter, A.S.; Çavuşoğlu, Ü.; Akgün, D. Blockchain-based IoT security solutions for IDS research centers. *Internet Things* **2024**, *27*, 101307. [CrossRef]
20. Jiang, D.; Wang, Z.; Wang, Y.; Tan, L.; Wang, J.; Zhang, P. A Blockchain-Reinforced Federated Intrusion Detection Architecture for IIoT. *IEEE Internet Things J.* **2024**, *11*, 26793–26805. [CrossRef]
21. Hu, B.; Zhou, C.; Tian, Y.C.; Qin, Y.; Junping, X. A collaborative intrusion detection approach using blockchain for multimicrogrid systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1720–1730. [CrossRef]
22. Liang, H.; Wu, J.; Mumtaz, S.; Li, J.; Lin, X.; Wen, M. MBID: Micro-blockchain-based geographical dynamic intrusion detection for V2X. *IEEE Commun. Mag.* **2019**, *57*, 77–83. [CrossRef]
23. Mirzaee, P.H.; Shojafar, M.; Bagheri, H.; Chan, T.H.; Cruickshank, H.; Tafazolli, R. A two-layer collaborative vehicle-edge intrusion detection system for vehicular communications. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Virtual, 27 September–28 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
24. He, X.; Chen, Q.; Tang, L.; Wang, W.; Liu, T. Cgan-based collaborative intrusion detection for uav networks: A blockchain-empowered distributed federated learning approach. *IEEE Internet Things J.* **2022**, *10*, 120–132. [CrossRef]
25. Alkhpor, H.K.; Alserhani, F.M. Collaborative Federated Learning-Based Model for Alert Correlation and Attack Scenario Recognition. *Electronics* **2023**, *12*, 4509. [CrossRef]
26. Zohourian, A.; Dadkhah, S.; Molyneaux, H.; Neto, E.C.P.; Ghorbani, A.A. IoT-PRIDS: Leveraging packet representations for intrusion detection in IoT networks. *Comput. Secur.* **2024**, *146*, 104034. [CrossRef]
27. Kalaria, R.; Kayes, A.; Rahayu, W.; Pardede, E.; Salehi, S.A. IoTPredictor: A security framework for predicting IoT device behaviours and detecting malicious devices against cyber attacks. *Comput. Secur.* **2024**, *146*, 104037. [CrossRef]
28. Malathi, S.; Begum, S.R. Enhancing trustworthiness among iot network nodes with ensemble deep learning-based cyber attack detection. *Expert Syst. Appl.* **2024**, *255*, 124528. [CrossRef]
29. Salim, M.M.; Camacho, D.; Park, J.H. Digital Twin and federated learning enabled cyberthreat detection system for IoT networks. *Future Gener. Comput. Syst.* **2024**, *161*, 701–713. [CrossRef]

30. da Silva Ruffo, V.G.; Lent, D.M.B.; Komarchesqui, M.; Schiavon, V.F.; de Assis, M.V.O.; Carvalho, L.F.; Proença, M.L., Jr. Anomaly and intrusion detection using deep learning for software-defined networks: A survey. *Expert Syst. Appl.* **2024**, *256*, 124982. [CrossRef]
31. Alserhani, F. Analysis of Encrypted Network Traffic for Enhancing Cyber-security in Dynamic Environments. *Appl. Artif. Intell.* **2024**, *38*, 2381882. [CrossRef]
32. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
33. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
34. Ren, S.; Cao, X.; Wei, Y.; Sun, J. Global refinement of random forest. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 723–730.
35. Rahman, M.; Kamal, N.; Abdullah, N.F. EDT-STACK: A stacking ensemble-based decision trees algorithm for tire tread depth condition classification. *Results Eng.* **2024**, *22*, 102218. [CrossRef]
36. Chen, M.; Liu, Z. Predicting performance of students by optimizing tree components of random forest using genetic algorithm. *Heliyon* **2024**, *10*, e32570. [CrossRef]
37. Norouzi, M.; Gürkaş-Aydın, Z.; Turna, Ö.C.; Yağci, M.Y.; Aydin, M.A.; Souri, A. A Hybrid Genetic Algorithm-Based Random Forest Model for Intrusion Detection Approach in Internet of Medical Things. *Appl. Sci.* **2023**, *13*, 11145. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.