Securing P2P resource sharing via blockchain and GNN-based trust

Bellaj Badr

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAS005

Thèse de doctorat

SECURING P2P RESOURCE SHARING VIA BLOCKCHAIN AND GNN-BASED TRUST

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Dénomination (EDIPP)
Spécialité de doctorat : Informatique

BADR BELLAJ

Composition du Jury :

Hanane EL BAKKALI Med V University. Morocco	Président/rapporteur
Mai-Trang Nguyen Sorbonne university. France	Rapporteur
Davide frey Inria Rennes. France	Rapporteur
Cigdem Sangul Brunel University. United Kingdom	Examinateur
Noel Crespi Telecom SudParis (SAMOVAR). France	Directeur de thèse
Abdellatif Mezrioui INPT(RAISS). Morocco	Directeur de thèse
Aafaf OUADDAH INPT (RAISS). Morocco	Co-directeur
Emmanuel BERTIN Orange Labs Cean. France	Co-directeur

SECURING P2P RESOURCE SHARING VIA BLOCKCHAIN AND GNN-BASED TRUST

BADR BELLAJ
PhD

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
INSTITUT POLYTECHNIQUE DE PARIS

2023

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Badr BELLAJ

Dedication

To my loving parents, supportive siblings, caring friends.
To my devoted wife, and wonderful kids.
Your unwavering encouragement and belief in me have been my guiding light throughout this journey.
Thank you all for being there every step of the way.

Acknowledgment

Throughout my PhD study period in the RAISS and SAMOVAR laboratories at INPT and Telecom SudParis, many people have kindly provided me with their help and unlimited support. It is a pleasure to convey my most profound gratitude to them all and I am particularly indebted to the following people. First and foremost, I want to express my deep and sincere gratitude to my supervisors, Prof. Noel Crespi and Prof. Mezrioui abdellatif for their continuous support and their constant guidance during my PhD study years. Your encouragements and personal guidance have provided a good basis for the present thesis. I was very fortunate to have you as supervisors.

Thank you for everything, it was a truly great experience working with you !

I wish to thank also the members of the dissertation jury :

- Hanane EL BAKKALI
- Mai-Trang Nguyen
- Davide Frey
- Cigdem Sangul

for accepting the invitation and for their valuable time and feedback. My special appreciation goes to the co-supervisors and to all my colleagues and friends inside INPT and Telecom SudParis for the excellent and truly enjoyable ambiance. I am grateful to all of you, for your encouragements and support.!

Last, I want to express my gratitude to my family for their unconditional support.

Abstract

The emergence of blockchain technology and cryptocurrencies has enabled the development of innovative peer-to-peer (P2P) models for resource allocation, sharing, and monetization. As these P2P models operate without inherent trust, the need for reliable trust and reputation mechanisms becomes crucial to minimize potential risks associated with engaging with malicious peers. Several trust management systems (TMS) have been proposed to establish trust in traditional P2P networks, aiming to facilitate the selection of dependable resources and deter peer misbehavior, with a significant focus on utilizing reputation as a guiding factor. Reputation-based trust systems (RTMS) play a fundamental role by leveraging community-based reputations to establish trust. They enable peers to assess the trustworthiness of others and evaluate the Quality of Service (QoS) based on shared reputations and past interactions. While these systems establish a peer-to-peer overlay trust network, the majority of these protocols are not tailored to suit blockchain-based networks, resulting in various shortcomings due to their outdated design.

This thesis presents our protocol BTrust, a novel decentralized and modular trust management system for large-scale P2P networks, leveraging blockchain technology and Graph Neural Network (GNN) for trust evaluation. BTrust introduces a multi-dimensional trust and reputation model to assess peer trustworthiness, dynamically deriving a single value from multiple parameters. The blockchain ensures reliable trust computation, dissemination, and storage without a central trust manager. An important breakthrough in our protocol is the resolution of the "cold start" or "initial trust score problem". To achieve this, the bootstrapping peer adopts random walks to select trustworthy peers among its neighbors, ensuring a decentralized approach without relying on any centralized entity or predefined peers. Unlike existing solutions, this method prevents overwhelming the most trusted peers in the network. Another challenge addressed in reputation systems is the reluctance of peers to provide negative feedback, often due to fear of retaliation or simply not providing feedback at all. To tackle these issues, we introduce an incentive mechanism that encourages truthful feedback and implement specialized mechanisms to penalize bad or lazy behavior. These innovations promote a more reliable and balanced trust evaluation process within the system. Furthermore, we propose a variant of BTrust called GBTrust, which improves

upon the original protocol by incorporating Graph Neural Networks (GNNs) and a novel attention-based mechanism specifically designed for trust management. This variant enhances the detection of dynamic malicious peers and strengthens the overall robustness and accuracy of trust evaluation. By leveraging GNNs, GBTrust effectively captures the complex relationships and dynamic behavior of peers in the network, enabling more accurate identification of malicious activities and better adaptability to changing trust dynamics. The attention-based mechanism further enhances the model's ability to prioritize and weigh different trust factors, leading to more reliable and precise trust assessments.

We demonstrate the efficiency of the proposed protocol in large-scale P2P networks using simulations of a P2P network and show that BTrust and its variant (GBTrust) are highly resilient to failures and robust against malicious nodes.

Key words: Trust management, Reputation system, P2P, RTMS, Blockchain, GNN, Resource sharing, Cold-start, Random walks.

Résumé

L'émergence de la technologie blockchain et des cryptomonnaies a ouvert la possibilité de créer de nouveaux modèles de gestion, de partage et de monétisation de ressources en pair-à-pair (P2P). Étant donné que ces modèles P2P sont sans confiance (trustless), des mécanismes de confiance et de réputation fiables et efficaces sont nécessaires pour minimiser le risque d'accès ou d'interaction avec des pairs malveillants. Plusieurs systèmes de gestion de confiance basés sur la réputation (RTMS) ont été proposés pour garantir la confiance dans les réseaux P2P, aider à choisir des ressources fiables et empêcher les comportements malveillants des pairs. Ces RTMS établissent la confiance en s'appuyant sur des réputations basées sur la communauté. Ils aident les pairs à évaluer la fiabilité des autres et à évaluer la qualité de service (QoS) en fonction de leur réputation et de leurs expériences passées mutuelles. Dans ces schémas, un réseau de confiance en surcouche pair-à-pair est établi.

Cette thèse présente BTrust, un nouveau système de gestion de confiance décentralisé et modulaire pour les réseaux P2P à grande échelle, exploitant la technologie blockchain et les GNN (Graph Neural Network) pour l'évaluation de la confiance. BTrust introduit un modèle de confiance et de réputation multidimensionnel pour évaluer la fiabilité des pairs, dérivant dynamiquement une valeur unique à partir de plusieurs paramètres. La blockchain garantit un calcul, une diffusion et un stockage fiables de la confiance sans gestionnaire de confiance centralisé, tandis que les GNN capturent efficacement les relations complexes entre les pairs, conduisant à des évaluations de confiance précises et robustes.

Une avancée importante dans notre protocole est la résolution du "problème de démarrage à froid" ou du "problème du score de confiance initial". Pour y parvenir, le pair d'amorçage adopte des marches aléatoires pour sélectionner des pairs fiables parmi ses voisins, garantissant une approche décentralisée sans dépendre d'une entité centralisée ou de pairs prédéfinis. Contrairement aux solutions existantes, cette méthode évite de submerger les pairs les plus dignes de confiance du réseau. Un autre défi abordé dans les systèmes de réputation est la réticence des pairs à fournir des rétroactions négatives, souvent par peur de représailles ou simplement en ne fournissant pas de rétroaction du tout. Pour résoudre ces problèmes, nous introduisons un mécanisme d'incitation qui encourage les rétroactions sincères et nous mettons en œuvre des mécanismes spécialisés pour sanctionner les comportements mauvais ou paresseux. Ces innovations favorisent un processus d'évaluation de confiance plus fiable et équilibré au sein du système. De plus, nous proposons une variante de BTrust appelée GBTrust, qui améliore le protocole original en incorporant des Graph Neural Networks (GNN) et un nouveau mécanisme basé sur l'attention spécifiquement conçu pour la

gestion de la confiance. Cette variante permet d'améliorer la détection des pairs malveillants dynamiques et renforce la robustesse et la précision globale de l'évaluation de la confiance. En utilisant les GNN, GBTrust capture efficacement les relations complexes et les comportements dynamiques des pairs dans le réseau, permettant ainsi une identification plus précise des activités malveillantes et une meilleure adaptabilité aux dynamiques de confiance changeantes. Le mécanisme basé sur l'attention renforce également la capacité du modèle à prioriser et à pondérer différents facteurs de confiance, conduisant à des évaluations de confiance plus fiables et précises.

Nous démontrons l'efficacité du système GBTrust proposé dans des réseaux P2P à grande échelle en utilisant des simulations d'un réseau P2P, et nous montrons que BTrust est hautement résilient aux pannes et robuste contre les nœuds malveillants.

Mots-clés: Gestion de confiance, Système de réputation, Blockchain, GNN, Partage de ressources, Démarrage à froid, Marches aléatoires.

Contents

List of Figures

List of Tables

List of Abbreviations

RTMS Reputation Trust Management Systems

GNN Graph Neural Networks

PoS Proof of Stake

PoW Proof of Work

DAG Directed Acyclic Graph

DLT Distributed Ledger Technology

P2P Peer-to-Peer

DoS Denial of Service

IoT Internet of Things

CNN Convolutional Neural Network

ML Machine Learning

RL Reinforcement Learning

API Application Programming Interface

CSV Comma-Separated Values

JSON JavaScript Object Notation

HTTPS Hypertext Transfer Protocol Secure

IPFS InterPlanetary File System

RPC Remote Procedure Call

SHA Secure Hash Algorithm

TPS Transactions Per Second

UCB Upper Confidence Bound

ANN Artificial Neural Network

BRTMS Blockchain-based Reputation Trust Management System

Chapter 1

Introduction

Contents

Peer-to-peer (P2P) systems have evolved from simple file sharing over the internet to advanced distributed resource sharing [1]. Blockchain technology enabled promoting trust, transparency, and fair compensation for resource providers through the use of cryptocurrencies.

The rapid expansion of cryptocurrencies has sparked a notable surge of interest in leveraging blockchain-based systems to monetize computing resources [2] [3]. Users enthusiastically embrace the idea of renting out their underutilized resources, including CPUs [4], surplus storage capacity [5], and even internet bandwidth [6], in exchange for crypto-tokens. Projects such as Storj[1], Golem[2], and Filecoin[3] capitalize on blockchain technology, enabling individuals to share their idle storage and computational power, thereby converting these dormant resources into valuable assets within a global sharing economy. By incentivizing resource providers with cryptocurrency tokens, these projects foster a mutually beneficial ecosystem. Furthermore, blockchain-based resource sharing initiatives like Akash Network[4], Substratum[5], and iExec[6] effectively address concerns related to centralization, censorship, and data privacy (1.1). They empower users to partake in sharing internet bandwidth and computing resources, offering an alternative to conventional cloud and hosting services.

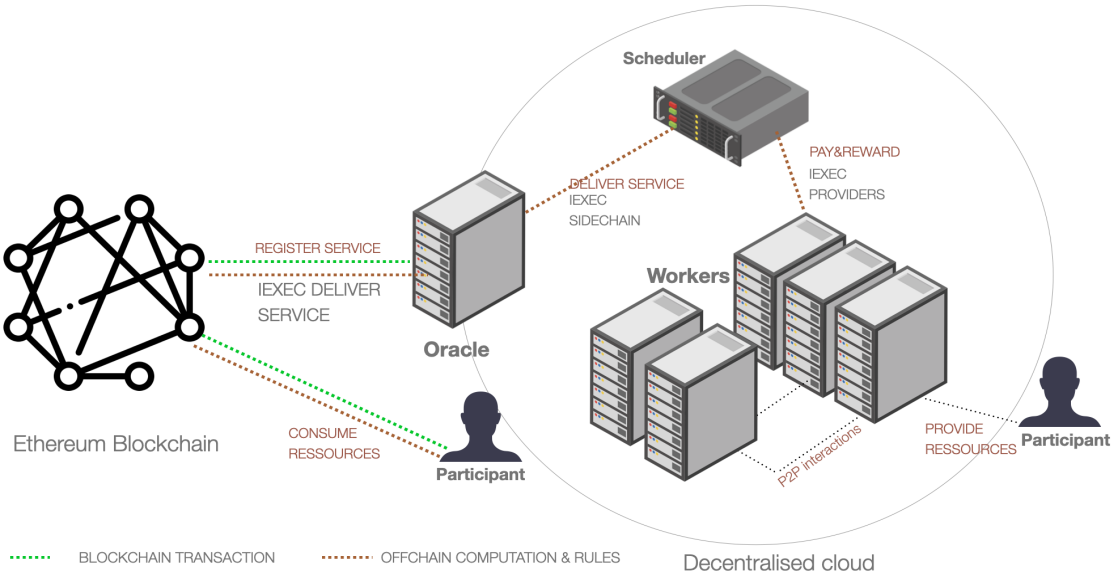


Figure 1.1: IExec resource sharing architecture

[1]https://www.storj.io/

[2]https://www.golem.network/

[3]https://filecoin.io/

[4]https://akash.network/

[5]https://substratum.net/index.html

[6]https://iex.ec/

In parallel to resource-sharing blockchain projects, projects like SETI@home[7] have engaged millions of volunteers worldwide to collectively analyze astronomical data, harnessing the combined processing power of participants' computers to search for signs of extraterrestrial intelligence. Similarly, distributed computing projects have emerged to address complex scientific problems, such as protein folding simulations or climate modeling, by aggregating computational resources from numerous volunteer contributors. These resource distribution systems has significantly reduced bandwidth costs for content providers and enhanced the efficiency of content delivery to end-users.

Nevertheless, these opportunities often involve engaging with unknown parties whose trustworthiness is uncertain, leading to substantial risks. Such risks may include the allocation of insecure and low-quality resources or the potential misuse of allocated resources for malicious purposes. Consequently, P2P resource sharing networks necessitate the implementation of suitable techniques to identify peers with valuable resources and the establishment of a robust trust model to ensure secure and efficient resource sharing.

Relying on established protocols, creating a dependable and secure peer-to-peer (P2P) resource sharing network is hindered by a multitude of challenging issues, including:

- Trust and Reputation Management: Assessing the trustworthiness of peers in a decentralized network, where interactions occur with unknown entities, poses a challenge. Developing effective trust and reputation models that can dynamically evaluate and quantify trust based on multiple parameters is essential to promote secure resource sharing.
- Corruption of Trust Managers: Trust managers play a pivotal role in the trust and reputation system of P2P networks. However, they can be susceptible to corruption or malicious manipulation. Ensuring the integrity and credibility of trust managers in submitting and processing feedback is essential to prevent the distortion of trust evaluations.
- Cold Start Problem: Bootstrapping a P2P network without relying on a centralized entity or predefined peers is another hurdle. Initiating the network and selecting trustworthy peers among neighbor peers, especially without access to prior reputation information, is a complex task and is frequently unsupported.
- Incentive Mechanisms: Encouraging truthful feedback and participation in the resource-sharing process necessitates the design of efficient incentive mechanisms. Addressing the reluctance of peers to provide negative feedback due to fear of retaliation or the inclination to avoid providing feedback altogether is crucial for maintaining a trustworthy network.

[7] https://setiathome.berkeley.edu/

- The Free-Rider Problem: Without proper incentivization mechanisms, some entities may engage in free-riding behavior, benefiting from shared resources without contributing back to the network adequately. Addressing this issue is crucial to maintain fairness and encourage active participation from all peers.
- Dynamic attacks: In the absence of a centralized authority, dynamic malicious nodes can lead to dishonest promotion or defamation of other actors in the network. Detecting and mitigating dynamic and Sybil attacks (using multiple identities) is essential to uphold the integrity and accuracy of the trust model.

To address these challenges, enhance trust within P2P networks, and mitigate peer misbehavior, researchers have proposed multiple RTMS [7][8]. These propositions aim to assess trust by relying on community-based reputations, enabling peers to assess the trustworthiness of others and evaluate the quality of service (QoS) based on their reputation and past interactions. In these schemes, a peer-to-peer overlay trust network is established, where each device (referred to interchangeably as a node or peer) acts as both a client and a service provider, necessitating evaluation of their QoS and trustworthiness. By leveraging reputation-based mechanisms, these systems facilitate the selection of reliable resources, promote honest behaviors among peers, and contribute to building a more trustworthy and efficient P2P network.

1.1 Problematic and objectives

In this thesis, we aim to address a fundamental challenge: **How can we ensure that users and peers in a blockchain resource-sharing network securely consume and provide resources?** This overarching question leads to several related inquiries: How can we efficiently manage trust computation, storage, and dissemination within blockchain-based resource-sharing environments? How can we incentivize users and device owners to participate? How can we secure new joining peers? How can we secure the network against both static and dynamic malicious participants? Additionally, how can we predict the trustworthiness of a given participant and forecast interactions among peers?

Our objective is to design a distributed protocol that leverages the unique capabilities of blockchain technology and Graph Neural Networks (GNNs) to enhance the trust management process. By integrating the properties of blockchain, such as decentralization, immutability, and transparency, we aim to ensure the reliability and integrity of trust computations and storage without relying on any central authority. Moreover, we specifically focus on the application of GNNs, which excel at capturing complex relationships and patterns in graph-structured data, to improve the accuracy and efficiency of trust evaluation. By harnessing the power of GNNs, we aim to extract meaningful insights from the network's topology, node attributes, and trust interactions, enabling more informed and precise trust assessments. Additionally, we try to address the challenge of efficient dissemination of trust information,

facilitating seamless and secure sharing of reputation data among peers while minimizing the overhead within the network.

1.2 Contributions

In this thesis, we present a novel protocol called BTrust which is designed as a generic solution applicable to various blockchain-based services. It introduces a trust overlay network that effectively addresses the challenges related to trust dissemination, incentivization, and the secure storage and retrieval of feedback in P2P networks. Notably, BTrust eliminates the need for trust managers or centralized operators, thereby enhancing the decentralized nature of the system. By leveraging the capabilities of the underlying blockchain to manage multiple services, such as identification, access control, and micropayments, it becomes possible to transparently assess, at a network level, the interactions of peers and mitigate risks related to centralized or on-edge trust computation and assessment. To the best of our knowledge, this approach is lacking in the literature, where the focus is more on using only smart contracts to outsource trust computation while keeping resource sharing services off-chain. This approach results in a resilient and adaptable reputation system capable of capturing new dimensions of trust, even enabling the detection of compromised devices that may otherwise appear to behave correctly. To amplify the accuracy of our trust management process, we've advanced the BTrust protocol, integrating Graph Neural Networks (GNN) to create the updated variant, GBTrust. This enhancement significantly boosts the system's ability to thoroughly analyze complex peer relationships within a P2P network, yielding more exact and robust trust evaluations.

Through multiple experimentation and analysis, we demonstrate that the proposed solution not only strengthens trustworthiness within these networks but also ensures data integrity, security, and reliable resource sharing among participants. By providing this innovative and generic BTrust solution, we contribute to the advancement of trust management mechanisms, fostering a trustworthy and efficient environment for decentralized networks.

This work extends the existing efforts on combining Blockchain and trust management systems [9]. The main contributions of BTrust (and GBTrust) could be listed below:

- Dynamic Trust Formula: BTrust's novelty lies in its adoption of a unique trust formula that combines recommendation and evidence-based approaches to accurately determine the trustworthiness level of peers within the network computed by distributed mechanism.
- Efficient Peer Selection through Random Walks: BTrust effectively tackles the "cold start problem" by employing random walks to identify trustworthy peers among neighboring peers. This approach ensures the selection of reliable peers without causing network overloads, enabling seamless integration of new nodes into the network.

- Usage of Remote Attestation: To safeguard the integrity of devices (both hardware and software), BTrust leverages Remote Attestation (RA). This mechanism allows authorized parties to detect any changes that may have occurred in remote devices, ensuring the security and authenticity of participating nodes.
- Incentive Mechanism for Truthful Feedback: BTrust proposes an incentive mechanism that encourages peers to provide honest and accurate feedback. By incentivizing truthful reporting, the system mitigates the risk of malicious or biased feedback, fostering a more reliable trust evaluation process.
- Improved RTMS precision and prediction: BTrust presents an upgraded iteration integrating Graph Neural Networks (GNN) alongside a novel attention mechanism. This proposition allows the network to dynamically adjust and allocate differing degrees of significance to diverse trust features, resulting in refined and more precise trust assessments.
- Resilience against dynamic malicious peers: BTrust and GBTrust present robust mechanisms to detect and mitigate the influence of such peers, ensuring the stability and security of the network.

1.3 Thesis Publications

1.3.1 Journal Papers

- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, **"BTrust: a new Blockchain-based Trust Management Protocol for Resource Sharing"**, in the Journal of Network and Systems Management, Springer https://dl.acm.org/doi/abs/10.1007/s10922-022-09674-4
- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, **"Drawing the boundaries between Blockchain and Blockchain-like systems: A Comprehensive Survey on distributed Ledger Technologies"**, for the journal Proceedings of the IEEE https://ieeexplore.ieee.org/document/10517413

1.3.2 Conference Papers

- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, "**SOK: A Comprehensive Survey on distributed Ledger Technologies**" in the 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Shanghai, China. https://hal.archives-ouvertes.fr/hal-03609651/
- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, "**DCEA: A Reference model for Distributed Ledger Technologies**," in the 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia. https://ieeexplore.ieee.org/document/9461129
- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, **"Untangling the overlap between Blockchain and DLTs."** in SAI Computing Conference 2022, London,UK. https://link.springer.com/chapter/10.1007/978-3-031-10467-1_30
- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, **"A Transpilation-Based Approach to Writing Secure Access Control Smart Contracts"** in the 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris,France, 2023.https://ieeexplore.ieee.org/abstract/document/10316873
- Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi and Abdellatif Mezrioui, **"GBTrust: Leveraging Edge Attention in Graph Neural Networks for Trust Management in P2P Networks"**, The 22nd IEEE International Conference on Trust, Security and Privacy in Computing and Communications), Exter,UK **(CORE-A conference)**.

1.4 Thesis Organisation

The thesis is organised as follows:

- Chapter 2 provides an extensive review of the background and relevant literature in the field of blockchain, and Reputation-Based Trust Management Systems (RTMS).
- Chapter 3 presents BTrust a blockchain based RTMS for securing resource sharing
- Chapter 4 presents GBTrust, a variant of BTrust based on GNN with a new attention mechanism.
- Chapter 5 concludes the thesis by summarizing the findings and insights obtained throughout the research and proposes potential research directions for future work.

Chapter 2

Background and Related Technologies

Contents

2.1 Introduction

This chapter provides an overview of the background and related technologies pertaining to trust management systems in P2P networks, with a focus on the integration of blockchain and GNN. First, the notion of trust and reputation is introduced, along with their relationship with RTMS, in section 2.2. Section 2.3 outlines the building blocks and properties of RTMS. Section 2.4 presents an overview of blockchain technology and its features that show promising potential for enhancing RTMS. Finally, section 2.6 delves into GNN, a powerful machine learning technique that operates on graph structures.

2.2 Trust and reputation-based trust

Trust can be defined as a subjective and intangible belief or confidence that an individual or entity possesses the capability and willingness to act in a reliable, honest, and responsible manner within a given context or domain. It is a dynamic and evolving notion, shaped by consecutive interactions and experiences between parties involved. Trust is context-related, meaning it may vary depending on the specific behavior, traits, or actions being evaluated. According to Gambetta's definition [10], trust is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.

Trust is crucial for establishing positive relationships, fostering cooperation, and mitigating risks in various social, economic, and technological interactions. In distributed systems and peer-to-peer networks, trust plays a pivotal role in assessing the credibility and trustworthiness of peers, enabling secure and efficient resource sharing, and facilitating decision-making processes. Trust is a fundamental aspect of establishing relationships and interactions among participants in distributed systems. It is often intertwined with the concept of reputation, which can be seen as the aggregated opinion or trust degree of an entity from other entities that have had prior interactions with the entity. While trust refers to a subjective belief towards an entity's behavior that evolves through repeated interactions, reputation provides a more objective and quantifiable representation of the entity's trustworthiness based on the experiences and recommendations of other peers.

2.2.1 Reputation and its significance in trust management

Reputation represents the collective perception and opinion of a community or group of individuals regarding the trustworthiness and reliability of an individual or an entity. In various real-world scenarios, individuals often rely on reputation to assess the level of trust they can place in an unfamiliar person or organization before engaging in any interaction. While reputation in the physical world may stem from word of mouth, media coverage, or tangible infrastructures, online communities present unique challenges in assessing reputation

due to their vast scale, prevalence of anonymous identities, and the ease of creating and maintaining digital personas.

To address the need for evaluating reputation in digital ecosystems, reputation systems play a vital role. The objective of an RTMS is to assess the accountability and trustworthiness of each participant using a quantitative approach. Trustworthiness is derived from direct experiences with the entity or recommendations from other peers and is represented as numerical scores, allowing for convenient measurement of the trust level. This trust and reputation score can serve as a safeguard to manage the associated risks when communicating and collaborating with other peers in dynamic and potentially hostile distributed systems. By leveraging trust metrics within RTMS, participants can make informed decisions, build reliable relationships, and foster a secure environment for resource sharing and collaboration.

Numerous applications of RTMS have been explored in the context of the Internet of Things (IoT). One such application is the TrustChain framework proposed in [11], which employs a three-layered trust management approach to tackle trustworthiness issues within supply chains. The TrustChain system evaluates the quality of commodities by analyzing multiple observations along the supply chain. In another study [12], researchers introduced a RTMS solution to enhance data validation in crowd sourcing scenarios. By incorporating trust management, this approach selects reliable validators based on the trustworthiness scores of participants in the crowd sourcing service, ensuring accurate and dependable data validation processes. These examples demonstrate the versatility and importance of RTMS in addressing trust-related challenges across various IoT applications.

2.2.2 Types of reputation systems: Network architecture

The network architecture of a reputation system plays a crucial role in shaping the fundamental activities that define its operation. Specifically, it governs the processes of feedback collection, feedback aggregation (reputation computation), and reputation dissemination. Two primary network architectures are commonly encountered in reputation systems: centralized and decentralized.

- Centralized Network Architecture: In a centralized reputation system, all critical functions such as feedback collection, aggregation, and reputation dissemination are managed by a central authority or a single entity. This central entity controls the entire process, making decisions and computations based on feedback received from participants. While centralized systems provide simplicity and ease of management, they also raise concerns regarding the trustworthiness and potential bias of the central authority due to its considerable influence over reputation assessments. A prominent example of this architecture is Google's PageRank [13], a link analysis algorithm used to determine the importance of web pages.
- Decentralized Network Architecture: In contrast to centralized systems, a decentralized

reputation system operates in a distributed manner, with no single entity having complete control. Feedback collection, aggregation, and reputation dissemination are spread among the nodes participating in the network. Each node independently contributes to the reputation computation process based on the feedback received from its interactions with other entities. Decentralized systems enhance transparency, trust, and resilience, as they eliminate a single point of failure and decisions are collectively made by the network participants. Notable examples of such systems include EigenTrust [14] and PowerTrust [8].

The choice of network architecture profoundly impacts the efficiency, security, and fairness of the reputation system. Depending on the specific requirements and characteristics of the ecosystem in which the reputation system operates, the selection between centralized and decentralized architectures becomes a critical consideration in ensuring the system's effectiveness and reliability.

2.2.3 Types of trust

In the context of trust management systems, trust can manifest in different forms and exhibit various characteristics. Understanding the different types of trust is essential for designing effective and context-specific trust models. In this section, we categorize trust into several types based on its nature and scope, shedding light on the nuances of trust in distributed systems.

- Reputation-based Trust: Reputation-based trust systems aggregate feedback and reputation scores from multiple interactions to determine an entity's overall reputation level. This approach utilizes the collective opinions and experiences of a community to assess the trustworthiness of an entity. Such systems are crucial in many distributed environments where they evaluate and compare entities based on their historical behavior and reliability. Prominent examples of reputation-based trust systems include SecuredTrust [15], EigenTrust [14], and PowerTrust [8].

- Direct Trust: Direct trust stems from direct interactions between individual entities within a network. As entities engage in activities such as sharing resources or exchanging information, they gain firsthand experiences of each other's behavior and reliability. This direct experience is integral to forming trust beliefs, enabling a personalized and specific assessment of an entity's trustworthiness. Notable implementations of direct trust models include TW-Trust [16], dTrust [17], and GenTrust [18].

- Indirect Trust: Indirect trust arises from the recommendations and opinions about an entity provided by other members within a network. Entities may not always have direct experiences with each other and often rely on feedback and reputation scores from their peers to evaluate the trustworthiness of someone unfamiliar. This mechanism is particularly valuable in scenarios where direct interactions are scarce or

when new entities join the network. Notable examples of indirect trust mechanisms include PeerTrust [19], as well as recent models discussed by Yuhong [20] and Xuemeng [21].

- Transitive Trust: Transitive trust involves the propagation of trust through a network of interconnected entities. In such scenarios, if entity A trusts entity B, and entity B trusts entity C, entity A may also extend some level of trust to entity C based on the existing trust with entity B. This chaining of trust relationships facilitates the assessment of trustworthiness for distant or unfamiliar entities by relying on intermediary trusted entities. Examples of this category include HonestPeer [22], AuthenticPeer++ [23], CuboidTrust [24], and T2D [25].

- Contextual Trust: Trust is inherently contextual, influenced by specific situations, tasks, or environments. Contextual trust acknowledges that an entity's trustworthiness may vary based on diverse factors such as the type of service provided, the sensitivity of the data shared, or the nature of the interaction. Trust models that adapt to various contexts can enhance the precision of trust assessments and support more effective decision-making. Notable studies in this area include works by Liu [26], Alhussain [27], and Altaf in multiple contexts [28], [29].

The table 2.1 highlights the characteristics, advantages, and limitations of each trust type, providing valuable insights into their suitability for specific use cases.

2.2.4 Trust aggregation

One of the key aspects in RTMS is the aggregation of trust evidence, which involves the accumulation and combination of trust ratings or feedback from various sources to calculate the final trust and reputation score of an entity. The aggregation process helps in synthesizing the dispersed trust data into a comprehensive measure of trustworthiness. Several models exist for aggregating feedback to derive reputation scores. Below, we summarize some of the common models. For a comprehensive survey of these methods, also known as reputation computation engines, Jøsang et al. [30] offer an excellent reference:

- Sum and Mean Model: A prevalent method in reputation scoring involves aggregating feedback through simple summation or by calculating the mean of feedback values, as detailed in Hasan's work [31]. The summation approach accumulates all feedback received, providing a comprehensive total score, whereas the mean approach offers a normalized representation by averaging the feedback. These methods are widely adopted in practical reputation systems, with notable examples including eBay.com, Epinions.com, and Amazon.com

- Markov Chain: The Markov Chain approach in reputation-based aggregation employs a probabilistic model to represent trust relationships among entities, as demonstrated in algorithms like EigenTrust [32] and PowerTrust [33]. This method utilizes transition

Table 2.1: Comparison of Types of Trust in Trust and Reputation Management

Type of Trust	Description	Pros	Cons
Direct Trust	Arises from direct interactions between individual entities within a network.	- Based on first-hand experiences.	- Limited to entities with whom direct interactions have occurred.
Indirect Trust	Emerges from recommendations and opinions provided by other entities within the network.	- Useful when direct interactions are scarce or unavailable.	- Relies on the accuracy and honesty of the sources providing the recommendations.
Transitive Trust	Involves trust propagation through a network of interconnected entities.	- Enables trust propagation beyond direct interactions.	- May be influenced by multiple intermediaries, potentially leading to inaccuracies.
Reputation-based	Derived from historical interactions and experiences with entities over time.	- Provides a comprehensive view of an entity's trustworthiness.	- May not account for changes in behavior over time.
Contextual Trust	Trust that is context-specific and can vary based on the circumstances of interactions.	- Offers adaptability to different scenarios.	- Requires accurate context identification, which can be challenging.

probabilities to propagate trust values throughout the network, aiming to converge on stable trust ratings. Markov Chain-based aggregation excels in its resilience against malicious behavior and its capability to manage inaccurate or outdated information effectively.

- Bayesian Aggregation: Bayesian methods, such as those detailed in Stereotrust [34] and explored further in the works of Hauke [35] and Burnett [36], utilize probabilistic models to iteratively update trust beliefs based on new evidence. Entities begin with initial prior trust beliefs and refine these beliefs as they receive fresh feedback. Bayesian aggregation is particularly advantageous in dynamic environments where the landscape of trust information is continually evolving.
- Flow Network Model: In the context of trust management, the flow network model (as explored in studies like Levien [37] and others [38], [39]) is represented by a weighted directed graph where each edge symbolizes the trust capacity between nodes. In this setup, nodes correspond to entities within a network that can extend and receive trust through their incoming and outgoing connections, ensuring that the trust extended equals the trust received. The source node, representing a highly trusted entity, has only outgoing connections with theoretically unlimited trust output. Conversely, the

sink node, which might represent a trust aggregator or evaluator, has only incoming connections and can accumulate an unlimited amount of trust inputs

- Fuzzy Logic : Unlike traditional binary logic, which only allows for precise and definitive values (true or false), Fuzzy Logic employs fuzzy membership functions to handle uncertain or imprecise evidence of trust. This approach enables the quantification of trust degrees by accommodating gradations of truth, as discussed in various studies [40], [41],[42]).

2.2.5 Trust dimensions

Trust is a complex and context-dependent concept, which means that trust cannot be universally applied across different contexts without proper adjustment and recalibration. Context-awareness is a crucial consideration when designing an RTMS. RTMS can be tailored to work with a single context or support multiple-context awareness in deriving trust from collected evidence, depending on their design.

Trust dimensions refer to the number of contexts that a RTMS can accommodate, and they can be broadly categorized into single-dimension and multi-dimension approaches:

- Single-Dimension RTMS: In single-dimension RTMS, trust is evaluated and computed based on a specific context or domain [43], [44]. The trustworthiness of an entity is assessed solely within that defined context, meaning that the trust score does not carry over to other contexts. These systems are particularly suitable when trust relationships and evidence are highly specialized and not applicable outside the designated context.
- Multi-Dimension RTMS: These systems consider various aspects and contexts in which trust is relevant, providing a more comprehensive evaluation of an entity's trustworthiness. By incorporating multiple dimensions, these RTMS [45] offer a more holistic view of trust, enabling broader application of trust assessments across diverse contexts. This approach allows entities to be evaluated more accurately, reflecting their performance and reliability in different areas.

The choice between a single-dimension or multi-dimension approach depends on the specific requirements and nature of the trust scenarios within a given system. While single-dimension RTMS provide focused and specific trust evaluations within well-defined contexts, multi-dimension RTMS can offer greater adaptability and coverage across a wide range of contexts.

2.3 Peer-to-Peer (P2P) networks

A Peer-to-Peer (P2P) [46] network is a decentralized and distributed network architecture in which individual nodes, called peers, collaborate and interact directly with each other to share resources, data, or services without the need for a centralized server or intermediary. In a P2P network, each node functions both as a client and a server, allowing direct communication

and data exchange between peers. Unlike traditional client-server models, where a central server facilitates communication and resource access for multiple clients, P2P networks rely on the collective computing power and storage capacity of all participating nodes. Each node in the network can act as both a provider and a consumer of resources, enabling a more efficient and scalable approach to resource sharing. P2P networks have been widely utilized for various applications [47], including file sharing [48], content distribution [49], real-time communication [50], collaborative computing [51], and decentralized systems [52]. They offer several advantages, such as increased resilience to failures, improved fault tolerance, better load distribution, and reduced dependency on single points of failure.

The absence of a central authority in P2P networks makes them more resistant to censorship and single points of control, making them suitable for applications that prioritize decentralization, privacy, and data autonomy. However, managing trust and security in P2P networks can be challenging due to the diverse nature of the participating peers and the need for efficient mechanisms to evaluate and maintain trust relationships among them.

2.4 Blockchain and DLTs

Blockchain technology was initially introduced by Satoshi Nakamoto in 2009 [53] as the underlying mechanism of Bitcoin, as a solution to the double-spending problem. Concretely, Blockchain can be defined as a replicated database (ledger) among the participants of a peer-to-peer network (Figure 2.1) and managed by a consensus mechanism. Transactions are packed into block units which are chronologically ordered and attached using cryptographic hashes to ensure data integrity.

One of blockchain's fundamental features is the distribution of the same ledger to all nodes participating in the network. This approach ensures transparency and high availability of data since each node possesses an identical copy of the ledger. Consequently, the need for a centralized entity to maintain and validate the ledger is eliminated, enhancing decentralization and trust within the system. To ensure the integrity of the ledger and prevent conflicting versions of data, a consensus mechanism is employed. This mechanism dictates that the majority of nodes must agree on the next block of data to be added to the chain. By achieving consensus, the network can maintain a single, agreed-upon version of the blockchain, eliminating potential discrepancies and ensuring the accuracy of transactions. Several consensus mechanisms exist in blockchain technology, each with its own unique characteristics and advantages. Some commonly used consensus mechanisms include Proof-of-Work (PoW) [54], where nodes compete to solve complex mathematical puzzles to validate transactions and add blocks to the chain [55]; Proof-of-Stake (PoS) [56], where validators are selected based on the number of tokens they hold and are responsible for validating transactions; and Proof-of-Authority (PoA) [57], where validators are pre-approved and known identities within the network responsible for validating transactions and adding new

blocks. Each of these consensus mechanisms brings different benefits and trade-offs in terms of security, energy efficiency, scalability, and decentralization [58]. The choice of consensus mechanism depends on the specific requirements and goals of the blockchain network being implemented.

Bitcoin's remarkable success as a global peer-to-peer network operating autonomously has paved the way for a revolutionary class of distributed systems known as blockchain. However, the growing popularity of this paradigm has led to a significant shift in the understanding of what constitutes a blockchain. The proliferation of systems marketed as "blockchain" that lack the core elements of the blockchain concept has led to a complex ecosystem within the industry. In response to this, a broader term, "Distributed Ledger Technology" (DLT), has emerged to encompass this category of projects that are heavily inspired by blockchain but do not strictly adhere to its principles. DLT serves as a more inclusive term when referring to these blockchain-like systems, acknowledging their similarities while acknowledging their deviations from the traditional blockchain model.

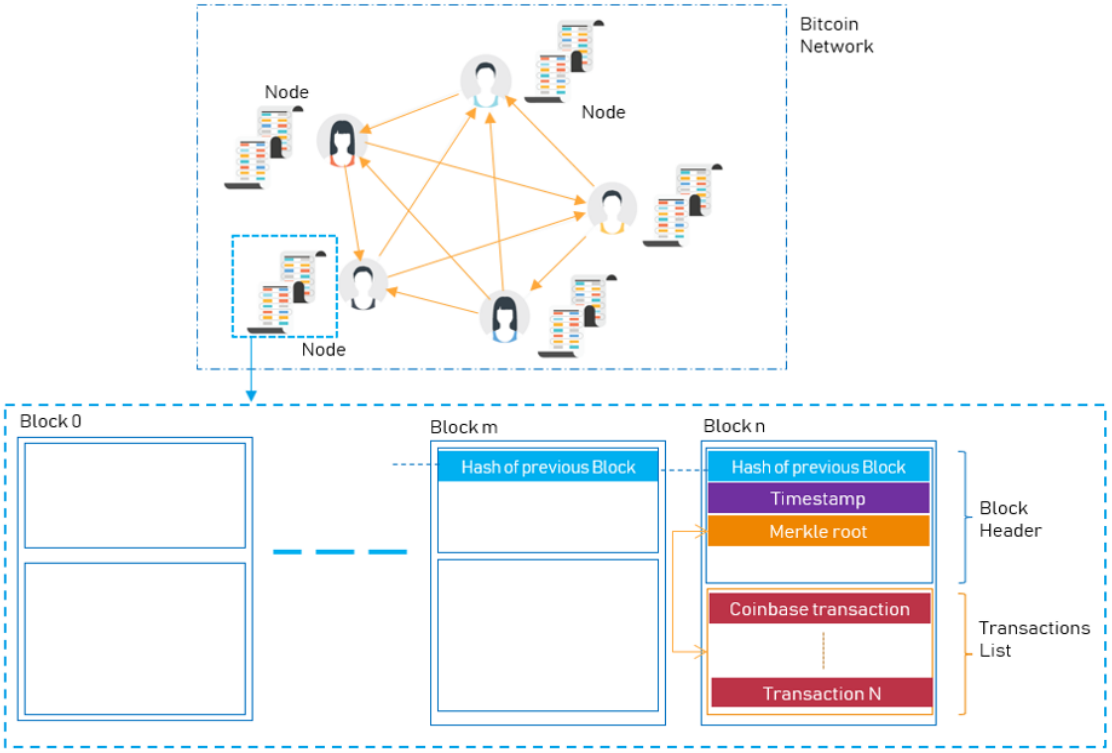


Figure 2.1: Basic blockchain network

Types of Blockchain

Blockchain systems can be classified into three main types based on several criteria and their usage in different application scenarios. These types are public, private, and consortium blockchains :

- Public Blockchains: Public blockchains [53] [59] [60] [61] offer an open and decentralized platform that allows users from various organizations and backgrounds to participate. Users can join the network, transact, mine, and perform read and write operations on the blockchain without any restrictions. In permissionless blockchains, there are no specific validators or pre-selected nodes; instead, all users can publish new blocks by solving computationally expensive puzzles or staking their own cryptocurrency. The blockchain's openness and transparency enable anyone to maintain a copy of the distributed ledger, making it secure and immutable. Additionally, public blockchains are tamper-resistant because each transaction incurs a processing fee, discouraging malicious attempts to alter the contents due to the high cost involved.
- Private Blockchains: Private blockchains [62] [63] [64] are designed to facilitate the private sharing and exchange of data among a defined group of known members. Unlike public blockchains, access to a private blockchain is restricted, and participation is limited to pre-approved entities. This controlled environment allows for more efficient and confidential transactions within the closed network. Private blockchains are often adopted by organizations seeking to maintain data privacy while still benefiting from the distributed ledger technology.
- Consortium Blockchains: Consortium blockchains [65] [66] represent a hybrid approach, combining elements of both public and private blockchains. In a consortium blockchain, no single organization controls the consensus process or block validation. Instead, a set of pre-selected nodes, typically representing different organizations, collaboratively handle these tasks. Consortium blockchains offer a more controlled and semi-private environment, providing benefits such as increased efficiency and reduced transaction costs compared to fully public blockchains. This type of blockchain is particularly useful when multiple organizations need to work together while maintaining a certain level of trust among the participants.

Smart contract

The Smart contract concept is one of the key innovations of Blockchain technology. The concept was proposed even before its emergence by N. Szabo who first coined the term smart contract and defined it as a "computerized transaction protocol that executes the terms of a contract. The general objectives of smart contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries" [67]. However, this objective was only truly concretized with the emergence of Blockchain technology. Concretely, a smart contract is a deterministic executable script –written in a high-level programming language– that codifies a given logic (e.g. a business logic) as a set of instructions for manipulating the states recorded in Blockchain. The script's clauses or functions are invoked by external transactions and executed by the validators in the network.

The execution result is then recorded in the blocks.

Blockchain's smart contract capabilities can facilitate the automation of trust-related processes, such as feedback collection, trust computation, and dissemination. Smart contracts are self-executing contracts with predefined rules and conditions, allowing trust-related interactions to be executed automatically once the specified criteria are met.

Consensus protocols

A consensus mechanism in the context of blockchain refers to the set of rules and protocols implemented within a decentralized network of nodes to achieve agreement on the validity and order of transactions and to maintain the integrity and consistency of the distributed ledger. It is a crucial component of blockchain technology that ensures consensus among network participants, who may be anonymous and untrusted, regarding the state of the blockchain. The consensus mechanism establishes a mechanism for verifying and validating transactions, resolving conflicts, and preventing malicious activities, allowing the network to achieve a shared, immutable, and tamper-proof record of transactions that all participants agree upon. Table 2.2 compare most known consensus mechanism used in blockchain networks.

Table 2.2: A comparison of consensus protocols

	Network Model	Adversarial model	Adversary mode	Fault tolerance	Identity Model	Safety	Liveness	Finality	transaction throughput	Type*
PBFT [68]	Asynchronous	Threshold Adversary	NA	f<n/3	Permissioned	Strong	Weak	Absolute	Very high	Blockchain-like
RAFT [69]	Asynchronous	Crash-failure	NA	f<n/2	Permissioned	Strong	Weak	Absolute	Very high	Blockchain-like
RIPPLE [70]	Asynchronous	Threshold Adversary	Adaptive	f<n/5	Permissionless	Strong	Weak	Absolute	High	Blockchain-like
STELLAR [71]	Asynchronous	Threshold Adversary	Adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very High	Blockchain-like
HONEYBADGER [72]	Asynchronous	Threshold Adversary	Non adaptive	f<n/3	Permissioned	Strong	Strong	Absolute	Very high	Blockchain-like
POW [53]	Partially-synchronous	Threshold Adversary	Strongly adaptive	f<n/2	Permissionless	Weak	Strong	Probabilistic	Low	Blockchain
BITCOIN-NG [73]	Partially-synchronous	Threshold Adversary	Strongly adaptive	f<n/2	Permissionless	Weak	Strong	Probabilistic	Very high	Blockchain
BYZCOIN [74]	Partially-synchronous	Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very high	Blockchain
GHOST [75]	Partially-synchronous	Threshold Adversary	Non adaptive	f<n/2	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
CASPER FFG [76]	Asynchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
CASPER TFG [76]	Asynchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	High	Blockchain
DPoS (EOS) [77]	Partially-synchronous	Stake Adversary	Non adaptive	f<n/3	Permissionless	Weak	Strong	Absolute	High	Blockchain-like
OUROBOROS [78]	synchronous	Stake Threshold Adversary	Middly adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
OUROBOROS [79] **PRAOS**	Partially-synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	weak	Strong	Probabilistic	High	Blockchain
OUROBOROS [80] **GENESIS**	Partially-synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
OUROBOROS [81] **CHRONOS**	Partially synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	weak	strong	Probabilistic	High	Blockchain
TENDERMINT [82]	Partially synchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissioned	Strong	Weak	Absolute	High	Blockchain-like
ALGORAND [83]	Partially synchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	High	Blockchain
THUNDERELLA [84]	Synchronous	Stake Threshold Adversary	Middly adaptive	f<n/3	Permissionless	Strong	Weak	Absolute (Fast Path)	Very high	Blockchain-like
HOTSTUFF [85]	Partially-synchronous	Threshold Adversary	Adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very high	Blockchain-like
LIBRABFT [86]	Partially synchronous	Threshold Adversary	Adaptive	f<n/3	Permissionless	Strong	Weak	Absolute	Very high	Blockchain-like
SPECTRE [87]	Partially synchronous	Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Probabilistic	High	Blockchain-like
IOTA [88]	Partially synchronous	Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Probabilistic	High	Blockchain-like
HASHGRAPH [89]	Asynchronous	Threshold Adversary	Non adaptive	f<n/3	Permissioned	Strong	Weak	Probabilistic	Very high	Blockchain-like
SNOW WHITE [90]	Asynchronous	Stake Threshold Adversary	Strongly adaptive	f<n/3	Permissionless	Weak	Strong	Probabilistic	High	Blockchain
AVALANCHE [91]	Partially synchronous	Stake Threshold Adversary	Non adaptive	f<n/3	Permissionless	Strong	Weak	Probabilistic	Very high	Blockchain

f : is the faulty nodes or actors and n: is the total number of nodes that are coming to consensus.

2.4.1 An overview and classification of Blockchains and DLTs

During our exploration of the blockchain domain, we introduced the DCEA model, which delineates DATA, Consensus, Execution, and Application, serving to differentiate blockchain systems from other frameworks. This model establishes a structured and layered heteroge-

neous stack catered for Distributed Ledger Technology (DLT), which encompasses blockchain and similar systems. From a design standpoint, our conceptual framework categorizes DLT technologies into four fundamental and discrete layers: data, distributed consensus protocols and network organization, execution, and application layers. Each layer plays a distinct and defined role within the architecture of DLT. The application of the DCEA framework leads to a two-dimension high-level taxonomy (blockchain and blockchain-like).

Presentation of DCEA framework

The framework consists of the DLTs components and their main properties (Table 2.3), with logically related functions grouped together. This layering approach is aligned with the DLT's modular architecture. That is, it will help to provide a better understanding of DLTs, and serves as a baseline to build a comparative analogy between different DLT variants.

Table 2.3: Layers and components of DCEA framework

Application Layer	Integrability			DLT orientation and purpose			Wallet and identity management		
Execution Layer	Execution environment		Turing-completeness		Determinism		Openness	Interoperability	
Consensus Layer	Safety	Liveness	Finality	Network model	Failure model	Adversary model	Governance model	Transaction ordering	Conflict resolution
Data Layer	Data structure		Data shareability		Data immutability		States storage		

In the following, we introduce the four layers that form the DLT stack.

- Data Layer: handles the data flowing through the distributed network and stored in the ledger. It includes entries recorded in the ledger, representing elements defined by protocols or data received from external sources. The data can be stored on the blockchain itself (on-chain storage) or in an auxiliary distributed database (off-chain storage).
- Consensus layer: Defines the global software-defined ruleset to ensure agreement among all participants, in a network, on a unified ledger. Consequently, this layer designates the formal rules that govern the system.
- Execution layer: Represents the components responsible for enforcing and executing distributed programs (e.g. smart contracts). Basically, these programs or contracts codify a given logic (e.g. a business logic) as a set of instructions for manipulating the states recorded in the ledger.
- Application layer: Represents an abstraction layer that specifies a variety of protocols and APIs provided by the DLT system to enable the building of distributed applications commonly called DApps. This layer also represents a communication link between the external actors or applications and the code hosted on the DLT ledger.

Based on the above layering, we propose a four-layered taxonomy (2.2), to categorize DLT systems. At each layer, DLTs adopt different settings for DCEA properties defined in Table 2.3. Based on their combinations, at the four layers, we can define different DLT classes.

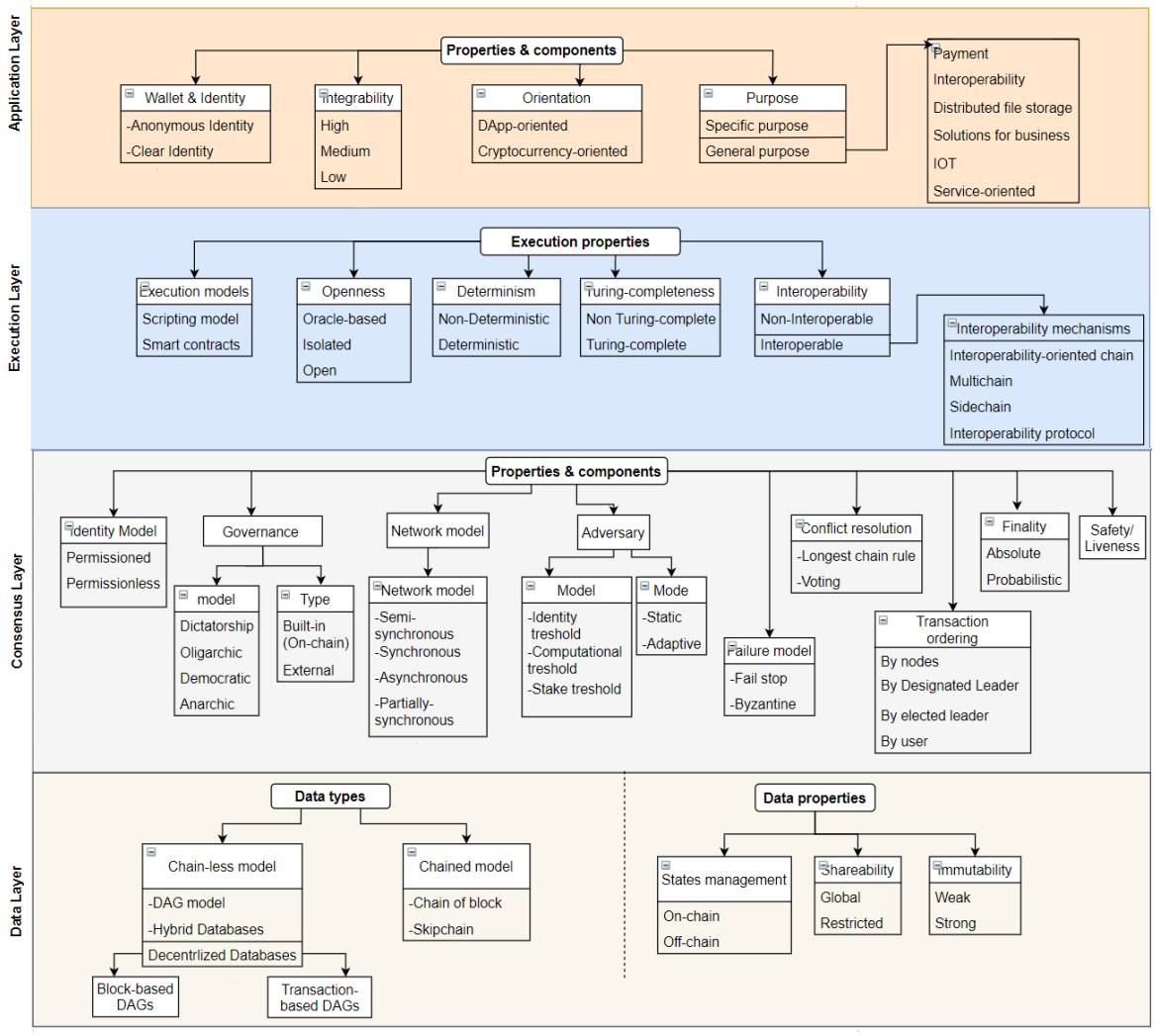


Figure 2.2: Layered taxonomy of DLT systems

2.4.2 The distinction between Blockchain and Blockchain-like systems

The distinction between Blockchain and Blockchain-like systems helps us to scope the targeted services because multiple Blockchain-like systems don't provide the required capabilities (decentralization, transparency, etc.) offered by BTrust. To make the distinction possible, we conducted a comprehensive survey [92] [93] [94] aims to provide an expanded and current review and evaluation of existing blockchains and their variants, while establishing clear boundaries between blockchain and blockchain-like systems. This has been achieved by deconstructing DLTs system using the DCEA framework and evaluating the differences between the two high-level taxons blockchain and the blockchain-like. After this deconstruction, we observe that they share many common characteristics, as well as distinguishing properties as

illustrated in Figure2.2.

Table 2.4: Distinctive settings of blockchain and blockchain-like in DCEA framework

	Components and properties	Blockchain	Blockchain-like
Data Layer	Data structure	Chain of block	Chainless model
	Shareability	Global	Restricted by design
	States management	On-chain	Off-chain
	Immutability	Strong	Weak
Consensus layer	Consensus identity model (membership)	Permissionless	Permissioned
	Governance	Democratic, Oligarchic	Dictatorship, Oligarchic
	Data ordering	Decentralized and open	Centralized or reserved
Execution layer	Conflict resolution	Longest-chain/ No-Forks	Longest-chain/No-Forks
	Turing completeness	Turing/Non-Turing complete	Turing and Non-Turing complete
	Openness	Closed/Oracle-based	Open/Oracle-based
	Interoperability	Non-interoperable/Interoperable	Non-interoperable/Interoperable
	Determinism	Deterministic	Non-deterministic
	Execution environment and rules enforcement	VM, Script runtimes	VM, Script runtimes
Application layer	Integrability	High, Medium, Low	High, Medium, Low
	DApp orientation	DApps, Cryptocurrency	DApps/ Cryptocurrency
	Wallet management	Built-in	Built-in or External

Table 2.5 presents a summary comparing a substantial representative sample of DLTs, totaling 44 projects, implemented within the industry or introduced in recent research literature. The comparison encompasses four key aspects: the composition within the DCEA framework, operational scope, level of decentralization, and the higher taxonomic classification to which each belongs. This comparison aids in understanding the diverse configurations of various blockchain technologies when formulating the trust protocol. Additionally, it assists in assessing the technical assumptions and the protocol's applicability scope.

Table 2.5: A summary table of comparison and analysis of the selected DLTs

	DATA LAYER				CONSENSUS LAYER					EXECUTION LAYER						APPLICATION LAYER		
DLT solution	Data Structure	Data shareability	States management	Immutability	Consensus	Governance	Governance nature	Transaction ordrer	Fork management (Non-malicious forks)	Turing completeness	Environment openness	Execution environment	Determinism	Languages	Interoperable	Orientation	Integrability	Purpose
Æternity	CoB	G	On	S	GHOST, Bitcoin-NG (for security) and PoS (for governance)	De	Bl	Randomly selected miner	Lc	Tc	Op (Built-in oracle)	Æternity VM	Dc	Sophia, Varna, solidity	No	Cy and DA	H	Ge
Algorand	CoB	G	On	S	Algorand	Ol	Ex	Randomly selected leader (stake weighted election)	Nf	NTc	Is (OB)	Algorand VM	Dc	TEAL	No	Cy and DA	M	Ge
Ardor	CoB (One parent chain with multiple child chains)	G	On	S	PoS	Ol	Bl	Forging account (NXT forging algorithm)	Lc	NTc	Op	Deterministic Java VM	Dc	Java	Yes (between child chains)	Cy and DA	M	Ge
Bigchaindb 2.0	HDS (a database and a CoB)	G	On/Off	W	Tendermint	Ol or De	Ex	Elected leader orders transactions by arrival time	Nf	Nsp					No	DA	M	Bo
Bitcoin	CoB	G	On	S	PoW	An	Bl and Ex	Randomly selected miner	Lc	NTc	Is	Script runtime	Dc	Bitcoin scripting, Miniscript	No	Cy	M	Po
BitShares	CoB	G	On	S	DPoS	De	Bl	Elected Witness	Lc	NTc	Is	Bitshares runtime	Dc	C++	No	Cy	L	So
Byzcoin[a]	Skipchain	G	On	S	Byzcoin	An	Ex	Randomly selected miner	Lc	NTc	Is	Byzcoin runtime	Dc	Go	No	Cy	L	Ge
Cardano	CoB	G&R	On	S	Ouroboros	De	Bl	Randomly selected leader	Lc	Tc	Is	IELE VM	Dc	IELE and Plutus	Yes (with cardano sidechain)	Cy and DA	M	Ge
Corda (R3)	DDB	G	Off	W	RAFT, BFT-SMaRt or KAFKA	Di, Ol or De	Ex	BFT-SMaRt distributed notary	Nf	Tc	Is (OB)	Java VM	NDc	Kotlin, Java	No	DA	H	Bo
Cosmos	CoB (Hub and multiple zones)	G&R	On	S	Tendermint	De (with Veto)	Bl	Elected block producer	Nf	Tc	Is	WebAssembly VM	Dc	Wasm languages(cosmos SDK)	Yes (Cross-chain Interoperability)	Cy and DA	M	Ge
Decred	CoB	G	On	S	PoW and PoS	De	Bl	Randomly selected miner	Lc	NTc	Is	Script runtime	Dc	Modified bitcoin scripting	No	Cy	L	Po
Elrond	SCoB (Metachain and shards)	G	On	S	Secure PoS (variant of Algorand)	De	Bl	Randomly selected proposer	Lc	Tc	Is (OB)	Elrond VM	Dc	IELE, Wasm languages	Yes (between Metachain and shards)	Cy and DA	L	Ge
EOS	CoB	G	Off	W	Transactions-as-PoS	Ol	Bl	Elected block producer	Lc	Tc	Is (OB)	WebAssembly VM	Dc	Wasm languages	Yes (with EOSYS sidechain)	Cy and DA	H	Ge
Ethereum 1.0	CoB	G	On	S	PoW (ETHASH)	An	Ex	Randomly elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	Solidity, Vyper, LLL, Julia	No	Cy and DA	H	Ge
Ethereum Enterprise	CoB	G&R	On	W	PoA, RAFT or IBFT	Di, Ol or De	Ex	Elected leader	Lc	Tc	Is	Ethereum VM	Dc	EVM languages	No	DA	H	Bo
Exonum Enterprise	CoB	G & R	On	S	Exonum protocol	Di, Ol or De	Bl	Predefined leader	Nf	Tc	Op (Built-in oracle)	Java VM and Rust runtime	NDc	Java, Rust	Yes (one way with Bitcoin)	DA	M	Bo
Elastos	CoB (main chain and sidechains)	G	On	S	DPoS and POW (merged mining with Bitcoin)[b]	De	Bl	Randomly elected miner	Lc	Tc	Is	CAR runtime, EVM (Ethereum Sidechain) , NEOVM(NEO sidechain)	Dc	C++, Java, Swift, JavaScript, Golang, solidity	Yes (Sidechains can transact with each other)	Cy and DA	M	Ge
Filecoin	CoB	G	On	S	Proofs-of-Spacetime	Ol	Bl	Elected leader (using Expected Consensus (EC))	Lc/ Hc[c]	Tc	Is	FilecoinVM	Dc	Golang	No[d]	DA	H	DSo
Hashgraph	DAG (Transaction-based DAG)	G&R	On	W	Hashgraph	Ol	Bl	Fair ordering via Consensus Time Stamping	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with Hyperledger Fabric)	Cy and DA	M	Ge
Hyperledger fabric	HDS	G&R	Off	W	PBFT	Ol or De	Ex	Ordering service node	Nf	Tc	Op	Java VM and Nodejs runtime	Dc	Go, JavaScript	No	DA	H	Bo
IOTA	DAG (Transaction-based DAG)	G	On	W	IOTA	Ol	Ex	End-user and coordinator node	Hb	Nsp (Qubic a smart contract protocol is under development)[e]					Yes (with Hyperledger Fabric)	Cy	M	IoT
Lisk	CoB (Main and sidechains)	G	On	S	DPoS	De	Bl	Elected leader	Lc	Tc	Is	NodeJs runtime	Dc	JavaScript	No	Cy and DA	H	Ge
Multichain	CoB	G&R	On/Off	W	Mutlichain protocol (variant of PBF)	Ol	Ex	Predefined leader	Lc	Multichain do not support smart contracts					No	DA	L	Ge
NEO	CoB	G&R	On	S	DBFT (Delegated Byzantine Fault Tolerance)	Ol, De	Bl and Ex	Elected leader	Nf	Tc	Is (OB)	NeoVM	Dc	.NET and JVM languages (Java, Kotlin)	Yes (Between private blockchains connected to NEO)	Cy and DA	M	Ge
Omniledger[f]	SCoB	G	On/Off	S	ByzCoinX (Variant of Byzcoin)	Ns	Ns	Randomly elected leader	Nf	Not supported					Yes (with its shards)	-	-	Ge
Parity substrate	CoB	G&R	On	W	Pluggable consensus (Hybrid PBFT, Aurand, Rhododendron, Shaft, ouroboros, PoW)	Ol	Bl	Depends on the chosen consensus mechanism	Nf	Tc	Is	WebAssembly VM	Dc	WASM languages	Yes (with Polkadot)	DA	H	Bo
Polkadot (Relay chain)	CoB (relay chain)[g]	G	On	S	GRANDPA and BABE (PoS)	De	Bl	Randomly elected leader	Lc	Tc	Is	WebAssembly VM[h]	Dc	WASM languages	Yes (Cross-chains interoperability)	Cy and DA	H	Io
Quorum	CoB	G&R	On	W	IBFT or RAFT or Clique POA	Di, Ol or De	Ex	Elected leader	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	No	DA	H	Ge
Qutum	CoB	G	On	S	PoS	De	Bl	Elected leader	Lc	Tc	Is (OB)	Ethereum VM and X86 VM	Dc	EVM languages, C, C++, Rust, Python	Yes (Atomic swap with bitcoin)	Cy and DA	Hign	Ge
Ripple	DDB (chain of ledgers stored as key-value)	G	On	S	Ripple (FBA)	Ol	Ex	Validating nodes converge toward a canonical order	Nf	NTc	Is	Built-in specialized payment types	Dc	JavaScript	No	Cy	M	Po
Rootstock	CoB	G	On	S	POW (merged mining with Bitcoin)	Ol	Bl	Randomly selected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with bitcoin)	Cy and DA	L	Ge
Steem	CoB	G	On	S	DPoS	De	Bl	Elected leader	Lc	Not supported						Cy	L	So
Stellar	DDB (chain of ledgers stored as key-value)	G	On	S	Stellar (FBA)	Ol	Ex	Validating nodes (using transactions sequence number)	Nf	NTc	Is	Stellar runtime	Dc	Java, JavaScript, Go	Yes (Atomic swap with other blockchains)	Cy	Hign	Po
Sia	CoB	G	On	S	PoW	An	Ex	Randomly elected miner	Lc	Not supported					No	DA	M	DSo
Stratis	CoB (Main chain and sidechains)	G	On	S	PoA or PoS	Ol	Ex	Randomly elected miner	Lc	Tc	Is	.NER runtime	Dc	.Net languages (e.g C#)	Yes (with its sidechains and with bitcoin)	Cy and DA	M	Ge
Nano	DAG (block-lattice)	G	On	S	Open Representative Voting (ORV) (based on DPoS)	Ol	Bl	Users (Sender and recipient)	Nf[i]	Not supported					No	Cy	L	Ge
Tezos	CoB	G	On	S	DPoS (Liquid PoS) and Emmy	De	Bl	Elected miner	Lc	Tc	Is (OB)	Tezos interpreter	Dc	Michelson	No	Cy and DA	M	Ge
Wanchain	CoB	G&R	On	S	PoS	De	Bl	Elected miner	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (Cross-Chain Communication Protocol)	Cy and DA	L	Io
Waves	CoB	G	On	S	Waves-NG (PoS based on Bitcoin-NG)	An	Bl	Elected miner (PoS)	Lc	NTc	Is (OB)	Waves runtime	Dc	Rideon	Yes (Atomic swap with other blockchains)	Cy and DA	L	Ge
Zilliqa	SCoB	G	On	S	PBFT and POW (Ethash)	De	Bl	Elected leader	Nf	Tc	Is (OB)	Zilliqa VM	Dc	Scilla	No	Cy and DA	M	Ge
Libra (Facebook)	DDB	G	On	W	LibraBFT	Ol	Ex	Elected leader	Nf	Tc	Is	Libra VM	Dc	Move	No	Cy	M	Po
Artis	CoB	G	On	S	HoneyBadgerBFT	Ol	Bl	Correct nodes	Nf	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	Yes (with Ethereum)	DA	M	Ge
VeChain	CoB	G	On	S	PoA	Ol	Bl	Elected leader (deterministic pseudo-random process)	Lc	Tc	Is (OB)	Ethereum VM	Dc	EVM languages	No	Cy and DA	M	Ge
Red Belly	CoB	G	On	S	DBFT Democratic BFT	Ns	Ns	Elected proposers	Nf	NTc	Is	Script runtime	Dc	Bitcoin-like scripting	No	Cy	L	Ge

LEGEND :
CoB: Chain of blocks, SCoB: Sharded Chain of blocks, DDB: Distributed database, HDS: Hybrid data structure
G: Global, S: Strong, R: Restricted, W: Weak, Cy: Cryptocurrency, DA: DApps, Ol: Oligarchic, De: Democratic, An: Anarchic, Di: Dictatorship
Bl: Built-in, Ex: External, Ns: Not specified, Nsp: Not supported, Nf: No forks, Lc: Longest chain, Hb: Heaviest branch, Hc: Heaviest chain,
Op: Open, Is: Isolated, OB: Oracle-based, Dc: Deterministic, NDc: Non-Deterministic, H: High, M: Medium, L: Low, Ge: General.
Bo: Business-oriented, Po: Payment-oriented, So: Service-oriented, DSo: Decentralized storage-oriented, IOT: IOT-oriented, Io: Interoperability-oriented

[a] Based on the implementation available on https://github.com/dedis/cothority/tree/master/byzcoin
[b] Elastos sidechains can have any consensus mechanism
[c] Filecoin gives weight to blocks that offer more storage power
[d] Filecoin ensures interoperability between different implementations of Filecoin protocol.
[e] https://qubic.iota.org
[f] Based on the minimalistic implementation available on https://github.com/dedis/student_18_byzcoin
[g] Parachains can have their own data structure
[h] Parachains are individual chains with their own runtime logic.
[i] In Nano, each user maintains its own DAG and a balance-weighted voting system is used to handle conflicting transactions.

2.5 Integration of Blockchain in reputation-based trust management systems (RTMS)

Traditional RTMS often face challenges such as centralized control, a lack of transparency, and a vulnerability to manipulation. The integration of blockchain technology offers a promising solution to address these limitations and enhance the effectiveness of RTMS.

- Decentralization and Consensus: One of the key advantages of blockchain in RTMS is its decentralized nature. By utilizing a distributed network of nodes, blockchain eliminates the need for a central authority, making the trust management system more resilient and resistant to single points of failure. The consensus mechanism employed by blockchain ensures that trust-related data is agreed upon by the network participants, further enhancing trustworthiness.

- Immutability and Transparency: Blockchain's inherent immutability ensures that once data is recorded on the blockchain, it cannot be altered or tampered with. This property is particularly valuable in RTMS, as it prevents malicious actors from manipulating reputation scores or feedback. Additionally, the transparent nature of blockchain allows all network participants to access and verify the integrity of trust-related data, promoting accountability and trust among peers.

- Trustworthiness Verification: Blockchain enables the verification of trust-related information through cryptographic techniques. Reputation scores, feedback, and transaction histories can be securely stored on the blockchain, allowing peers to independently verify the trustworthiness of their counterparts. This verifiability strengthens the

credibility of the RTMS and promotes trust in the network.

- Smart Contracts and Incentives: Smart contracts, programmable code deployed on the blockchain, can be leveraged in RTMS to automate trust management processes. These contracts can define rules and conditions for reputation scoring, feedback validation, and dispute resolution. Additionally, blockchain-based RTMS can introduce incentive mechanisms, such as token rewards or reputation-based incentives, to encourage honest behavior and discourage malicious activities.

The integration of blockchain technology in RTMS presents a paradigm shift in trust management, providing enhanced security, transparency, and reliability in P2P networks. By leveraging the decentralized and immutable nature of blockchain, trust-related information can be securely stored and verified, promoting a more trustworthy and resilient network ecosystem.

2.6 Background on GNN

2.6.1 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) [95] are a class of deep learning models specifically designed to process and analyze data structured as graphs. A graph consists of nodes (also known as vertices) and edges (also known as links or connections) that represent the relationships between nodes. GNNs leverage the inherent structure of graphs to learn and infer information about the nodes and their relationships, making them a powerful tool for various tasks in different domains.

The concept of GNNs can be traced back to the late 2000s, but their popularity and advancements surged with the development of more sophisticated architectures and training techniques [96]. They were introduced as a response to the limitations of Convolutional Neural Networks (CNNs) [97], which struggled to achieve optimal results in scenarios involving arbitrary graph sizes and complex structures. The fundamental idea behind GNNs is to allow nodes to aggregate and exchange information with their neighboring nodes iteratively, propagating and refining this information across the entire graph. At the core of a GNN is the message-passing mechanism, where each node receives messages from its neighbors, aggregates them, and uses the aggregated information to update its own representation. This process is performed iteratively over multiple layers, enabling the network to capture complex and high-level patterns in the graph data. The architecture and processing procedures of a typical and basic Graph Neural Network (GNN) involve several steps (see Figure 2.3). Firstly, GNN selects neighbors with a certain strategy. Then, an aggregate function is applied to extract information around the central node. Finally, the aggregated information passes through a neural network to undergo a nonlinear transformation. The output represents the updated representation of the central node. Different GNN architectures have been proposed,[98]

which implement different flavors of message passing[99] [100],started by recursive [101] or convolutional constructive [98] approaches

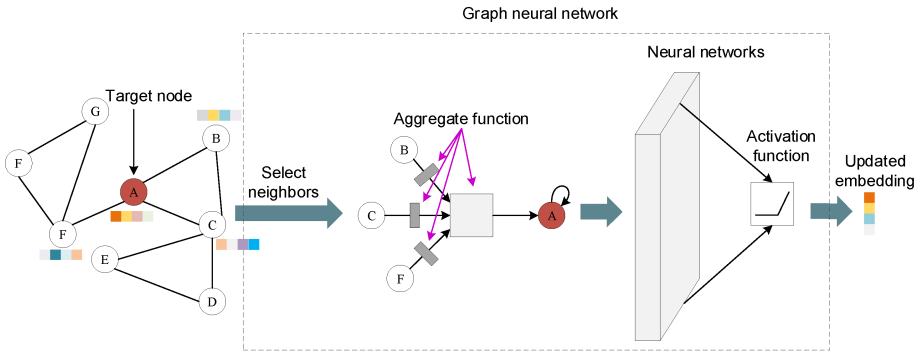


Figure 2.3: Illustration of the architecture of a Graph Neural Network (GNN).

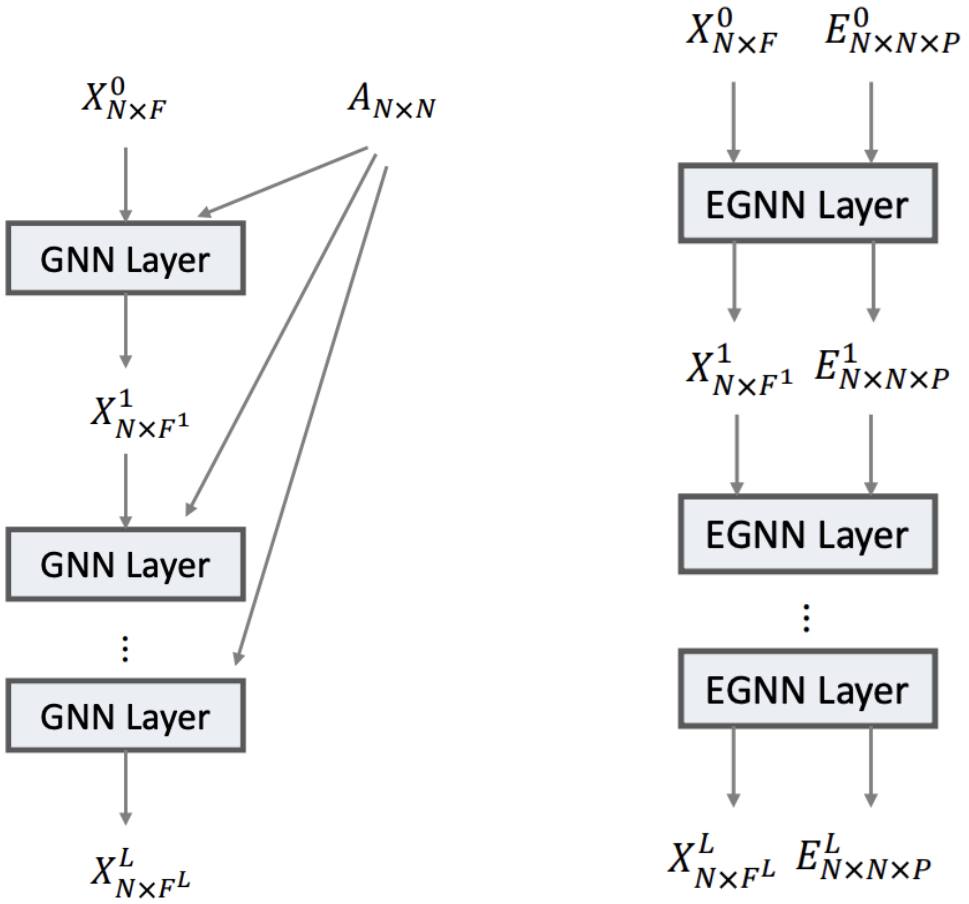
GNNs have proven to be highly effective in various applications, including social network analysis, recommendation systems, drug discovery, computer vision, and natural language processing. Their ability to handle non-Euclidean data and capture relational dependencies makes them well-suited for problems involving graphs and structured data.

However, GNNs also present challenges, such as scalability to large graphs, over-smoothing of node representations, and the need for careful handling of graph irregularities and noise. Researchers continue to explore and develop novel architectures, training techniques, and regularization methods to address these challenges and further improve the performance of GNNs.

2.6.2 Edge Graph Neural Networks (EGNNs):

EGNN [102], or Edge Graph Neural Network, is a novel framework for a family of graph neural network models that aims to better utilize edge features in graph learning tasks. Current state-of-the-art neural network models designed for graph learning, such as graph convolutional networks (GCN) [103] and graph attention networks (GAT) [104], often inadequately leverage edge features, especially multidimensional edge features. The EGNN framework addresses this limitation by introducing several key innovations.

Firstly, EGNN uses doubly stochastic normalization of graph edge features, deviating from the commonly used row or symmetric normalization approaches in current graph neural networks. This normalization technique allows for better utilization of edge features across network layers. Secondly, EGNN introduces new formulas for the operations in each individual layer to handle multidimensional edge features effectively. This enables the incorporation of rich and diverse edge information into the graph neural network models. Thirdly, the proposed framework ensures that edge features are adaptive across network layers, enhancing the ability of the models to learn from evolving edge information. Lastly, EGNN encodes edge directions using multidimensional edge features, enabling the modeling of directional



Original GNN architecture Proposed EGNN architecture

Figure 2.4: Schematic illustration of the proposed EGNN architecture (right), compared with the original GNN architecture (left). Figure from the original paper.

relationships within the graph.

EGNN is applied to various graph learning tasks, including graph node classification on citation networks, whole graph classification, and regression on molecular datasets. Experimental results demonstrate that EGNN outperforms current state-of-the-art methods, such as GCNs and GAT, emphasizing the significance of effectively utilizing edge features in graph neural networks.

2.7 Attention Mechanism in GNNs

In recent years, attention mechanisms [105] have emerged as powerful tools in various machine learning tasks, including natural language processing and computer vision. Attention allows the model to focus on specific parts of the input, giving more weight to relevant information. The idea of attention has also been successfully applied to GNN, enhancing their ability to

capture important graph structures and improve performance in various graph-related tasks.

The attention mechanism in GNNs enables the network to assign different weights to the neighboring nodes or edges during information aggregation. This allows the GNN to dynamically learn the importance of different parts of the graph while processing node representations. By leveraging attention, the GNN can effectively capture the most informative and relevant features, leading to more accurate and robust predictions.

In the context of GNNs, attention can be applied in different ways. One common approach is to use self-attention, where each node learns its attention weights based on its own features. This allows the node to selectively attend to its neighbors and adjust the importance of their contributions. The attention mechanism can be formulated as follows:

For node i in layer l:

$$\alpha_i^l = \text{softmax}\left(\text{LeakyReLU}\left(\mathbf{W}_1\mathbf{x}_i^l + \mathbf{W}_2 \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j^{l-1}\right)\right)$$

where $\mathbf{x}_i^l$ represents the hidden state of node i in layer l, $\mathcal{N}(i)$ denotes the set of neighboring nodes of node i, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weight matrices, and LeakyReLU is the activation function.

Another approach is graph attention, where attention weights are learned for both the nodes and their corresponding edges. The attention mechanism can be formulated as follows:

For edge (i, j) in layer l:

$$\alpha_{ij}^l = \text{softmax}\left(\text{LeakyReLU}\left(\mathbf{W}_1\mathbf{x}_i^l + \mathbf{W}_2\mathbf{x}_j^l + \mathbf{W}_3\mathbf{e}_{ij}^{l-1}\right)\right)$$

where $\mathbf{e}_{ij}^{l-1}$ represents the edge features between node i and j in layer $l-1$, and $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ are learnable weight matrices.

These attention weights can then be used to aggregate the information from neighboring nodes or edges, allowing the GNN to adaptively focus on important features and relationships within the graph. This mechanism facilitates more effective information aggregation and improves the GNN's ability to capture relevant graph structures, leading to improved performance in a wide range of graph-based tasks.

2.8 Integration of GNN in RTMS

GNNs have demonstrated remarkable capabilities in modeling graph-structured data and capturing intricate patterns and dependencies. By leveraging GNNs, reputation trust management systems can benefit from improved accuracy, robustness, and adaptability in

trust evaluation.

The integration of GNNs in reputation trust management systems brings several advantages:

- Graph Neural Networks (GNNs) can effectively capture both network structure and node attributes, enabling a comprehensive representation of trust relationships. This allows for a more accurate assessment of trustworthiness, considering both local and global contexts within the network [106].
- GNNs excel at handling the dynamic nature of networks. As trust relationships evolve over time, GNNs can adaptively update the trust assessments by leveraging temporal and sequential information. This dynamic modeling capability allows reputation trust management systems to reflect the evolving trust dynamics in real-time, enhancing the system's responsiveness and adaptability [107].
- GNNs enable the incorporation of multidimensional edge features into trust evaluations. By considering diverse edge attributes, such as trust ratings, transaction history, and social connections, GNN-based reputation trust management systems can provide a more nuanced and accurate assessment of trustworthiness. This approach facilitates a comprehensive understanding of trust relationships and mitigates the limitations of relying solely on node-level information [108].
- GNNs can improve the resilience of reputation trust management systems against malicious entities. By capturing patterns of malicious behaviors and detecting anomalies, GNNs enable more effective identification and isolation of untrustworthy entities. This enhancement contributes to the security and reliability of the system, making it more robust against various forms of attacks [109].

2.9 Conclusion

In conclusion, this chapter has provided the necessary background and highlighted the significant benefits of integrating blockchain and GNN in RTMS. The discussion emphasized how blockchain's decentralized and tamper-proof nature can facilitate the secure storage and computation of trust-related data. Furthermore, GNNs were shown to be effective in leveraging this data to learn and infer trust scores for network peers. The synergistic combination of these technologies has the potential to revolutionize trust management in P2P networks, offering improved scalability, transparency, and resistance against malicious activities. The insights gained from this chapter serve as a foundation for the subsequent chapters, which will delve into the design, implementation, and evaluation of a novel trust management system. This system will harness the power of blockchain and GNNs to enhance trustworthiness and security in P2P networks, contributing to the advancement of decentralized trust management solutions.

Chapter 3

Presentation of BTrust : A Blockchain-based RTMS

Contents

3.1 Introduction

In this chapter, we present the innovative trust protocol known as BTrust, which leverages blockchain technology to establish trustworthiness in large-scale P2P networks. We begin by providing an overview of the related work, including a comparison with existing blockchain based reputation trust management systems (BRTSM) and traditional approaches. Next, we delve into the details of the BTrust protocol, explaining its underlying architecture and algorithms. Moreover, we describe the optimized trustless bootstrapping process incorporated in the protocol, which enables the selection of trustworthy peers from neighboring peers. Additionally, we discuss the incentive mechanism designed to encourage truthful feedback. To evaluate the effectiveness of the BTrust protocol, we have conducted simulations, which demonstrate its resilience to failures and robustness against malicious nodes

3.2 Related work and comparative analysis

In this section, we explore the existing literature and underline the necessity of adopting BRTMS to enhance the security of P2P resource-sharing networks. Since in this thesis we present two versions of the RTMS protocol BTrust, one utilizing blockchain technology (BTrust) and its evolved form incorporating GNN (GBTrust), the related work is divided into two distinct sections, one focusing on the integration of blockchain in RTMS and the other on the integration of GNN alongside blockchain for RTMS. This later will be discussed in chapter 4, section 4.3.

3.2.1 BRTMS State of the art

Thanks to its promising features, Blockchain has been investigated intensively by the research community to build new trust management systems. Dennis and Owen [110] presented one of the early BRTMS. They proposed a reputation system to store reputation from completed transactions on a new Blockchain network in which transactions are validated by Bitcoin miners using merged mining. They propose a simple schema, where after receiving the correct file, the user sends an encrypted transaction consisting of the reputation score. This score is calculated using a single-dimensional reputation based on the non-satisfactory transactions in which the user received the file they requested. To reduce malicious transactions on the network, they propose a proof-of-stake system, where a user with a low, or no reputation, stakes a small amount of currency (Bitcoins) into a triple signed wallet. However, the use of Bitcoin as a validation network would cause important latency since Bitcoin takes up to 10 minutes to process each block [111] and there is no guarantee that reputation transactions will be mined sequentially in order because miners are free to choose which transaction to validate. Another issue is the ability of a single user to generate multiple identities and promote his reputation since they link the indemnity creation of an identity to the IP address of a user. Moreover, the approach adopted for selecting the peers from which the user will

download the file is insecure. The choice of the source depends on the friend's peer reputation, thus malicious nodes could focus their activity on proposing friendship to the newly joining peers and impacting the computation of other peers' reputations. Furthermore, operating this solution on a large scale is unpractical since each peer has to operate a resource-expensive full Bitcoin node. These properties make it unlikely that a network with a high amount of low resourced users, such as IoT devices, would implement this reputation system.

Another BRTMS targeting resource sharing in P2P networks is presented in [112]. The authors proposed a multi-level reputation scoring system for a Cluster Of Non-Dedicated Interoperating Kernels (Clondike). In a Clondike system, each participating user contributes computing performance of their machines and uses the computing performance of the other workstations for his computing. In such systems, there is a need for an RTMS to ensure fair usage of resources among all nodes of an inter-organisation cluster, as well as to identify and eliminate nodes that tend to overuse resources of the whole cluster and do not contribute by their computation resources or contribute by false results. To achieve these goals, they base the trust system on a relation between a node and the system instead of building an interpreted trust based on the feedback of other nodes. Instead of trusting reputation data that single nodes exchange, each node interprets behaviour data, which is stored in a blockchain, with its strategy. This approach presents some drawbacks. The reputation is built only on positive feedback (kudos), therefore peers cannot rate the bad behaviour of their counterparts and this bad behaviour is not logged into the blockchain. Moreover, the authors did not experiment with their BRTMS on a large-scale network to prove its scalability, since they experimented on a 3 node cluster both 2 fair nodes successfully penalized an abusive one.

In [113], a Proof-of-Trust (PoT) consensus protocol for enhancing data validation and accountability in crowdsourcing services is proposed. The authors introduced a hybrid blockchain architecture, based on a consortium blockchain acting as the underlying deployment network, while a public blockchain is used to ensure validation for the novel consensus protocol. The proposed PoT selects transaction validators to validate collected data based on their trust values while leveraging RAFT leader election [114] and Shamir's secret sharing algorithms [115]. The consortium is connected to the public blockchain through a set of gateways. Each consortium member has a consortium ledger management node and a gateway node. The gateway nodes are situated in a demilitarized zone (DMZ), providing isolation of the private consortium network from the open Internet environment. However, with the limited number of gateways, there is a risk of disconnection of the consortium network from the public blockchain, if the gateways are down or under DoS attack. Consequently, such disconnection will break the consensus within the network.

In [116], the authors proposed a BRTMS for the Autonomous System (ASes). The proposed BRTMS is devoted to evaluating network providers based on their adherence to Service Level Agreements (SLAs) regarding interconnection agreements. The method used to calculate

reputation is defined by a pre-agreed and publicly known scoring function and the results are written in a private Blockchain. They propose the use of SLA scores, which are quantified using a smart contract, for helping ASes choose their business partners. They introduce a fair scoring protocol that allows the scores to be deterministically computed from measurements of forwarding performance. The protocol requires each SLA score to be written on the blockchain and achieves privacy preservation by adopting an order-preserving encryption mechanism.

In [117] the authors argue that Distributed Ledger Technology (DLT) can be leveraged to create and manage the trust relationship between peers in a decentralized manner. They propose the LegIoT framework, which utilizes a DLT to store, manage and process trust information, enabling mutually distrusting parties to participate in a network.

In [118], authors leveraged the IoT with Ethereum's Blockchain to provide a reputation-based monetization system for IoT data, whose quality is ensured for consumers through reviews and ratings. They proposed a publish-subscribe model based on smart contracts, whereby a data owner shares information about the topics and subscribers make deposits, consume data and rate the service quality.

In [119], Bitcoin blockchain was proposed to be used as a public platform to manage the trust for decentralized sensor networks, as well as for logging nodes activities. These logs are then used as an indication of a node behaviour and thus a basis from which to compute the node trust score.

In [120], the authors introduced a distributed credit-based Blockchain system with a built-in reputation mechanism. They proposed a distributed ledger –obligation chain– for storing obligations of commitments. The service provider checks the obligation chain and the payment chain (Bitcoin blockchain) to assess the credibility of the obligation issuer by relying on the credit history of consumers and their ability to pay off their obligations.

3.3 Reputation and trust management in BTrust

In this section, we provide an overview of the various components that constitute the BTrust protocol within a Peer-to-Peer (P2P) network. We will discuss the smart contracts employed, the trust factors considered, the trust metrics utilized, the BTrust algorithms employed, and the selection of neighboring peers. These components play crucial roles in the functioning and effectiveness of the BTrust model, enabling it to accurately evaluate and establish trust relationships among peers in the P2P network. By understanding these components and their interactions, we gain insights into the inner workings of the BTrust system, its mechanisms for trust assessment, and its selection of neighboring peers.

3.3.1 BTrust components

In our model we envision a P2P network, which involves the following entities (Figure 3.6) playing various roles:

- **Network operator (NO)** : An NO is a community-based entity operating a service-oriented network. The NO is responsible for the initialization, provisioning and updating of BTrust agents (fig 3.1), and for establishing the first nodes in the network. The NO is also responsible for deploying and updating the different BTrust smart contracts. We envision NO as a decentralized entity operating similarly to the maintainers of Blockchain (BC) projects such as Ethereum or Bitcoin.
- **Certificate Authority (CA)** The CA is an entity providing valid identities and certificates for the members of the blockchain network. All the actions of the CA (Creation, Validation, Revocation, etc.) are recorded transparently in the blockchain.
- **Blockchain for reputation (BC)** The BC acts as a shared database for storing reputation and computing feedback data. We assume that the BC can store transparent and immutable data as well as execute smart contracts. In BTrust the underlying BC can be either public, private, permissioned or permissionless.
- **BTrust agent (TA)** Every peer-to-peer node hosts an agent that maintains the BTrust protocol rules and evaluates the security of the device through interactions with built-in security tools. This agent ensures the communication between the peers and between a peer and the different BTrust smart contracts (Reputation, Patch, Access control and Identification). The TA also manages peer security by regularly communicating with a patch distribution server to identify new security patch information and assesses the peer based on its adherence to patching compliance activities.
- **The device manufacturer (DM)** The DM is the entity that creates each device. The DM securely installs in each device the bootstrapping credentials (e.g. the endorsement key) needed for the Remote attestation.
- **Patch distribution server (PDR)** A PDR is an entity that informs and communicates to BTrust agents (Patch clients in this case) security patches available for each device using a patch distribution protocol. The PDR manages a patch DB which is updated by the network operator and by security vendors.
- **Remote attestation server (RAS)**: Remote attestation [121] is a security mechanism that enables a remote entity (the verifier) to authenticate the integrity of a system or application running on another machine (the prover or attester). In simpler terms, it allows for remote verification of whether a device or software has been tampered with or is trustworthy. The RAS oversees the entire attestation process (3.2). It maintains all the necessary proofs for validating the integrity of devices (such as BIOS or bootloader integrity, along with other system measurements) and evaluates the trustworthiness

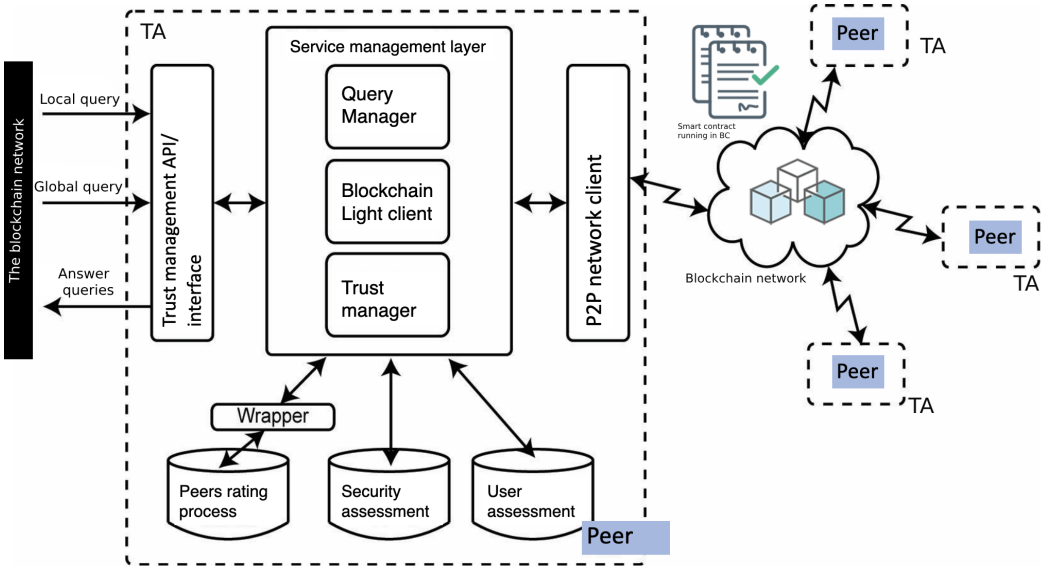


Figure 3.1: An overview of BTrust agent (TA) structure

of the remote attestation client. In BTrust, remote attestation isn't performed on demand; rather, the BTrust agent, acting as the prover, periodically measures and securely records its own hardware and software state, transmitting this information to the RAS.

- (or attester): The BTrust agent in our scenario, who serves as the prover, is responsible for gathering evidence about the device state, such as software versions and hardware configurations. It then generates a cryptographic proof using a challenge provided by the verifier.
- **Verifier (RAS)**: The Remote Attestation Service (RAS) serves as a server that facilitates the verification process. It is deployed and operated by the Network Operator (NO) and performs the following tasks:
  - Generating challenges for the prover.
  - Validating the proofs sent by the attester.
  - Communicating the results (trusted/untrusted) to third parties, such as other devices in the network, upon request.

- **Device owner (DO)** A DO is the person or entity that physically owns the device and is ultimately responsible for that device and how it is being used. The DO is responsible for onboarding his devices, as well as for transferring ownership of his device to another individual or entity. A device may have only one owner at any given time.

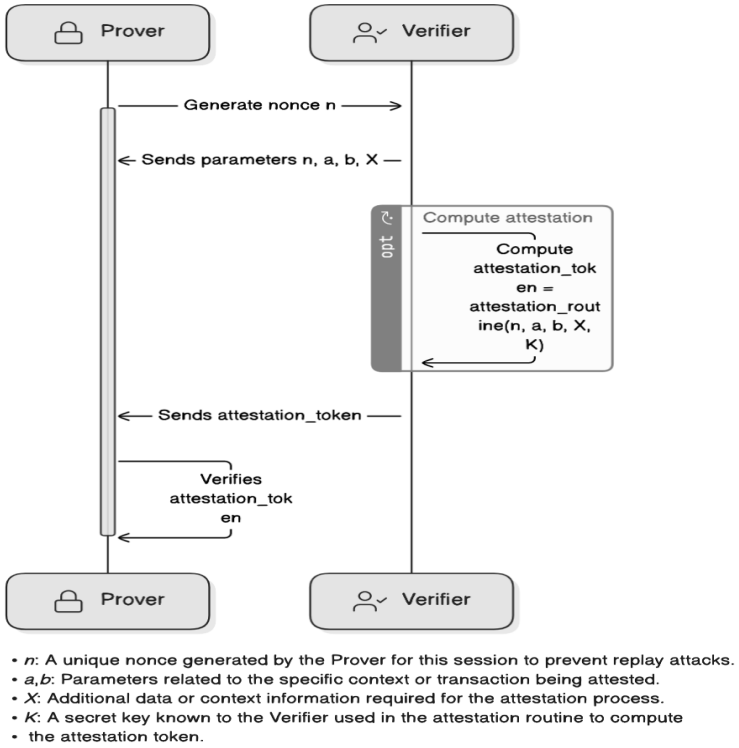


Figure 3.2: Remote attestation protocol

- **The device user (DU)** A DU is the individual or entity that uses a device. In BTrust's network, the DU (or the DO) is represented by his or her cryptographic credentials (an address and public key pairs).

3.3.2 BTrust smart contracts

To perform its different roles, BTrust relies on four different smarts contracts:

- **Reputation smart contract (RSC)** Builds the trust graph, capturing the relationships among peers, and logs the trust of each peer in the network. Besides, it manages the peers' feedback and the incentivisation process (3.3).
- **Identification smart contract (ISC)** Manages the identification and enrolment of the peers, device owners and users as well as the remote attestation. Device identification includes information about a device (the device profile) that helps the patch server send appropriate patches to the BTrust client when new patches are available in a patch DB (3.4).
- **Access control smart contract (ASC)** Determines the access rules defined by the DO. Each DO defines an access control policy for their devices, otherwise, a default policy is applied. In BTrust, we provide a transpiler that transpiles access control

```go
type Peer struct {
    ID             string  `json:"id"`
    GlobalTrust    float64 `json:"globalTrust"`
    Transactions   int     `json:"transactions"`
    UserBehavior   float64 `json:"userBehavior"`
    SecurityAssess float64 `json:"securityAssessment"`
}
// Define the structure for storing peer relationships
type TrustGraph struct {
    Peers map[string]Peer `json:"peers"` // Map of peer IDs to peer data
}

// Function to add or update peer feedback
func (rc *ReputationContract) AddOrUpdateFeedback(ctx contractapi.TransactionContextInterface, peerID string, targetID string, feedback int) error {
    // Retrieve the trust graph from the ledger
    trustGraphBytes, err := ctx.GetStub().GetState("trustGraph")
    if err != nil {
        return fmt.Errorf("failed to read trust graph from world state: %v", err)
    }
    // Initialize trust graph if not present
    if trustGraphBytes == nil {
        trustGraph := TrustGraph{
            Peers: make(map[string]Peer),
        }
        trustGraphBytes, err := json.Marshal(trustGraph)
        if err != nil {
            return fmt.Errorf("failed to marshal trust graph: %v", err)
        }
        err = ctx.GetStub().PutState("trustGraph", trustGraphBytes)
        if err != nil {
            return fmt.Errorf("failed to update trust graph: %v", err)
        }
    }
    var trustGraph TrustGraph
    err = json.Unmarshal(trustGraphBytes, &trustGraph)
    if err != nil {
        return fmt.Errorf("failed to unmarshal trust graph: %v", err)
    }
    // Check if peer exists in the trust graph
    peer, exists := trustGraph.Peers[peerID]
    if !exists {
        // Initialize peer if not present
        peer = Peer{
            ID:       peerID,
            Feedback: make(map[string]int),
        }
    }
    peer.Feedback[targetID] = feedback
    trustGraph.Peers[peerID] = peer
    trustGraphBytes, err = json.Marshal(trustGraph)
    if err != nil {
        return fmt.Errorf("failed to marshal trust graph: %v", err)
    }
    err = ctx.GetStub().PutState("trustGraph", trustGraphBytes)
    if err != nil {
        return fmt.Errorf("failed to update trust graph: %v", err)
    }
    return nil
}
```

Figure 3.3: A simplified example of RSC

```go
// Define the structure for storing peer identity data
type PeerIdentity struct {
    ID           string   `json:"id"`
    Role         string   `json:"role"` // Possible roles: peer, device owner, user
    PublicKeyHash string  `json:"publicKeyHash"`
    DeviceProfile []string `json:"deviceProfile"` // Device profile for device owners
}

// Define the structure for storing device profile
type DeviceProfile struct {
    DeviceID     string `json:"deviceId"`
    Manufacturer string `json:"manufacturer"`
    Model        string `json:"model"`
    OS           string `json:"os"`
}

// Function to enroll a new peer, device owner, or user
func (imc *IdentityManagementContract) Enroll(ctx contractapi.TransactionContextInterface, id string, role string, publicKeyHash string, deviceProfile
    // Check if identity already exists
    exists, err := imc.IdentityExists(ctx, id)
    if err != nil {
        return fmt.Errorf("failed to check if identity exists: %v", err)
    }
    if exists {
        return fmt.Errorf("identity already exists with ID %s", id)
    }
    identity := PeerIdentity{
        ID:            id,
        Role:          role,
        PublicKeyHash: publicKeyHash,
        DeviceProfile: deviceProfile,
    }
    identityBytes, err := json.Marshal(identity)
    if err != nil {
        return fmt.Errorf("failed to marshal identity data: %v", err)
    }
    ctx.GetStub().PutState(id, identityBytes)
    return nil
}

// Function to check if an identity exists
func (imc *IdentityManagementContract) IdentityExists(ctx contractapi.TransactionContextInterface, id string) (bool, error) {
    identityBytes, err := ctx.GetStub().GetState(id)
    if err != nil {
        return false, fmt.Errorf("failed to read identity data from ledger: %v", err)
    }
    return identityBytes != nil, nil
}
// Function to get identity details
func (imc *IdentityManagementContract) GetIdentity(ctx contractapi.TransactionContextInterface, id string) (*PeerIdentity, error) {
    identityBytes, err := ctx.GetStub().GetState(id)
    if err != nil {
        return nil, fmt.Errorf("failed to read identity data from ledger: %v", err)
    }
    if identityBytes == nil {
        return nil, fmt.Errorf("identity not found with ID %s", id)
    }
    var identity PeerIdentity
    err = json.Unmarshal(identityBytes, &identity)
```

Figure 3.4: A simplified example of ISC

policies written in Alfa or XACML language into smart contract code in Ethereum smart contracts or Hyperledger Fabric chaincode [122].

- **Patch and security smart contract (PSSC)** Provides the security patches and evaluates if the recommended patches have been applied; stores the evaluation of the security assessment of a device sent by the BTrust agent (3.5).

The ISC, RSC and PSSC smart contracts are developed, deployed and upgraded solely by the NO, whereas the ASC can be defined and deployed by the device owner or the NO. In order to utilise BTrust, we assume that the device should have (1) sufficient performance for the required public-key cryptographic operations, (2) a sufficient energy supply to perform the required operations, (3) enough non-volatile storage space to store the blockchain data and cryptographic keys, and (4) hardware features to support remote attestation. If the device lacks the last feature a software-based attestation [123] mechanism can be used.

```go
func (psc *PatchSecurityContract) assessDeviceSecurity(ctx contractapi.TransactionContextInterface, deviceID string,
    patchIDs []string) (int, error) {
    // Predefined security controls map
    predefinedSecurityControls := map[string]func(ctx contractapi.TransactionContextInterface, deviceID string) (bool, error){
        "FirewallConfiguration":  psc.checkFirewallConfiguration,
        "RegularUpdatesAndPatches": psc.checkRegularUpdatesAndPatches,
        "UserAuthentication":       psc.checkUserAuthentication,
        "SecureRemoteAccess":       psc.checkSecureRemoteAccess,
        "DataEncryption":           psc.checkDataEncryption,
        "BackupAndRecovery":        psc.checkBackupAndRecovery,
        "MonitoringAndLogging":     psc.checkMonitoringAndLogging,
        "VulnerabilityScanning":    psc.checkVulnerabilityScanning,
        "ApplicationWhitelisting":  psc.checkApplicationWhitelisting,
    }
    // Initialize overall security score
    overallSecurityScore := 0
    // Check patch application for each patch
    for _, patchID := range patchIDs {
        isApplied, err := psc.IsPatchApplied(ctx, patchID, deviceID)
        if err != nil {
            return 0, fmt.Errorf("failed to check if patch %s is applied: %v", patchID, err)
        }
        if !isApplied {
            return 0, fmt.Errorf("patch %s is not applied to device %s", patchID, deviceID)
        }
    }
    // Check predefined security controls
    for controlName, controlFunc := range predefinedSecurityControls {
        // Evaluate security control
        isFulfilled, err := controlFunc(ctx, deviceID)
        if err != nil {
            return 0, fmt.Errorf("failed to evaluate predefined security control %s: %v", controlName, err)
        }
        if !isFulfilled {
            return 0, fmt.Errorf("predefined security control %s is not fulfilled for device %s", controlName, deviceID)
        }

        // Add severity score of the control to overall security score
        overallSecurityScore += predefinedSecurityControlSeverities[controlName]
    }

    return overallSecurityScore, nil
}
```

Figure 3.5: A simplified example of PSSC

Before joining the network, on-device attestation keys (Endorsement Key) are injected into the device during the manufacturing process and signed by the NO or the CA. The device's Trusted Platform Module (TPM) signs the PCRs (Platform Configuration Registers), and registers for securely maintaining measurements inside the TPM with various attestation Identity keys(AIKs) that it generates. The BTrust agent extends the PCRs at runtime by writing a hash code into them. We consider that at any point in time, the number of active peers may be different, and not known in advance.

3.3.3 Trust factors

In BTrust, a peer's trustworthiness is defined by a combination of the evaluation of the peer it receives from other peers in the past, alongside behavioural monitoring and detection of abnormal activities. In developing BTrust, we consider four important factors for such evaluation:

- The feedback a peer obtains from other peers;

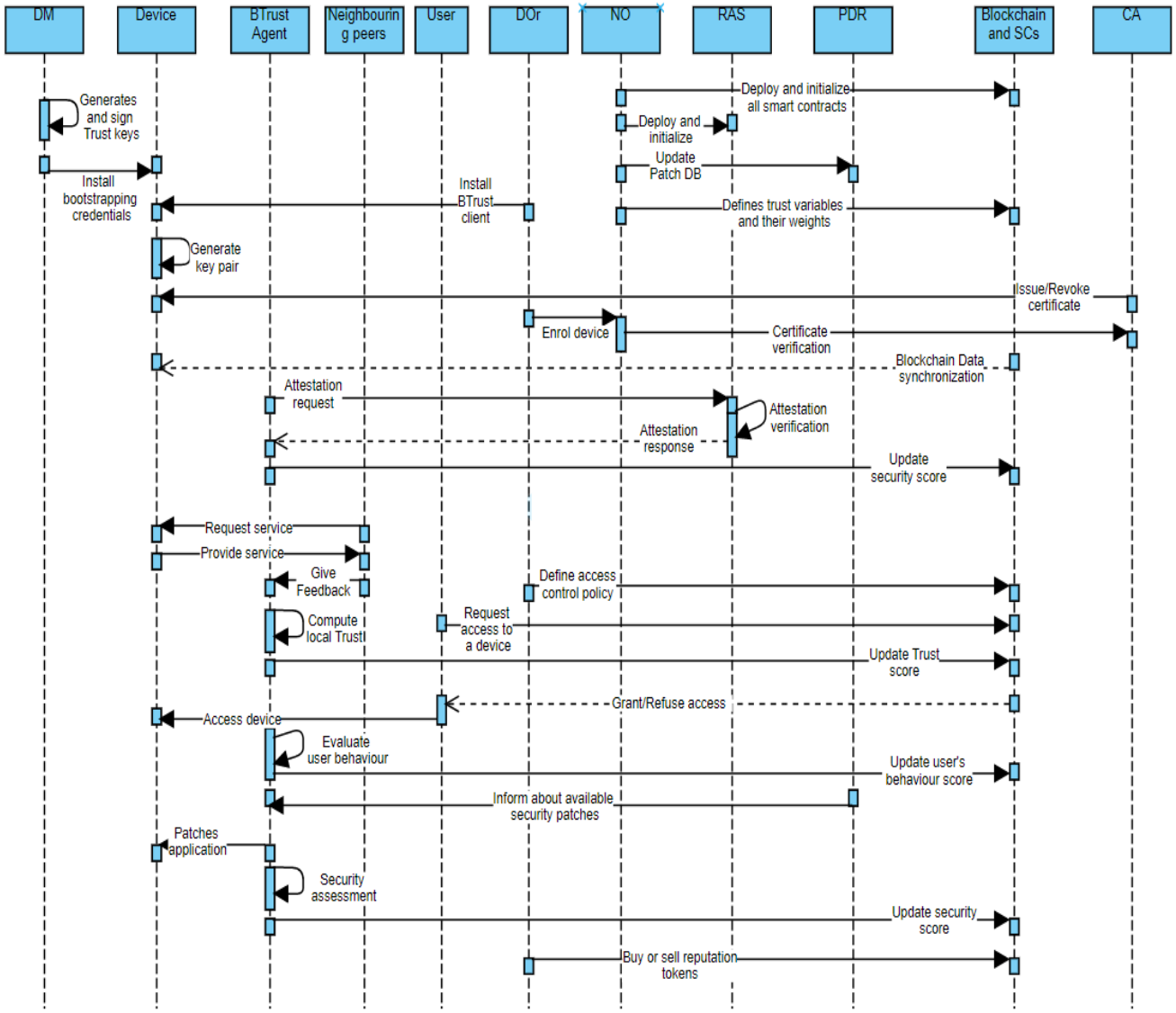


Figure 3.6: A simplified overview of interactions workflow between the entities in BTrust architecture

- The total number of transactions that a peer performs;
- The security assessment of a peer for discriminating vulnerable devices from less or secure ones; and
- The user's behaviour.

Peers feedback : In BTrust, trust is based on feedback gained directly and indirectly from other peers, and the total trust score is calculated based in part on the average of all recommendations, weighted by the trust degrees of the other peers.

Number of transactions : The transaction volume is an important factor that reflects the degree of satisfaction among different peers. We consider the ratio of the total amount of satisfactory transactions received over the total number of transactions received by a peer from another peer.

The user's behaviour : BTrust aggregates user data from a set of devices to build a reputation score for the users, creating the possibility of tracking a user's behaviour across many devices. At each device, the behaviour is evaluated based on the log of successful and failed events for the services managed by the blockchain using smart contracts. For example, BTrust monitors and evaluates the user behaviour based on three major services:

- Integrity and attestation services: Since devices and their system software can be replaced by malicious users or a device owner, BTrust relies on the Remote attestation (RA) technique, to detect compromised entities. Such that, each device's identity is primarily attested before the device can access a network and afterwards the device is periodically assessed and attested to ensure its integrity.
- Financial transactions: A user can be evaluated regarding their financial transaction records. For instance, a malicious user could try to perform a double-spending attack or launch a DDoS (Distributed Denial of service) attack on the network with malicious transactions.
- Access control service: The user's access activity whether for his own resources or those owned by others is evaluated. If the user access activity deviates from the rules defined by the access control smart contract, he is deemed suspicious and thus the peer reputation.

It is worth noting that other aspects can be considered to evaluate user behaviour such as Spamming activity or behavioural patterns. For simplicity, we choose to adopt only the three aforementioned services.

The security assessment of the device : We cannot evaluate the trust of a device without considering its security assessment. In this regard, BTrust attempts to determine the security status of a device either through behavioural monitoring, detection of abnormal activities or compliance to required security policies which are updated regularly by the NO (patching security issues, updating vulnerable systems and software, etc.). To conduct the security assessment, each peer is equipped with a BTrust agent (fig 3.1), which interacts with a locally hosted anomaly-based IDS (intrusion detection system) to detect anomalies, gather evidence and communicate this information to the RSC to adjust the peer's trust score. The agents also evaluate the compliance of each peer to the security rules and guidelines defined by the NO.

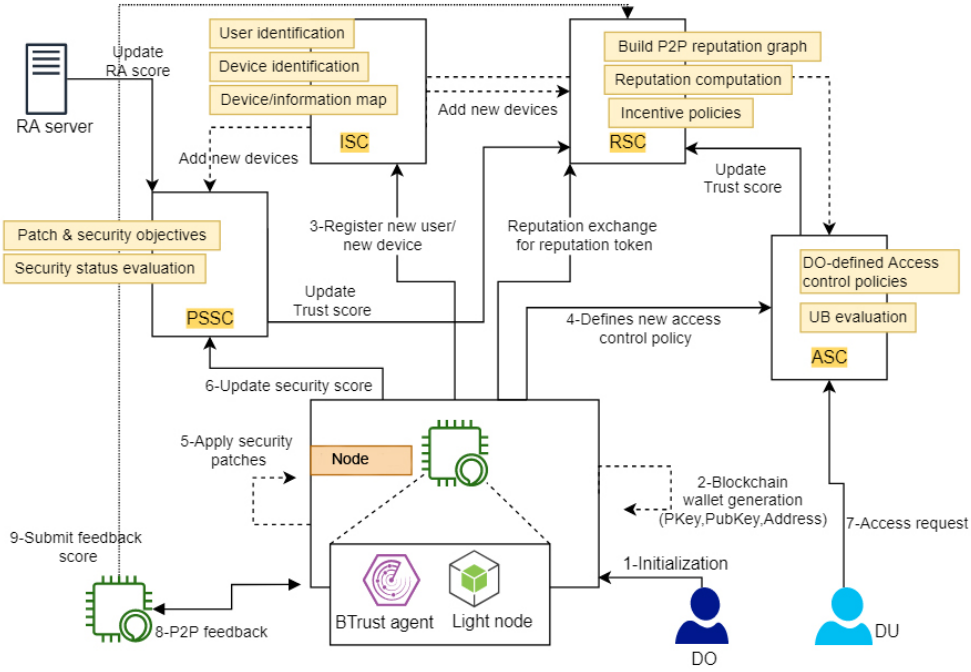


Figure 3.7: An overview of the interactions between devices, users and smart contracts in BTrust

3.3.4 General trust metrics

After discussing the importance of the trust parameters involved in BTrust, we formalize the BTrust parameters and present the formula we adopted to compute the trust for each parameter.

P2P networks are decentralized, BTrust builds a virtual trust overlay on top of these networks. Figure 3.9 shows a trust overlay network. BTrust network is modelled as a directed edge-weighted graph $G = (V, E)$, where each vertex describes a node in the network, and directed edges are the feedback between peers. We associate a weight to the edges for both directions to express the local trust between the source and the destination peers. To model the peer-to-peer interactions, we assume that peers exchange and rate exchanged transactions. For example, peers can exchange blockchain information (such as block headers) and rate the quality of received data.

In the graph G, the local trust value, a peer accords to others is indicated by the weight of the outgoing edge from the node, whereas the weight of the in-going edges to a peer represents the local trust received from other peers as depicted in figure 3.9. Thus, the directed edge P to Q reflects how much P trusts Q. Let $Tr(P)$ denote the trust score for peer P. The formula is defined in (3.1) and involves the following parameters:

- $GT(P)$ the global trust value of peer P;

- $T(P,Q)$ the total number of transactions sent by peer Q to P
- $UB(P)$ the user's behaviour score
- $ST(P)$ the security assessment score for peer P.

$$Tr(i) = \alpha * (GT(i) * UB(i)) + \beta * (ST(i)) \tag{3.1}$$

The coefficients α and β are normalized weight factors that serve to adjust the global trust value. The product between the global trust of a peer and the user behavior score enables a nuanced evaluation of trust, accounting for their interaction and relative magnitudes, thus adjusting the overall trust even in scenarios where one factor is low or zero. This approach anticipates that an insecure user within a trusted peer network holds a similar weight as an untrusted peer used by a trustworthy user, ensuring a comprehensive assessment of trustworthiness.

The adopted trust formula consists of two parts. The first part reflects the degree of trust that other peers have in the subject peer, based on their past experiences, and the second part assesses the security situation of a given peer, reflecting its likelihood of becoming a malicious peer. In the next three subsections, we present how the different parts of the formula are calculated.

P2P Reputation feedback

Similarly to most RTMS, we rely on positive and negative feedback to determine a local trust value that helps to compute the global trust of peers. That is, each peer first calculates the local trust values for other peers as described below:

Each time a peer P transacts with another peer Q, it may rate the response received from Q as positive ($tx(P,Q) = +1$) if P is satisfied with the transaction or as negative ($tx(P,Q) = -1$) if the transaction was not satisfactory.

Let $S(P,Q)$ denote the normalized sum of the feedback received by P from Q indicating the amount of satisfaction peer P has with Q. $S(P,Q)$ is a normalized value between 0 and 1. $S(P,Q)$ will be calculated as follow :

First, let $Sum(P,Q)$ denote the sum of the feedback given by Q to P.

$$\text{Sum}(P,Q) = \max\left(0, \sum^{T(P,Q)} tx(P,Q) * AR(P,Q)\right) \tag{3.2}$$

such that $AR(P,Q)$ denotes the ratio of the total number of good transactions received by P from Q. This ratio ensures that the sum accurately reflects the satisfaction level relative

to the total number of transactions exchanged between the peers, effectively adjusting for varying numbers of unsatisfactory transactions.

$$AR(P,Q) = Ng(P,Q)/T(P,Q) \tag{3.3}$$

The ratio ensures that the sum accounts for the disparity in satisfaction levels between these cases. If we take the case where P received 10 good transactions and 5 bad transactions, the $Sum(P,Q)$ will be counted as in the case where P received only 5 good transactions and 0 bad transactions. Then, we normalize the result to obtain a score between 0 and 1

$$S(P,Q) = Sum(P,Q)/\sum_{i \in N(P)} Sum(P,i) \tag{3.4}$$

The local trust value that peer P has about peer Q –denoted as $LT(P,Q)$– can be defined as following :

$$LT(P,Q) = S(P,Q) \tag{3.5}$$

where $Ng(P,Q)$ denotes the total number of good transactions performed by peer Q with P. Once the local trust with the neighbouring peers is calculated, we can compute the feedback-based global trust value of a peer P as shown in the following formula:

$$GT(P) = \sum_{i \in N(P)} LT(i,P) * (GT(i)/\sum_{j \in N(P)} GT(j)) * (1/1 + e^{-k * Tx_i}) \tag{3.6}$$

$N(P)$ is the set of neighbouring peers that interacted with P (peers that received transactions from P). $GT(P)$ is not updated unless card$(N(P))$ is greater than a specific threshold. In the previous formula, Tx_i denotes the total number of transactions provided by peer i to other peers, and k is a parameter that determines how steeply the feedback impact of peers rises with the number of served transactions. The logistic function $(1/1 + e^{-k * Tx_i})$ is leveraged in (3.6) as a factor to reduce the impact of malicious feedback on honest peers' global trust in the bootstrap phase.

User Behaviour score

In BTrust, we analyze how a user engages with various services in the network. This assessment is represented as $UB(P)$, which reflects user behavior and is defined as:

$$UB(P) = \sum_{i \in servs(P)} \alpha_i * Us(i) \tag{3.7}$$

$UB(P)$ quantifies the user's behavior by evaluating their interactions with different services denoted by $Us(i)$, weighted by α_i.

$Us(i)$ represents the score obtained from using service i from a set of defined services for the peer P (denoted as $servs(P)$). Different weights α_i can be assigned (after being normalized) by the network operator according to the importance of a given network service. For instance, an NO can define as basic services (as $servs(P)$) the following :

- Access Control (AC) that scores users based on how frequently they attempt to access restricted areas or perform unauthorized actions.
- Authentication sevice (AU) assesses the strength of user credentials and detects any suspicious login attempts.
- Resource Utilization (RU) monitors the amount of resources (like bandwidth or CPU) a user consumes and flags any abnormal usage patterns.

A user whose interactions with these services yield the following scores over a given time period (computed by BTrust agent):

- $Us(AC) = 0.8$ (indicating a high number of access attempts to restricted areas)
- $Us(AU) = 0.9$ (indicating successful logins with strong credentials)
- $Us(RU) = 0.5$ (indicating moderate resource usage)

To assess user's overall behavior, we apply the weights assigned by the network operator (α_i) to each service's score:

$\alpha_{AC} = 0.33$, $\alpha_{AU} = 0.22$, $\alpha_{RU} = 0.44$ and the final result will be UB(P)=0.776

Peers security assessment score

The security assessment score, denoted as $ST(P)$, aims to quantitatively reflect the current security situation of a peer P. We define a peer's security assessment score by aggregating the sum of the required objectives, weighted by the severity of each objective. Such objectives could encompass various security measures like applying security patches, system upgrades, and ensuring correct configurations. The $ST(P)$ is defined as follows:

$$ST(P) = \begin{cases} (\sum_{i \in rules(P)} (Fa(i) * Sv(i) * Da(i))) * RA(P) \\ 0, \text{ if } ST(P) < 0 \end{cases} \tag{3.8}$$

$rules(P)$ denotes the collection of security objectives that peer P is tasked with fulfilling. $Fa(i)$ represents the fulfillment factor, indicating whether a specific objective i was achieved (1) or not (0). $Sv(i)$ denotes the severity level of objective i, ranging from 1 to 10. $Da(i)$ indicates the patch application status for objective i, where 0 signifies that a patch needs

to be applied but wasn't for this particular objective. For instance, if a device correctly configures the firewall but fails to apply an announced firewall security patch, the score for this rule is 0. RA denotes the result of the remote attestation, such that $RA = 0$ if the operation fails, otherwise $RA = 1$. The security objectives are not predetermined; instead, they are left for the NO to define the most appropriate objectives based on their specific use case and the unique requirements of the system or environment. Therefore, applying patches relevant to security rules and adequately fulfilling security objectives increases the peer's security score.

When a peer first joins the network, it must first fulfil a minimal security policy (e.g 3.8) defined by the NO to acquire an initial trust score (IS). This includes configuring recommended security features, performing security-related tasks and addressing the recommended action with a third-party application or software, or an alternate mitigation. Then, the peer will only be able to interact with and rate other peers if its reputation score is greater than a threshold IS value defined by the NO.

Table 3.1: Example of security objectifs

Security Considerations	**Description**	**Severity Score**
Enable Firewall	Configure and activate the firewall to control incoming and outgoing network traffic.	High (0.9)
Regular Updates and Patches	Maintain regular system updates and security patches for addressing vulnerabilities.	Critical (0.95)
User Authentication	Implement secure user authentication, such as strong passwords or SSH keys.	High (0.9)
Secure Remote Access	Secure remote access by configuring SSH with key-based authentication and other measures.	High (0.9)
Data Encryption	Encrypt sensitive data or directories using tools like LUKS or other encryption methods.	Medium (0.7)
Backup and Recovery	Set up regular backups of critical data and configurations for recovery in case of issues.	Medium (0.7)
Monitoring and Logging	Enable system logs and monitoring to track and review system activity for potential issues.	High (0.9)
Vulnerability Scanning	Use vulnerability scanning tools to identify and address security weaknesses.	Critical (0.95)
Application Whitelisting	Allow only approved and necessary applications to run on the peer.	Medium (0.7)

3.4 BTrust Algorithms

Generally, BTrust proceeds in three phases: an initialization phase, the enrolment phase, and the processing phase. These phases are explained below.

The initialization phase : This is a one-time stage during protocol and network bootstrapping. In this phase, the Network Operator (NO) constructs an information database containing security details and available patches for a wide range of devices, encompassing various models and versions. It defines the minimal security policy in the security smart contract and the required threshold for a device to start rating other peers. At the same time, the NO defines the initial security actions to be fulfilled by the DO in order to achieve a trusted reputation and indicates their corresponding severities and scores. The NO then defines the exchangeability rules of the BTrust token-based incentive mechanism presented in section 3.5. The manufacturer defines the information about the devices as well as how to evaluate their integrity using remote attestation, whereas the DO installs and sets up a BTrust agent in his devices.

The enrolment phase : In the enrollment phase, the Network Operator (NO) designates the initial set of peers, managed either by users or the NO itself. During this phase, the NO utilizes a Public Key Infrastructure (PKI) to generate a unique public/private key pair for each user. Subsequently, users are enabled to enroll their devices into the Identity and ISC. The ISC maps a device's identity (public key or wallet address) to several attributes, including a device ID, an IP address, a blockchain wallet, a Device Owner (DO), and a patch ID. Enrollment is an ongoing process. However, during the first enrollment phase, the RSC

```
{
  "properties": {
    "displayName": "IoT Devices must have Secure Boot Enabled",
    "policyType": "Custom",
    "mode": "All",
    "description": "All IoT devices must have secure boot enabled to ensure that only signed and trusted boot components are used during the boot process.",
    "metadata": {
      "category": "IoT Security",
      "version": "1.0.0"
    },
    "version": "1.0.0",
    "parameters": {
      "effect": {
        "type": "string",
        "defaultValue": "AuditIfNotExists",
        "allowedValues": [
          "AuditIfNotExists",
          "Disabled"
        ],
        "metadata": {
          "displayName": "Effect",
          "description": "Enable or disable the execution of the policy"
        }
      }
    },
    "policyRule": {
      "if": {
        "field": "type",
        "equals": "IoTDevices"
      },
      "then": {
        "effect": "[parameters('effect')]",
        "details": {
          "type": "SecurityAssessment",
          "name": "IoT Secure Boot Assessment",
          "severity": "High",
          "severity_scrore":5
        }
      }
    }
  },
  "id": "iot-secure-boot-policy"
}
```


Figure 3.8: Minimal IoT Security Policy: Secure Boot

constructs an initial network graph comprising full nodes and the first devices, which are initially considered trusted devices.

The processing phase : In the processing phase peers and users can join and leave the network, the new joining nodes retrieve a list of live peers from a bootstrap DNS server or a cached node list and then select a set of bootstrapping peers using the random walk function defined in algorithm (2). BTrust agents start assessing the device security and the user behaviour, as well as rating other peers using algorithm (1). While computing the trust, the global score is computed iteratively, since the global trust of a peer depends on the global trust of the rating peers, till it converges below a specified threshold(γ). Initially, the algorithm starts with default trust values. As peers obtain feedback from each other, the trust value is updated regularly according to BTrust formulas. The trust computation is performed fully on-chain via smart contracts. BTrust agents (peers) communicate all trust data to the reputation smart contract to perform the trust computation by invoking the dedicated function "$addOrUpdateTrust$" (fig 3.3). However, the trust computation can be performed off-chain. In this case, each BTrust agent computes its own global reputation, except for the last calculation, which is performed on the smart contract when updating the trust score to ensure the veracity of the calculation. The reputation smart contract calculates and checks the convergence of the global trust—locally computed by the peers—then updates the global trust score in a single transaction by invoking a dedicated function "$checkandUpdateTrust$", which accepts as input all intermediary computations for each iteration before convergence. The intermediary values are used by the smart contract to inspect and verify the trust computation performed by the peer.

Figure 3.6 provides a global overview of different interactions between the components involved in assessing and evaluating a peer's trust in the BTrust network during these three phases.

3.4.1 Trust-Based peer selection using random walks

In this subsection, we describe how BTrust protocol selects bootstrapping peers or counterparts with whom peers can exchange transactions. Intuitively, an honest peer will tend to interact with closest peers that have a higher level of reputation and a higher trust score. However, this approach will incur a heavy workload for the most reputable and trustworthy nodes in large-scale networks. To avoid this problem, our selection process considers nodes' reputation and capacity (free inbound connections). Furthermore, we want to assist new joining peers to randomly select their neighbors avoiding peers with a high number of incoming connections (peer's In-degree). The focus of our work is on unstructured P2P systems, where peers select neighbors randomly without any knowledge about the network topology. The selection is performed through random walks over an overlay network based

Algorithm 1 BTrust algorithm for computing peer's trust

$Tr(P)$
Require α, β, γ, N, FeedbackThreshold
Initialization $ST(P) \leftarrow$ RetrieveSecurityScore(P)
$UB(P) \leftarrow$ RetrieveUserBehaviorScore(P)
if $N >$ *FeedbackThreshold* **then**
  for $i = 1;\ i < N;\ i = i + 1$ **do**
    $LT(P, i) \leftarrow$ RetrieveLocalTrust(P, i)
    $GT(i) \leftarrow$ RetrieveGlobalTrust(i)
    $Tr(i) \leftarrow$ Default
  end
  while $|GT'(P) - GT(P)| > \gamma$ **do**
    for $i = 1;\ i < N;\ i = i + 1$ **do**
      $GT'(P) \leftarrow \sum_{i \in N(P)} LT(i, P) * (GT(i) / \sum_{j \in N(P)} GT(j))$
    end
  end
end
return $Tr(P) \leftarrow \alpha * (GT(P) * UB(P)) + \beta * (ST(P))$

on the reputation graph and peers' degrees.

To evaluate the availability of appropriate peers, we define the pertinence ratio PR as the global trust of a peer 'i' divided by its in-degree (d_i):

$$PR(i) = \begin{cases} Tr(i)/d_i, & \text{if } d_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

New nodes should preferably connect to peers with similar or higher pertinence ratios. The probability of a new joining node choosing the peer 'i' is defined as the following:

$$P(i) = PR(i) / \sum_{j \in O(t)} PR(j) \tag{3.10}$$

such that $O(t)$ denotes the set of online nodes at a given time. The probability distribution $P(X = i)$ can be therefore represented as :

$$P(X = i) = P(i) = 1/Z * PR(i) \tag{3.11}$$

$$Z = \sum_{j \in O(t)} PR(j) \tag{3.12}$$

As it is impractical to calculate $P(X)$ in wide networks due to the large number of peers, we use a Markov chain Monte Carlo method (MCMC) for sampling from P. More specifically, we

use the Metropolis-Hastings (HM) algorithm [124][125] which works by simulating a Markov Chain with a stationary distribution Π. This means that, after enough time, the samples from the Markov chain resemble those of the samples from Π. That is, we model a second

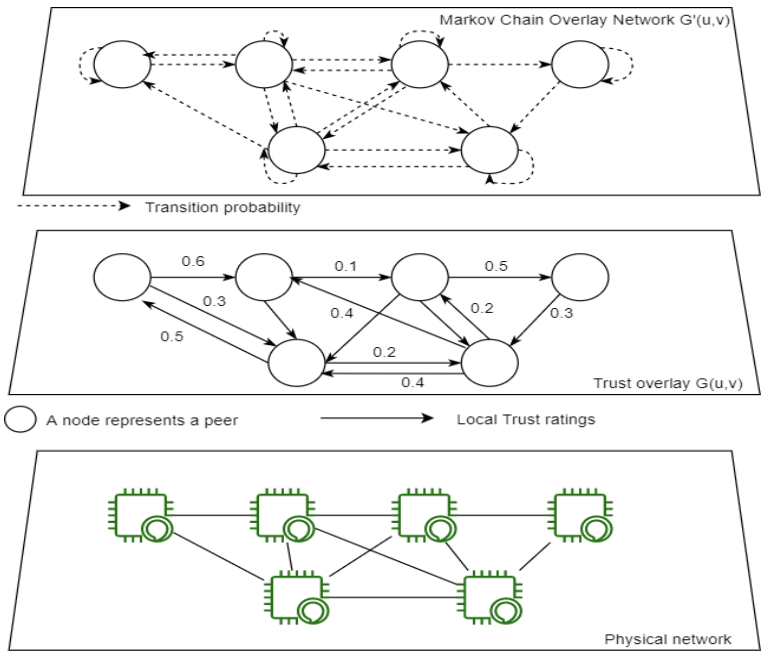


Figure 3.9: Trust overlay network in BTrust

overlay P2P network as a connected graph $G' = (V, E)$ with finite node set $V = \{1, 2, .., N\}$ and an edge set E that belongs to V^2. In G', We assign a transition probability and create a self-loop at each node (Fig.3.9), such that the total transition probability is 1.

Using HM, we then construct a Markov transition matrix P as :

$$Pij = P(Xn = j/Xn = i) \tag{3.13}$$

$$Pij = \begin{cases} q(i,j) * \alpha(i,j), & \text{if } j \neq i \\ q(i,j) + \sum_{k \neq i} q(i,k) * (1 - \alpha(i,k)), & \text{otherwise} \end{cases} \tag{3.14}$$

such that q is the transition kernel, which represents the probability of proposing a move to some state j (peer j) given the current state i(peer i). In our case

$$q(i,j) = 1/(d_i + 1) \tag{3.15}$$

and

$$\alpha(i,j) = min\{1, p(j) * q(j,i)/p(i) * q(i,j)\} \tag{3.16}$$

Algorithm 2 Random walk algorithm for peer selection with connection constraints

```
PeerList
Require m, P, TTL, Graph G', MaxIn, MaxOut
Initialization w ← P // The walker starts at P
for i = 1; i < m; i = i + 1 do
    inboundCount ← CountInboundConnections(w)
    while w is connected to P do
        while TTL > 0 do
            w ← ForwardWalker(TransitionProbability) //Position after moving one step
            TTL ← TTL − 1
            inboundCount ← CountInboundConnections(w)
            if inboundCount < MaxIn then
                continue // Skip the rest of this iteration if the inbound connection limit is
                exceeded
        end
        if w is connected to P and TTL = 0 then
            w ← ForwardWalker(TransitionProbability)
        else if w is not connected to P then
            outboundCount ← CountOutboundConnections(w)
            if outboundCount < MaxOut then
                PeerList[i] ← w  break // Exit the loop once a suitable peer is found
    end
end
return PeerList
```

is the acceptance probability for accepting a proposed move from state i (peer i) to state j (peer j). By its definition, the defined MC is reversible, aperiodic and irreducible

By using this algorithm, new joining nodes retrieve a list of neighbouring peers. Each node broadcasts its current connectivity degree to its neighbours to allow other peers to calculate the defined pertinence ratio $PR(i)$ using the global trust retrieved from the reputation smart contract. The pertinence ratio is then assigned as a transition probability to their edges as shown in figure 3.9. Next, each node starts multiple walkers, where the number is equivalent to the node's Out-degree. In order to avoid long walks, each walk is limited in time using a TTL value equivalent to the number of iterations in the HM algorithm. The walker moves from one node to another based on the edge probability and the walker's TTL is decremented until it stops ($TTL = 0$). If the node where the walker stops is already connected to the starting node, then the walker moves some additional steps. The walk repeats until the walker discovers a suitable node for the new node to join. Each node has a maximum number of allowed inbound ($MaxIn$) and outbound connections ($MaxOut$).

3.5 Feedback quality

BTrust protocol relies partially on p2p feedback rating to compute trust scores. However, two major problems stand in the way of having a reliable feedback exchange; Lack of trustworthy ratings and false feedback. In this section, we propose our solution to address these 2 issues.

3.5.1 Reputation tokenization and incentivisation

BTrust introduces a financial incentive mechanism that aims to incentivize users and peers in the P2P network to enhance their trust scores by promoting honest behavior and providing accurate feedback. The core concept behind this mechanism is to monetize reputation by converting it into tangible financial tokens known as reputation tokens.

Conversion to reputation tokens

The RSC enables each device owner to convert a fraction of their device's trust score into reputation tokens. These tokens are then allocated to the user's wallet, which is managed directly by the contract. The conversion rate is determined by NO via a predefined function that takes into account the current trust score and predefined thresholds. The formula for conversion might look something like:

$$\text{Reputation Tokens} = \frac{\text{Trust Score} - \text{Base Threshold}}{\text{Conversion Factor}} \tag{3.17}$$

where the Base Threshold is a minimum trust score required for conversion, and the Conversion Factor is a scaling parameter set by the Network Operator (NO). The conversion process involves the following steps:

- Conversion Request: The device owner (DO) initiates a conversion request via their BTrust agent by calling the $convertTrustToToken$ function of the RSC. The request specifies the amount of trust score to be converted.
- Trust Score Calculation: The RSC calculates the corresponding number of reputation tokens based on a predefined conversion rate.
- Trust Score Deduction: The specified amount of trust score is deducted from the DO's current trust score. Trust score deduction is performed by updating the DO's trust score in the RSC's storage.
- Token Allocation: The equivalent number of reputation tokens is minted and allocated to the DO's wallet. This involves calling the mint function to create new tokens and the transfer function to transfer them to the DO's address.

Trading and Use of reputation tokens

Once converted, these reputation tokens become tradable assets among users, facilitating transactions and granting access to services offered by the NO on a dedicated marketplace or with other users within the ecosystem. For example, users can use reputation tokens to pay for premium backup services provided by the network operator. The marketplace allows for the exchange of services and reputation tokens, ensuring that peers with higher trust scores have greater access to valuable services. The smart contract ensures secure and transparent transactions within the marketplace, with functions to list services, price them in reputation tokens, and handle payments.

Purchase and impact on reputation

It's important to note that purchasing reputation tokens by a Device Owner (DO) does not directly impact the reputation of their own device or entity. Instead, reputation tokens serve as a means of transferring trustworthiness from one peer to another. This mechanism includes rules to prevent abuse, such as:

Reputation tokens can only be transferred between wallets that have previously interacted or have an established trust relationship. The reputation tokens can be used for services but cannot be reconverted back into trust scores for the device they originated from. Once used they are burned by RSC.

Role of the reputation smart contract

The reputation smart contract, often built on a blockchain like Ethereum, manages the lifecycle of reputation tokens. Using the ERC721 standard [1], each token represents a unique and non-fungible reputation score. Key functions of the contract include:

- Minting Tokens: Creating new reputation tokens when trust scores are converted.
- Transferring Tokens: Securely transferring tokens between wallets.
- Burning Tokens: Destroying tokens when used for services or penalized for misbehavior.
- Verification: Ensuring the legitimacy of tokens and transactions.

Incentives and penalties

To ensure the incentivization mechanism promotes the desired behavior, additional rules and policies are implemented:

- **Incentives for positive behavior**: Users and peers are rewarded with reputation tokens for honest behavior, accurate feedback, and contributions to the network's

[1] https://eips.ethereum.org/EIPS/eip-721

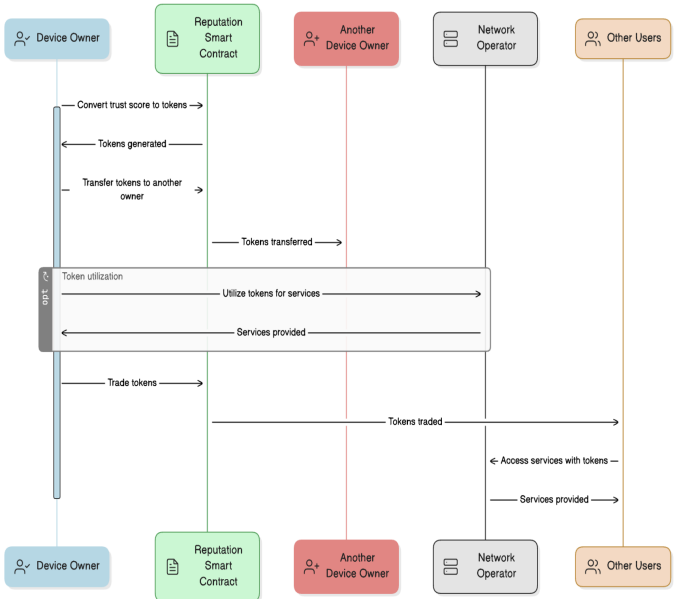


Figure 3.10: Reputation tokenization workflow

```go
// ReputationToken represents a reputation token compatible with ERC721 standard.
type ReputationToken struct {
    ID          string `json:"ID"`
    Owner       string `json:"owner"`
    Transferable bool   `json:"transferable"`
}

// MaxConversionAmount represents the maximum amount of reputation that can be converted into tokens.
const MaxConversionAmount = 100
// ConvertTrustScoreToTokens initiates the process by converting trust score into reputation tokens.
func (rc *ReputationContract) ConvertTrustScoreToTokens(ctx contractapi.TransactionContextInterface,
    peerID string, reputationAmount int) (string, error) {
    // Validate the reputation amount against the predefined limit
    if reputationAmount <= 0 || reputationAmount > MaxConversionAmount {
        return "", fmt.Errorf("invalid reputation amount: must be between 1 and %d", MaxConversionAmount)
    }
    // Check if the invoking identity is the owner of the peer
    peerOwner, err := rc.GetPeerOwner(ctx, peerID)
    if err != nil {
        return "", fmt.Errorf("failed to get peer owner: %v", err)
    }
    if peerOwner != ctx.GetClientIdentity().GetID() {
        return "", fmt.Errorf("only the owner of the peer can convert its trust score into tokens")
    }
    // Generate a unique ID for the reputation token
    tokenID := ctx.GetStub().GetTxID()
    // Create a new reputation token
    token := &ReputationToken{
        ID:          tokenID,
        Owner:       peerOwner,
        Transferable: true,
    }
    // Save the reputation token on the ledger
    if err := rc.UpdateToken(ctx, token); err != nil {
        return "", fmt.Errorf("failed to save reputation token: %v", err)
    }
    return tokenID, nil
}
```

Figure 3.11: code managing tokenization of RSC

security. The rewards are periodically distributed based on the trust score improvements recorded on the blockchain.

- **Penalties for misbehavior**: If a peer is found to be dishonest or if their device's security is compromised, their reputation score can be reduced, leading to a devaluation of their reputation tokens. This is enforced by the smart contract, which can automatically deduct tokens or burn them based on predefined conditions.

Example of a Penalty Consider a peer that has been detected engaging in dishonest behavior. The penalty mechanism can be described as follows:

1. **Detection**: The system detects that Peer A has provided false feedback or has been involved in malicious activity.
2. **Penalty calculation**: The smart contract calculates the penalty based on the severity of the misbehavior. For instance, let P_s be the penalty severity score and T_s be the current trust score.

$$\text{Penalty Amount} = \alpha \times P_s$$

$$\text{New Trust Score} = T_s - \text{Penalty Amount}$$

Here, α is a scaling factor determined by the NO.

3. **Token deduction**: The equivalent amount of reputation tokens is deducted from

Peer A's wallet. If the penalty exceeds the available tokens, the remaining penalty is subtracted from future earnings or results in a temporary ban.

$$\text{Tokens Deducted} = \frac{\text{Penalty Amount}}{\text{Conversion Factor}}$$

By integrating this financial incentive mechanism and leveraging the reputation smart contract, BTrust establishes a dynamic and economically-driven reputation system within the P2P network. This system encourages peers to engage in trustworthy behavior and discourages malicious activities, as the direct economic consequences of reputation loss serve as powerful deterrents. Overall, the combination of reputation tokens and the reputation smart contract enhances trust management in the P2P network, facilitating a more reliable and robust ecosystem for trust evaluation and decision-making.

3.5.2 Mitigation of lack of rating and bad behaviour

In order to limit false feedback and mitigate Bad-mouthing attacks [126], as well as to deter bad behaviour, BTrust relies on special entities we call Watchdogs. The Watchdogs are special BTrust peers –operated by the NO– responsible for inspecting the transactions and evaluating the feedback exchanged between peers. Each WatchDog peer hosts a complete updated copy of the Blockchain. Thus, a sender peer P can request the Watchdog to inspect the delivered transaction and feedback rating. If P does not receive the due rating feedback or it gets false feedback from the requesting peer Q, the watchdog peer can ask peer Q to provide the correct due feedback ($tx(P,Q)$) and subsequently both peers to correctly computing their local trust ($Sum(P,Q)$ defined in 3.2). This is possible since the reputation feedback provided by the requesting peer Q is stored in the RSC and because service requests and responses are digitally signed by both peers 3.

In fact, in BTrust when a peer Q requests data from another peer P, the former digitally signs the request. P then responds with a transaction that conveys alongside the requested data, a timestamp, a digital signature (or a message authentication code) of P and the signed request initiated by peer Q. The received transaction is stored in the memory pool of the recipient peer Q, for a limited time –equivalent to the average time needed for a blockchain transaction to be validated–. In the case where a peer is not able to store the received transaction for any reason (e.g. a full pool or unavailable free storage), it can request the watchdog to store them.

When peer P does not receive the correct feedback, it can provide the watchdog with the signed request and response. Subsequently, the watchdog can request the peer that missed giving the feedback to provide it to the peer providing the service. If the requesting peer Q refuses, the watchdog blocks its ability to tokenize reputation score in a slashing process. However, the limitation of this mechanism lies in the capacity of WatchDogs to assess the quality of service (QoS) provided by peers.

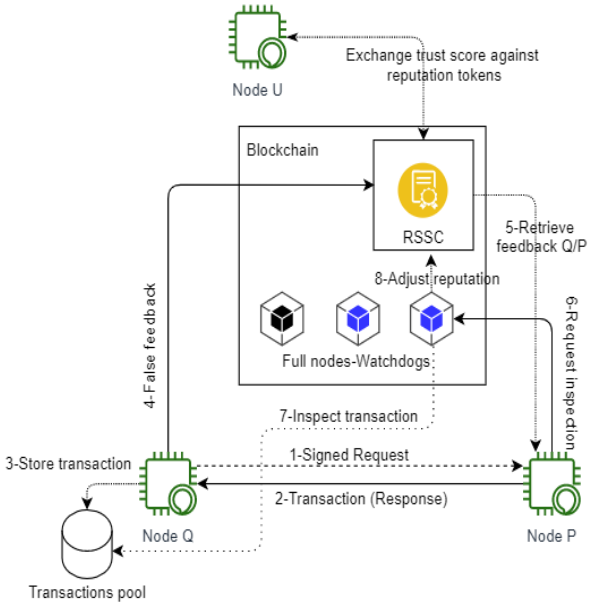


Figure 3.12: BTrust watchdog mechanism

Algorithm 3 Watchdog Mechanism in BTrust

Verified Transactions and Penalized Peers

Data Structures:

- **Transaction** { sender: string, receiver: string, timestamp: int64, signedRequest: string, signedResponse: string, feedback: string }
- **Feedback** { from: string, to: string, score: int }

Procedure: RegisterTransaction txID, sender, receiver, timestamp, signedRequest, signedResponse, feedback tx ← **Transaction**{sender, receiver, timestamp, signedRequest, signedResponse, feedback} Store tx in ledger with key txID

Procedure: VerifyTransaction txID, sender, receiver, signedRequest, signedResponse tx ← Get transaction from ledger using txID **if** *tx is NULL* **then**
| **return** *False, "Transaction does not exist"*
end
if *tx.sender* ≠ *sender tx.receiver* ≠ *receiver tx.signedRequest* ≠ *signedRequest tx.signedResponse* ≠ *signedResponse* **then**
| **return** *False, "Transaction details do not match"*
end
return *True, "Transaction verified"*

Procedure: PenalizePeer peer Reduce peer's trust score Block peer's ability to tokenize reputation score

Procedure: StoreTransaction txID, sender, receiver, signedRequest, signedResponse tx ← **Transaction**{sender, receiver, timestamp, signedRequest, signedResponse} Temporarily store tx in Watchdog storage with key txID

Procedure: WatchdogProcess **for** *each tx in Watchdog storage* **do**
| Verify tx details **if** *tx is not verified* **then**
| | Penalize peer involved in tx
| **end**
end

3.6 Experimental evaluation

In this section, we describe our simulation setup and configurations, and we present the result of the experiments conducted to study the effectiveness of the proposed Protocol. The simulation is based on different experiments that evaluate the proposed trust management algorithm and examine its accuracy in a variety of scenarios.

3.6.1 Simulation setup

To evaluate the BTrust model, we implemented a simulation model based on the Netlogo 6.1.1 environment [2]. Netlogo is a multi-agent modelling environment for emulating large-scale networks. This environment constructs networks of agents that can be programmed, using a high-level language, to behave and interact with each other according to the defined program. Using Netlogo we designed a cycle-based simulation model for BTrust. For simulation purposes, we assume a P2P file-sharing network as the application scenario of our trust model. At the start of each cycle, we assume that a random number of peers may start a new request for a file, respond to incoming requests, or rate the interactions. For each simulation configuration, we perform five randomized runs for 100 cycles each, on a community of 100 peers. The trust score (global and local trust) is computed through a deployed reputation smart contract. For simulating possible behaviours, we consider three types of peers :

- Honest peer: which always behaves honestly, cooperates with other peers and provides honest feedback afterwards.
- Static malicious peer: which always delivers bad transactions and gives wrong feedback to other peers.
- Dynamic malicious peer: which probabilistically behave maliciously by delivering bad transactions and feedback. We assume that all dynamic malicious peers act honestly in the beginning in order to enhance their trust score.

We use an unstructured P2P architecture with peers responding to incoming requests and providing feedback. We assume the network provides the same service (p2p file sharing). Table 3.2 describes the main parameters adopted in our simulations.

In this simulation, nodes are initialized with a global trust value of $1/N$. We assume that dynamic malicious peers act approximately with the same rate denoted as $maliciousness\_rate$. We consider that each node has a maximum of 8 outgoing connections. If one of these outgoing connections is disconnected, the node will try to replace the lost connection by trying to connect to another peer. At the same time, a node may accept up to 50 incoming connections from other peers.

[2] http://ccl.northwestern.edu/netlogo/

Parameters	Default values
α	0.7
β	0.3
$ST(i)$	1
$UB(i)$	1
Maliciousness rate ($maliciousness\_rate$)	100%
Trust score threshold	0.3
Total number of cycles	100
Number of services	1
Peers providing the service	All peers
Convergence threshold (γ)	10^{-4}
Feedback threshold (Witnesses)	5
Percentage of malicious peer	20%
Type of malicious peer	Static

Table 3.2: Trust simulation default settings

3.6.2 Experiment 1 : Evaluation of accuracy

In this first experiment, we evaluate the accuracy and effectiveness of BTrust against malicious peers through the calculation of root-mean-square error (RMSE) of aggregated total trust of all peers. RMSE is an estimator of the overall deviations between the predicted and measured values. The RMSE is defined by the following :

$$RMSE = \sqrt{\sum_{i=1}^{N}((T_i - Tc_i)/T_i)^2/N} \tag{3.18}$$

Where N is the total number of peers in the network, and T_i and Tc_i are the correct and evaluated trust of a peer i, respectively. The RMSE is a good indicator that is inversely proportional to the accuracy of the trust models, such that the lower the RMSE, the more the accuracy of the trust evaluation. The plots in Figure 3.13 and 3.14 show respectively the average of RMSE under different rates of malicious static peers and malicious dynamic peers. Within both scenarios, a malicious proportion of 45% produces a low RMSE below a value of 0.2. Thus, the BTrust approach of computing the trust score remains robust when we have a large fraction of dishonest nodes. This result can be explained by the fact that BTrust uses a personalized and adaptive formula to compute trust for each entity. Another

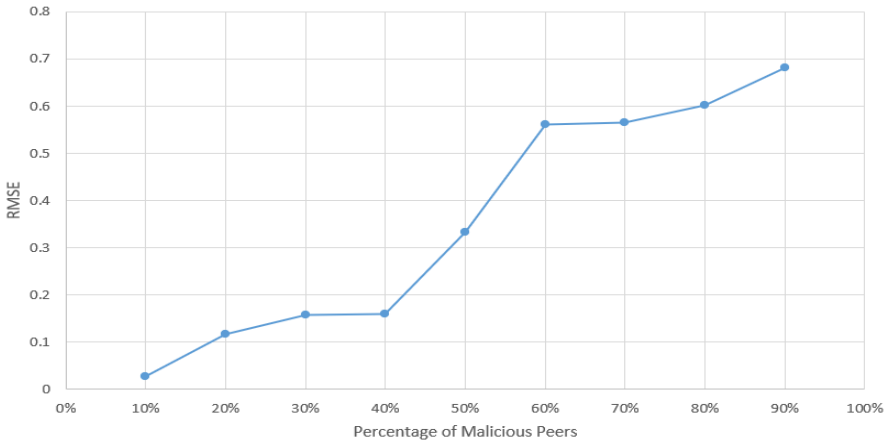


Figure 3.13: Trust estimation error (RMSE) in presence of different portions of static malicious peers

reason is the fact that our system, unlike existing decentralized trust systems, replaces score managers with a trustworthy smart-contract based score management system. This obviates the inherent risks of having malicious score managers; i, e., a malicious trust manager could intentionally or under attack affect the computation of the trust value of a given peer. Moreover, in the BTrust model, we leverage watchdogs for feedback verification to filter out malicious feedback.

3.6.3 Experiment 2: Effectiveness against dynamic and static malicious Peers

In this second experiment, we focus on evaluating the effectiveness of the BTrust model in resisting dynamic attacks without leveraging the feedback correction mechanism (Watchdogs). In that regard, we evaluate the successful transaction rate (STR) in the presence of dynamic

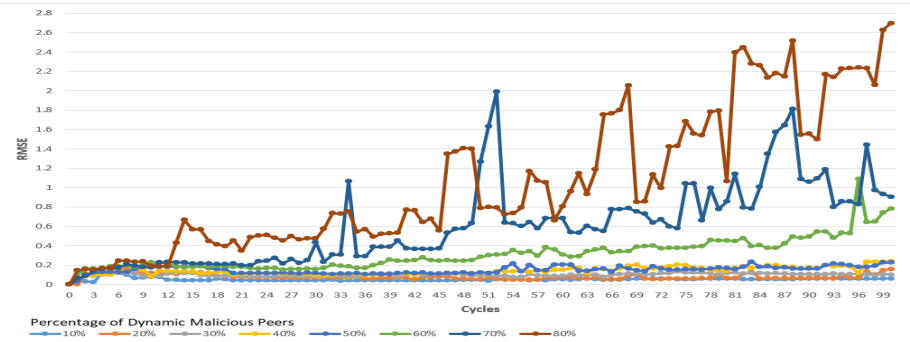


Figure 3.14: Trust estimation error(RMSE) in presence of different portions of dynamic malicious peers, during 100 cycles

malicious nodes that oscillate between malicious and honest behaviour at random with a probability of 0.5. The STR is the ratio of the number of successful transactions over the total number of transactions. It is a typical metric used to evaluate the efficiency of trust models. We assume these peers behave honestly up to some time (10th cycle) to accumulate trust before they start behaving maliciously. Within this experiment, we maintain the default percentage of malicious nodes in the network (20%).

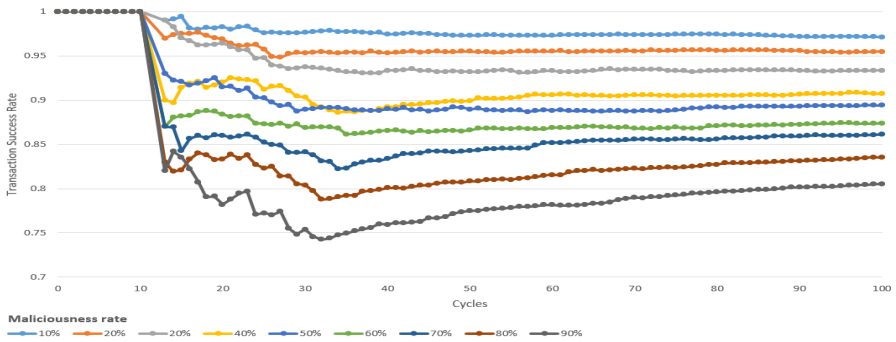


Figure 3.15: Successful transaction rates under different scales of dynamic attacks

Figure 3.15 illustrates how the level of trust evolves during 100 cycles. We observe that the STR decreases proportionally to the $maliciousness\_rate$. In the context of a low and moderate $maliciousness\_rate < X$ within the interval[0.1,0.2], we observe a weak variation of STR (0.1%). When the malicious peers switch from behaving honestly to behaving maliciously, the STR decreases quickly and, after an average of 40 cycles, the STR stabilizes at a high rate. This cadence can be explained by the fact that the trust score of dynamic malicious peers decreases, even though they behave correctly from time to time in order to regain the trust score. This result reflects that the correctness of the BTrust model is very high even in the presence of dynamic malicious peers. Moreover, the STR remains below 1 as the malicious peers continue to exchange malicious transactions and feedback between them without affecting honest peers.

In a second scenario, we compute the STR in the presence of different proportions of static malicious peers. As shown in Figure 3.16, a high proportion (over 60%) destabilises the level of trust within the system and thus decreases significantly the STR. Inversely, the STR remains at a high level of 0.8 when we have a malicious proportion of less than 40%. This result reflects the resilience of the BTrust model against a large proportion of static malicious actors.

3.6.4 Experiment 3 : Convergence of BTrust

In this simulation, we focus on evaluating the convergence speed of BTrust and its scalability with regard to the increasing network. The convergence speed is measured as the number of iterations needed before the trust score converges. Thus, a lower value of the convergence

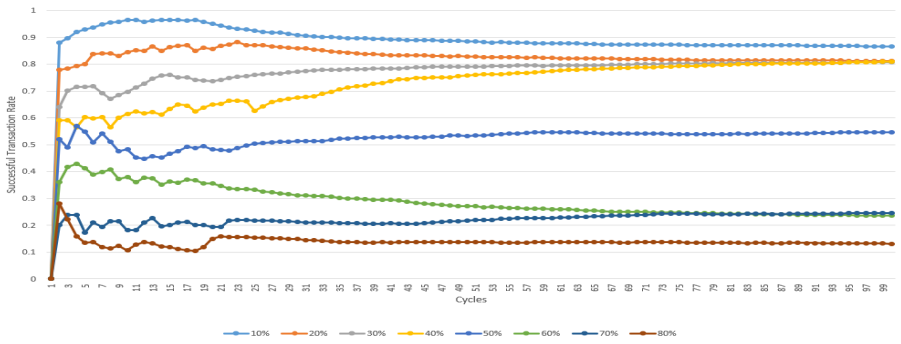


Figure 3.16: Successful Transaction Rate(RMSE) in the presence of different proportions of static malicious peers

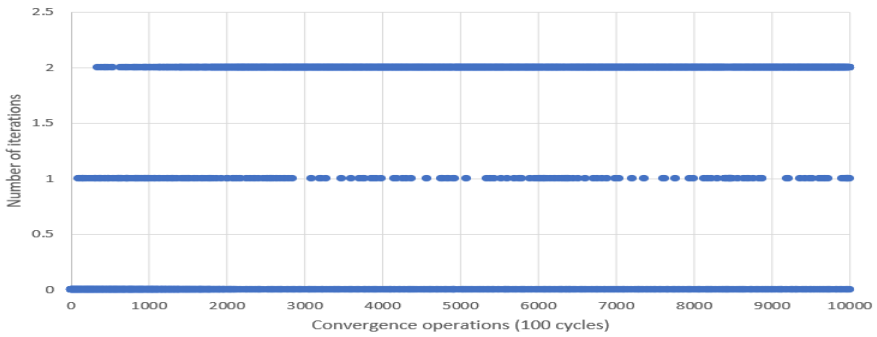


Figure 3.17: Number of iterations required for convergence by each node within 100 cycles

iterations means a higher convergence speed. Interestingly, the results shown in Figure 3.17, confirm that the convergence speed is very fast since each node needs a maximum of 2 iterations before trust score computation converges. The results prove the scalability of BTrust concerning the number of iterations needed to converge since the latter does not grow substantially with the increase (from 100 to 10000) of the number of peers in the network.

3.6.5 Load distribution

In this experiment, we evaluated the load of individual peers incurred by BTrust. The metric used for this evaluation is the number of times a particular peer (service) is requested to deliver a transaction. To measure the amount of variation across the network, we compute the standard deviation of the load among all the peers. The simulation is performed after 10000 transactions, the equivalent of an average of 100 transactions per peer, in the presence of a varying-size minority of malicious peers (5-80%). The load computation is performed only on honest peers since they are the preferred targets of other peers and thus the most likely to incur heavy load. Figure 3.18, shows that the standard deviation of the load distribution is very minimal and does not change significantly as the size of the malicious

minority increases. Thus, BTrust does not incur a heavy load on the peers, since the adopted random walk process privileges peers with a high pertinence ratio.

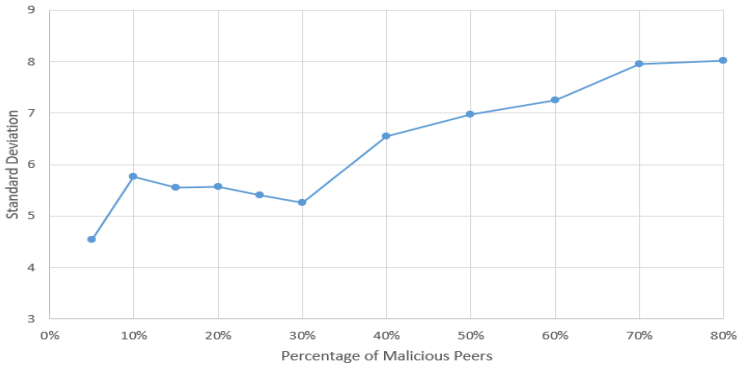


Figure 3.18: The standard deviation of the load distribution in the network

3.6.6 Comparison with Current BRTMs: An analytical Study

In Table 3.3, we present a comparative study of our work with the state of the art BRTMS solutions, and show that the proposed solution outperforms other similar approaches. BTrust presents several key advantages over the existing RTMS. First, it is robust to outliers and Sybil attacks, which ensures the integrity and trustworthiness of the system. This is a crucial feature, as outliers and Sybil attacks can severely compromise the trust management process. Second, BTrust demonstrates good scalability, allowing for its effective implementation in P2P file sharing, social networks, and e-commerce domains. Furthermore, BTrust is a modular protocol that can be easily adapted to multiple use cases. Its generic nature allows for flexibility in implementing trust management solutions across various domains and applications. This modularity enables BTrust to incorporate additional services, factors, or extensions to enhance the trust evaluation process as needed. By offering a customizable and adaptable framework, BTrust provides a robust foundation for building trust management systems that cater to specific requirements and diverse scenarios.

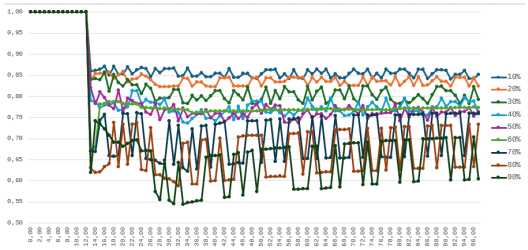


Figure 3.19: Successful transaction rates under different scales of dynamic attacks of HonestPeer (non-blockchain based)

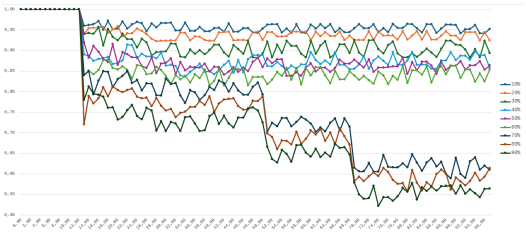


Figure 3.20: Successful transaction rates under different scales of dynamic attacks in case of Proof-of-trust(blockchain based)

Actually, BTrust is a comprehensive and generic solution that tackles various trust-related challenges in peer-to-peer networks. Unlike other cited solutions, BTrust addresses the

Table 3.3: Comparison of BTrust with other BRTMS

Solution	Field of application	Intended Improvement						Performance Measured
		Cold start problem	Incitivation	Assessment of bad behavior	Extensibility of Trust Formula	Metrics of trust	Trust Dissemination	
[110]	P2P	No	No	Yes	No	Bad or good transactions	Client side	Not Provided
[112]	Open multi-agent system in P2P clusters	Yes	No	Yes	No	Immigration request actively rejected, Immigration request confirmed Task result verified	Consortium Blockchain	Number of kudos delivered
proof-of-trust [113]	Croud-sourcing/ Sensing	Yes	Yes	Yes	No	Total Transaction Amount validation Times	Blockchain	Accuracy Scalability
[116]	Autonomous systems	No	No	Yes	No	SLA score of an agreement the SLA score of a network	Oracle and permissioned blockchain	Not provided
[118]	Monetization of data in IoT	No	No	No	No	Review and rating	Smart Contract on Ethereum Blockchain	Not provided
[120]	IoT /Sensor Network / Edge Computing	No	No	No	No	History of obligation fulfillment	Obligation chain	Reputation
[119]	IoT /Sensor Network / Edge Computing	No	No	Yes	No	Miner approval Blame Payload Renew Payload Ban Payload	Blockchain	Trust level
BTrust [93]	Generic	Yes	Yes	Yes	Yes	The feedback a peer obtains from other peers; The total number of transactions that a peer performs; The security assessment of a from less or secure ones peer for discriminating vulnerable devices The user's behavior	Smart contract	Accuracy Effectiveness against dynamic and static Malicious Peers Convergence of BTrust

cold start problem, provides incentives, assesses bad behavior, allows flexibility in trust formula, and employs multiple metrics for evaluating trust. It leverages smart contracts for trust dissemination, with the aim of attaining high accuracy and effectiveness against both dynamic and static malicious peers. In fact, in figures 3.13, 3.15, 3.19, 3.21 we observe a significant difference in term of the percentage of successful transactions and RMSE compared to proof-of-trust in the presence of various scales of dynamic attacks, as detailed in section 3.6.3. The successful rate of proof-of-trust drops significantly, and trust estimation error rises significantly in the presence of 60% dynamic malicious peers, as the protocol adopts a majority vote mechanism. This technical comparison was feasible with proof-of-trust but not with other solutions, as it's the only blockchain-RTMS offering clear algorithms for implementation and testing.

BTrust distinguishes itself through its comprehensive approach, which incorporates an

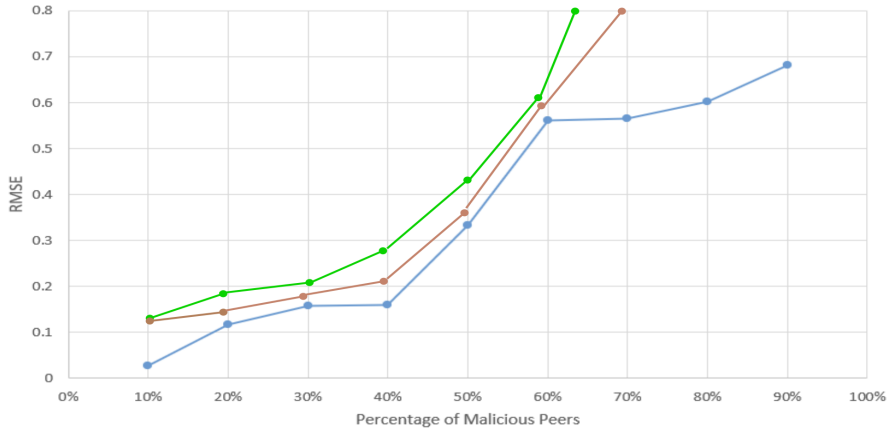


Figure 3.21: Trust estimation error (RMSE) in presence of different portions of static malicious peers (proof-of-trust in brown, BTrust in blue, HonestPeer in green)

analysis of user behavior, evaluates the security status of vulnerable devices, and explores the dynamics of peer interactions. By meticulously examining these factors, BTrust aims to encompass all elements that influence trust within a resource-sharing network.

3.6.7 Comparison with traditional RTMS

In Table 3.4, various RTMS are compared with BTrust, focusing on attributes such as advantages, disadvantages, complexity, resistance to dynamic malicious peers, peer selection mechanism, scalability, and domain of application. BTrust demonstrates robustness against malicious dynamic peers and Sybil attacks, making it suitable for P2P resource sharing [127]. Despite its complexity in implementation, BTrust offers a high level of security against evolving threats. Comparatively, other RTMS like PeerTrust are simpler to understand and implement but lack resilience against malicious peers [127]. EigenTrust and PowerTrust show robustness against specific threats but are complex to implement [22, 128]. CuboidTrust and VectorTrust offer expressive capabilities but also require a high level of understanding and implementation complexity [24, 129]. HonestPeer and AuthenticPeer++ provide robustness against Sybil attacks but at a higher complexity [22, 23]. In fact, in figures 3.13, 3.19, 3.15, and 3.21, we observe a significant discrepancy in both the percentage of successful transactions and RMSE when compared to HonestPeer, particularly in the context of various scales of dynamic attacks outlined in Section 3.6.2. BTrust outperforms HonestPeer in terms of success rate and trust estimation error. Overall, BTrust distinguishes itself with its comprehensive security features, making it well-suited for P2P resource-sharing environments leveraging blockchain technology.

Table 3.4: Comparison of Reputation Trust Models

RTMS	Advantages	Disadvantages	Complexity	Resistance to Dynamic Malicious Peers	Peers Selection Mechanism	Scalability	Domain of Application
PeerTrust [127]	Easy to understand and implement.	Sensitive to malicious peers.	Low	Low	No	Good	P2P file sharing, social networks
EigenTrust [22]	Robust to malicious peers.	Complex to understand and implement.	Medium	Medium	No	Good	P2P file sharing, social networks
PowerTrust [128]	Robust to collusion.	Complex to understand and implement.	Medium	Medium	No	Good	P2P file sharing, social networks
HonestPeer [22]	Robust to Sybil attacks.	Complex to understand and implement.	High	High	Yes	Good	P2P file sharing, social networks
AuthenticPeer++ [23]	Robust to Sybil attacks and collusion.	Complex to understand and implement.	High	High	Yes	Good	P2P networks
CuboidTrust [24]	More expressive than traditional RTMS.	Complex to understand and implement.	High	Medium	Yes	Good	P2P file sharing, social networks, e-commerce
VectorTrust [129]	More expressive than traditional RTMs.	Complex to understand and implement.	High	Medium	Yes	Good	P2P file sharing, social networks
BTrust	Robust to malicious dynamic peers and Sybil attacks.	Complex to implement.	High	High	Yes	Good	P2P resources sharing

3.7 Conclusion

In this section, we have introduced BTrust, a pioneering trust model that amalgamates multiple factors to quantify and compare the trustworthiness of peers within peer-to-peer (P2P) networks. Our approach incorporates a three-layered trust system, leveraging P2P feedback mechanisms, user behavior analysis, and device security assessments. Additionally, we have outlined an efficient peer selection method based on random walks, which mitigates computational burdens in large networks while facilitating the identification of secure peers for newcomers.

At its core, BTrust is designed to be extensible and modular, enabling the seamless integration of additional services such as spamming activity detection or behavioral pattern analysis to enhance the accuracy of user behavior evaluation through representative metrics. Through simulations, we have validated that our trust model effectively fulfills the design considerations outlined in the introduction. It adeptly disseminates trust information, identifies and isolates malicious peers within P2P networks autonomously, without reliance on a central authority, and maintains efficiency and efficacy against a spectrum of attacks.

Chapter 4

GBTrust: An Edge Weight-Shared Graph Neural Network for Trust management in P2P networks

Contents

4.1 Introduction

Recent advancements in Machine Learning (ML) and Artificial Intelligence (AI) have led to the development of new trust management protocols. These models introduce enhanced adaptability and predictive capabilities, presenting potential solutions to the challenges encountered in P2P networks, as surveyed in [130] [131, 132] [133]. In fact, the integration of AI into trust management systems has introduced innovative approaches, yielding enhanced results in tackling longstanding issues. Various methodologies highlighted in the literature exemplify this progress:

- Anomaly Detection: AI-based algorithms leverage historical data to discern patterns indicative of malicious behavior [134]. By continuously learning from past instances, these algorithms enhance the system's ability to identify and flag anomalies in real-time, enhancing network security.
- Predictive Reputation Scoring: AI-based models analyze past interactions and user behaviors to forecast future trustworthiness [135]. Unlike traditional static scoring systems, which may overlook evolving user dynamics, predictive reputation scoring adapts to changes, providing a more accurate assessment of a user's reliability over time.
- Community Detection: AI-based algorithms can detect communities or clusters within the network based on patterns of interactions and connections between peers [136]. By identifying cohesive groups of users, community detection facilitates targeted trust assessments within these communities, improving the granularity and accuracy of trust management in complex network structures.
- Collaborative misbehavior detection: AI-enabled trust management systems can detect instances of collaborative misbehavior, where multiple peers coordinate to deceive or manipulate the network [137] [138]. By analyzing patterns of cooperation and behavior, these systems can identify suspicious collaborations and mitigate the impact of coordinated attacks on trust within the network.

  Among AI approaches, Graph Neural Networks (GNNs) [98] [139] have recently become the standard tool for machine learning on graphs. These architectures effectively combine node features and graph topology to build distributed node representations. GNNs can be used to solve node classification [139] and link prediction [140] tasks, or they can be applied to downstream graph classification [139]. In literature, such models are usually evaluated on chemical and social domains [141]. Given their appeal, an ever increasing number of GNNs is being developed [142]. Their capacity to discern complex relationship patterns within networks is unparalleled. By leveraging graph structures, GNNs adeptly capture the subtle dynamics of trust interactions, marking a significant advancement in the field of trust management.

In this chapter, we present GBTrust, a GNN-based RTMS that harnesses these abilities to provide a better trust assessment on the network. In this following section of the thesis, we present a novel approach to enhance RTMSs by incorporating an Edge-Feature Attention Mechanism into the Edge Graph Neural Network (EGNN) model, which takes into account the directionality of edges. GBTrust represents a progression from BTrust, where the graph and its embeddings (table 4.2) are produced through the execution of the latter.

4.2 Contributions

We hypothesize that the proposed GBTrust model could potentially outperform both traditional and ML-based trust management systems in terms of accuracy and adaptability in detecting malicious peers, thereby significantly improving the overall security and reliability of P2P networks.

This model makes two primary contributions:

- We present an AI model intended to complement traditional reputation-based trust protocols, enhancing their performance using an enhanced version of EGNN.
- We propose a new graph attentive network model, which introduces a newly designed attention mechanism that accounts for the direction and influence from different features of edges within the network, enabling a more granular and context-aware analysis of peer interactions.

Also, we validate the effectiveness of our proposed model through experiments conducted on a synthetic dataset. The results demonstrate that the GBTrust model significantly outperforms other baseline models while concurrently providing valuable information about fraudulent users in the network.

4.3 Related Work: Application of GNN to trust evaluation

This section presents an overview of the existing research relevant to trust management in P2P networks, focusing specifically on those utilizing GNN models. Indeed, there is a growing interest in applying GNNs to trust management in P2P networks, with recent studies beginning to explore this application.

The first method we discuss is Guardian [143], which applies GNNs to trust evaluation. Guardian divides neighboring nodes into in-degree neighbors and out-degree neighbors to represent the roles of trustee and trustor, respectively. It utilizes a Graph Convolution Network (GCN) to aggregate information from these nodes and their corresponding trust interactions. Guardian employs a mean aggregator to combine information from first-order neighbors and stack multiple GCN layers to enable effective trust propagation. Guardian

outperforms traditional trust evaluation models and neural network-based models in terms of accuracy and efficiency. However, it fails to capture the dynamic nature of trust and the mean aggregator cannot differentiate the importance of different neighbors accurately, leading to inaccurate trust evaluation.

GATrust [144], addresses the limitations of Guardian by incorporating node features, such as personal hobbies, which are essential for trust evaluation. Similar to Guardian, GATrust establishes trust asymmetry based on out-degree and in-degree concepts and propagates trust using GNN's message-passing mechanism. GATrust integrates multi-facet properties of nodes, including contextual features, network structural information, and trust relationships. It utilizes Graph Attention Network (GAT) to assign varying weights to different properties of each node. By combining and aggregating these properties, GATrust learns node embeddings that contain rich information for accurate trust relationship prediction. However, like Guardian, GATrust does not consider the evolving nature of trust (dynamicity), and it lacks robustness

TrustGNN [145] takes a different approach by constructing various types of trust chains based on a RotatE-like approach [146] to achieve trust asymmetry and propagation. These trust chains are aggregated to form comprehensive representations (embeddings) of nodes for trust relationship prediction. TrustGNN employs learnable attention scores to differentiate the contributions of different trust chains, providing spatial explainability through visualization. However, TrustGNN focuses on a specific snapshot and does not account for the dynamic nature of trust, resulting in a lack of temporal explainability. It also ignores attacks on trust evaluation, raising concerns about its robustness.

The aforementioned models overlook the dynamicity of trust, leading to less effective and questionable predictions. To address this issue, Medley [147] incorporates temporal features along with other essential features to capture hidden and time-aware trust relationships. Medley utilizes a functional time encoding module to capture temporal information and assigns different weights to timestamped trust interactions using an attention mechanism. By incorporating temporal information, Medley demonstrates significant improvement compared to Guardian [148] . However, this design relies on fine-grained timestamps, which may be challenging to obtain in practice. It also incurs high storage consumption and computational overhead when nodes and edges are frequently updated. Additionally, Medley only considers the role of trustors when learning node representations, leading to a lack of effective information about trustees. The rationality of evaluation results derived from Medley is not well explained, resulting in poor explainability. Furthermore, model robustness was not addressed in this work.

Recentely, TrustGuard, a GNN-based model for robust and explainable trust evaluation was proposed [149]. TrustGuard addresses the limitations of existing trust evaluation models by considering attack resistance, dynamicity of trust, and explainability. It uses a layered

architecture with spatial and temporal aggregation layers to facilitate the robust aggregation of local trust relationships and capture temporal patterns from a sequence of snapshots. TrustGuard has been extensively tested and demonstrated superiority over state-of-the-art GNN-based trust evaluation models in terms of trust prediction across single-timeslot and multi-timeslot, with or without attacks. However, TrustGuard represents several potential limitations. One significant drawback is its reliance on the quality of the data it uses. If the data contains biases or inaccuracies regarding trust relationships, TrustGuard's predictions may not be reliable. Additionally, while TrustGuard provides visualizations to enhance explainability, understanding how it arrives at trust assessments can still be challenging for users. Moreover, the computational costs of training TrustGuard can be high, especially for large trust networks, which could limit its scalability and practical use. Furthermore, TrustGuard may struggle to generalize well to different trust networks with unique characteristics.

All these GNN-based RTMS fail to consider multi-dimensional edge features in their trust calculations, nor do they differentiate between the directionality of edges in the graph. This results in a lack of nuanced understanding and representation of trust dynamics within the network. Without considering multi-dimensional edge features, the models may overlook important factors influencing trust relationships, leading to potentially inaccurate trust assessments. Similarly, ignoring edge directionality limits the models' ability to capture the asymmetrical nature of trust interactions, where the direction of trust may differ between nodes. Consequently, the failure to account for these aspects limits the overall effectiveness and reliability of GNN-based RTMS in accurately capturing and predicting trust dynamics in complex networks. Furthermore, all the proposed RTMS are use-case designed, while GBTrust is generic and can be applied to different use cases. This flexibility allows GBTrust to adapt to various trust management scenarios, offering a versatile solution that addresses the diverse needs of different applications (4.1).

Table 4.1: Comparison of GBTrust with existing models.

″✓″: satisfy a criterion; ″×″: do not satisfy a criterion; ″-″: not available.

Model	Asymmetry	Dynamicity	weighted multi-dimensional edge features	Robustness	domain application
Guardian [143]	×	-	×	×	social network
GATrust [144]	×	-	×	×	(OSNs)
TrustGNN [145]	×	-	×	×	social network
Medley [147]	×	×	×	×	social network
TrustGuard [149]	✓	✓	×	✓	-
GBTrust	✓	✓	✓	✓	Generic

4.4 Graph Attentive Network Model for trust management

This section provides a detailed architectural description of the proposed GBTrust model. The model is designed to enhance trust management in P2P networks by effectively capturing

the importance of different neighbors and edge features within the network graph.

4.4.1 GBTrust : Methodology

GBTrust leverages information generated by BTrust [93] or other RTMS to effectively capture the complex dynamics of the peer-to-peer network. We begin by modeling the network as a graph G , where each node represents a peer within the network. These nodes are characterized by a set of features that capture different attributes of the corresponding peer, generated by running these protocols (BTrust or other), including transaction history, reliability, and interactions with other peers (see Table 4.2). On the other hand, the edges connecting these nodes in the graph carry local features that provide additional information about the nature and quality of peer interactions, including the direction of the interactions. These edge features encompass parameters such as the frequency of interactions, the reliability of past transactions, and other variables reflecting the quality and trustworthiness of the connections between peers. By considering node and edge features and the direction of interactions, the GBTrust model can effectively capture the complex dynamics of the peer-to-peer network and enable a more nuanced approach to trust management. This comprehensive representation enhances the model's ability to assess and manage trust in the network accurately, taking into account the directional nature of interactions between peers.

4.4.2 GBTrust graph Construction and attributes transformation

In GBTrust, we model the interactions between peers as a directed graph denoted as $G = (V, E)$, where $V = v_1, ..., v_n$ represents the set of nodes and $E = e_{ij}$ represents the set of edges. Each node in the graph corresponds to a peer, and the edges represent the connections between the peers.

The node feature information is represented as a matrix $X = x_1, x_2, ..., x_N$, where N is the number of nodes in the graph. Each x_i represents the feature vector of node v_i. These features capture various attributes of the peers, such as their transaction history, reliability, and interactions with other peers as indicated in this table 4.2.

In addition, we consider P as the number of edge features adopted in the graph. The edge features are represented by a tensor $E \in \mathbb{R}^{N \times N \times P}$, where E_{ijp} (where $1 \leq i, j \leq N$ and $1 \leq p \leq P$) denotes the value of the p-th edge feature between node v_i and node v_j.

Moreover, $E_{ij} \in \mathbb{R}^P$ represents the P-dimensional feature vector associated with the edge between node vi and node vj. Each dimension of this vector corresponds to the value of a specific edge feature. To describe the neighboring nodes of node vi in the graph G, we use $\mathcal{N}i$ to denote the set representing the neighbors of node vi. Similarly, $\mathcal{N}ip$ represents the set of neighbors of node vi with respect to the edge feature p in the graph G.

Since the graph G is directed, where direction represents an important factor, we plan to deconstruct the original graph G into two subgraphs, $G_{i_{in}}$ and $G_{i_{out}}$. The subgraph $G_{i_{in}}$

Table 4.2: Example of Node and Edge Features for GBTrust Model

Type	Feature	Description	Computation Method
Node	Historical BTrust Scores (or any other RTMS)	Trust scores of a peer in the past.	Extracted from historical transaction data.
Node	Transaction Frequency	The number of transactions a peer is involved in.	Counting transactions involving the peer in blockchain log.
Node	Average Transaction Value	The average value of transactions a peer is involved in.	Computed as the total value of transactions involving the peer divided by transaction frequency.
Node	Dispute Frequency	The frequency of disputes involving a peer.	Computed by counting disputes involving the peer.
Node	Global Trust	The aggregate trustworthiness of a peer considering its interactions with all other peers.	Calculated using the BTrust algorithm with input from all transactions involving the peer.
Node	Pertinence Ratio	The pertinence ratio (PR) is a quantitative metric employed to assess the relative relevance or appropriateness of a peer within a given network.	Computed by BTrust as global trust of a peer 'i' divided by its in-degree
Edge	Transaction Value	The value of a transaction between two peers.	Directly extracted from the transaction data.
Edge	Transaction Recency	How recent a transaction is between two peers.	Computed based on the timestamp of the transaction.
Edge	Transaction Frequency	The frequency of transactions between two peers.	Computed by counting transactions between the two peers.
Edge	Local Trust	The trustworthiness of a peer considering only its interactions with another specific peer.	Calculated using the BTrust algorithm with input from transactions between the two peers only.
Edge	Disputes	The number of disputes associated with the ratings provided by the client peer (j).	Extracted from BTrust log.

represents the incoming edges incident to vertex i and encompassing neighboring peers, while the subgraph $G_{i_{out}}$ includes the outer edges connecting to a vertex i and encompassing neighboring peers. By splitting the graph into these subgraphs, we can analyze the local interactions and relationships between the vertex i and its neighbors in each specific direction,

enabling a more detailed examination of the graph structure.

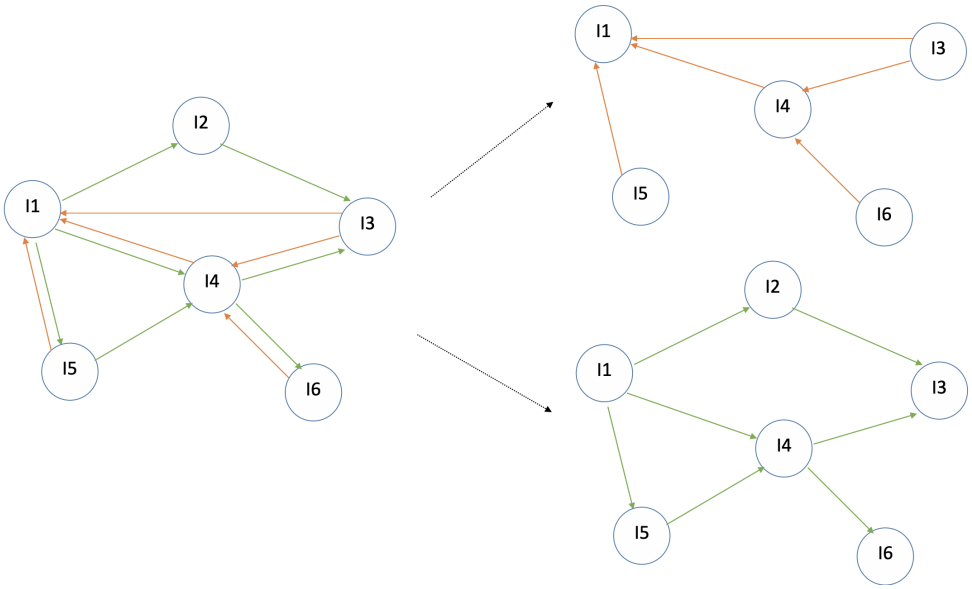


Figure 4.1: Transformation of the directed graph G into two sub graphs $G_{i1_{in}}$ and $G_{i1_{out}}$ for vertex i_1

4.4.3 Steps and overview

GBTrust constructs from the outputs of the first stage (trust scores and features) a trust graph G. For each peer, we deconstruct the graph G into $G_{i_{\text{in}}}$ and $G_{i_{\text{out}}}$. In each subgraph, we calculate the embeddings of every peer by considering both the peer features and the edge features. The embedding of a peer vi in the $G_{i_{\text{in}}}$ subgraph is denoted as $x_{i_{\text{Giin}}}$, while the embedding of the same peer in the $G_{i_{\text{out}}}$ subgraph is denoted as $x_{i_{\text{Giout}}}$. The embeddings

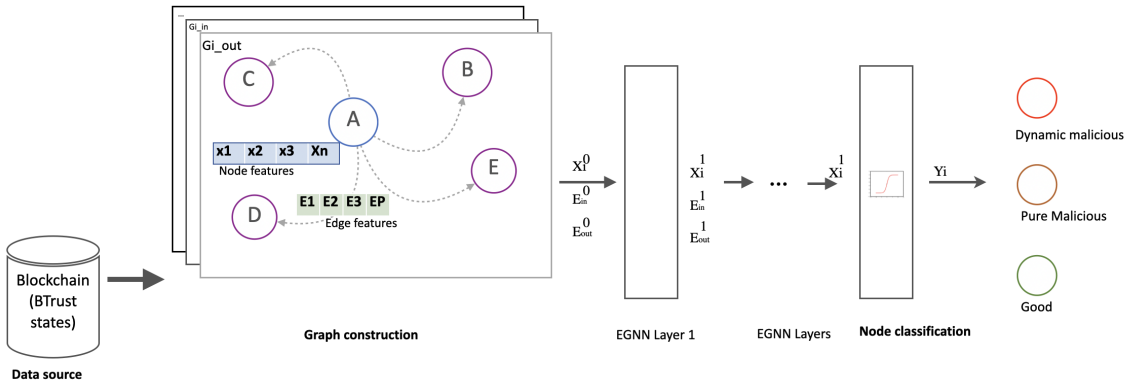


Figure 4.2: The GBTrust architecture.

are updated as follows:

$$x_{i_{\text{Gin}}} = f_{\text{in}}(i, x_i \mathcal{N}_{\text{i}_{\text{in}}}, E_{\text{i}_{\text{in}}}) \tag{4.1}$$

$$x_{i_{\text{Giout}}} = f_{\text{out}}(i, x_i\mathcal{N}_{\text{i}_{\text{out}}}, E_{\text{i}_{\text{out}}}) \tag{4.2}$$

where f_{in} and f_{out} are embedding functions that consider the embeddings of neighboring peers $x_i\mathcal{N}_{\text{i}_{\text{in}}}$ or $x_i\mathcal{N}_{\text{i}_{\text{out}}}$, and the edge features $E_{\text{i}_{\text{in}}}$ or $E_{\text{i}_{\text{out}}}$ within the corresponding subgraph.

The final embedding of each peer is obtained by concatenating the two representations:

$$x_i = concat(x_{i_{\text{Gin}}}, x_{i_{\text{Giout}}}) \tag{4.3}$$

This approach enables the incorporation of both the individual characteristics of the peers and the contextual information provided by the edges, resulting in a comprehensive and informative representation for each peer, denoted as xi. GBTrust uses a multilayer perceptron layer to predict trust relationships based on node pair embedding, i.e., the reputation trust classification layer.

4.4.4 Node and Edge Attention mechanism

To compute the embedding of each peer in graph G we proceed to compute them for both subgraphs then concatenate the results. As in EGNN proposed by [102] model, in GBTrust, edge features are used in attention mechanism. For each layer l, node vi representation vector $x_{iG'}^{(l)}$ is obtained from aggregation on its neighbor's node representation vectors, i.e. $\left\{x_j^{(l-1)}, j \in \mathcal{N}_i\right\}$ in subgraph G'($G_{i_{out}}$ or $G_{i_{out}}$). Therefore, for each neighboring node vj and each feature p, node representation vector $x_j^{(l-1)}$ and edge feature $E_{ijp}^{(l-1)}$ are incorporated.

Attention Vector Calculation

The attention vector $\hat{a}_{ijp}^{(l)}$, which represents the influence neighboring node vj makes on node vi for feature p at layer l is calculated as following:

$$\hat{a}_{ijp}^{(l)} = \exp\left\{\sigma\left[E_{ijp}^{(l-1)} v^T \left(W^{(l)} x_i^{(l-1)} \| W^{(l)} x_j^{(l-1)}\right)\right]\right\} \tag{4.4}$$

where σ is a non-linear activation, v is a parameter vector, $W^{(l)}$ is a parameter matrix, and $\|$ means concatenation operation.

The edge feature $E_{ijp}^{(l-1)}$ is combined with the concatenated node features $x_i^{(l-1)}$ and $x_j^{(l-1)}$, transformed by the weight matrix $W^{(l)}$

Normalized Attention and Node Feature Update

The attention is normalized by a doubly stochastic normalization function $DS(\cdot)$ and node feature and edge feature are both updated as shown below

$$x_{iG'}^{(l)} = \sigma \left\{ \|_{p=1}^{P} \sum_{j \in N} DS\left(\hat{a}_{ijp}^{(l)}\right) W^{(l)} x_j^{(l-1)} \right\} \tag{4.5}$$

$$E_{ijp}^{(l)} = DS\left(\hat{a}_{ijp}^{(l)}\right) \tag{4.6}$$

This equation updates the node feature $x_{iG'}^{(l)}$ by aggregating the features of neighboring nodes, weighted by the normalized attention coefficients $DS\left(\hat{a}_{ijp}^{(l)}\right)$, and applies an activation function . The edge features are similarly updated.

Alternative Node Feature Update

The following equation computes the updated node features $x_{iG'}^{(l)}$ for node i in graph G' at layer l.

$$x_{iG'}^{(l)} = \sigma \left(\sum_{p=1}^{P} \sum_{j \in N} \left[DS\left(\hat{a}_{ijp}^{(l)}\right) \cdot (W_1^{(l)} x_j^{(l-1)}) \right] \right) \tag{4.7}$$

where:

- $x_{iG'}^{(l)}$ represents the node features of node i in the graph G' at layer l.
- σ denotes an activation function.
- $\|_{p=1}^{P}$ represents a concatenation over different features p.
- $\sum_{j \in N}$ represents a summation operation over neighboring nodes j in the graph.
- $\hat{a}_{ijp}^{(l)}$ represents the attention coefficient between node i and node j for message type p at layer l.
- $W^{(l)}$ represents the weight matrix for layer l of the neural network.
- $x_j^{(l-1)}$ represents the node features of neighboring node j at layer $l-1$.
- $E_{ijp}^{(l)}$ represents the attention-softmaxed coefficients, scaled by D, for message type p between nodes i and j at layer l.

It aggregates messages from neighboring nodes, weighted by attention coefficients $\hat{a}_{ijp}^{(l)}$ and transformed by the weight matrix $W^{(l)}$, and then applies an activation function σ. The attention coefficients are normalized and computed using a softmax function S, and the aggregated messages are scaled by a normalization factor D. This approach incorporates edge features when calculating the attention coefficients. It begins by transforming the edge features into a higher-dimensional space. These transformed edge features are then used to compute the attention coefficients used in updating the node features. Thus this approach only requires one layer, as both the node and edge embeddings are updated simultaneously. The edge embeddings are determined based on the calculated attention coefficients. This

approach enables EGNN to capture relationships between nodes and edges, enhancing its ability to learn complex patterns within the graph structure. However, a drawback of EGNN is that, apart from the original edge features used in the first layer, the attention coefficients between nodes in the $l-th$ layer are used as the edge features in the $(l+1)-th$ layer. This can lead to a loss of important edge information. Additionally, EGNN does not explicitly consider the influence of edge feature directions on different edge features.

Enhanced Attention Mechanism

To overcome the previous limitations, we propose a new attention function that takes into account the influence of different edge features individually, as well as the direction of the edges. This allows us to discriminate between the directions of edges and capture the nuanced relationships within the graph. Additionally, we introduce a global normalization operation to determine the relative importance of each edge feature, further enhancing our understanding of their significance in the graph.

We first, introduce the edge-feature coefficient $\alpha_{ijp}^{(l)}$ for a directed graph in order to consider edge features in both directions. This coefficient is computed using the equation:

$$\alpha_{ijp}^{(l)} = \sigma\left\{\mathbf{u}_p^T\left[\mathbf{W}_1^{(l)}\mathbf{E}_{ijp}^{(l-1)} \| \mathbf{W}_1^{(l)}\mathbf{E}_{jip}^{(l-1)}\right]\right\} \tag{4.8}$$

where σ denotes the activation function, $\mathbf{u}_p$ represents the weight vector, $\mathbf{W}_1^{(l)}$ is the weight matrix, $\mathbf{E}_{ijp}^{(l-1)}$ represents the edge feature from node vi to node vj at layer $l-1$, and $\mathbf{E}_{jip}^{(l-1)}$ represents the edge feature from node vj to node vi at layer $l-1$. The equation computes the weighted sum of these edge features, followed by the application of the activation function to obtain the final coefficient value. The proposed edge-feature coefficient enables the consideration of bidirectional edge information, enhancing the modeling capability for directed graphs.

New Attention Coefficient Calculation

We compute a new attention coefficient by incorporating the previous edge embedding and the edge-feature coefficient.

$$\hat{\beta}_{ijp}^{(l)} = e_{ijp}^{(l-1)}\alpha_{ijp}^{(l)} \tag{4.9}$$

Substituting the expression for $\alpha_{ijp}^{(l)}$ from the first equation:

$$\hat{\beta}_{ijp}^{(l)} = e_{ijp}^{(l-1)}\sigma\left\{\mathbf{u}_p^T\left[\mathbf{W}_1^{(l)}\mathbf{E}_{ijp}^{(l-1)} \| \mathbf{W}_1^{(l)}\mathbf{E}_{jip}^{(l-1)}\right]\right\} \tag{4.10}$$

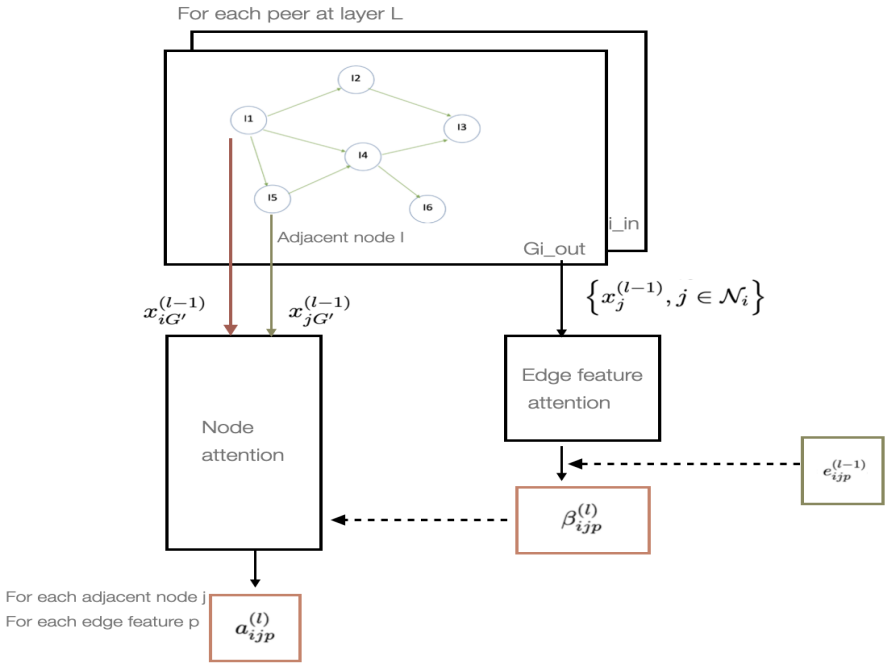


Figure 4.3: Construction of attention mechanism in GBTrust

This coefficient is then normalized via the $softmax$ function

$$\beta_{ijp}^{(l)} = softmax\left(\hat{\beta}_{ijp}^{(l)}\right) = \frac{\exp\left(\hat{\beta}_{ijp}^{(l)}\right)}{\sum_{k=1}^{P}\sum_{n=1}^{N}\exp\left(\hat{\beta}_{ink}^{(l)}\right)}, \tag{4.11}$$

we use the $Softmax$ function to measure the importance of edge feature p over all vi's neighbor nodes and all edge features. In this way, we can compare the result to know which edge feature is more important for node vi.

Final Attention Calculation

Considering the influence of edge features, and normalize it over all neighboring nodes. We calculate each neighboring node vj 's attention in terms of feature p $\hat{a}_{ijp}^{(l)}$, by using $\beta_{jip}^{(l)}$ in the exponent:

$$\hat{a}_{ijp}^{(l)} = \exp\left\{\sigma\left[\beta_{jip}^{(l)} v^T \left(W_2^{((l))} x_i^{(l-1)} \| W_2^{(l)} x_j^{(l-1)}\right)\right]\right\} \tag{4.12}$$

where $W_2^{(l)}$ is a parameter matrix and v is a parameter vector. In order to normalize the attention scores $a_{ijp}^{(l)}$ over all neighboring nodes of node i for feature p, we divide $\hat{a}_{ijp}^{(l)}$ by the sum of attention scores over all neighbors k:

$$a_{ijp}^{(l)} = \frac{\hat{a}_{ijp}^{(l)}}{\sum_{k \in N_p(i)} \hat{a}_{ikp}^{(l)}} \tag{4.13}$$

Finally, we get the following node vi 's representation vector $x_i^{(l)}$ in subgraph G'($G_{i_{out}}$ or $G_{i_{out}}$).

$$x_{iG'}^{(l)} = \sigma \left\{ \|\|_{p=1}^{P} \sum_{j \in N} a_{ijp}^{(l)} W_2^{(l)} x_j^{(l-1)} \right\} \tag{4.14}$$

then the global embedding of each peer x_i is obtained by concatenating its two representations (in both subgraphs $G_{i_{out}}$ and $G_{i_{out}}$) :

$$x_i^{(l)} = concat(x_{i_{\text{Giin}}}^{(l)}, x_{i_{\text{Giout}}}^{(l)}) \tag{4.15}$$

4.4.5 Classification Module

The classification module is used to predict the trustworthiness of each peer. We use two fully connected layers with specific activation functions to obtain these predictions. First, we apply the $Tanh$ activation function in the first fully connected layer. The $Tanh$ activation function maps the output values to the range [-1, 1], enabling the representation of both positive and negative trust values. This layer captures and processes the input features of each peer, transforming them into a meaningful representation.

Next, we employ the *Sigmoid* activation function in the second fully connected layer. The *Sigmoid* function maps the output values to the range [0, 1], which is suitable for representing trustworthiness probability. This layer refines the transformed representations from the previous layer and produces the final predicted trustworthiness values for each peer.

To train the trust network classification module, we define the loss function using the binary cross-entropy loss. The loss function penalizes the difference between the predicted trustworthiness ($\hat{y}_i$) and the actual trustworthiness (y_i) for each peer. The binary cross-entropy loss is given by:

$$\text{Loss} = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{4.16}$$

By minimizing this loss function during the training process enables the classification module to accurately predict the trustworthiness values for each peer, facilitating the detection of honest peers as well as distinguishing between static and dynamic malicious behavior within the trust network.

4.5 Performance Evaluation

In this section, we present an experimental evaluation to compare the performance of the proposed GBTrust model with existing GNN-based trust models. The experiments aim to assess the effectiveness of the GBTrust in trust management for P2P networks. We implemented the EGNN algorithm using the Python programming language and TensorFlow library. The models were trained on a machine equipped with a $GTX1080$ graphic card, which provided 8 GB of graphics memory. The model used $L = 2$ updated EGNN layers, indicating the depth of the model.

4.5.1 Experimental Setup

Dataset

We conducted experiments on a specially generated dataset. This dataset was created by simulating a network of 1000 peers running the BTrust algorithm, with peers labeled for evaluation purposes. The simulation was done using Netlogo 6.1.1 environment [1] which is a multi-agent modeling environment for emulating large-scale networks. During the simulation, each peer's interactions and trust ratings were recorded, providing a rich source of data for training and evaluating our model. The resulting dataset consists of 1000 nodes and 10000 directed edges. By leveraging this simulated dataset, we were able to assess the performance of our proposed model in accurately predicting the trustworthiness of peers in the context of file sharing. File sharing was chosen for its simplicity, its representation of resource exchange and its prevalent use as a case study in literature.

Baseline Models

We compare the GBTrust with three state-of-the-art GNN-based trust models: TrustGuard [149] , GATrust[144], TrustGNN [145] and EGNN [102]. These models represent existing approaches in trust management for P2P networks and serve as baselines for performance comparison.

4.5.2 Evaluation Metrics

To evaluate the performance of the trust models, we employed the following evaluation metrics:

- **Precision**: This metric measures the proportion of correctly classified trustworthy peers among all peers predicted as trustworthy. It indicates how accurate our model is in identifying trustworthy peers.
- **Recall**: The recall metric calculates the proportion of correctly classified trustworthy

[1] http://ccl.northwestern.edu/netlogo/

peers among all actual trustworthy peers. It assesses the model's ability to capture all trustworthy peers in the dataset.

- **F1-Score**: The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of performance by considering both precision and recall. A higher F1-score indicates a better balance between precision and recall.
- **Accuracy**: The accuracy metric measures the overall correctness of the predictions. It is calculated as the proportion of correctly classified peers, regardless of their trustworthiness. A higher accuracy score indicates a higher percentage of correct predictions.

In addition to these metrics, we also evaluated the performance of the models on the following tasks:

- **File Recommendation**: This task involves recommending files to each peer based on their past behavior, load, and shared neighbors. The goal is to assess the model's ability to effectively utilize the peer's historical data, consider their current workload, and leverage information from their neighboring peers to make accurate file recommendations. File recommendation serves as a prediction aiding peers in assessing whether the files provided by their counterparts are malicious or not, consequently influencing their decision to request the file or not from a given peer.

For these tasks, we compute precision, recall, F1-score, and accuracy as the evaluation metrics.

4.5.3 Experimental Design

We conduct a series of experiments to compare the performance of the GBTrust with TrustGuard, GATrust, EGNN and TrustGNN. Each experiment involves the following steps:

1. **Model Training**: We train each trust model using the training dataset and optimize their parameters using appropriate optimization algorithms such as gradient descent. The models are trained to learn the trust relationships, classify the trustworthiness of peers, and perform the additional tasks of peer classification and file recommendation.
2. **Model Validation**: We evaluate the performance of each model on the validation dataset and tune the hyperparameters, such as learning rate, batch size, and number of layers, to achieve optimal performance. This step ensures that the models are well-optimized and generalize well to unseen data.
3. **Model Testing**: We assess the performance of each model on the testing dataset, which contains unseen data. The evaluation metrics (precision, recall, F1-score, and accuracy) are calculated to quantify the effectiveness and robustness of each trust model. Additionally, the metrics for the peer classification and file recommendation

tasks are computed and analyzed.

Before training and testing, the dataset undergoes a random shuffling procedure to remove any biases or sequential patterns. This step aimed to promote fairness and impartiality in subsequent model training and testing phases. Following the shuffling process, the dataset was partitioned into three subsets: training, validation, and test sets. The allocation was structured to adhere to a well-balanced ratio of **70:15:15**, respectively. This distribution strategy was chosen to strike a harmonious balance between model training, validation, and evaluation, thereby facilitating effective learning and performance assessment.

4.5.4 Experimental Results

From table 4.3, several interesting phenomena can be observed. Overall, our approach 'GBTrust' outperforms three baselines in all metrics. Especially for Recall, it reaches to 0.81, much higher than any of the baseline model. 'GBTrust' had better performance which means that aggregating edge features to calculate node attention has strong influence on the prediction. Meanwhile, comparison of 'GBTrust' with 'EGNN' demonstrates the effectiveness of edge-feature directionality.

Table 4.3: Performance Comparison of Trust Models (Trust Assessment)

Model	Precision	Recall	F1-Score	Accuracy
GBTrust	0.83	0.81	**0.82**	0.86
TrustGuard	0.79	0.78	0.78	0.82
GATrust	0.80	0.79	0.80	0.84
TrustGNN	0.81	0.79	0.80	0.79
EGNN	0.81	0.80	0.78	0.81

Table 4.4: Performance Comparison of Trust Models (File Recommendation/transaction)

Model	Precision	Recall	F1-Score	Accuracy
GBTrust	0.79	0.78	**0.80**	0.81
TrustGuard	0.72	0.70	0.71	0.75
GATrust	0.75	0.73	0.74	0.77
TrustGNN	0.74	0.78	0.79	0.76
EGNN	0.78	0.76	0.78	0.80

4.5.5 Discussion

The experimental results clearly demonstrate that the GBTrust model excels in various evaluation metrics across multiple tasks, such as trust assessment, peer classification, and

file recommendation. Specifically, GBTrust consistently outperforms baseline models like TrustGuard, GATrust, EGNN, and TrustGNN in terms of precision, recall, F1-score, and accuracy. The highest recall achieved by GBTrust (0.81) underscores its superior capability in accurately identifying trustworthy peers, which is essential for maintaining the reliability and security of a peer-to-peer network.

In the trust assessment task, GBTrust's precision of 0.83 and F1-Score of 0.82 indicate a remarkable balance between identifying trustworthy peers and minimizing false positives. The accuracy of 0.86 further confirms its robust performance. Similarly, in the file recommendation task, GBTrust maintains a strong performance with a precision of 0.79, recall of 0.78, F1-Score of 0.80, and accuracy of 0.81, demonstrating its versatility and effectiveness across different scenarios.

Comparatively, other models like TrustGuard and GATrust show reasonable performance but fail to match GBTrust in key areas. TrustGuard's lower F1-Score and accuracy suggest difficulties in balancing precision and recall, potentially due to its simpler handling of peer interactions and edge features. GATrust, while leveraging graph attention mechanisms, does not fully capture the directional nature of interactions, leading to slightly lower performance metrics.

EGNN performs well with balanced precision and recall but lacks the detailed modeling of directed edges that GBTrust employs, which limits its accuracy in trust assessments. TrustGNN, despite solid performance, falls short due to its limitations in exploiting edge features and handling the dynamic nature of peer interactions.

The superior performance of GBTrust can be attributed to its advanced directed Edge Graph Neural Network architecture, which effectively captures the complex relationships and dynamics within P2P networks. This model's ability to integrate edge features and model the directionality of interactions provides a nuanced and detailed understanding of peer trustworthiness. Additionally, GBTrust's robust mechanism for dynamically adjusting trust scores ensures high accuracy and reliability, even as network conditions change.

The insights from this comparative analysis suggest that while other models have their strengths, they may not fully leverage the complexities of P2P networks as effectively as GBTrust. The detailed edge-feature aggregation and directed graph modeling in GBTrust set a new standard for trust management in decentralized systems.

4.6 Comparison to BTrust and traditional RTMS

In this section, we present the results of comparing GBTrust with BTrust and conventional RTMS, illustrating the influence of GNN on trust assessment.

As detailed in section 3.6.2, RMSE (root-mean-square error) serves as a reliable metric inversely linked to the accuracy of trust models, with lower RMSE values indicating higher

precision in trust evaluation. Following the simulation setup outlined in section 3.6.1, we evaluate identical scenarios under consistent conditions. Comparing RMSE values in both static and dynamic peer environments reveals that GBTrust significantly enhances the accuracy of trust assessments compared to BTrust, as depicted in Figures 4.4 and 4.5. This improvement can be attributed to GBTrust's capability to swiftly identify malicious peers across numerous interaction cycles, leveraging insights gleaned from BTrust interactions. Additionally, GBTrust incorporates supplementary parameters facilitating peer trustworthiness prediction based on network embeddings.

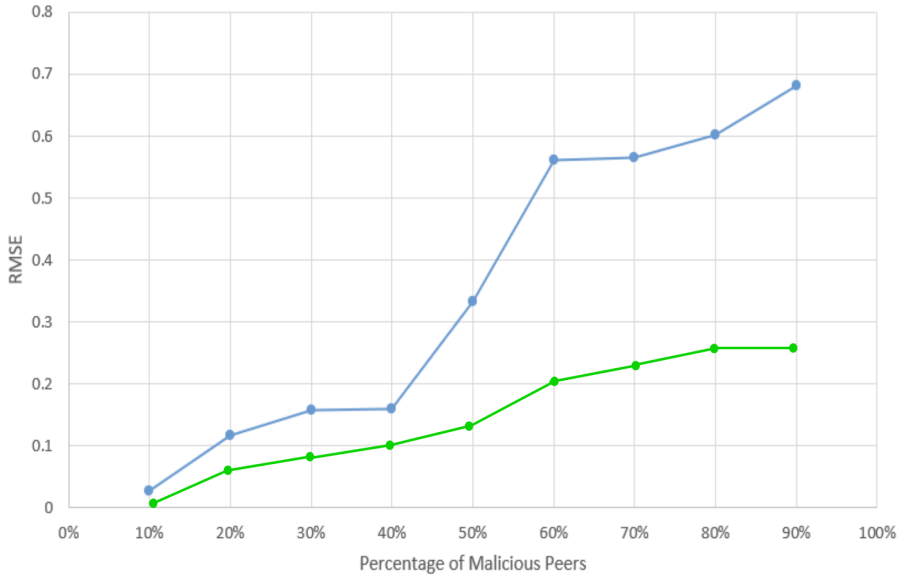


Figure 4.4: Trust estimation error (RMSE) GBTrust(green) and BTrust(blue) in presence of different portions of static malicious peers

To emphasize GBTrust's capacity to combat evolving malicious activity, we compare it with several established P2P RTMS platforms. In this comparison, we assess protocols based on two criteria: trust consistency and resilience to NMA, CRA, NRA, MSA, and CBA attacks [150]:

1. **Naive Malicious Attack (NMA):** A compromised node may offer improper services without accurately reporting its Local Trust Feedback (LTF), which is the feedback about a peer's behavior within a network service protocol.
2. **Collusive Rumor Attack (CRA):** Malicious nodes not only provide improper services but also collude to falsely report LTFs, aiming to disrupt trust or reputation assessment by spreading misleading information.
3. **Non-collusive Rumor Attack (NRA):** Malicious nodes act independently to report false LTFs contrary to observed evidence. For instance, if an LTF is assessed as positive (p), the malicious node may deceitfully report a negative

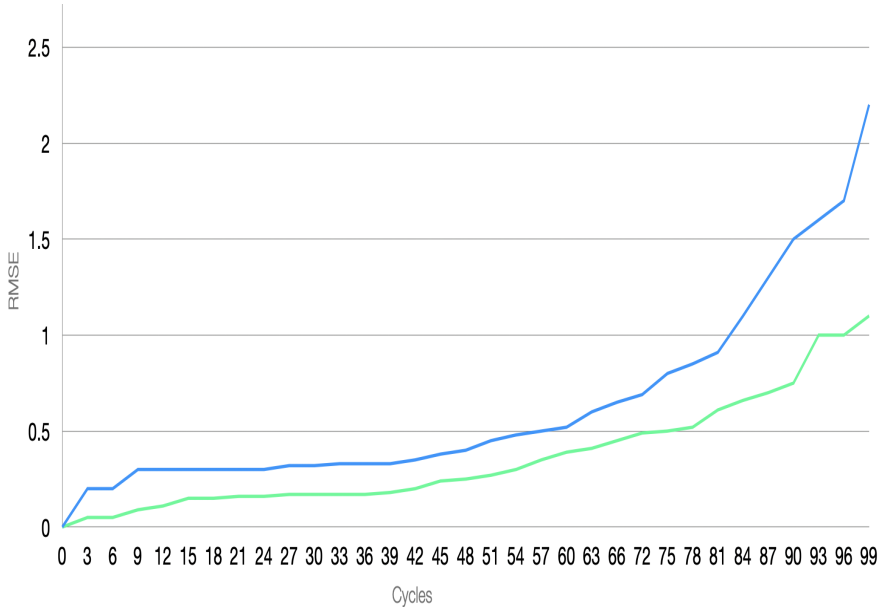


Figure 4.5: Trust estimation error (RMSE) average for GBTrust(green) and BTrust (blue) in presence of different portions of dynamic malicious peers, during 100 cycles

$(1-p)$ LTF.

4. **Malicious Spy Attack (MSA):** While some malicious nodes misbehave openly, others, known as malicious spies, behave normally to blend in. These nodes collaborate to spread misinformation by reporting false LTFs, thereby undermining the trust and reputation system.

5. **Conflicting Behavior Attack (CBA):** Malicious nodes adopt inconsistent behavior towards different parties, aiming to create confusion by disseminating conflicting LTFs. For example, they may target specific honest nodes with misbehavior to amplify the disparity in LTFs between targeted and non-targeted nodes, reducing the overall impact of the attack through intermittent misbehavior.

In network reputation assessment, consistency denotes the uniformity of evaluations across the network. In cooperative evaluations, consistency necessitates that a node's reputation remains consistent throughout the network. Conversely, in independent evaluations, reputations toward a node may vary based on how evaluators differentiate between direct and indirect observations. The analysis indicates that a robust RTMS like EigenTrust can partially resist CRA. This is achievable with the aid of pre-trusted nodes; nevertheless, for nodes that pre-trusted nodes have not interacted with or observed (i.e., high uncertainty), the reputation assessment could be significantly distorted.

Moreover, MSA presents a significant threat to EigenTrust for two primary reasons: Firstly, EigenTrust lacks the ability to detect spies, as their reputations are inflated with high values.

Protocol	Consistency	NMA	NRA	CRA	MSA	CBA
EigenTrust [22]	Yes	✓	✓	⋆	•	⋆
HonestPeer [22]	No	✓	✓	•	•	•
Robust [151]	No	✓	✓	✓	⋆	•
PeerTrust [127]	No	✓	✓	✓	⋆	✓
BTrust	Yes	✓	✓	✓	⋆	✓
GBTrust	Yes	✓	✓	✓	✓	✓

Table 4.5: Security Analysis of BTrust and other trust management protocols
Legend: ✓ - resilient; ⋆ - partially vulnerable; • - vulnerable

Consequently, false recommendations provided by spies are perceived as credible since the spy nodes exhibit no abnormal behavior apart from disseminating false information. Secondly, EigenTrust may be susceptible to CBA, particularly when pre-trusted nodes are deceived by malicious nodes displaying inconsistent behavior. While trust models like Robust and PowerTrust aim to counter collusion attacks effectively, they overlook the credibility of recommendations in reputation evaluation, thus failing to identify spies in MSA. Additionally, Robust is prone to CBA due to its inability to detect malicious nodes exhibiting inconsistent behavior, as honest nodes lack the capability to identify them.

In contrast, GBTrust employs an attention mechanism to filter out false recommendations effectively. This mechanism prioritizes relevant information while simultaneously screening out noise introduced by malicious actors. By assigning higher attention weights to trustworthy sources of feedback and reducing the influence of unreliable peers, GBTrust can effectively mitigate the impact of malicious spies on the reputation system.

4.7 Conclusion

Assessing the trustworthiness of peers in peer-to-peer (P2P) systems is a critical task within the domain of reputation trust systems. Existing approaches often overlook the utilization of graph information and lack interpretability in their models when tackling this challenge. Moreover, previous studies on graph attentive networks fail to properly account for the importance of neighbors and different edge features in the context of reputation trust systems.

To overcome these limitations, we propose GBTrust, a novel graph attentive network model specifically designed for reputation trust assessment in P2P systems. GBTrust incorporates an edge-feature attention mechanism to effectively handle the diverse characteristics of edges in the graph. Additionally, we introduce node attention and feature attention to leverage the directional aspects of the graph, enabling the identification of influential neighbors

and significant features within the reputation trust system. By combining these elements, GBTrust aims to provide an enhanced and comprehensive approach to reputation trust assessment in P2P systems.

Experimental evaluations demonstrate the superiority of our proposed model over existing approaches. Notably, our model achieves enhanced accuracy in trustworthiness detection and provides valuable insights and explanations. The effectiveness of GBTrust can be leveraged by traditional RTMS to enhance their performance and predictability. By integrating GBTrust as an overlay system, existing RTMS can benefit from the advanced capabilities of GBTrust to improve trust assessment outcomes and overall system reliability.

Chapter 5

Conclusion and future work

In this work, we have presented BTrust, a novel Reputation Trust Management System (RTMS) that combines multiple determinant factors for quantifying and comparing the trustworthiness of peers in P2P systems. BTrust dynamically incorporates a feedback system, user behavior analysis, and device security assessment to provide a comprehensive trust evaluation approach. We have also introduced an efficient peer selection method based on random walks, which proves to be suitable for large-scale networks without excessive computational burden.

BTrust presents several key strengths, including its extensibility and modularity. The generic nature of the underlying trust computation concepts allows for easy extension and the inclusion of additional services or factors to enhance the accuracy of trust evaluations. In addition, BTrust introduces a novel mechanism that distributes the load across the network, ensuring efficient and balanced trust management. Furthermore, an incentive mechanism is incorporated to encourage peers to provide trustworthy feedback and behave honestly. Moreover, the simulation results confirm that BTrust successfully fulfills the design considerations outlined in the introduction. It effectively isolates malicious peers within P2P networks, eliminating the need for a central authority to oversee trust management. The proposed trust model demonstrates its effectiveness and efficiency in countering various attacks, ensuring the robustness and security of the system. The extensibility, load distribution mechanism, and incentive mechanism of BTrust contribute to its effectiveness and applicability in P2P networks.

Moving forward, there are several avenues for further research and development. Firstly, we envision a decentralized evaluation of remote attestation, enhancing the security and reliability of trust assessments. Furthermore, we are investigating off-chain models for logging local trust scores, leveraging solutions like lightning networks to reduce validation latency and minimize blockchain overhead.

In our efforts to enhance BTrust, we have proposed GBTrust, a graph attentive network model designed specifically for reputation trust assessment and predictions in P2P systems. GBTrust addresses the limitations of existing similar approaches by incorporating an edge-feature attention mechanism, introducing node attention and feature attention, and leveraging

the directional aspects of edges. This mechanism allows GBTrust to capture and exploit the rich information embedded within the edges of the graph, resulting in more accurate and comprehensive trust evaluations. Moreover, GBTrust incorporates the concept of directionality in edges. By considering the direction of edges in P2P networks, GBTrust is able to capture asymmetry in interactions between peers. This consideration of directional edges enhances the model's ability to accurately assess trust relationships, capturing the nuances of trust dynamics in P2P systems. A noteworthy contribution is that GBTrust was designed to be used for any existing RTMS where trust data can be modeled as a directed graph. Experimental evaluations have demonstrated the performance of GBTrust over existing models in terms of trustworthiness detection accuracy and predictions.

Future work entails refining and optimizing GBTrust through several avenues. Firstly, the integration of time-aware GNNs can be explored to enhance the model's ability to capture temporal dynamics in trust relationships. By incorporating time as a factor in the GNN framework, GBTrust can provide more accurate and timely trust evaluations. Additionally, efforts can be focused on reducing the training and inference time of GBTrust. This can involve investigating techniques such as model compression, parallel computing, or distributed training to accelerate the learning process and improve efficiency.

Furthermore, expanding the evaluation of GBTrust on various datasets and real-world scenarios would provide a comprehensive understanding of its performance and generalizability. By testing the model on diverse domains and considering different network characteristics, the robustness and effectiveness of GBTrust can be further validated.

Bibliography

[1] Mahmood Hosseini, Constantinos Marios Angelopoulos, Wei Koong Chai, and Stephane Kundig. Crowdcloud: a crowdsourced system for cloud infrastructure. *Cluster Computing 2018 22:2*, 22:455–470, 8 2018.

[2] H Yao, T Mai, J Wang, Z Ji, C Jiang IEEE Transactions on . . . , and Undefined 2019. Resource trading in blockchain-based industrial internet of things. *ieeexplore.ieee.org*, 2019.

[3] JJ Sikorski, J Haughton, M Kraft Applied Energy, and Undefined 2017. Blockchain technology in the chemical industry: Machine-to-machine electricity market. *Elsevier*, 2017.

[4] Kiran Kumar Kondru, R. Saranya, and Annamma Chacko. A review of distributed supercomputing platforms using blockchain. *Lecture Notes in Networks and Systems*, 127:123–133, 2021.

[5] Ben Fisch, Joseph Bonneau, Nicola Greco, and Juan Benet. Scaling proof-of-replication for filecoin mining. 2018.

[6] Rajeev Singh, Sudeep Tanwar, and Teek Parval Sharma. Utilization of blockchain for mitigating the distributed denial of service attacks. *Security and Privacy*, 3:e96, 5 2020.

[7] Lydia Garms and Elizabeth A. Quaglia. A new approach to modelling centralised reputation systems. volume 11627 LNCS, pages 429–447. Springer Verlag, 2019.

[8] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing *, 2005.

[9] BELLINI EMANUEL, IRAQI youssef, and DAMIANI ERNESTO. Blockchain-based distributed trust and reputation management systems: A survey. *IEEE ACCESS*, 8, 2020.

[10] Diego Gambetta et al. Can we trust trust. *Trust: Making and breaking cooperative relations*, 13(2000):213–237, 2000.

[11] Pim Otte, Martijn de Vos, and Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems*, 107:770–780, 2020.

[12] Jun Zou, Bin Ye, Lie Qu, Yan Wang, Mehmet A Orgun, and Lei Li. A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services. *IEEE Transactions on Services Computing*, 12(3):429–445, 2018.

[13] Fan Chung. A brief survey of pagerank algorithms. *IEEE Trans. Netw. Sci. Eng.*, 1(1):38–42, 2014.

[14] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. pages 640–651, 2003.

[15] Anupam Das and Mohammad Mahfuzul Islam. Securedtrust: A dynamic trust computation model for secured communication in multiagent systems. *IEEE transactions on dependable and secure computing*, 9(2):261–274, 2011.

[16] Zhenquan Qin, Lei Wang, Mingchu Li, and Weifeng Sun. An efficient trust mechanism in p2p network. In *2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 731–734. IEEE, 2010.

[17] Quang-Vinh Dang and Claudia-Lavinia Ignat. dtrust: a simple deep learning approach for social recommendation. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 209–218. IEEE, 2017.

[18] Ugur Eray Tahta, Sevil Sen, and Ahmet Burak Can. Gentrust: A genetic trust management model for peer-to-peer systems. *Applied Soft Computing*, 34:693–704, 2015.

[19] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[20] ZHAO Yuhong and CHEN Jie. A p2p trust model based on trust factor and feedback aggregation. In *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, pages 214–219. IEEE, 2019.

[21] Ke Xuemeng, Zhou Guofu, and Du Zhoumin. Trust evaluation model for p2p networks based on time and interaction. In *MATEC Web of Conferences*, volume 208, page 05005. EDP Sciences, 2018.

[22] Heba A Kurdi. Honestpeer: An enhanced eigentrust algorithm for reputation management in p2p systems. *Journal of King Saud University - Computer and Information Sciences*, 27:315–322, 7 2015.

[23] Sarah Alkharji, Heba Kurdi, Rasha Altamimi, and Ebtesam Aloboud. Authenticpeer++: a trust management system for p2p networks. In *2017 European Modelling Symposium (EMS)*, pages 191–196. IEEE, 2017.

[24] Ruichuan Chen, Xuan Zhao, Liyong Tang, Jianbin Hu, and Zhong Chen. Cuboidtrust: a global reputation-based trust model in peer-to-peer networks. In *International Conference on Autonomic and Trusted Computing*, pages 203–215. Springer, 2007.

[25] Rachid Saadi, Jean-Marc Pierson, and Lionel Brunie. T2d: A peer to peer trust management system based on disposition to trust. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1472–1478, 2010.

[26] Xin Liu and Anwitaman Datta. Contextual trust aided enhancement of data availability in peer-to-peer backup storage systems. *Journal of Network and Systems Management*, 20:200–225, 2012.

[27] Alanoud Alhussain, Heba Kurdi, and Lina Altoaimy. Managing trust and detecting malicious groups in peer-to-peer iot networks. *Sensors*, 21(13):4484, 2021.

[28] Ayesha Altaf, Haider Abbas, and Faiza Iqbal. Context based trust formation using direct user-experience in the internet of things (iot). In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 424–430. IEEE, 2019.

[29] Ayesha Altaf, Haider Abbas, Faiza Iqbal, Farrukh Aslam Khan, Saddaf Rubab, and Abdelouahid Derhab. Context-oriented trust computation model for industrial internet of things. *Computers & Electrical Engineering*, 92:107123, 2021.

[30] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43, 2007.

[31] Omar Hasan, Lionel Brunie, and Elisa Bertino. Privacy-preserving reputation systems based on blockchain and other cryptographic building blocks: A survey. *ACM Computing Surveys (CSUR)*, 55(2):1–37, 2022.

[32] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

[33] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on parallel and distributed systems*, 18(4):460–473, 2007.

[34] Xin Liu, Anwitaman Datta, Krzysztof Rzadca, and Ee-Peng Lim. Stereotrust: a group based personalized trust model. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 7–16, 2009.

[35] Sascha Hauke, Sebastian Biedermann, Max Mühlhäuser, and Dominik Heider. On the application of supervised machine learning to trustworthiness assessment. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 525–534. IEEE, 2013.

[36] Chris Burnett, Timothy J Norman, and Katia Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[37] Raph Levien. Attack-resistant trust metrics. In *Computing with Social Trust*, pages 121–132. Springer, 2009.

[38] Raph Levien and Alex Aiken. Attack-resistant trust metrics for public key certification. In *Usenix security symposium*, pages 229–242, 1998.

[39] C-N Ziegler and Georg Lausen. Spreading activation models for trust propagation. In *IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004*, pages 83–97. IEEE, 2004.

[40] Giovanni Acampora, Daniyal Alghazzawi, Hani Hagras, and Autilia Vitiello. An interval type-2 fuzzy logic based framework for reputation management in peer-to-peer e-commerce. *Information Sciences*, 333:88–107, 2016.

[41] Jonathan Perez, Fevrier Valdez, Oscar Castillo, Patricia Melin, Claudia Gonzalez, and Gabriela Martinez. Interval type-2 fuzzy logic for dynamic parameter adaptation in the bat algorithm. *Soft Computing*, 21:667–685, 2017.

[42] Walayat Hussain, Farookh Khadeer Hussain, and Omar Khadeer Hussain. Maintaining trust in cloud computing through sla monitoring. In *Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part III 21*, pages 690–697. Springer, 2014.

[43] Yousef Elsheikh. A trust and reputation model for quality assessment of online content. *International Journal of Advanced Computer Science and Applications*, 8(3), 2017.

[44] Hady W Lauw, Ee-Peng Lim, and Ke Wang. Quality and leniency in online collaborative rating systems. *ACM Transactions on the Web (TWEB)*, 6(1):1–27, 2012.

[45] Hadeel T El Kassabi, Mohamed Adel Serhani, Rachida Dssouli, and Boualem Benatallah. A multi-dimensional trust model for processing big data over competing clouds. *Ieee access*, 6:39989–40007, 2018.

[46] Geoffrey Fox. Peer-to-peer networks. *Computing in Science & Engineering*, 3(3):75–77, 2001.

[47] Hongyong Huang, Haiyan Wu, and Guozheng Wang. Study of distributed p2p information sharing system. In *2009 Third International Symposium on Intelligent Information Technology Application*, volume 2, pages 300–303. IEEE, 2009.

[48] Hala Amin, Mohamed Khaled Chahine, and Gianluca Mazzini. P2p application for file sharing. In *2012 19th International Conference on Telecommunications (ICT)*, pages 1–4. IEEE, 2012.

[49] Hai Jiang, Jun Li, Zhongcheng Li, and Jing Liu. Efficient hierarchical content distribution using p2p technology. In *2008 16th IEEE International Conference on Networks*, pages 1–6. IEEE, 2008.

[50] Andrey V Luppov, Anton S Kudryavtsev, Dmitry A Marenkov, and Jury V Lanskikh. Improving the efficiency of p2p real-time communications networks. In *2013 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–2. IEEE, 2013.

[51] Francisco de Asís López-Fuentes and Gerardo García-Rodríguez. Collaborative cloud computing based on p2p networks. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 209–213. IEEE, 2016.

[52] Jan Skodzik, Peter Danielis, Vlado Altmann, Jens Rohrbeck, Dirk Timmermann, Thomas Bahls, and Daniel Duchow. Dude: A distributed computing system using a decentralized p2p environment. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 1048–1055. IEEE, 2011.

[53] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. pages 1–9, 2008.

[54] Chandranshu Gupta and Asmita Mahajan. Evaluation of proof-of-work consensus algorithm for blockchain networks. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2020.

[55] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[56] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19(1), 2012.

[57] Md Mainul Islam, Mpyana Mwamba Merlec, and Hoh Peter In. A comparative analysis of proof-of-authority consensus algorithms: Aura vs clique. In *2022 IEEE International Conference on Services Computing (SCC)*, pages 327–332. IEEE, 2022.

[58] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *Ieee Access*, 7:22328–22370, 2019.

[59] Hye-Yeong Shin, Meryam Essaid, Sejin Park, and Hongtaek Ju. A survey on public blockchain-based networks: structural differences and address clustering methods. In

2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), pages 57–60. IEEE, 2021.

[60] Mohammadreza Rasolroveicy and Marios Fokaefs. Performance and cost evaluation of public blockchain: An nft marketplace case study. In *2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 79–86. IEEE, 2022.

[61] Dharam Buddhi and Abhishek Joshi. Survey on analysis of blockchain technology. In *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, pages 880–885. IEEE, 2022.

[62] Hugo Figueiredo Caramelo Santos Martins. Exploring permissioned blockchains for decentralizing access control. 2018.

[63] Rodelio Arenas and Proceso Fernandez. Credenceledger: A permissioned blockchain for verifiable academic credentials. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–6, 2018.

[64] Shantanu Vidwans, Amogh Deshpande, Pratiksha Thakur, Aditya Verma, and Sushila Palwe. Permissioned blockchain voting system using hyperledger fabric. In *2022 International Conference on IoT and Blockchain Technology (ICIBT)*, pages 1–6, 2022.

[65] Baohua Huang, Huiying Zheng, Xi Qu, and Tongyi Xie. Consortium blockchain distributed storage protocol. In *2022 14th International Conference on Communication Software and Networks (ICCSN)*, pages 126–130, 2022.

[66] Junho Hong, Eunhwa Oh, Yeojin Yang, Kyungwoo Roh, Minji Oh, and Jaesung Kim. Consortium blockchain-based v2g energy trading system using tokens. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 677–682, 2020.

[67] N Szabo First Monday and undefined 1997. Formalizing and securing relationships on public networks. *firstmonday.org*, 1997.

[68] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. volume 99, pages 173–186, 1999.

[69] Diego Ongaro and John Ousterhout. *In Search of an Understandable Consensus Algorithm.*

[70] David Schwartz, Noah Youngs, and Arthur Britto. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5, 2014.

[71] Stellar. Intuitive stellar consensus protocol - developers blog.

[72] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. pages 31–42, 2016.

[73] I Eyal, AE Gencer, and EG Sirer. Bitcoin-ng: A scalable blockchain protocol. *13th USENIX Symposium*, 2016.

[74] P Jovanovic. Byzcoin: Securely scaling blockchains. *Hacking, Distributed, August*, 2016.

[75] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. pages 507–527. Springer, 2015.

[76] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[77] Ian Grigg. Eos-an introduction. *White paper. https://whitepaperdatabase. com/eos-whitepaper*, 2017.

[78] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. pages 357–388. Springer, 2017.

[79] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. volume 10821 LNCS, pages 66–98. Springer Verlag, 2018.

[80] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. pages 913–930, 2018.

[81] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros chronos: Permissionless clock synchronization via proof-of-stake, 2019.

[82] Ethan Buchman. Buchman, e. (2016). tendermint: Byzantine fault tolerance in the age of blockchains.tendermint: Byzantine fault tolerance in the age of blockchains, 2016.

[83] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. pages 51–68. Association for Computing Machinery, Inc, 10 2017.

[84] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. pages 3–33. Springer, 2018.

[85] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. pages 347–356, 2019.

[86] Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. State machine replication in the libra blockchain, 2019.

[87] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016.

[88] Serguei Popov. The tangle, 2017.

[89] Leemon Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance, 2016.

[90] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. pages 23–41. Springer, 2019.

[91] Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. Scalable and probabilistic leaderless bft consensus through metastability. 6 2019.

[92] Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi, and Abdellatif Mezrioui. Dcea: A reference model for distributed ledger technologies. *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2021*, 5 2021.

[93] Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noel Crespi, Abdellatif Mezrioui, and Khalid BELLAJ. Btrust : a new blockchain-based trust management protocol for resource sharing. *Journal of Networkand Systems Managemen*, 2022.

[94] Badr Bellaj, Aafaf Ouaddah, Emmanuel BERTIN, Noel Crespi, and Abdellatif Mezrioui. Drawing the boundaries between blockchain and blockchain-like systems : A comprehensive survey on distributed ledger technologies. *Proceeding of IEEE*, 2022.

[95] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[96] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[97] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[98] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[99] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[100] Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K Dey, Soham Mukher-

jee, Shreyas N Samaga, et al. Topological deep learning: Going beyond graph data. *arXiv preprint arXiv:2206.00606*, 2022.

[101] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[102] Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9211–9219, 2019.

[103] Danfeng Hong, Lianru Gao, Jing Yao, Bing Zhang, Antonio Plaza, and Jocelyn Chanussot. Graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 59(7):5966–5978, 2020.

[104] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[106] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34:249–270, 2018.

[107] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[108] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[109] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29:626–688, 2015.

[110] Richard Dennis and Gareth Owen. Rep on the block: A next generation reputation system based on the blockchain. *2015 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015*, pages 131–138, 2 2016.

[111] T McConaghy, R Marques, A Müller ... paper, undefined BigChainDB, and undefined 2016. Bigchaindb: a scalable blockchain database. *git.berlin*, 2016.

[112] Josef Gattermayer and Pavel Tvrdik. Blockchain-based multi-level scoring system for p2p clusters. *Proceedings of the International Conference on Parallel Processing Workshops*, pages 301–308, 9 2017.

[113] Jun Zou, Bin Ye, Lie Qu, Yan Wang, Mehmet A. Orgun, and Lei Li. A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services. *IEEE Transactions on Services Computing*, 12:429–445, 5 2019.

[114] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm (extended version). In *Proceeding of USENIX annual technical conference, USENIX ATC*, pages 19–20, 2014.

[115] Ittai Abraham, Danny Dolev, Rica Gonen, and Joe Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. pages 53–62, 2006.

[116] Yousef Alowayed, Marco Canini, Pedro Marcos, Marco Chiesa, and Marinho Barcellos. Picking a partner: A fair blockchain based scoring protocol for autonomous systems. *ANRW 2018 - Proceedings of the 2018 Applied Networking Research Workshop*, pages 33–39, 7 2018.

[117] Jens Neureither, Alexandra Dmitrienko, David Koisser, Ferdinand Brasser, and Ahmad Reza Sadeghi. Legiot: Ledgered trust management platform for iot. volume 12308 LNCS, pages 377–396. Springer Science and Business Media Deutschland GmbH, 2020.

[118] Atia Javaid, Maheen Zahid, Ishtiaq Ali, Jalees Ul, Hussen Khan, Zainib Noshad, and Nadeem Javaid. Reputation system for iot data monetization using blockchain. *Springer*, 97:173–184, 2020.

[119] Axel Moinet, Benoit Darties, and Jean Luc Baril. Blockchain based trust authentication for decentralized sensor networks, 6 2017.

[120] Roberto Di Pietro, Xavier Salleras UPF, Matteo Signorini, and Erez Waisbard. A blockchain-based trust system for the internet of things. *dl.acm.org*, pages 77–83, 6 2018.

[121] Mauro Conti, Edlira Dushku, Luigi V Mancini, Masoom Rabbani, and Silvio Ranise. Remote attestation as a service for iot. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 320–325. IEEE, 2019.

[122] Badr Bellaj, Aafaf Ouaddah, Noel Crespi, Abdellatif Mezrioui, and Emmanuel Bertin. A transpilation-based approach to writing secure access control smart contracts. In *2023 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–7. IEEE, 2023.

[123] Boyu Kuang, Anmin Fu, Willy Susilo, Shui Yu, and Yansong Gao. A survey of remote attestation in internet of things: Attacks, countermeasures, and prospects. *Computers & Security*, 112:102498, 2022.

[124] Ilker Yildirim. Bayesian inference: Metropolis-hastings sampling, 2012.

[125] KW Kwong, DHK Tsang IEEE/ACM transactions on, and undefined 2008. Building heterogeneous peer-to-peer networks: protocol and analysis. *ieeexplore.ieee.org*, 2008.

[126] Z. Banković, J. C. Vallejo, D. Fraga, and J. M. Moya. Detecting bad-mouthing attacks on reputation systems using self-organizing maps. volume 6694 LNCS, pages 9–16, 2011.

[127] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities nnexus view project deep learning on graph view project. *ieeexplore.ieee.org*, 2004.

[128] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18, 2007.

[129] Huanyu Zhao and Xiaolin Li. Vectortrust: trust vector aggregation scheme for trust management in peer-to-peer networks. *The Journal of Supercomputing*, 64:805–829, 2013.

[130] Jingwen Wang, Xuyang Jing, Zheng Yan, Yulong Fu, Witold Pedrycz, and Laurence T Yang. A survey on trust evaluation based on machine learning. *ACM Computing Surveys (CSUR)*, 53(5):1–36, 2020.

[131] Houda Amari, Zakaria Abou El Houda, Lyes Khoukhi, and Lamia Hadrich Belguith. Trust management in vehicular ad-hoc networks: Extensive survey. *Ieee Access*, 11:47659–47680, 2023.

[132] The bittorrent protocol what is bittorrent ?

[133] Zhiqi Li, Weidong Fang, Chunsheng Zhu, Zhiwei Gao, and Wuxiong Zhang. Ai-enabled trust in distributed networks. *IEEE Access*, 2023.

[134] Kashif Naseer Qureshi, Gwanggil Jeon, and Francesco Piccialli. Anomaly detection and trust authority in artificial intelligence and cloud computing. *Computer Networks*, 184:107647, 2021.

[135] Benedikt Veith, Dennis Krummacker, and Hans D Schotten. The road to trustworthy 6g: A survey on trust anchor technologies. *IEEE Open Journal of the Communications Society*, 4:581–595, 2023.

[136] Santa Agreste, Pasquale De Meo, Giacomo Fiumara, Giuseppe Piccione, Sebastiano Piccolo, Domenico Rosaci, Giuseppe ML Sarne, and Athanasios V Vasilakos. An empirical comparison of algorithms to find communities in directed graphs and their application in web data analytics. *IEEE transactions on big data*, 3(3):289–306, 2016.

[137] Sohan Gyawali, Yi Qian, and Rose Qingyang Hu. Deep reinforcement learning based dynamic reputation policy in 5g based vehicular communication networks. *IEEE Transactions on Vehicular Technology*, 70(6):6136–6146, 2021.

[138] Marios Thoma and Christoforos N Hadjicostis. Detection of collaborative misbehaviour in distributed cyber-attacks. *Computer Communications*, 174:28–41, 2021.

[139] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019.

[140] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[141] Davide Bacciu, Federico Errica, and Alessio Micheli. Contextual graph markov model: A deep and generative approach to graph processing. In *International conference on machine learning*, pages 294–303. PMLR, 2018.

[142] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[143] Wanyu Lin, Zhaolin Gao, and Baochun Li. Guardian: Evaluating trust in online social networks with graph convolutional networks. *Proceedings - IEEE INFOCOM*, 2020-July:914–923, 7 2020.

[144] Nan Jiang, Wen Jie, Jin Li, Ximeng Liu, and Di Jin. Gatrust: A multi-aspect graph attention network model for trust assessment in osns. *IEEE Transactions on Knowledge and Data Engineering*, 7 2022.

[145] Cuiying Huo, Di Jin, Chundong Liang, Dongxiao He, Tie Qiu, and Lingfei Wu. Trustgnn: Graph neural network based trust evaluation via learnable propagative and composable nature. 7 2022.

[146] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[147] Wanyu Lin and Baochun Li. Medley: Predicting social trust in time-varying online social networks. *Proceedings - IEEE INFOCOM*, 2021-May, 7 2021.

[148] Wanyu Lin, Zhaolin Gao, and Baochun Li. Guardian: Evaluating trust in online social networks with graph convolutional networks. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 914–923. IEEE, 2020.

[149] J Wang, Z Yan, J Lan, E Bertino, W Pedrycz arXiv preprint arXiv, and undefined 2023. Trustguard: Gnn-based robust and explainable trust evaluation with dynamicity support. *arxiv.org*, 2023.

[150] Eleni Koutrouli and Aphrodite Tsalgatidou. Taxonomy of attacks and defense mechanisms in p2p reputation systems—lessons for reputation system designers. *Computer Science Review*, 6(2-3):47–70, 2012.

[151] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for mobile ad-hoc networks. 2003.