Advancing Blockchain-based Reputation Systems : Enhancing Effectiveness, Privacy Preservation, and Scalability

Mouhamed Amine Bouchiha

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORALE EUCLIDE

LABORATOIRE L3i

THÈSE présentée par :

Mouhamed Amine BOUCHIHA

soutenue le : **18/07/2024**

pour obtenir le grade de : **Docteur de La Rochelle Université**

Discipline : **Informatique et Applications**

Vers des systèmes de réputation basés blockchain améliorés: efficacité, protection de la vie privée et évolutivité.

Rapporteurs	Abdelhakim Hafid	Professeur des universités	Université de Montréal
	Maria Cristina Onete	Maîtresse de conférences HDR	Université de Limoges
Examinateurs	Leila Merghem-Boulahia	Professeure des universités	Université de Troyes
	Maria Potop-Butucaru	Professeure des universités	Sorbonne Université
	Pascal Estraillier (Président)	Professeur des universités	La Rochelle Université
	Raja Jurdak	Professeur des universités	Queensland Université
Invité	Yacine Ghamri-Doudane	Professeur des universités	La Rochelle Université
Direction	Mourad Rabah	Maître de conférences	La Rochelle Université
	Ronan Champagnat	Maître de conférences HDR	La Rochelle Université

Thèse réalisée au Laboratoire L3i
Faculté des Sciences et Technologies
Avenue Michel Crépeau
17042 La Rochelle cedex 01

Tel: +33 5 46 45 82 62
Web: http://l3i.univ-larochelle.fr

Sous la direction de

Mourad Rabah	Maitre de conférences
Ronan Champagnat	Maitre de conférences HDR

Financement Allocation de recherche de la région Nouvelle-Aquitaine

Résumé

La gestion décentralisée de la réputation connait une demande croissante et importante ces dernières années. Ceci a conduit à la mise en place de nouvelles solutions sous la forme de systèmes de réputation basés sur la blockchain (BRSs). Un BRS étant tout simplement un système de réputation qui fonctionne sur un réseau blockchain. Son introduction répond au besoin d'une gestion transparente, immuable et décentralisée de la réputation qui contribue à promouvoir la confiance, la crédibilité et la responsabilité. Cependant, les performances et la sécurité des BRSs actuels sont limitées et empêchent leur adoption à grande échelle. En particulier, les BRSs existants sont peu performants en cas de charge accrue et peuvent perdre des transactions (c.-à-d. qu'ils ne supportent pas des taux de transaction très élevés). Ils sont également très limités dans leur capacité à protéger la vie privée des utilisateurs (au sens de la divulgation de données et d'activités personnelles) sans compromettre l'évolutivité de la blockchain sous-jacente. En outre, les défis associés à la mise en œuvre de cadres de réputation basés sur la blockchain dans le monde réel n'ont pas fait l'objet d'un examen approfondi. En particulier, les solutions existantes ne parviennent pas à relever de manière adéquate le défi de l'incorporation efficace de données du monde réel dans une blockchain pour la gestion de la réputation. Ceci est dû au fait que les blockchains traitent principalement des informations qui sont propres au réseau.

Dans cette thèse, nous présentons plusieurs contributions visant à améliorer à la fois les l'éfficacité, la protection de la vie privée mais aussi les performances des BRSs afin d'élargir leur adoption. Pour améliorer l'éfficacité et les performances des BRSs, nous présentons tout d'abord GuRuMarket. GuRuMarket est un BRS qui introduit la gestion de la réputation à la fois au niveau applicatif (pour des interactions au monde réel) ainsi qu'au niveau consensus. Il favorise la confiance et la responsabilité entre les entités grâce à une gestion transparente de la réputation et des garanties sur la chaîne. Il améliore l'évolutivité et l'équité de la blockchain sous-jacente grâce à un protocol de consensus léger appelé preuve de garantie et de réputation (PoGR).

Pour répondre aux problèmes de la protection de la vie privée dans les BRSs actuels, nous présentons DARS, un système de réputation anonyme décentralisé. Ce système permet aux utilisateurs et aux entités d'utiliser plusieurs pseudonymes dans leurs interactions, protégeant ainsi leur véritable identité tout en maintenant leur réputation exacte et à jour. Pour ce faire, DARS utilise des preuves zkSNARK sur deux grands livres différents, séparant ainsi la gestion de l'identité des opérations commerciales.

Le consensus PoGR, proposé dans GuRuMarket comme technique de mise à l'échelle de la couche 1, n'est pas suffisant pour assurer à la fois la gestion de la réputation et le volume de transactions commerciales. C'est pourquoi nous présentons RollupTheCrowd, une extension des BRSs proposés dans GuRuMarket ainsi que DARS. Le cadre proposé exploite les zkRollups en tant que technique de mise à l'échelle de la couche 2 pour améliorer l'efficacité et les performances et réduire les coûts. RollupTheCrowd présente un cadre pour une application commune des BRSs, qui est le crowdsourcing. Il introduit un modèle de réputation efficace et respectant la vie privée qui mesure la crédibilité des participants en évaluant leurs interactions dans le cadre du crowdsourcing.

Poursuivant nos efforts pour élargir l'adoption des BRSs, nous explorons son utilisation dans un domaine émergent, l'évaluation des modèles large du language (LLMs). Par conséquent, pour une évaluation transparente et éfficace des LLMs, nous présentons LLMChain. LLMChain présente un nouveau cadre basé sur la blockchain qui permet aux fournisseurs de LLM de partager l'accès à leurs modèles. Les utilisateurs possédant des compétences différentes peuvent alors interagir avec ces LLMs et fournir un retour d'information. Comme cette évaluation humaine dépend de la volonté des utilisateurs de fournir un retour d'information, et afin d'assurer la continuité de l'évaluation, nous utilisons une approche automatique supplémentaire mais efficace. Celle-ci montre une bonne corrélation avec l'évaluation humaine.

This thesis is dedicated to my father, who left us 15 years ago, driven by a singular ambition: to witness my success. I aspire to make him proud. To my mother, my brother, and my sister, whose boundless love has sustained me in difficult times.

Advancing Blockchain-based Reputation Systems: Enhancing Effectiveness, Privacy Preservation, and Scalability

Abstract

Decentralized reputation management has seen significant growth in recent years, leading to new solutions known as blockchain-based reputation systems (BRSs). A BRS is a reputation system that runs on top of a blockchain network. Its introduction addresses the need for transparent, immutable, and decentralized reputation management that helps promote trust, credibility, and accountability. However, the performance and privacy protection of current BRSs are limited and prevent their widespread adoption. Specifically, existing BRSs perform poorly under increased load and may lose transactions. They are also severely limited in their ability to handle user privacy without compromising the scalability of the underlying blockchain. Furthermore, the challenges associated with implementing blockchain-based reputation frameworks in the real world have not been thoroughly examined. In particular, existing solutions fail to adequately address the challenge of effectively incorporating real-world data into a blockchain for reputation management. This is because blockchains mainly process information that is native to the network.

In this thesis, we present several contributions aimed at improving the effectiveness, privacy protection, and performance of BRSs in order to broaden their adoption. To enhance the effectiveness and performance of BRSs, we first introduce GuRuMarket. GuRuMarket is a BRS that introduces reputation management at both the application and consensus layers. It promotes trust and accountability among entities trading services among each other through transparent on-chain reputation and guarantee management. It enhances the scalability and fairness of the underlying blockchain with a lightweight consensus protocol called Proof-of-Guarantee&Reputation (PoGR).

To address privacy concerns in current BRSs, we present DARS a Decentralized Anonymous Reputation System. DARS allows users and entities to use multiple pseudonyms in their interactions, protecting their true identities while keeping their reputations accurate and up-to-date. To ensure this, DARS uses zkSNARK's proofs on two different ledgers, separating identity management from business operations.

PoGR, proposed in GuRuMarket as a Layer-1 scaling technique, is not sufficient to serve both reputation and business workloads. Therefore, we present RollupTheCrowd, an extension to the BRSs proposed in GuRuMarket and DARS. This one leverages zkRollups as a Layer-2 scaling technique for improved performance and reduced cost. RollupTheCrowd presents a framework for a common application of BRSs which is crowdsourcing. It introduces an effective and privacy-aware reputation model that measures the trustworthiness of participants by evaluating their crowdsourcing interactions.

Continuing our efforts to broaden the adoption of BRSs, we explore their use in an emerging area, namely the evaluation of large language models (LLMs). Therefore, for a transparent and effective evaluation of LLMs, we present LLMChain. LLMChain introduces a new blockchain-based framework that enables LLM providers to share access to their models. Users with different expertise can therefore interact with these LLMs and provide feedback information for human evaluation. As this evaluation depends on users' willingness to provide feedback and to ensure the continuity of LLM behavior monitoring, we employ an additional but effective automatic approach. This one demonstrates a good correlation with human evaluation.

Author Publications

The research findings presented in this dissertation have been published/submitted in the five publications listed below:

Journals

- M. A. Bouchiha, Y. Ghamri-Doudane, M. Rabah, R. Champagnat, and G. Bailly-Maitre, "GuRuMarket: Fostering Trust in Blockchain-Enabled Real-World Marketplaces via Guarantee and Reputation," *Under Review*, 2024.

Conferences

- M. A. Bouchiha, Y. Ghamri-Doudane, M. Rabah, and R. Champagnat, "GuRuChain: Guarantee and Reputation-based Blockchain Service Trading Platform," in *2023 IFIP Networking Conference (IFIP Networking)*, Barcelona, Spain, 2023, pp. 1–9.
- M. A. Bouchiha, Y. Ghamri-Doudane, M. Rabah, and R. Champagnat, "DARS: Empowering Trust in Blockchain-Based Real-World Applications with a Decentralized Anonymous Reputation System," in *Advanced Information Networking and Applications (AINA)*. Kitakyushu, Japan: Springer Nature Switzerland, 2024, pp. 48–61.
- A. M. R. Bendada, M. A. Bouchiha, Y. Ghamri-Doudane, and M. Rabah, "RollupTheCrowd: Leveraging ZkRollups for a Scalable and Privacy-Preserving Reputation-based Crowdsourcing Platform," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)* Osaka, Japan. IEEE, 2024.
- M. A. Bouchiha, Q. Telnoff, S. Bakkali, R. Champagnat, M. Rabah, M. Coustaty, and Y. Ghamri-Doudane, "LLMChain: Blockchain-based Reputation System for Sharing and Evaluating Large Language Models," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)* Osaka, Japan. IEEE, 2024.

Patents

The results of GuRuMarket, in particular the proposed *Proof-of-Guarantee&Reputation (PoGR)* consensus, have led to the filing of:

- A priority patent application referenced in France with the following specifications: Filing date: 03/02/2023 - Application number: FR2301038
- An international PCT application referenced as follows: Filing date: 31/01/2024 - Application number: PCT/FR2024/050129

Acknowledgements

Throughout my 40-month Ph.D. journey, many people have provided invaluable assistance, encouragement, and belief in my abilities. I want to express my sincere gratitude to each of you. I truly appreciate your unwavering support, without which I would not have reached this point.

Firstly, I would like to thank my supervisors, Ronan and Mourad. I am extremely grateful for your patience, understanding, and most importantly, your belief in me. You have been kind and patient with me throughout this journey and have always been a source of support. Your scientific approach with insightful suggestions and detailed feedback throughout the research and writing of this thesis. It has been a great honor to be your Ph.D. student.

Aside from being my advisor, I am deeply indebted to Yacine for his unwavering belief in me and his invaluable scientific insights. His attentive guidance and tireless counsel have left an indelible mark on my journey. Without his encouragement, I would not have considered submitting a patent application, particularly for my first research endeavor—a daunting prospect that, with his support, became a reality. Yacine consistently went above and beyond to provide technical support and financial assistance. Thanks to his steadfast support, I had the opportunity to engage with technical partners and participate in scientific conferences.

With heartfelt sincerity, I thank my colleagues Quentin, Souhail, and Mickael from L3i for their exceptional collaboration. Our work together on the emerging subject of LLM evaluation was not only a valuable experience but also a pleasant memory. I hope this marks just the beginning of a fruitful and enduring collaboration. I would also like to thank Mounsf, who did his internship with us and continues his journey as a research engineer in the lab. He has done a great job in a short time and with many technical challenges. I hope my journey encourages you to pursue a Ph.D. as well. I want to thank all my colleagues at L3i, especially Youcef and Muzzamil. Youcef for being there during my struggles and being the person I open up to. And Muzzamil, for all his technical support in building the L3iB4IoT platform.

I extend my deepest gratitude to my mother, my rock, my pillar of strength during these hard times. Despite facing serious health challenges after my first year, she never once complained or revealed her struggles, all so I could concentrate on my research. Dear Mom, your selflessness knows no bounds, and I am forever indebted to you. Finally, I would like to thank my brother and sister for shouldering responsibilities that were meant to be mine. I owe much of who I am to you and I love you very much.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

Contents

Reputation systems have long been a cornerstone of human interaction, serving as a means of assessing trustworthiness and reliability in various contexts. Dating back to early civilizations where word-of-mouth and personal recommendations held great importance, the concept of reputation has evolved significantly over time. With the advent of the Internet, online reputation systems emerged, enabling users to gauge the credibility of entities in the digital world [3]. From e-commerce platforms to social media networks, reputation systems have become indispensable tools for navigating the vast landscape of the Internet. Over the years, various frameworks and architectures incorporating reputation management have been proposed for several domains such as crowdsourcing [4], e-commerce [5], and supply chains [6]. However, traditional reputation systems such as eBay[1] and Airbnb[2]) are not without their shortcomings. Centralization, one of the main problems of these traditional systems, poses major concerns about trust [7]. Centralized entities hold immense power over reputation metrics, leading to issues of bias, manipulation, and censorship. Moreover, the lack of transparency in how reputation scores are calculated undermines the integrity of these systems, leaving users skeptical of their reliability. Lastly, privacy preservation remains a pressing concern, as centralized reputation systems often collect and store sensitive user data, raising fears of surveillance and misuse [1]. Even when cryptographic primitives are used to enhance privacy, vulnerabilities remain, casting doubt on the security of existing solutions [8]. These challenges underscore the need for a paradigm shift in how reputation is managed and verified.

The evolution of blockchain technology, particularly in Decentralized Applications

[1] https://www.ebay.fr/

[2] https://www.airbnb.fr/

(DApp) development, has led to the emergence of new reputation systems powered by decentralized networks [9–11]. The integration of blockchain with reputation systems aims to foster trust and accountability by harnessing blockchain's potential to provide crucial features such as transparency, immutability, and decentralization. Transparent reputation systems allow users to observe how their scores are updated by system operators, thus building trust among users. In addition, the immutability and tamper resistance in a reputation system assures users that their scores cannot be manipulated, which is critical to achieving greater accountability and trust. Furthermore, the decentralization achieved through consensus mechanisms allows network participants to collectively validate reputation data, eliminating the need for a central authority to maintain or manage reputations.

Essentially, a blockchain embodies the characteristics of a fully distributed state machine replication (SMR) protocol [12]. Hence, at a high level, a blockchain can be defined as a collection of machines executing user requests in a mutually agreed order to maintain a globally consistent state. To establish agreement on the execution order, machines must communicate, proposing and voting on batches of user requests arranged in a specific order (*i.e.*, blocks). Machines consent to execute a block once it attains a majority of votes, a process commonly known as distributed consensus. Since its inception as a decentralized payment system [13], blockchain has experienced widespread adoption over the years. Currently, over 9024 active cryptocurrencies are operating on blockchains, and more than 300 Million people worldwide use blockchain[3]. In 2014, Ethereum [14] introduced the capability to interact with code and execute functions on the blockchain through transactions. This innovation led to the development of applications that run on blockchain, known as Decentralized Applications (DApps). Specifically, a DApp comprises an interface (*e.g.*, web front-end) communicating via Remote Procedure Calls (RPC) with a back-end piece of code executing on the blockchain. The execution of the DApp's back-end code on the blockchain inherently decentralizes these applications, as there is no single authority controlling blockchain execution. The significant increase in blockchain users recently can be attributed to the popularity of DApps. Most of these DApps encompassed applications in Decentralized Finance (DeFi) (*e.g.*, Decentralized Exchanges), decentralized games (*e.g.*, Matr1x Fire), and Non-Fungible Tokens (NFTs) (*e.g.*, CryptoPunks) executed on blockchains.

As mentioned above, centralized reputation systems, exemplified by platforms such as eBay and Airbnb, lack essential features for robust, transparent, and fair operation. Blockchain and the broader Web3 ecosystem, meanwhile, are emerging as promising alternatives to centralized reputation-based applications [1, 7]. These decentralized solutions address the issues of data manipulation and content censorship by distributing control and decision-making across a network of participants. Blockchain aims to foster transparency, immutability, and a more democratic approach to reputation management. However, for a wider integration of Web3 into reputation systems, one must address existing performance and security challenges in these systems. In particular, blockchain-based reputation systems face scalability hurdles that hinder their ability

[3] https://www.demandsage.com/blockchain-statistics

to handle increasing workloads. In addition, vulnerability to multiple threats such as Sybil attacks, whitewashing attacks, and the validity of off-chain data undermine the security and effectiveness of these systems. Furthermore, privacy concerns within current blockchain-based reputation systems need to be addressed. This includes ensuring that users can interact and provide feedback without fear of being tracked or retaliated against. By addressing these challenges, the vision of decentralized and robust Web3-based reputation systems can be realized, fostering a more trustworthy and user-centric environment.

This thesis focuses on improving the effectiveness, scalability, and privacy protection aspects of blockchain-based reputation systems. In doing so, we aim to contribute to the broader use of these systems. Our work includes several innovative contributions to improve these aspects in Web3-powered reputation systems. The motivation and goal of this thesis can be thus summarized as follows:

Motivation: Widening the adoption of Blockchain-based Reputation Systems (BRSs)

Goal: Enhance Blockchain-based reputation systems effectiveness, privacy preservation, and scalability

Improving the effectiveness and performance of blockchain-based reputation systems (BRSs) by scaling the mainchain (Layer-1) expands their potential applications. However, while Layer-1 scaling is essential, it is still insufficient for large-scale deployment. Additionally, addressing privacy issues within BRSs is critical to their widespread adoption. However, just like effectiveness and transparency, ensuring privacy involves trade-offs that affect blockchain performance. Therefore, solving scalability challenges in BRSs to increase performance while ensuring privacy and maintaining effectiveness is essential for their broader application. Therefore, based on these motivations, we list in the following section the main objectives of this thesis.

1.1 Objectives

Objective 1. Enhance Blockchain-based Reputation Systems effectiveness and Layer-1 scalability

To update the ledger securely and efficiently, blockchain networks must employ a complex consensus mechanism to validate new blocks. Several consensus schemes have been proposed over the years. Competitive consensus such as *Proof-of-Work (PoW)* requires participants to perform an important amount of computation to compete for the right to create and add new blocks. *Proof-of-Stake (PoS)*, another competitive consensus, that uses auction results of virtual stakes held by participants to elect a winner who will have the right to produce the next block. PoS-like schemes grant more influence to participants with higher stakes or wealth, leading to a concentration of

power among a few stakeholders. *Byzantine Fault Tolerance/Practical Byzantine Fault Tolerance (BFT/PBFT)*, are cooperative consensuses that aim to reach agreement in an asynchronous environment with bounded message delays and less than one-third ($n/3$) of Byzantine nodes. However, BFT/PBFT and its variants support a small set of players known in advance and become very slow when their number exceeds a certain threshold [15]. Algorand [16] incorporates a Byzantine Agreement (BA$\star$) with *PoS* to build a scheme called *Pure-Proof-of-Stake* that aims to achieve decentralization and security without compromising scalability. Recently, numerous studies have investigated the integration of trust and reputation management at the consensus layer to enhance blockchain maintenance and overall scalability [17–19]. Additional initiatives have focused on leveraging blockchain to establish decentralized and transparent reputation systems for real-world applications [10, 20, 21]. However, only a few have explored the simultaneous integration of application trust and consensus trust management. Specifically, implementing a reputation system on top of a blockchain network introduces significant computational overhead, leading to accelerated blockchain congestion. Therefore, the effectiveness of the reputation system is constrained by the inherent performance limitations of the underlying blockchain. Thus, it is imperative to improve blockchain performance to efficiently handle both business and reputation workloads. Hence, our first goal of improving the effectiveness and scalability of BRSs encapsulates the following sub-goals that enable it to support real-world applications.

1.1. *Design and develop a secure, effective, and Layer-1 (L1) scalable BRS.*

1.2. *Assess the security aspects of the proposed solution.*

1.3. *Evaluate the proposed framework on a local blockchain to prove its feasibility and observe its limitations*

Objective 2. Enhance BRSs privacy preservation

Reputation systems aim to build trust, ensure credibility, and guarantee accountability. This typically requires gathering and assessing certain information (*e.g.*, feedback, proof of delivery, etc.) to evaluate interactions and display reputation scores. However, this process often raises privacy concerns due to the personal nature of this information. In addition, users in reputation systems may hesitate to interact with each other and offer feedback due to the fear of potential retaliation [1]. A straightforward solution to this problem involves adopting feedback-independent reputation models [9]. However, the necessity of linking reputation scores or tokens to a single master key still causes concerns regarding potential tracking in current BRSs. A recent approach to address this issue involves the development of decentralized privacy-preserving reputation models, allowing users to interact and share feedback confidentially. The primary objective here is to offer robust reputation management without compromising user privacy. The use of cryptographic techniques with decentralized systems such as blockchain [11, 22] can help reputation systems guarantee privacy preservation and effectiveness simultaneously. However, existing solutions present some limitations, as they fall short of being entirely

decentralized since they depend either on a centralized entity or a group of trusted peers to handle identities, credentials, and security parameters. Additionally, despite the use of blockchain in numerous research efforts [23–25] to develop decentralized and privacy-centric reputation systems, the proposed solutions do not fully leverage its capabilities. Furthermore, the issues associated with the implementation of real-world BRSs, particularly the Oracle problem [26], have not undergone thorough examination. In particular, current solutions fail to adequately tackle the challenge of securely and privately incorporating real-world data into the blockchain for robust reputation management. Thus, our goal of improving BRS's privacy preservation encapsulates the following sub-objectives, which provide anonymity without compromising the reliability and effectiveness of the reputation model.

2.1. *Design and develop a secure, fully decentralized anonymous reputation system.*

2.2. *Demonstrate the security features provided by the system.*

2.3. *Evaluate the developed system on a running blockchain to prove its feasibility.*

Objective 3. Layer-2 scaling to achieve privacy preservation and effectiveness in BRSs

Current Blockchain-based reputation solutions for trust-related real-world applications fail to tackle the challenge of ensuring both efficiency and privacy without compromising the scalability of the blockchain [1]. L1 scaling techniques (*e.g.*, scaling the consensus) are essential as transaction finality depends on the state of the mainchain. However, relying solely on these techniques has proven insufficient for large-scale deployment [27]. Therefore, to support a large ecosystem of millions of users to broaden the adoption of BRSs, one needs to match the performance of current centralized applications. Additionally, there is a need to improve the performance of BRSs to achieve privacy preservation and effectiveness. As a result, continuing our efforts to improve BRSs scalability and performance, our third objective encapsulates the sub-objectives below that enable these systems to support both application and reputation workloads in a privacy-centric manner.

3.1. *Design and develop a secure and L2 scalable BRS.*

3.2. *Prove the security features provided by the system.*

3.3. *Evaluate the developed framework on a running blockchain to demonstrate its feasibility and performance.*

Objective 4. Examine our BRSs adaptability and extensibility

The previous chapters propose different frameworks to enable effective, privacy-preserving, and scalable blockchain-based reputation management. The designed solutions aim to foster trust and accountability through transparent yet privacy-preserving reputation management. Our primary objective in this chapter is to explore the extensibility of

the work proposed in these chapters and its adaptability to emerging usages. One such usage is the evaluation of Large Language Models (LLMs).

LLMs have witnessed rapid growth in emerging challenges and capabilities of language understanding, generation, and reasoning. Despite their remarkable performance in natural language processing-based applications, LLMs are susceptible to undesirable and erratic behaviors, encompassing hallucinations [28], unreliable reasoning [29], and the generation of harmful content [30]. These flawed behaviors undermine trust in LLMs and pose significant challenges to their adoption in large-scale real-world usages, such as legal assistance and medical diagnostics, where precision, reliability, and ethical considerations are paramount. All existing approaches for evaluating LLMs are centralized and thus lack essential features of transparency and immutability. Therefore, to continue our efforts to broaden the adoption of BRSs by demonstrating their ability to build trust in innovative real-world use cases, our final goal includes the following sub-goals that enable the integration of BRSs into an emerging application, namely the evaluation of LLMs.

4.1. *Examine the extensibility and adaptability of the frameworks developed in the previous chapters to new applications.*

4.2. *Design and develop a secure and transparent BRS to evaluate the credibility of LLMs.*

4.3. *Evaluate the proposed framework on a running blockchain to demonstrate its feasibility and performance.*

1.2 Contributions

In this dissertation, we present four novel contributions that are in line with the previously presented research objectives and our goal of improving BRSs effectiveness, privacy preservation, and scalability. First, we investigate the intersection between blockchain and decentralized reputation management and show why traditional blockchains cannot concurrently support both business and reputation workloads. Then, present an effective and L1 scalable blockchain that supports both workloads. Second, we present a novel decentralized anonymous reputation system that fosters trust in blockchain-based real-world applications. Third, introduce an L2 scalable variant of the initially presented BRS that improves the system's performance and allows for a large-scale cost-effective deployment. Finally, we present a novel BRS for evaluating LLMs credibility to demonstrate the extensibility and adaptability of our work.

In summary, our thesis presents the following contributions:

1. *GuRuMarket: Guarantee and Reputation-based Blockchain Marketplace* (Objective 1) - We explore the intersection between blockchain and decentralized reputation management and show why traditional blockchains are unable to support effective on-chain reputation management due to its additional workload (computational overhead). We then show the potential of introducing reputation scores at the consensus level

and its impact on the overall scalability. To support both business (Market DApp) and reputation workloads, we develop a provably secure reputation-based blockchain known as GuRuMarket [9,31]. GuRuMarket employs a novel consensus called *Proof-of-Guarantee&Reputation (PoGR)*. Unlike PoS-based blockchains, GuRuMarket (1) ensures a high degree of decentralization and fairness by preventing wealth centralization using reputation and guarantee and (2) enables effective and transparent on-chain reputation management (3) and fosters accountability and credibility among traders when exchanging real-world services via a robust incentive mechanism. The results obtained from running our solution on a local blockchain network demonstrate the feasibility, effectiveness, and scalability of GuRuMarket.

2. *DARS: Decentralized Anonymous Reputation System* (Objective 2) - To mitigate privacy concerns and fear of potential retaliation in BRS without compromising their security and effectiveness, we present a decentralized blockchain-based anonymous reputation system. In our framework, users can use different pseudonyms when interacting with each other allowing them to hide their digital identities. In DARS design, all pseudonyms of a specific user, yet, are cryptographically linked to the same access token, allowing honest users to maintain their reputation and preventing malicious ones from starting over [32]. This is achieved through the use of zkSNARK proofs for set membership via Merkle trees over commitments. We extended our framework with an efficient reputation model that respects all the security and privacy properties of our formal model. We build the proposed framework using emerging technologies (*e.g.*, Hyperledger Besu) and cryptographic tools (*e.g.*, Circom and Snarkjs libraries). The results of the evaluation of the proposed DARS on a local blockchain network demonstrate its feasibility and effectiveness.

3. *RollupTheCrowd: Leveraging ZkRollups for a Scalable and Privacy-Preserving Reputation-based Crowdsourcing Platform* (Objective 3) - To improve BRSs scalability while protecting user privacy, we present RollupTheCrowd. This novel blockchain-powered crowdsourcing framework leverages zkRollups to reduce gas costs[4] and improve performance [33]. This allows for concurrent crowdsourcing tasks and reputation update management. RollupTheCrowd introduces an effective and privacy-preserving reputation model that gauges workers' trustworthiness by assessing their crowdsourcing interactions. To alleviate the load on the mainchain, we employ an off-chain storage scheme, optimizing RollupTheCrowd's performance. To prove the feasibility of the proposed framework, we developed a proof-of-concept implementation using cutting-edge tools. Utilizing smart contracts and zero-knowledge proofs, our Rollup layer achieves a significant 20x reduction in gas consumption. Overall, our results demonstrate the effectiveness and scalability of RollupTheCrowd, validating its potential for real-world application scenarios.

4. *LLMChain: Blockchain-based Reputation System for Sharing and Evaluating Large Language Models* (Objective 4) - To effectively and transparently evaluate the cred-

[4]Each operation in a smart contract has a specific gas cost

ibility of LLMs, we design LLMChain, a decentralized blockchain-based reputation system that combines automatic evaluation with human feedback to assign contextual reputation scores that accurately reflect LLM's behavior [34]. LLMChain not only helps users and entities to identify the most trustworthy LLM for their specific needs but also provides LLM's developers with valuable information to refine and improve their models. To our knowledge, this represents the first work to introduce a blockchain-based distributed framework specifically designed for assessing LLMs. LLMChain showcases its efficacy and scalability in evaluating LLMs across two benchmark datasets.

1.3 Thesis Organization

Our thesis, which aims to improve the effectiveness, scalability, and privacy protection of blockchain-based reputation systems, is organized as follows:

Chapter 2 presents the background. It explains in detail the intersection between reputation and blockchain and their concepts, as well as some related work that forms the foundation of the concepts upon which this thesis builds.

Chapter 3 presents the first contribution of this thesis, GuRuMarket. GuRuMarket is a secure and efficient Layer-1 (L1) scalable blockchain marketplace based on guarantee and reputation. In GuRuMarket, we first analyze the existing literature at the intersection of blockchain and reputation management. We then describe the proposed framework that uses reputation and guarantee to promote accountability and trust. We then present the proposed lightweight consensus scheme, which allows GuRuMarket to achieve better scalability and fairness than existing protocols. Then, we analyze the security of GuRuMarket. Finally, we evaluate its performance and provide an analytical comparison with existing solutions.

Chapter 4 introduces a Distributed Anonymous Reputation System. DARS fosters trust by allowing users to use multiple pseudonyms in their interactions. In DARS, we first present related work on privacy preservation in decentralized reputation systems. We then discuss the threats model employed in our work. Then the proposed framework is explained before providing its security analysis. Finally, we evaluate DARS performance and present an analytical comparison with existing solutions.

Chapter 5 introduces RollUpTheCrowd, a Layer-2 (L2) scalable blockchain-based reputation system for crowdsourcing. In RollUpTheCrowd, we first analyze existing techniques for scaling blockchain-based reputation systems. We then detail the proposed design of RollUpTheCrowd. Next, we describe the proposed reputation model for evaluating crowdsourcing tasks. Finally, we present a detailed evaluation of the performance of RollUpTheCrowd.

Chapter 6 explores the adaptability of the models designed in the previous chapters by introducing LLMChain, a novel blockchain-based reputation system for sharing and evaluating Large Language Models (LLMs). In LLMChain, we first discuss related work on LLMs evaluation and present some blockchain-based reputation solutions. We then detail the proposed architecture of LLMChain. Next, we discuss the proposed

reputation model for evaluating LLMs behavior. Finally, we present a detailed evaluation of LLMChain regarding reputation model effectiveness and blockchain performance.

Chapter 2

Blockchain-based Trust & Reputation Management

Contents

In this chapter, we introduce the concepts of reputation management, the fundamentals of blockchain, and the concepts necessary to understand the content presented in the following chapters. More precisely, we first present the general context of reputation management and the role of blockchain in advancing these systems. Next, we discuss blockchain preliminaries and principles that define the core components of a blockchain.

The following section defines the concepts of decentralized, blockchain-based, reputation management. Finally, to provide a security background for the following chapters, we present common blockchain and reputation attacks.

2.1 Trust and Reputation Management

Reputation systems are essential for determining the trustworthiness or credibility of users/entities in environments lacking pre-established trust [1]. The reputation of a particular entity is often calculated by aggregating local reputations, usually termed subjective trust or direct opinion provided by other users. In other words, an entity's reputation is the collective perception of its conduct, formed through the trust established by other entities [3].

While manifestations of trust are readily recognizable in our daily experiences, trust is a complex concept with a definition that can be challenging, encompassing elements of ethics, morals, emotions, and values, and spanning various fields. Additionally, trust is inherently contextual. For example, an e-commerce seller may be trusted to sell a product but may not be trusted to perform a medical diagnosis [7]. Several definitions of trust have been introduced to the literature over the years. For instance, trust has been described as the *"willingness to rely on an exchange partner in whom one has confidence"* [35]; the *"willingness to place oneself in a relationship that establishes or increases vulnerability with the reliance upon someone or something to perform as expected"* [36]; the*"willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that party"* [37]; and *"the subjective probability by which an individual A, expects that another individual, B, performs a given action on which its welfare depends"* [38].

The primary goal of reputation systems is to hold users accountable for their actions [1]. The secondary goal of these systems is to provide insight into the credibility and trustworthiness of users and entities in the system. Achieving the second goal depends primarily on the presence of the first goal because, without accountability, reputation scores will not accurately reflect the behavior of users. Thus, we cannot achieve trust and credibility. We expect a reputation system to implement an effective and robust reputation model where users with good behavior are fairly rewarded and those with bad (incorrect or malicious) behavior are punished.

The two most prevalent applications of reputation systems are in E-commerce marketplaces and crowdsourcing platforms. Notable examples include platforms such as eBay[1], Fiverr[2], and Uber[3]. Additionally, academic approaches to reputation management in E-commerce, Crowdsourcing, and Internet of Things environments have been proposed [9, 21, 39–41]. Airbnb[4] stands out as a renowned online reputation-based mar-

[1] https://www.ebay.fr/
[2] https://www.fiverr.com/
[3] https://www.uber.com/
[4] https://www.airbnb.fr/

ketplace for vacation rentals. The platform allows independent hosts to offer their private accommodations to guests for short stays. The reputation system on Airbnb is pivotal, as guests seeking satisfactory accommodations rely solely on the reputation of hosts and their offerings, derived from reviews by previous guests. Similarly, hosts, concerned about lending their lodgings to well-behaved guests, also depend on the reputation system.

Reputation management can be approached in various ways, incorporating different parameters based on the specific use case and business logic. For instance, rating workers on platforms like Fiverr involves different considerations compared to rating hosts on Airbnb. Furthermore, the architecture of reputation systems can influence the design of the reputation model (*i.e.*, centralized versus decentralized models).

2.1.1 Centralized Reputation Systems (CRS)

A centralized reputation system relies on a central authority to act as a judge or assessor of trust. In this setup, a centrally controlled computation facility (*i.e.*, a server) is responsible for collecting and processing all ratings. This approach proves effective when a business case demands the involvement of a reputable third party (*e.g.*, eBay, Airbnb). While relying on a central institution can facilitate reputation evaluation and partially address the trust issue, it does not eliminate the underlying source of mistrust. Additionally, relying on an intermediary introduces several challenges, including a single point of failure, information asymmetry, compromised user privacy, lack of transparency, and most of all the necessity for the users to trust the central authority. Thus, the questions raised by Juvenal in his Satires "Quis custodiet ipsos custodes?" (Who watches the watchers?), persists [7].

2.1.2 Decentralized Reputation Systems (DRS)

Decentralized reputation management has been present nearly as long as Peer-to-Peer (P2P) systems themselves. Reputation systems in P2P networks serve diverse purposes, including selecting dependable resources, ensuring honest peer behavior, and rating the quality of shared data. Despite the advantages of a distributed approach, they encounter additional challenges associated with the decentralized nature of the system. These issues include maintaining accurate, up-to-date reputation scores over a dynamically changing network [7].

Although using a distributed method addresses many of the problems associated with traditional centralized approaches, it is vulnerable to attack and potential manipulation. These vulnerabilities can compromise user privacy and possibly the entire system's security, especially when reputation is used to select entities to perform some critical tasks (*e.g.*, selecting validators or initiating shards in a blockchain protocol). Therefore, it is crucial to strengthen the security of DRS with schemes capable of preventing or mitigating the short and long-term effects of reputation attacks. In this respect, Distributed Ledger Technologies (DLT), in particular blockchain, is emerging as a promising solution [1]. In particular, blockchain technology can offer a robust framework for building

trust using reputation systems that are transparent, secure, and resistant to manipulation, thereby fostering greater trust and accountability in various applications [7].

The rest of this chapter is organized as follows. Section 2.2 presents blockchain preliminaries, its consensus problem, Web3, and blockchain performance and scalability. Section 2.3 discusses the intersection between blockchain and reputation management. Section 2.4 presents common attacks in on-chain reputation management. Finally, Section 2.5 concludes this chapter.

2.2 Blockchain Context

A blockchain operates as a decentralized and distributed system with numerous validating nodes that communicate over a peer-to-peer network. Its primary functions include (1) establishing consensus on the order of transaction execution initiated by its clients/participants, (2) executing the agreed transactions, and (3) recording the completed transactions in blocks. Each block in this chain references its predecessor, forming the characteristic structure known as a *"Blockchain"*.

2.2.1 Blockchain Fundamentals

Over the past years various blockchain implementations have been proposed. Nevertheless, common concepts are prevalent in many blockchains. In this section, we explore these fundamental concepts that constitute a blockchain.

1. *Nodes.* A blockchain system is made up of machines called *nodes* or *peers*, such as PCs, servers, or other hardware devices that are connected over a network. Blockchain's *clients* refer to software components hosted by nodes, responsible for transmitting read and write requests across the network. Note that the term *"Client"* is used by the Ethereum community to designate Ethereum implementations, such as the Geth and Besu, which represent Ethereum's Golang and Java implementations, respectively [14].

   Validators, alternatively referred to as miners in certain blockchain implementations, are essential nodes that undertake various critical functions. These include (1) validating client write requests, (2) presenting validated client write requests in batches (blocks) to the network of validators, (3) determining the execution order of client write requests, and (4) executing write requests in total order. Note that validators can also act as clients, sending write or read requests to the network. According to their behavior, validators can be classified into two types, *Correct* and *Byzantine*. We refer to validators that follow the blockchain rules (*i.e.*, protocol) as *correct* validators. For instance, correct validators do not produce blocks with invalid transactions. Validators that deviate from the blockchain protocol are termed as *Byzantine* validators. The term *Byzantine* is derived from the Byzantine Generals' Problem [42], which is a classic computer science and distributed computing problem that explores the challenges of reaching consensus among a group of entities, referred to as generals, in the presence of faulty or traitorous participants.

2. *Account.* A blockchain account is a digital identity or wallet associated with a user/participant on a blockchain network. It enables users to submit transactions, to store digital assets, and to interact with decentralized applications (DApps) on the blockchain. The key components of a blockchain account are, (i) *Address.* a cryptographic hash derived from the public key, serving as the unique identifier for the account (*i.e.*, other users can send digital assets to this address), (ii) *Public Key.* The publicly shared part of the cryptographic key pair, used for verifying digital signatures, (iii) *Private Key.* the secret part of the cryptographic key pair, known only to the account owner, used for signing transactions and providing proof of ownership, and (iv) *Balance.* the amount of digital assets (cryptocurrency or tokens) associated with the account.

   Wallet-initiated requests necessitate a valid signature from the wallet's private key. Blockchain nodes exclusively process write requests after confirming the signature's validity, matching it with the corresponding public key of the sending wallet, and verifying the request's origin from the private key owner. Additionally, in the majority of blockchains [14,16,43], wallets are assigned a sequence number that increments each time the wallet owner submits write requests to the blockchain. This sequence number also referred to as a *nonce* in Ethereum blockchains, aids in the orderly execution of requests within the system. Specifically, if a wallet sends two requests, the one with the lower sequence number takes precedence and is executed before the request with the higher sequence number.

3. *Transactions.* They represent write requests initiated by blockchain clients and submitted to the network. In contemporary blockchains, transactions typically fall into three primary types: native payments, facilitating the transfer of funds between accounts; code deployments, involving the uploading of code to be executed by the blockchain; and code executions, which trigger functions within the uploaded code.

   A transaction within blockchains adheres to a distinct structure. Typically, a transaction includes key elements: (i) the sender's wallet address, representing the account initiating the transaction, (ii) the recipient's wallet address, indicating the destination for the funds, (iii) the sequence number (nonce), and (iv) the transferred funds amount in the case of a native payment transaction. Additionally, for specific transaction types, such as code deployment or code execution, the transaction may encompass the bytecode of the code to be uploaded or the function invocation bytecode, respectively.

4. *State Machine.* It functions as a virtual machine equipped with an instruction set, commonly referred to as a stack machine. It possesses the capability to execute transactions, uphold the blockchain state, and retain records of executed transactions. Upon executing a transaction, the state machine modifies the corresponding state mentioned in the transaction, such as the sender's wallet balance, the receiver's wallet balance, or the state of any associated code.

Ethereum Virtual Machine (EVM) is a decentralized, Turing-complete virtual machine. It serves as the runtime environment for executing code across an Ethereum network. Its diverse implementations and optimized versions find applications in several blockchains [14, 43]. The EVM uses a specialized tree, the Merkle Patricia Tree (MPT), to store the blockchain state. Upon the execution of a transaction, the EVM updates the MPT. The nodes of the MPT are stored in memory, with branches periodically flushed to disk when the MPT's size exceeds a specific threshold. This in-memory storage ensures swift access to the blockchain state for clients.

5. *Blocks.* A block in a blockchain is the integral component of the blockchain structure. It represents a collection/batch of transactions grouped and added to the blockchain in a sequential and chronological order. A block commonly consists of transactions, a hash that encapsulates the block's contents, the hash of the preceding block (referred to as the *parent block*), establishing a linkage to its precursor, an index (block number) denoting the position of the block within the sequence of blocks, and a timestamp indicating the time when the block was added to the blockchain.

   The initial block in a blockchain is termed the *genesis block*, setting the initial state during the blockchain's initialization. Unlike subsequent blocks, *genesis block* does not reference a prior block. Once validators reach a consensus on a block's position in the chain, it is deemed final or confirmed. Transactions within confirmed blocks are executed, leading to global state updates.

6. *Transaction Pool.* The terms "transaction pool" and "mempool" refer to the temporary storage location for transactions upon receipt from a client, before their incorporation into a block. Typically, the transaction pool is found on each validator node. A validator node assembles blocks from transactions within its transaction pool and disseminates these blocks to other validators [14].

7. *Transaction Model.* There are typically two main transaction models used in blockchains: the Unspent Transaction Output (UTXO) model and the Account-based model. Blockchains such as Bitcoin [13] and Redbelly [43] use the UTXO model, where each transaction creates a set of unspent outputs that can be used as inputs for future transactions. When a user spends a UTXO in a new transaction, it gets consumed and new UTXOs are created as outputs. In the account-based model, user account balances are maintained, and transactions involve updating those balances. Each user has an account with a balance and transactions transfer value between accounts. Ethereum primarily uses an account-based model, where each account has an associated balance and state. Transactions involve updating the balances and state of these accounts.

8. *Blockchain Types.* Three primary types of blockchains exist:

   - *Permissionless* blockchains have restrictions on nodes joining or leaving the network, allowing anyone to function as a client or validator. These blockchains,

termed public permissionless blockchains, are open to the public. Bitcoin [13] and Ethereum [14], the two largest blockchains, both fall under the category of public permissionless blockchains.

- *Open* blockchains permit any node to join or exit the network, though specific tasks, such as proposing blocks or determining transaction order, are confined to a fixed set of nodes. To transition an open blockchain to a permissionless state, validator rights can be periodically granted to nodes, enabling any network participant to become a validator. Algorand [16] employs this approach.

- *Permissioned* blockchains incorporate an access control layer. Any node granted permission can access and join the network, typically limited to an exclusive set of nodes. Unlike permissionless blockchains, permissioned blockchains operate with a restricted number of nodes. Examples of permissioned blockchains include Hyperledger [44] and Quorum [45].

9. *Transaction Validation.* Validators on the network receive transactions generated by clients and check their syntax and structure to ensure that they comply with the rules of the blockchain protocol. Transaction validation within the blockchain is a two-stage process. (i) Eager validation, which takes place when a validator receives a transaction from another validator or a client, and (ii) Lazy validation, occurring before the execution of transactions within a block.

   Executing an invalid transaction does not initiate a state transition; instead, it results in an exception being thrown. In essence, after the lazy validation of transactions, the state machine re-evaluates additional validity criteria (such as transaction signature verification and size limit checks) during the transaction execution phase. If any invalidity is detected, an exception is thrown.

10. *Incentives.* Blockchains, particularly open and public blockchains such as Bitcoin [13], Ethereum [14], and Algorand [16], achieve decentralization by allowing any node to propose blocks for consensus through resource-intensive processes such as solving proof-of-work (PoW) puzzles or staking coins. However, since there is no inherent motivation for nodes to become validators without a reward, these blockchains implement incentive mechanisms to encourage active participation and maintain decentralization. Rewards for validators are typically given for suggesting blocks that are successfully appended to the mainchain. Additional incentives are paid by clients in the form of transaction fees. Some blockchains also include penalties or punishments as negative incentives for validators who deviate from the blockchain protocol [14,46]. Designing a thoughtful incentive mechanism is critical to prevent nodes from compromising the security of the blockchain to maximize rewards.

2.2.2 The Blockchain Consensus Problem

2.2.2.1 Consensus Problem

The Byzantine Generals' problem addresses the challenge of achieving consensus among a group of processes in the presence of faulty elements [47]. A faulty process may experience a crash failure (e.g., hardware failure) or behave Byzantine, demonstrating arbitrary actions. The term "Byzantine" is derived from the Byzantine Generals' problem [42], where processes engage in equivocation to induce disagreement. A Byzantine process can exhibit any arbitrary behavior, including equivocation, sending invalid messages, or being unresponsive. Conversely, a correct process adheres to the protocol consistently. In a distributed system, proposing a value to the consensus is termed *"proposing"*, and achieving agreement on a value is referred to as *"deciding"* [16, 43].

For a system aiming to achieve agreement, it must adhere to the following three properties [16, 42, 43]:

- *Liveness:* Every correct process should eventually reach a decision.
- *Safety:* No two correct processes decide on different values.
- *Validity:* If all correct processes propose the same value, no alternative value is decided.

Achieving consensus is essential for distributed systems to reach agreements, and this requirement extends to specialized distributed systems like blockchains. However, adapting the consensus problem to the blockchain context introduces intricacies due to the unique primitives employed, including transactions, blocks, a chain of blocks, and validators.

2.2.2.2 Blockchain Consensus Problem

The blockchain consensus problem is defined as the challenge of guaranteeing the liveness, safety, and validity of a blockchain. The safety and liveness criteria are derived from the definition by Garay *et al.* [48], while the validity property is retained from its conventional definition [16, 47].

The Blockchain Consensus Problem. aims to guarantee that a distributed group of validators upholds a sequence of transaction blocks with the following properties:

- *Liveness:* (*i.e.*, Termination). A valid transaction received by a correct validator is reliably stored in the block sequence of all correct validators over time.
- *Safety:* (*i.e.*, Agreement) The local chains of blocks maintained by any two correct validators are either identical or one is a prefix of the other.
- *Validity:* Each block added to the blockchain of every correct validator includes a set of valid and non-conflicting transactions.

Several blockchain consensus protocols have been proposed in the literature in recent years, the Proof of Work (PoW) and Proof of Stake PoS being the most famous due to their use in Bitcoin [13] and Ethereum [14] public networks. Other protocols referred to as "PoX" stand for "Proof of X", where the "X" can represent different factors or resources used to reach consensus in a blockchain network. The "X" can stand for Authority, Space, Storage, Reputation, etc. Each of these variants represents a different consensus mechanism, adapted to specific needs or goals.

2.2.3 Web3

Web3 refers to the vision of the third generation of the internet and web applications, characterized by the integration of decentralized technologies such as blockchain, smart contracts, Dapps, and peer-to-peer protocols. In contrast to the traditional, centralized model of Web2, Web3 envisions a more open, trustless, and user-centric internet where individuals have greater control over their data and interactions [14, 49].

Smart Contract. A smart contract is a piece of code or program with predefined rules and conditions encoded in high-level languages (*e.g.*, Solidity). It operates on a blockchain platform, executing and enforcing the terms of a contract when specific conditions are met. Smart contracts eliminate the need for intermediaries, providing a decentralized and transparent way to facilitate or enforce agreements between parties [49]. These code segments are uploaded and executed on the blockchain through transactions involving smart contract deployment and invocation. The execution of these pieces of code is replicated across each node in a blockchain, utilizing the instruction set of the respective virtual machine on each node (*i.e.*, EVM). Contrary to their original purpose of facilitating conditional payments upon meeting specific criteria, smart contracts, particularly since the advent of the Ethereum blockchain [14], have evolved. Instead of exclusively executing straightforward payment contracts, contemporary smart contracts now function as integral components within Decentralized Applications (DApps).

Decentralized Applications (DApps). Software applications that run on a distributed network of nodes, typically a blockchain, where no single entity has control over the system, and smart contracts and consensus protocols enforce the operations. DApps comprise a blockchain user interacting with a smart contract back-end through Remote Procedure Calls (RPC), utilizing languages like JavaScript, Golang, Java, and Rust. A DApp may encompass a collection of backend smart contracts. Examples of common DApps include Decentralized Finance (DeFi) applications, Non-Fungible Tokens (NFTs) marketplaces, and Decentralized Exchanges (DEX) such as UniSwap.

2.2.4 Blockchain Performance

2.2.4.1 Performance Metrics

- *Throughput* is the number of transactions committed per unit of time (second) by a blockchain. A committing of a transaction refers to a transaction being executed, written to a block, and irreversibly appended to the blockchain. When this happens,

blockchains usually generate a notification known as a transaction receipt notifying the client that the sent transaction was committed. Upon receipt of the notification, the client gains assurance that the transaction has been successfully committed. When measuring throughput, the client's perspective is taken into account. Meaning, the calculation involves determining the number of transactions committed per unit of time using the receipts received by the client.

- *Latency* refers to the time required to commit an individual transaction successfully. This duration is calculated by taking the difference between the time the transaction is sent and the time it is committed, as perceived by the client. Essentially, the commit time is when the client receives a notification confirming the transaction was committed. Typically, in assessing blockchain latency, measuring the latency of a single transaction is inadequate. Therefore, the average of all latencies is computed to gauge the overall latency of a blockchain.

- *Gas* is the unit used to measure the computational effort required to process transactions. Each operation in a smart contract has a specific gas cost associated with it. Gas costs are determined by factors such as the complexity of the smart contract code, the amount of data being processed, and network congestion. Higher gas costs are typically required for more complex operations or during periods of high network activity.

2.2.4.2 Blockchain Congestion

Blockchain congestion is a phenomenon that arises when the influx of requests surpasses the processing capacity of the blockchain system. This overload causes transaction queues within the blockchain to become saturated, resulting in a range of adverse effects. During periods of blockchain congestion, one can observe transaction losses and a noticeable decline in overall system performance.

In practical terms, the saturation of transaction queues implies that the blockchain is unable to handle incoming requests promptly. Consequently, transactions may face delays or, in severe cases, might be unable to proceed, leading to transaction losses. This congestion-induced bottleneck manifests as a reduction in throughput, meaning the system processes fewer transactions per unit of time. Simultaneously, latency, or the time delay between the initiation and completion of a transaction, tends to increase. This increase in latency further contributes to the observable performance degradation during periods of heightened blockchain congestion.

2.2.4.3 Performance Evaluation Tools

- *Hyperledger Caliper*[5] is a tool for assessing blockchain performance. It is capable of evaluating Ethereum as well as various blockchains developed within the Hyperledger foundation, such as Besu and Fabric. The metrics supported by Caliper for

[5] https://github.com/hyperledger/caliper-benchmarks

blockchain evaluation encompass transaction throughput, transaction latency, and resource usage, which include CPU, memory, and network utilization. Caliper facilitates predefined workloads, specifying the calling smart contract, contract function, and transaction sending rate. It is important to note that the predefined workloads in Caliper, while both synthetic and user-defined, may not necessarily offer a realistic evaluation of blockchains in a real-world context.

- *BlockZoom* [50] is a blockchain testbed operating on a highly adaptable and controllable HPC platform. It offers a reproducible setting for experimenting with distributed ledger technologies and smart contract applications. By employing various configuration scenarios, the platform allows for assessing the performance of decentralized applications on a scale comparable to a production environment. BlockZoom is deployed on the Grid'5000[6] platform, a resource-rich environment for research-driven experiments spanning eight geographic sites in France and Luxembourg.

- *Chainhammer*[7], as a blockchain evaluation tool, primarily focuses on evaluating the throughput of EVM-based blockchains when subjected to intensive workloads. However, it lacks support for varying transaction sending rates, as it assesses blockchains exclusively under a consistently high workload. Additionally, Chainhammer lacks flexibility in adjusting transaction workloads to replicate realistic scenarios.

2.2.5 Blockchain Scalability

The blockchain scalability problem, also known as the *"Blockchain Trilemma"*, refers to the challenge of increasing the capacity of a blockchain network to handle a higher volume of transactions or operations per second while maintaining efficiency, security, and decentralization. There are two approaches to solving the scalability problem in blockchain networks: Layer-1 (L1) and Layer-2 (L2) scaling solutions [51].

2.2.5.1 L1 scaling

L1 scaling solutions focus on improving the blockchain protocol's base layer (Layer-1). These techniques aim to increase the throughput and capacity of the underlying blockchain network by making fundamental changes to the protocol. L1 scaling solutions typically require protocol upgrades or hard forks and involve trade-offs in decentralization and security. Examples of L1 scaling solutions include increasing the block size, optimizing consensus algorithms (*e.g.*, sharding in Ethereum 2.0, Delegated-PoS [46]), and implementing new blockchain architectures (*e.g.*, Directed Acyclic Graphs - DAGs) [51].

2.2.5.2 L2 scaling

L2 scaling solutions operate on top of the existing blockchain network and aim to alleviate scalability issues by offloading some transaction processing from the mainchain.

[6] https://www.grid5000.fr/w/Grid5000:Home
[7] https://github.com/drandreaskrueger/chainhammer

These techniques introduce additional layers (Layer-2) or protocols that process transactions off-chain or more efficiently, while still leveraging the security of the underlying blockchain. L2 scaling solutions offer the advantage of scalability improvements without requiring changes to the underlying blockchain protocol. However, they may introduce complexities in terms of interoperability and security guarantees. Examples of L2 scaling solutions include payment channels (*e.g.*, Lightning Network for Bitcoin), state channels, sidechains (*e.g.*, Plasma), and rollups (*e.g.*, Optimistic Rollups and zkRollups) [27].

2.3 Blockchain and Reputation Management

The convergence of blockchain and reputation systems refers to the integration and utilization of blockchain technology in the design and implementation of reputation systems [7]. Blockchain, a decentralized and tamper-resistant ledger, is combined with reputation systems to enhance transparency, security, and reliability in tracking and evaluating the reputation of users and entities within a network. This integration often addresses issues of trust, accountability, and data integrity, offering a decentralized and verifiable foundation for managing and validating reputation information [1]. There are two distinct concepts within the field of on-chain reputation management, each with its own focus and approach. Blockchain-based Reputation Systems (BRSs) refer to reputation systems built on top of blockchain technology. These systems leverage the transparent and immutable nature of blockchain to establish and maintain reputation scores or ratings for users and entities [20, 52]. Reputation-based Blockchain Systems (RBS), on the other hand, refer to blockchain systems where the consensus mechanism or other aspects of the blockchain protocol incorporate reputation-based aspects [2, 17].

Blockchain-based Reputation Systems (BRSs) aim to provide more robust, trustworthy, and decentralized solutions compared to traditional, centralized reputation systems [20, 52]. Research in this area has some limitations, including how to securely and effectively incorporate off-chain data needed to evaluate real-world interactions into the blockchain. Also, due to the transparent nature of the blockchain, some privacy concerns such as tracking and retaliation have arisen in existing BRSs [1, 23]. In addition, most existing BRSs do not consider the impact of on-chain reputation management on the overall performance and scalability of the underlying blockchain. Throughout this thesis, we will present several BRSs with different applications to address these issues.

The other meeting ground between blockchain and reputation is Reputation-based Blockchain Systems (RBSs). These systems involve introducing reputation management into the consensus mechanism. In traditional consensus algorithms, nodes or participants can be considered equal, and decisions are often made based on the computational (PoW) or stake-based (PoS) power of each participant. In reputation-based consensus, the historical behavior, reliability, or trustworthiness of participants, often quantified by a reputation score, plays a key role in the decision process [53]. Nodes with higher reputation scores may have more influence or voting power in the validation process [2, 18].

The integration of reputation into consensus mechanisms aims to enhance the overall

Table 2.1: The objectives of the intersection between blockchain and reputation management.

Objective	Blockchain-based Reputation Systems	Reputation-based Blockchain Systems
Transparency & Trust	✓	✓
Decentralization	✓	✓
Security & Immutability	✓	✓
Accountability	✓	✓
Incentive Alignment	✓	✓
Privacy Protection	✓	
Protocol Resilience		✓
Network Fairness		✓
Scalability		✓
Network Governance		✓

Table 2.2: The challenges of the intersection between blockchain and reputation management.

Challenge	Blockchain-based Reputation Systems	Reputation-based Blockchain Systems
Sybil Attacks	✓	✓
Collusion Attacks	✓	✓
Computational Cost	✓	✓
Scalability	✓	✓
Initial Trust	✓	✓
Data Accuracy & Reliability	✓	✓
Subjectivity & Bias	✓	✓
Interoperability	✓	✓
Dynamic Environment		✓
Privacy Concerns	✓	
Integration with off-chain data	✓	
Adaptability	✓	
Regulatory Compliance	✓	

reliability, security, and resilience of the network by considering the past behavior and trustworthiness of participants [17,53]. This approach aligns with the principles of decentralized systems, where trust is distributed based on observed actions. Reputation-based consensus, when complemented by robust incentive mechanisms, can be particularly beneficial in blockchain networks. This ensures that decisions are influenced by the track record of nodes, building a more trustworthy and resilient consensus process. In chapter 3, we explore existing RBSs, discuss their limitations, and propose a novel blockchain protocol based on reputation. Our proposed protocol aims to enhance fairness and scalability by improving consensus using reputation scores.

Table 2.1 and 2.2 shows the main objectives and challenges of BRSs and RBSs. We describe them in detail in the following.

2.3.1 Common Objectives and Challenges

Blockchain-based reputation systems (BRSs) and Reputation-based blockchain systems (RBSs) are designed to achieve several goals, but they face several challenges to

widespread adoption. Their common goals and challenges are detailed below.

2.3.1.1 Common Objectives

- *Transparency & Trust:* Both BRSs and RBSs aim to instill trust by providing a transparent and immutable record of participants' actions and contributions. The decentralized and tamper-resistant nature of blockchain ensures that reputation information is trustworthy and verifiable.

- *Decentralization:* Within the blockchain context, reputation systems aim to operate in a decentralized manner, reducing reliance on central authorities. This decentralization contributes to increased resilience, fairness, and resistance to manipulation.

- *Security and Immutability:* Blockchain's security features, including its cryptographic primitives and protocols, provide a secure foundation for reputation data. Once recorded on the blockchain, reputation information becomes immutable and resistant to unauthorized alterations.

- *Accountability:* Both BRSs and RBSs aim to achieve this property as it strengthens the system's integrity, encourages positive behavior, and creates more trustworthy and resilient decentralized applications and networks.

- *Incentive Alignment:* Both categories often adjust incentives to encourage positive behavior and contributions. Participants can earn and enhance their reputation by acting in ways that benefit the system, thereby building a positive and productive ecosystem.

2.3.1.2 Common Challenges

- *Sybil Attacks:* The biggest threat to BRSs and RBSs are Sybil attacks. This is where malicious actors create multiple pseudonymous identities to gain a disproportionate amount of reputation. Implementing effective mechanisms to detect and mitigate Sybil attacks is critical. The impact of Sybil attacks on RBSs is even more important, as the security of the entire system depends on the ability of the reputation-based protocol to withstand these attacks.

- *Collusion Attacks:* Malicious participants can collude to manipulate the reputation system. Detecting and preventing collusion among users and nodes can be complex, requiring sophisticated access and permission control algorithms and monitoring mechanisms.

- *Computational Cost:* The computational cost of implementing reputation systems in blockchain environments is a significant challenge, given the need for these systems to be efficient, scalable, and secure. In addition, storing reputation-related data, including transaction history, feedback, and reputation scores, directly on the blockchain

can be expensive due to the high costs associated with on-chain storage. More importantly, when reputation models are implemented on-chain for transparency, executing smart contracts that manage reputation incurs computational costs. Complex contracts with multiple functions and conditions can be particularly costly.

- *Scalability:* As the number of users and nodes in BRSs and RBSs grows, managing reputation scores for a large number of participants becomes a challenge. Ensuring that the reputation system can handle a large volume of interactions while maintaining efficiency is critical. Scalability concerns may also arise in maintaining an efficient and timely consensus process.
- *Initial Trust:* Establishing an initial level of trust or reputation for new participants in the system can be challenging. Without a history of interactions, it is difficult to assess their reliability, which can lead to inconsistencies in the network.
- *Data Accuracy and Reliability:* Ensuring the accuracy and reliability of data entered into the blockchain for reputation aggregation is challenging. Misinformation or manipulation of reputation data can compromise the effectiveness of the system.
- *Subjectivity and Bias:* Determining what factors contribute to a node's reputation and how those factors are weighted can introduce subjectivity and bias. Different stakeholders may have varying opinions on what constitutes trustworthy behavior.
- *Interoperability:* Achieving interoperability between different blockchain networks and their reputation systems is crucial for creating a global and unified reputation standard. Developing protocols that enable interaction between diverse reputation-based blockchains is a challenge.

2.3.2 Blockchain-based Reputation Systems (BRSs)

2.3.2.1 BRSs Objectives

The goals of blockchain-based reputation systems include several key objectives that leverage the unique characteristics of blockchain technology. In addition to the common goals outlined above, some BRSs incorporate privacy-preserving techniques that allow participants to control their personal information while still contributing to the reputation ecosystem [1].

2.3.2.2 BRSs Challenges

Blockchain-based reputation systems also face additional challenges that must be addressed for widespread adoption.

- *Privacy Concerns:* The balance between transparency and privacy is delicate. While the immutability of the blockchain increases transparency, it can also expose sensitive information. Designing reputation systems that protect user privacy while providing useful information is complex.

- *Integration with Off-chain Data:* Many reputation systems rely on off-chain data and real-world events to assess user behavior. Integrating this off-chain data with on-chain reputation systems without compromising decentralization and security is challenging.
- *Adaptability:* Reputation systems need to be adaptable to different use cases and industries. Designing a one-size-fits-all reputation mechanism that accommodates diverse requirements is a significant challenge.
- *Regulatory Compliance:* Adhering to legal and regulatory frameworks, especially concerning data protection and privacy, poses challenges for blockchain-based reputation systems.

2.3.3 Reputation-based Blockchain Systems (RBSs)

Reputation-based blockchain systems (RBSs) are designed to achieve several goals but face several challenges in practical implementation. In addition to the general goals outlined above, the specific goals and challenges of RBSs are described in detail below.

2.3.3.1 RBSs Objectives

- *Network Fairness:* Reputation-based consensus schemes aim to allocate influence or voting power based on nodes' past behavior and reliability. This ensures that participants with positive reputations have a fair and proportionate say in the consensus process, preventing concentration of power.
- *Protocol Resilience:* Improve the resilience of the consensus mechanism by considering nodes' past behavior. This can help mitigate the impact of malicious actors and ensure that decisions are made with a focus on the reliability of nodes.
- *Scalability:* The blockchain can scale more efficiently when a reputation system is in place. A subset of reputable nodes can be authorized to execute the current instance of the consensus. Unlike public blockchains, this system prevents all nodes from providing the same service simultaneously, to minimize resource consumption.
- *Network Governance:* Facilitate community governance by involving participants in decision-making processes. Reputation-based consensus can empower participants to have a say in network governance based on their historical contributions and trustworthiness.

2.3.3.2 RBSs Challenges

In dynamic systems like blockchains, participants' behavior can change over time. Adjusting reputation scores to reflect current network configuration is crucial but challenging, especially in rapidly changing environments. In particular, frequent entry and exit of nodes can lead to rapid changes in reputation scores, making it difficult to establish long-term trust and reliability.

2.3.4 Reputation Modeling

Reputation modeling in BRSs describes the main dimensions and properties of how reputation scores are calculated. The authors in [7] present a comprehensive taxonomy on reputation computation in these systems including eleven properties:

- *Information:* denotes the information that is gathered and administered by the BRSs. Three types of information can be collected, Transaction score, Trust/Reputation score, or both Transaction & Reputation scores.
- *Dimension:* enables differentiation between domain-specific systems (focused on a single aspect) and general-purpose systems that can handle multiple aspects. For instance, being a good doctor should not imply being a good seller.
- *Computation:* trust scores can be calculated in different ways. The choice of method can have an impact on both system performance and the trust value at a particular moment. For example, a BRS can implement a full history calculation, a transactional update, or a period-based calculation.
- *Aggregation:* outlines the various approaches for calculating a reputation score, which are plentiful in literature and tailored to specific applications. The most common approaches are Deterministic, Probabilistic, and Flow-based models.
- *Logic:* encapsulates the logic used to calculate the reputation score: Bad behavior *vs* good behavior, local trust *vs* recommended reputation (*i.e.*, gathered from other users), recent actions *vs* old actions, good transaction count *vs* bad transaction count.
- *Value Control:* defines the entity responsible for managing the trust calculation. Trust values can be controlled collectively or locally in a subjective manner.
- *Data Aging:* the scheme that determines how the reliability of information diminishes over time. A decay function can assign greater weight to recent actions and less weight to actions that occurred long ago.
- *Selection:* specifies the approach used to select an entity to initiate a transaction with. Several methods can be used: Rank-based, Threshold-based, or Probabilistic.
- *Interoperability:* refers to the extent of the system. Is reputation earned in one system transferable to another or not?
- *Control:* specifies how a reputation system motivates and controls entities to act in a desired manner. BRSs can employ, Incentives (An entity is motivated or guided using rewards and punishments) and/or Rules (An entity is forced or limited to act only within a prescribed manner).
- *Actor:* identifies if the entities involved in the system are humans (*e.g.*, organizations, sellers, workers) or machines (*e.g.*, IoT devices, vehicles).

2.3.5 Privacy Preservation

Concerns about privacy in reputation systems emerged in the mid-2000s. Evidence suggests that users may be reluctant to provide honest feedback for reasons ranging from fear of retaliation or negative reviews to concerns about disclosing sensitive personal information [54]. Furthermore, the lack of anonymity on reputation-centric platforms like Airbnb has been found to create a sense of pressure for individuals to submit reviews with a positive bias. Concealing the identities of the users has been recommended as a solution to this problem [1].

Early work by Pavlov et al. [55], Kinateder and Pearson [56], among others, laid the foundation of privacy preservation in reputation systems. Since then, privacy-preserving reputation systems have undergone continuous evolution to adapt to new fields of application such as Industrial IoT-enabled retail marketing [22] and Crowdsourcing [57]. Moreover, the emergence of blockchain technology has recently sparked research in this domain. The integration of blockchain with other cryptographic components has facilitated the creation of privacy-preserving reputation systems, imbued with significant new attributes including decentralization, transparency, and immutability. However, research on privacy-preserving blockchain-based reputation systems (PPBRS) is still hampered by unresolved issues, in particular, the lack of crucial security properties and defenses against common reputation attacks such as Sybil, Whitewashing, and Free-riding attacks [1]. Hasan *et al.* present a detailed analysis of existing solutions. Their study highlights the shortcomings of these systems and summarizes the main future research directions. We agree with them that more work needs to be done in the area of privacy-preserving blockchain-based reputation systems in terms of defending against attacks other than privacy violations. This is because blockchain has not been used to its full potential to build truly trustless systems, as almost all existing solutions introduce some degree of centralization. In the following, we present a classification of PPBRS and their security objectives.

2.3.5.1 Classification of PPBRS

Privacy-preserving reputation systems can be classified into two main categories based on their security objectives. The first category aims to maintain user anonymity, while the second category focuses on safeguarding the confidentiality of user-provided feedback [1].

- *Privacy-preserving reputation* systems with a focus on **user anonymity** hide the true identities of users. Thus feedback providers are represented by one or more unlinkable pseudonyms rather than their real identity. This arrangement enables users to conduct transactions and provide feedback anonymously, eliminating the need for feedback confidentiality.

- *Feedback confidentiality-based* systems do not conceal the identities of users but rather assign each user a single pseudonym. Furthermore, these systems do not hide the fact that a user has provided feedback to another user. However, the information

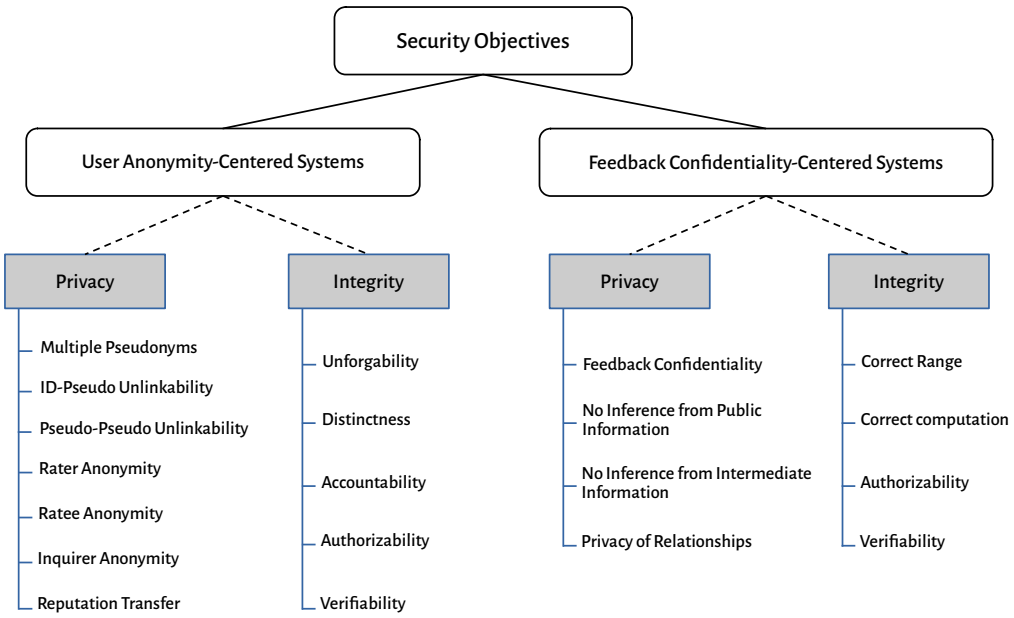


Figure 2.1: Security objectives of PPBRS [1].

contained in the feedback, as well as any related data, is considered private. This type of system is necessary as complete anonymity is not always feasible in real-world interactions. For instance, even if users remain anonymous on an e-commerce platform, exchanging physical goods bought and sold through the platform could expose their true identities. Thus, preserving the confidentiality of feedback provides a practical alternative that encourages users to provide honest feedback without fear of retaliation.

2.3.5.2 Security Objectives of PPBRS

The security objectives of a privacy-preserving blockchain-based reputation system can be divided into categories: those addressing privacy; and those addressing integrity or correctness. Privacy objectives involve concealing information about users, such as preserving the anonymity of the rater and ratee. Conversely, integrity objectives focus on upholding the accuracy of the functions of the reputation system while safeguarding user privacy. For instance, allowing users to use multiple pseudonyms within an anonymity-oriented reputation system may enable malicious ones to conduct self-promotion attacks. Figure 2.1 describes the privacy and integrity security objectives of both categories mentioned above and introduced in [1].

2.4 Common Attacks in On-Chain Reputation

Although on-chain reputation management can be vulnerable to various attacks depending on their implementation. In what follows, we present the most common attacks against these systems.

2.4.1 Blockchain Attacks

- *Sybil Attack.* Involves an adversary assuming multiple identities to overwhelm the blockchain, leading to governance disputes and potential disruptions. For instance, by impersonating a significant number of validators, the adversary can form a coalition within the consensus committee, resulting in disagreements (forks) that jeopardize blockchain safety or passive behavior that affects blockchain liveness. Probabilistic methodologies such as PoS and PoW provide a certain level of resistance against Sybil attacks in blockchains. They achieve this by introducing intricacies that make it challenging for adversaries to adopt multiple identities. In these approaches, adversaries are required to divide their staking or computing power to impersonate multiple users. However, this strategy diminishes their probability of being selected for specific governance tasks. On the contrary, deterministic schemes like voting-based approaches, where selection is determined by the number of votes, are susceptible to Sybil attacks. Therefore, it becomes crucial to implement appropriate mechanisms to mitigate these vulnerabilities.

- *Double Spending Attack.* A double-spending attack occurs in a blockchain network when an actor attempts to spend a certain amount of cryptocurrency (*i.e.*, coin or token) twice, exploiting the digital nature of the currency and attempting to deceive the decentralized consensus mechanism. The attacker aims to create conflicting transactions that appear valid individually but result in the unauthorized duplication of the same cryptocurrency units. This undermines the integrity of the blockchain by violating the fundamental principle of ensuring that each unit of cryptocurrency is spent only once. To prevent double-spending attacks, blockchain networks typically rely on consensus algorithms, cryptographic techniques, and validation mechanisms to reach an agreement on the valid transaction history, ensuring the uniqueness and authenticity of each transaction.

- *Updating Monopolization.* Also called *Blockchain Oligarchy*, and *Wealth Centralization* in PoS blockchains. Monopolization in a blockchain network occurs when a singular entity or a coalition of entities substantially controls a significant portion of the network's resources, governance, or decision-making processes. This dominance may result in an imbalance of power, potentially compromising the decentralized and trustless nature of the blockchain. Monopolization undermines the principles of fairness and decentralization that are fundamental to many blockchain architectures.

2.4.2 Reputation Attacks

- *Sybil attack.* In reputation systems, a Sybil attack is a type of attack where a malicious actor creates multiple fake identities (known as Sybil identities) to manipulate or undermine the reputation system's trustworthiness and reliability. By controlling multiple identities, the attacker can manipulate ratings, reviews, or feedback to their advantage. Legitimate users may be misled by the attacker's artificially inflated reputation, leading to bad decisions or transactions. To protect against Sybil attacks, reputation systems must implement robust identity verification. The success of this attack depends on the cost of obtaining such an identity. Therefore, the risk of a Sybil attack can be reduced as the cost of creating new identities increases. However, the most effective countermeasure is to associate the identity with a real-world identity [7].

- *Whitewashing attack.* A whitewashing attack in a reputation system occurs when a participant, with a history of negative or low reputation, strategically tries to reset his reputation. The attacker resets his poor reputation by rejoining the system with a new identity. Unexpectedly, the efficiency of this attack stems not only from the minimal entry cost to the network but also from the system's evaluation, which places a user with a zero reputation score higher than a user with negative ratings. This setup creates an incentive for the user to discard such an account. One approach to mitigate this issue is to ensure the attack remains costly. This can be achieved by consistently linking the identity of the user or by employing high re-entry costs [1].

- *Bad-Collusion attack.* Bad-collusion attacks in reputation systems refer to a form of collusion where malicious entities or participants actively coordinate their efforts to manipulate the reputation scores in a way that undermines the integrity of the system. Unlike benign collaboration, bad collusion involves deceptive actions with harmful intent, often aiming to boost the reputation of malicious actors or degrade the reputation of honest participants. More Formally, Bad-Collusion attacks in a reputation system occur when malicious entities or participants conspire to engage in coordinated and deceptive actions with the intent of manipulating reputation scores. This collusive effort aims to disrupt the fairness, accuracy, and reliability of the reputation system by artificially inflating or deflating the reputations of specific participants. Bad collusion is characterized by its harmful nature, seeking to undermine the trustworthiness and effectiveness of the reputation mechanism [7].

- *Self-Promotion attack.* A self-promotion attack in a reputation system occurs when a participant engages in manipulative actions to artificially enhance his/her reputation. This may involve creating and promoting positive feedback, ratings, or interactions about oneself, with the intent to deceive the reputation system and gain undeserved trust or advantages. Such attacks undermine the integrity and accuracy of the reputation system, as they compromise the authenticity of the reputation scores [1].

As outlined in the challenges section above 2.3.1, the primary threat in both BRSs and RBSs is Sybil attacks. The implementation of robust identity management and access

control is essential for building secure reputation-aware systems. Therefore, throughout this thesis, we will present several reputation frameworks powered by permissioned blockchains to prevent these attacks. Our solutions introduce various permissions and access control techniques, including on-chain permissioning via smart contracts (whitelisting and blacklisting) and token-based access control.

2.5 Conclusion

The areas where decentralized reputation systems have used blockchain are very diverse. In this thesis, we will focus on real-world applications such as crowdsourcing, e-commerce platforms, community evaluation, and other applications such as energy trading and IoT-based applications. From what had been discussed above several challenges to overcoming in such a realm of reputation management in real-world scenarios. More precisely, this is related to the question of how to collect off-chain data needed to evaluate real-world interactions to effectively and accurately update reputation scores stored on-chain. This thesis answers this question through four complementary contributions. In the next chapters of this thesis, we explore the issues of effectiveness, privacy preservation, and scalability in on-chain reputation management and propose novel blockchain-based reputation systems (BRS) to address them.

In chapter 3, we dig into the effectiveness and Layer-1 scalability issues and propose a blockchain-based reputation system (BRS) called GuRuMarket that addresses them. In particular, GuRuMarket takes a step further and leverages the reputation model proposed to evaluate participants' behavior at the consensus layer. This allows the selection of validators to be optimized without adding computational overhead (*i.e.*, a lightweight RBS).

Chapter 4 focuses on privacy concerns in real-world BRSs. It addresses the problem of effectively and privately managing reputation scores in real-world scenarios. The chapter proposes DARS an anonymity-oriented privacy-preserving BRS (PPBRS). The proposed framework allows users to interact using multiple pseudonyms to hide their true identities while maintaining effective reputation management.

Chapter 5 discusses scalability issues in BRSs. To achieve decentralization and transparency, reputation scores must be maintained on-chain, usually using smart contracts. However, in this case, instead of just managing business transactions, validators are tasked with handling an additional workload associated with reputation management. Therefore, in this chapter, we propose RollupTheCrowd a scalable blockchain-based reputation system that uses an L2 scaling technique, namely zkRollups. The key aspect of zkRollups is that they inherit the security of the mainchain (*i.e.*, provide almost the same level of security as the mainchain), which is crucial for robust reputation management.

In the final chapter 6 of this thesis, we explore the extensibility and adaptability of the models proposed in the previous chapters through their usage in emerging applications. In particular, the chapter proposes LLMChain a new reputation system for sharing and evaluating large language models using blockchain and other decentralized building blocks. The system aims to provide transparency and traceability to the process of

evaluating generative AI models. It allows individuals and small companies to identify the most appropriate LLMs for their specific needs based on contextual, fine-grained reputation management. Additionally, it allows LLM developers to gather valuable information to refine and correct their models.
This set of contributions is meant to answer the issues identified and discussed in this chapter.

Chapter 3

Effectiveness and Layer-1 Scalability of Blockchain-based Reputation Systems

Contents

In the previous chapter, we discussed the challenges associated with on-chain reputation management. We elucidated the significant impact of on-chain reputation management on the scalability of the underlying blockchain. Additionally, we discussed the inherent difficulty in achieving privacy and effectiveness in blockchain-based reputation systems without compromising scalability and performance.

This chapter addresses the challenge of achieving effective and scalable on-chain reputation management. Our primary objective in the present chapter is to develop a fully decentralized, blockchain-based reputation system applicable to diverse real-world scenarios. In particular, the focus here is on (1) designing an effective reputation model for blockchain-based real-world marketplaces and (2) improving the scalability of the mainchain (L1).

3.1 Introduction

Blockchain (BC) has attracted considerable attention in recent years due to its inherent ability to maintain privacy and security. By eliminating the need for a centralized intermediary, this technology can be used in various trust-related applications such as supply chains [20], crowdsourcing [2], and Internet of Things [10]. Some recent initiatives have explored using blockchain to build decentralized and transparent reputation systems for real-world applications [10,21,41]. Other studies have delved into using trust and reputation management at the consensus layer to optimize blockchain maintenance and improve its overall scalability [17–19]. However, only a handful have explored the integration of application trust and consensus trust management to accomplish both goals simultaneously. In addition, building a reputation system on top of a blockchain introduces significant computational overhead. In this scenario, the network is tasked not only with managing business transactions (*e.g.*, crowdsourcing or trading), but also with evaluating interactions and updating reputations. Therefore, finding a suitable solution to minimize the cost of on-chain reputation management without compromising the security and reliability of the model is critical to maintaining/improving the overall performance and scalability of the system.

To address the issues raised by traditional consensus protocols, new mechanisms are emerging [2,18,21,53]. These novel approaches are grounded in trust and reputation. In this context, "trust" refers to the mutual reliance between two nodes and may be seen as a local reputation. It is established by evaluating and recording interactions between nodes/validators. Conversely, "reputation" signifies the overall opinion that nodes within the system hold toward a specific node [3]. It is usually determined by aggregating local reputation scores. By integrating trust management, a reputation-based approach can

be implemented to replace or enhance existing consensus mechanisms. In particular, the use of global reputation scores helps to screen out active nodes, improving the selection of validators and speeding up the consensus process. This naturally improves the scalability of the system. Moreover, adopting a strategy that prioritizes a reputation score over computational or staking power alone greatly fosters fairness among peers in the network, effectively addressing concerns about monopolizing updates (*e.g.*, wealth centralization) [2, 9, 18]. However, while reputation-based blockchains show potential, current solutions lack a detailed blueprint for implementing their reputation models. In addition, several systems use reputation frameworks from other protocols and applications. This approach faces compatibility issues with the blockchain environment due to the complexity or centralized, semi-honest design of the models employed.

The fusion of consensus and application trust aims to cater to diverse trust management needs for decentralization, transparency, and efficiency without compromising the scalability of the blockchain. Nevertheless, a shared concern in current approaches revolves around the calculation of trust and the incentivization of entities through reputation. Most existing systems rely on reputation models that assess the trustworthiness of entities based on their past behavior and offer incentives based on reputation alone. We advocate for anchoring trust to tangible assets as this offers stronger incentives and aligns better with real-world scenarios. In addition, an often neglected element of most reputation-based applications developed on blockchain is their adaptability to real-world situations. A robust BRS should not ignore off-chain tasks. It must consider whether the off-chain part of the interactions is performed properly, *e.g.*, whether the goods purchased are delivered properly or not.

To address the aforementioned challenges, in this first chapter, we propose GuRuMarket: a Guarantee and Reputation-based Blockchain Marketplace. To foster accountability and trust we design a new effective and automated reputation model that merges subjective and objective trust. The proposed model is complemented by a guarantee and reputation-based incentive mechanism to control participant behavior. The core part of GuRuMarket is the proposed Lightweight consensus mechanism, called *Proof-of-Guarantee-and-Reputation (PoGR)*. PoGR improves the mainchain scalability and fairness (*i.e.*, prevents wealth centralization) by employing a block producer selection based on guarantee and reputation.

Thus, the rest of this chapter is structured as follows: Section 3.2 presents related work. The proposed GuRuMarket framework is presented in section 3.3. The designed reputation model is detailed in Section 3.4. The proposed incentive-based trading logic is described in section 3.5, followed by the consensus protocol in section 3.6. Sections 3.7 and 3.8 discuss the security analysis and performance evaluation of GuRuMarket, respectively. Section 3.9 presents an analytical comparison between GuRuMarket and some relevant studies. Finally, we conclude the chapter in section 3.10

3.2 Related Work

The intersection of blockchain and reputation management is promising, but existing solutions have several technical and theoretical limitations. These solutions, as discussed previously (Chapter 2), fall into two main categories: Blockchain-based Reputation Systems (BRS) and Reputation-based Blockchain Systems (RBS). We discuss their shortcomings in the following.

3.2.1 Blockchain-based Reputation Systems (BRSs)

Blockchain-based reputation systems aim to build a reputation system on top of a blockchain network using smart contracts and other Web3 ingredients.

TrustChain [20] is a three-layered blockchain-based framework for trust management in blockchain IoT-supported Supply Chains. The proposed solution is a service platform implemented on a permission-based blockchain network that uses smart contracts to automate reputation calculations, along with a reward and punishment-based incentive mechanism to encourage users to behave properly. The proposed model calculates a seller's rating by aggregating a weighted sum of three inputs: the reputation score of the traded commodity, the regulator's assessment of the seller, and the buyer's evaluation of the seller. However, the proposed solution does not discuss the consensus protocol, some entities are considered honest, and the business network administrator has strong control over the network, which creates a central point of failure and a security threat.

The authors in [21] introduce a trust-based permissioned blockchain that replaces *PoW* by trust assessment using a *Proof-of-Trust* (PoT) consensus. The proposed solution incorporates a trust architecture that uses detailed formulas and additional modules to compute the trust level and predict the behavior of each participant before creating smart contracts and before starting interactions. However, the designed architecture includes several complex modules such as machine learning and artificial intelligence modules that consume a lot of resources to predict user behavior. This poses significant challenges to building such a complex system.

Reputable [39] is a decentralized reputation system for assessing service providers' activity within a blockchain-based ecosystem. The work model reputations as a collection of attributes without discussing their aggregation together. The proposed solution integrates a centralized oracle, which prevents the system from achieving full decentralization. The authors do not discuss the consensus mechanism in their study and use PoW to evaluate the proposal, which lacks scalability, especially when considering the additional reputation workload.

ValidatorRep [58], introduces a verification scheme that utilizes blockchain with trust management to foster accountability within crowdsourcing systems. The proposal entails a decoupled blockchain model designed for the distinct storage of business and log transactions throughout data interaction. It uses a trust model encompassing the reputation of participants and the trust relationships among them. However, the research does not discuss the consensus protocol, and the impact of managing reputation on-chain.

Truth [59] is a blockchain-assisted secure reputation system that removes trusted third

parties from the e-commerce platform while guaranteeing authenticity. In particular, it uses a hybrid framework that relies on a centralized e-commerce platform to provide traditional trading services, while using the blockchain only to authenticate the correctness of feedback. The platform allows buyers to confirm a seller's feedback score, thereby exposing fake scores. However, the system lacks resistance to collusion attacks between buyers and sellers.

TRUSTD [60] builds an ecosystem powered by blockchain and collective signatures, designed to support content creators in garnering community backing for their content. Additionally, it aids users in assessing the credibility and accuracy of these contents. The authors emphasize that machines can detect fake content to some extent, but there is no substitute for human intervention. However, their feedback-based trust model is also subject to collusion attacks. Additionally, the content could be signed by an entity that is not trusted by the user. Furthermore, the proposed approach lacks incentive for the entities designated as reviewers to quickly sign off on the content.

Other solutions for specific use cases are proposed in [10, 61–65]. Their lack of scalability and adaptability prevents them from being used as general frameworks in other real-world scenarios.

3.2.2 Reputation-based Blockchain Systems (RBSs)

Reputation-based blockchain systems are blockchain protocols that integrate trust and reputation at the consensus layer to improve the governance of the mainchain.

Delegated-Proof-of-Reputation (DPoR) [18], is an improvement of *Delegated-Proof-of-Stake (DPoS)* [46]. To enhance decentralization and fairness, DPoR uses a set of parameters to select the block producer instead of the staked amount alone. In particular, it replaces pure staking with a reputation ranking system based on ranking theories. However, like DPoS, DPoR relies on a small number of elected nodes or delegates to validate transactions and create blocks. This concentration of power in a limited number of nodes raises centralization concerns. If these nodes collude or the group is compromised, the security of the network may be compromised. In addition, the proposed system requires more analysis and experimentation to assess the relevance of certain choices (*e.g.*, a reputation ranking based on HodgeRank theory). Finally, the implementation details of the protocol and their costs are not discussed in the work, which causes doubts about the feasibility of the solution.

The authors in [2] combine crowdsourcing with a blockchain consensus scheme. The work introduces a reputation model adapted to crowdsourcing and uses an incentive mechanism based on game theory to ensure the honesty of nodes within an improved *Proof-of-Trust* consensus [53]. The consensus selects nodes with high credibility using a reputation model based on subjective logic [66]. However, the model first used in [67] combines direct opinions with recommended opinions using a complex mathematical model, which poses significant cost challenges. In addition, the implementation of the model is not discussed in their work, which also raises questions about its feasibility. Furthermore, the proposed consensus lacks determinism in selecting the leader of the next index (*i.e.*, the probabilistic algorithm does not guarantee the uniqueness of the

chosen leader). It also does not discuss the process of resolving forks that are likely to occur.

The authors in [17] propose a reputation model to evaluate the reputation of nodes based on their historical behavior. The system assigns each new node an initial reputation value close to zero. It then uses the reputation model to update the reputation score at the end of each epoch. The model uses a function to score a node's proposing and voting behavior and then combines them into a single normalized rating. However, the proposed model is implemented locally in each node, which lacks transparency and introduces potential vulnerabilities since any node can alter the code. Furthermore, the protocol requires an additional phase for evidence propagation. Finally, the research does not discuss the implementation and experimentation of the proposed reputation-based SMR, which raises questions about its feasibility.

Proof-of-Reputation [68], introduces a consensus mechanism that evaluates a node's reputation by considering its assets, transaction history, and participation in the consensus process. Under this proposed mechanism, the leader node with the highest reputation is responsible for generating a new block. Subsequently, the validation and confirmation of this new block are conducted through reputation-based voting. The rewards accrued from block generation are then distributed among validators according to their respective reputation scores. However, the study lacks implementation details and does not present experimental results to validate the proposal. Further analysis is necessary to address security concerns, including potential reputation-related issues such as whitewashing and bad-collusion attacks, as well as blockchain-specific threats.

The Blockchain Reputation-Based Consensus (BRBC) [69] mechanism uses a reputation score requirement for nodes wishing to become validators. That is, the score of a given node must exceed a predefined network trust threshold. A group of randomly selected judges oversees each node's conduct in the consensus process and adjusts their reputation scores accordingly. Positive actions are rewarded, while any non-cooperative or malicious behavior incurs penalties. However, the incentive scheme relies on reputation only, which leaves room for selected validators to behave maliciously. The proposed mechanism does not incentivize judges to act appropriately, which can lead to collusion attacks. Furthermore, the authors do not provide a comprehensive mathematical formulation for calculating the reputation score.

Author studies [19, 70–72] employ incentive mechanisms centered on reward and punishment. Participants or nodes receive reputation rewards for positive behavior, while negative behavior may result in punishment or expulsion from the system. However, using reputation alone to incentivize nodes is not sufficient, as an elected leader may produce malicious blocks that harm the system and then leave the network permanently.

3.2.3 Positioning our work

Given the challenges inherent in both categories, there is a need to design a blockchain system tailored to real-world trust-related applications that meets the following objectives:

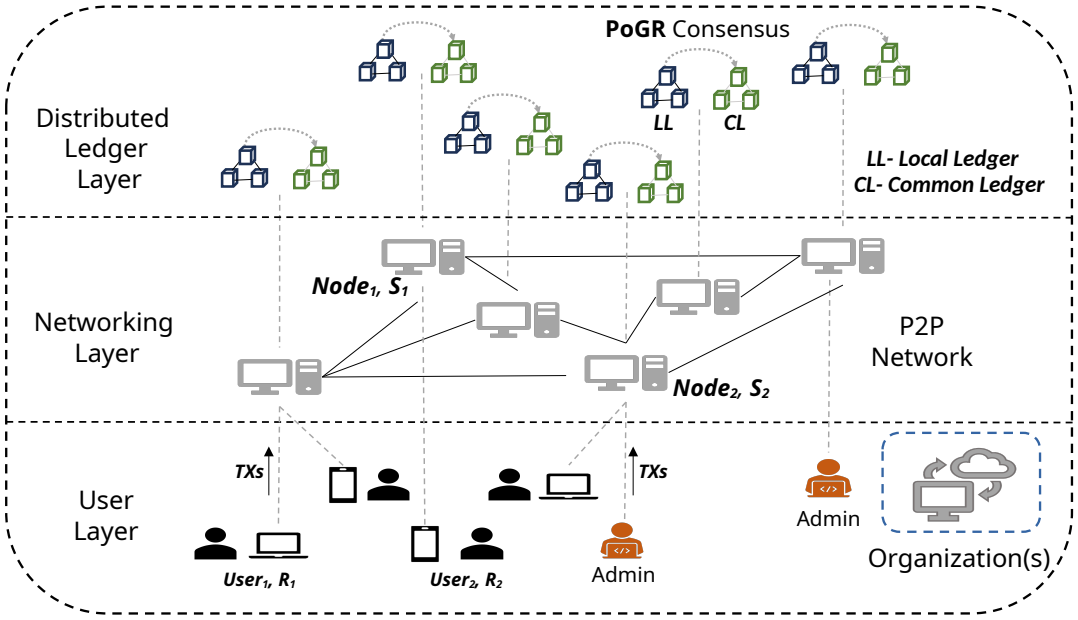


Figure 3.1: Layered Architecture of GuRuMarket

- To build an effective reputation model at the application layer, one needs to improve the scalability of the mainchain. One potential approach is to use participants' reputation scores as metrics in the design of a trust-based consensus protocol to scale the underlying blockchain.
- The consensus algorithm must be implemented securely and effectively to ensure a high degree of decentralization and fairness among operating nodes. It has to prevent any node or group from monopolizing updates.
- To control the behavior of participants and nodes robust incentive mechanisms should be integrated at both the application (business) and consensus levels. These mechanisms must use deposits along with reputation rewards and penalties to secure both the trading process and the blockchain protocol (*i.e.*, ensure accountability).

3.3 GuRuMarket Framework

In this section, we introduce GuRuMarket, a blockchain-based real-world marketplace framework. GuRuMarket mitigates reputation attacks through effective on-chain feedback-free reputation management; and fosters trust and accountability through guarantee and reputation-based incentives. GuRuMarket integrates a novel effective reputation model to evaluate traders' behavior when exchanging real-world products and services, and a reward-punishment mechanism to secure the trading process by ensuring accountability. In what follows, we first present GuRuMarket's architecture, then describe its main components, and finally detail its reputation model.

3.3.1 Design Overview

Figure 3.1 depicts the layered architecture of our proposed system. Our framework consists of individual users/participants, each owning a key pair (Sk, Pk), a private key and a corresponding public key, and at least one End Device (ED) to interact with the system. Nodes connect via a secure peer-to-peer (P2P) protocol to form a well-connected network supporting the Distributed Ledger (DL) layer above. All nodes collaboratively maintain a Common Ledger (CL) by executing the consensus algorithm to update their Local Ledgers (LL). Central to the system is the consortium, represented on-chain by participants who serve as administrators. Their primary responsibility includes managing user entries and exits from the system. They have the authority to add or remove participants or nodes through voting mechanisms. The consortium acts as the initiator of the marketplace. Entities seeking to join must register with the consortium, providing proof of legitimacy. This information is stored off-chain to uphold privacy and comply with regulatory requirements. To guarantee this, a distinct ledger for managing identities and credentials could be employed in addition to the primary ledger, aiming to preserve privacy and protect user data [41,73]. The identity management ledger aims to provide sybil-resistant privacy-preserving decentralized credentials for any valid user or entity. The consortium is also responsible for converting real-world currency into virtual funds “stablecoins". However, it does not get involved in the trading process.

3.3.2 System Model

We consider an *open permissioned* blockchain model [74], where a subset of the distributed nodes have permission to execute the consensus's current instance and maintain the resulting state. This model is called “open” because permission can be revoked (*e.g.*, current reputation below the threshold), and there are no barriers preventing a particular node from obtaining permission at a later time. Unlike permissionless blockchains, we simply prevent all nodes from providing the same service at the same time to minimize resource waste.

3.3.3 Threat Model

In GuRuMarket, it is assumed that all entities, including traders and nodes, can be malicious. These involved participants do not trust each other and always seek to maximize their benefits by committing malicious activities or joining forces to carry out attacks. Furthermore, we assume that there is no trust among the administrators representing the consortium in our framework. We assume the existence of a vote-based governance protocol for making decisions such as adding and deleting participants and nodes. Under these assumptions, there may be a wide range of potential attacks that GuRuMarket aims to counter, as described below:

- **Sybils attack:** An attacker creates multiple identities (*i.e.*, Sybils) and attempts to exploit them to manipulate its reputation score.

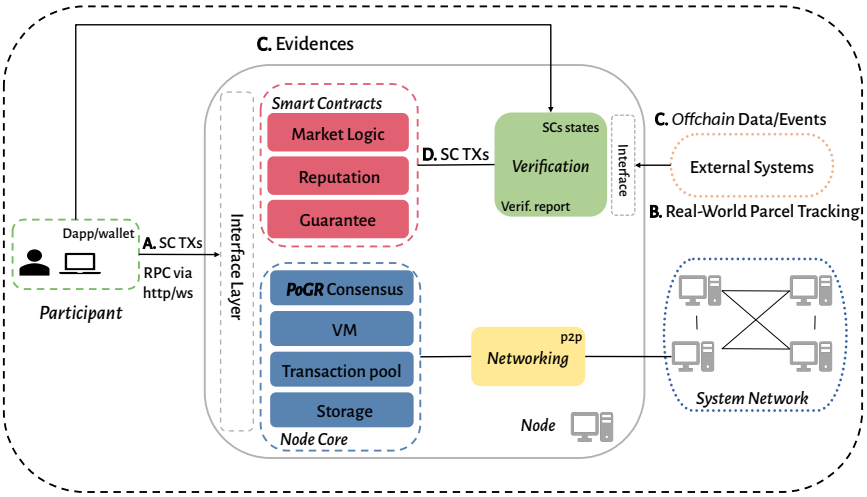


Figure 3.2: GuRuMarket Modules

- **Double spending attack:** An attacker tries to spend some tokens more than once. We have two cases here according to the attacker role:
  Case-1: The attacker is a seller who tries to sell the same product more than once.
  Case-2: The attacker acting as a buyer, attempts to submit multiple payment transactions, using the same funds (only one TX is valid).

- **Whitewashing attack:** A malicious trader tries to reset his bad reputation score by re-entering the system with a new identity and getting the initial reputation score. This form of attack directly impacts the marketplace's reliability and can erode trust among traders in the system.

- **Bad-collusion attack:** This type of attack occurs when a group of traders colludes together to lower the reputation of a target trader or improve their reputation. Defending against this type of attack is essential for robust and accurate reputation management.

- **Malicious updates:** In this attack scenario, the attacker, acting as a participant running a node, attempts to create invalid blocks (*i.e.*, containing at least one invalid TX, such as double spending) by exploiting the blockchain protocol. The security of the entire system depends on the security of the underlying blockchain protocol.

3.3.4 GuRuMarket Modules

Two primary parameters are considered in GuRuMarket, Guarantee, and Reputation, their definitions are the following:

- **Guarantee:** represents the amount the participant desires to lock in the blockchain as a deposit/collateral to secure a task, such as a trading interaction.

- **Reputation:** refers to the opinion that blockchain participants have towards another participant. It is represented by a score $R \in [0 - 1]$ and calculated based on past interactions.

Within GuRuMarket, individuals, termed participants, gain access to the network upon fulfilling two precise conditions: (i) upholding a sufficient reputation level and (ii) depositing at least the minimum necessary guarantee. Figure 3.2 shows an overview of GuRuMarket, highlighting the blockchain node launchable by a participant and illustrating the various interconnections between its main components. It includes the following functional modules:

3.3.4.1 Node Core Module

It includes all the components of a blockchain client, specifically the consensus algorithm, the storage, the transaction pool, and the virtual machine (VM). Consensus algorithms enable transaction verification, block generation, and block validation. Storage represents the key-value database schema used to store blockchain data locally. A blockchain's VM enables the deployment and execution of smart contracts via transactions. Finally, the transaction pool is a temporary storage unit where valid but unconfirmed transactions are held before being added to a block and subsequently confirmed by the network. When a participant initiates a transaction, it is broadcast to the network and added to this pool. It remains in the pool until it is included in a block or is deemed invalid.

3.3.4.2 Networking Module

The network module enables inter-node communication using P2P protocols. Each node must be connected to at least one other node to ensure that the system network is well-connected. It is the only way for nodes to communicate with other nodes. The communication protocol used by the network module must ensure that participants' private information does not leak across the network. Its main functions are peer discovery and connection, message(transactions and blocks) propagation, synchronization, and communication security through encryption.

3.3.4.3 Verification Module

It is used to check and evaluate whether the real-world (off-chain) part of the interaction is properly executed. This module can verify immediate transactions as well as track the quality of services over time. It provides participants with a certificate to confirm whether the requested services/products have been correctly delivered. To issue the service provision certificate, the verification module is required to gather trading data, including real-world data and events, from an external system. The buyer, seller, and independent controller must submit the necessary data to authenticate the delivery, allowing the verification module to generate the verification report. The module verifies data through corresponding measurements and generates a certificate that all nodes can recognize based on these verification results.

The controller in our system may vary depending on the service provided. It can be a standalone smart device with a dedicated interface to capture required information, or an independent physical entity equipped with advanced end devices for data entry. Given the system's diverse services, the types of proofs collected and measurement methods employed may differ. Moreover, the verification process and its outcomes must be hands-off. The system operates under the assumption that the verification module consistently produces identical certificates when valid proofs are provided as input. The verification process can be implemented using a Decentralized Oracle Network (DON[1]). Verification using a DON supports hybrid smart contracts. It combines on-chain code and off-chain infrastructure to build advanced Decentralized Applications (DApps) that react to real-world events and interoperate with traditional systems [26]. Hybrid Smart contracts on the DON automate the process of confirming delivery. For instance, once a parcel reaches its destination and is confirmed through sensors, QR codes, or digital signatures, the smart contract could execute, triggering actions like confirming delivery by generating the delivery certificate, releasing payment, and updating reputation.

3.3.4.4 Smart Contract Module

A smart contract is a piece of code or program with predefined rules and conditions encoded in high-level languages (*e.g.*, Solidity). It runs on a blockchain platform, executing and enforcing the terms of a contract when specific conditions are met [14]. Smart contracts are triggered by addressing transactions to them. The smart contract module in our system implements the entire trading process. It has three main sub-modules: Business logic, Reputation, and Guarantee sub-modules. The Business Logic sub-module implements the complete open market trading logic. The Reputation sub-module is responsible for calculating trust scores after each interaction and updating the overall reputation of the participants. The Guarantee module ensures the deposit and revocation of the guarantee. We explain the functions of these sub-modules in detail in the following.

1. **Business Logic Sub-module** is the application component that implements all the functions required for a participant to interact with the system. It is mainly responsible for:

   - Managing participant access to the various functions of smart contracts through role-based access control.
   - Allowing participants to manage their independent wallet accounts *e.g.*, generate transfer and refill transactions.
   - Managing various business transactions such as service announcements and order placements sent to the system through the participant ED or the networking module.

2. **Reputation Sub-module** is responsible for trust and reputation management, implementing two main functions that enable first the evaluation of the interaction and then

[1] https://chain.link/education/blockchain-oracles

the updating of the overall reputations of the participants involved in that interaction.

3. **Guarantee Sub-module** is devoted to the guarantee management. In our system, buyers can reject the delivered service after verification. To protect sellers from such behavior, we allow them to set the proportion they want to get as a service delivery fee in case of rejected service. They are free to set this value, which we call the guarantee proportion P_g. Let Val be the value of the service, the amount transferred to the seller in case of a rejected delivery is $G = P_g Val$. Buyers, on the other hand, are free to choose the services offered by different sellers. To encourage buyers to choose its service, a seller must choose an appropriate P_g. Services with low P_g will be preferred, while those with high P_g will be obviously avoided.

3.4 Reputation Modeling

3.4.0.1 Trust Value Calculation

Within our framework we evaluate the trust of all participants per interaction. More precisely, the system's reputation module assigns a trust value to each trader (seller or buyer) after each trading interaction. The trust value assigned to a seller T_i^s after a real-world interaction i is given by [9]:

$$T_i^s = (1 - \theta).T_{b \to s} + \theta.T_{v \to s} \tag{3.1}$$

where $T_{b \to s}$ represents the subjective trust value, $T_{v \to s}$ is the verification (objective) trust value, and $\theta \in [0, 1]$ is the weight that gives more relevance to the objective trust.

The trust value that represents the direct opinion of the buyer on the seller $T_{b \to s}$ is computed using the subjective trust logic [66]. Subjective logic is a framework for probabilistic information fusion that operates on subjective beliefs about the environment. It formulates a participant's reputation score based on historical interactions. Subjective logic uses the term "opinion" to denote the representation of a subjective belief, and models positive and negative statements and uncertainty [67]. We use it in our reputation formulation to automatically assess subjective trust without requiring feedback from the buyer. This allows the system to mitigate bad-collusion and bad-mouthing attacks.

When a buyer b interacts with a seller s, an opinion denoted by $O_{b \to s} = (t, d, u)$ is given to express b's belief in the trustworthiness of s with t, d, and u representing trust, distrust, and uncertainty, respectively; $t + d + u = 1$ and t, d, $u \in [0, 1]$.

$$\begin{cases} t = (1 - u)\frac{m}{m+n} \\ d = (1 - u)\frac{n}{m+n} \\ u = 1 - I_f \end{cases} \tag{3.2}$$

where, m and n are the number of valid and invalid interactions between b and s, $I_f \in [0, 1]$ denotes the normalized interaction frequency (*i.e.*, The higher interaction

frequency results in a lower uncertainty). The interaction frequency between b and s is determined by the ratio of the number of times b interacts with s to the average number of interactions b has with other sellers.
The value of $T_{b\to s}$ after the i^{th} interaction:

$$T_{b\to s} = t + \psi u \tag{3.3}$$

where ψ is the uncertainty weight.

The objective trust value $T_{v\to s}$ depends on the results of the verification process and the measures used in that process. As mentioned before, according to the types of services or products provided on the platform, different metrics are required to verify and evaluate the provision of these services, such as compliance, quality, delivery time, etc. Our solution provides a versatile platform framework adaptable to various applications. Hence, the objective trust formula $T_{v\to s}$ may vary based on the application. Nonetheless, the calculation method remains consistent, relying on three primary factors: the presence of the delivery certificate, the interaction value/amount, and the time between interactions. The value and timing metrics are introduced to counter coordinated attacks. In such attacks, a buyer collaborates with sellers to increase their reputation by exploiting the affordability of the delivery certificate. This strategy involves purchasing a large number of low-cost products from the same or different sellers in a short period, which rapidly increases reputation. Therefore, it is imperative to link trustworthiness to the value and timing of interactions. The calculation formula of $T_{v\to s}$ is as follows [9]:

$$T_{v\to s} = C\ [\sigma_c + \sigma_t F_t + \sigma_a F_a] \tag{3.4}$$

$$\sigma_c, \sigma_t, \sigma_a \in [0,1]\ ;\ \sigma_c + \sigma_t + \sigma_a = 1$$

where, C is a boolean that refers to the presence of the certificate '1' or not '0', σ_c is the weight of the certificate itself. σ_t and σ_a are the weights of the time t and the amount a of the interaction, respectively. F_t and F_a are the functions that normalize t and a, respectively (F_a, $F_t \in [0,1]$). They both have a positive correlation with t and a. Note that additional evaluation measures can be added to this formula depending on the context.
The combination of objective and subjective trust results in the overall trust value of the seller at the ith interaction. The same model will be used to calculate the trust value of the buyer T_i^b.

3.4.0.2 Reputation Update

In GuRuMarket, each new participant is assigned an initial reputation value R_{init}. This value is assumed to be the critical threshold of trust T_{min} so that all participants whose reputation is lower than R_{init} are considered untrustworthy. We update a trader's overall reputation after each real-world interaction. We believe that the only way to demonstrate good intentions is through proper real-world interaction. Online interactions managed by our platform are instantaneous and negate the need for mutual

trust between participants. In contrast, in real-world interactions, buyers rely on sellers to trust them for post-payment services. For example, if the seller fails to deliver, the buyer will end up wasting time and incurring additional fees to secure a replacement. Therefore, within GuRuMarket traders' reputation scores improvement relies on their successful real-world interactions with one another.

Let us now explain the updating process, the reputation model is triggered after each offline interaction to reevaluate the overall reputation score of both parties. First, it computes the trust value T_i as described in formula (3.1), then it uses this value to update the overall reputation score of the participants involved in this interaction as follows [9]:

$$R_{new} = \begin{cases} (1-\omega)R_{old} + \omega.T_i & T_i \geq T_{min} \\ \omega R_{old} + (1-\omega)T_i & T_i < T_{min} \end{cases} \tag{3.5}$$

$$\omega = F(T_i, N_b) = \kappa T_i \frac{1 - e^{-\lambda.N_b}}{1 + e^{-\lambda.N_b}} \tag{3.6}$$

where R_{old} is the old reputation value, *i.e.* before the last interaction, ω is a weighting function and N_b refers to the number of blocks added to the ledger since the trader joined the network.

The variable weighting factor ω ensures that older participants have increased opportunities to enhance their reputations without granting them the freedom to misbehave. Thus, the longer the users stay in the system, the more likely they are to improve their reputation, provided they maintain correct behavior. The maximum value of the weighting, which can be reached when N_b becomes significantly high, is $\kappa \in [0, 1]$. Moreover, since ω does not depend only on N_b but primarily on the trust score T_i, any misconduct indicated by a low T_i ($T_i < T_{min}$) will exert a significant impact on the overall reputation score, regardless of the value of N_b.

3.5 Guarantee and Reputation-based Trading Logic

After introducing the GuRuMarket framework, which primarily consists of independent participants and their nodes, this section presents the trading process within this framework. This process is designed to protect traders from potential malicious actions.

GuRuMarket is an open marketplace where participants act independently, free to buy and sell products and services. This means participants can be buyers, sellers, or both. This process allows for parallel interactions with multiple traders, without restrictions.

As described in section 3.3, each new participant must register with the system organization. Then, upon successful registration, participants are given the ability to set up their individual "digital wallet" for performing various trading operations. Additionally, the organization offers users the option to refill their wallets with stablecoins, *i.e.* 1 coin is equivalent to 1$.

To interact with the system participants must submit transactions through their digital wallets. The general structure of a transaction in GuRuMarket is described as follows:

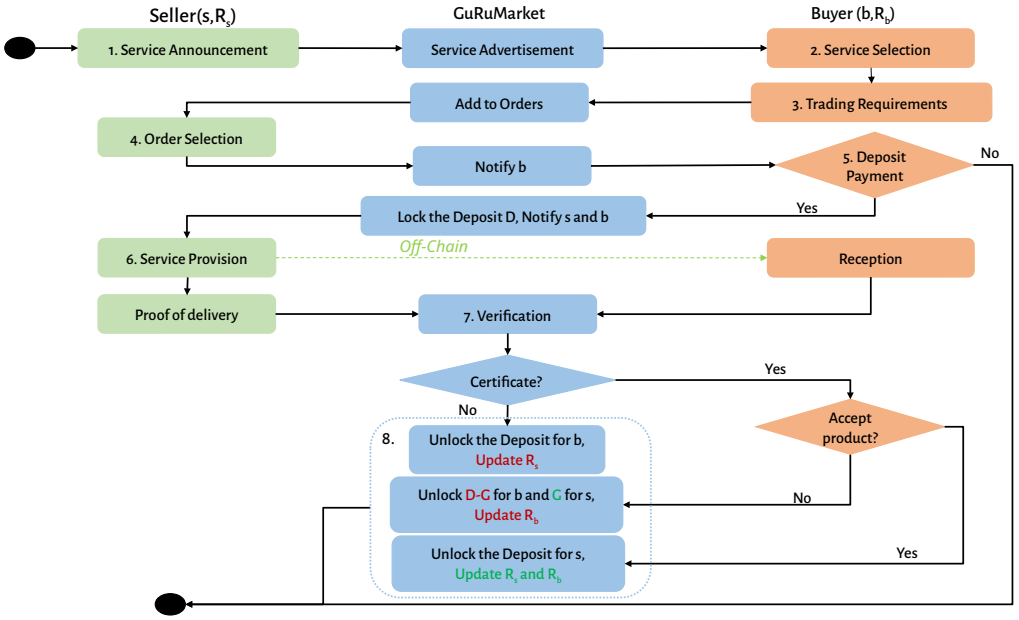


Figure 3.3: The complete trading process

$$TX_x = \mid \text{from} \mid \text{to} \mid \text{value} \mid \text{nonce} \mid data_x \mid \text{fee} \mid$$

where $from$ is the initiator of the transaction, to is the receiver, $value$ is the amount of money to be transferred, $nonce$ is the transaction sequence number and fee indicates the amount the sender must pay as a transaction fee. Besides, the $data_x$ field of the transaction has the following general format:

$$data_x = \mid txname_x \mid (TX_x \text{ input data}) \mid$$

where $txname_x$ is the hash of the transaction name/signature (*i.e.* the invoked SC function name and argument types), (TX_x *input data*) is the hash of transaction inputs (that varies depending on the transaction type x).

The transaction object needs to be signed using the sender's private key.

We explain hereafter the complete trading process in the system, depicted in Figure 3.3. The process is divided into eight steps to secure the trading interaction and guarantee the service delivery.

1. *Service Announcement:* A seller who wishes to add a new product/service can simply invoke the *createProduct* function via his user interface. The resulting transaction will include the price of the service to be provided and the guarantee proportion P_g that the seller is willing to obtain if the corresponding service is rejected by the buyer after delivery. The transaction is then propagated to all nodes via the network layer for verification. If the transaction is deemed valid, it will be stored in the BC and the list of available services will be updated. Note that sellers can fill in the expected

selling price and guarantee in their announcement freely. The specific structure of the data field of service/product creation transaction TX_{crP} is:

$$data_{crP} = \mid \text{createProduct} \mid (\text{productID, Price, } P_g) \mid$$

Details about the product can be stored using an off-chain storage system like IPFS [75] to reduce the cost and enhance storage efficiency. The resulting hash, known as the content identifier (CID), is unique and can serve as the product identifier (productID).

2. *Service Selection:* Buyers are free to pick among the services proposed by different sellers. However, it's important to emphasize that the most affordable service may not always be the optimal choice. Buyers should take into account factors such as the proposed guarantee P_g and the seller's reputation when making their decision.

3. *Trading Requirements:* Once the buyer has identified the service he/she wishes to proceed with for trading, he/she must generate a trade request transaction containing his/her trading requirements *e.g.* order quantity and delivery type. The specific structure of the data field of this transaction, which we call the *createOrder* transaction TX_{crO} is:

$$data_{crO} = \mid \text{createOrder} \mid (\text{productID, Trading Requirements}) \mid$$

4. *Order Selection:* Similar to buyers, sellers will have the flexibility to accept buyer requests received within a specified timeframe. Provided the seller can fulfill the orders, they may opt to engage with multiple buyers simultaneously to complete the necessary transactions. The format of the data field in the order acceptance transaction is outlined below:

$$data_{acO} = \mid \text{acceptOrder} \mid (\text{orderID}) \mid$$

5. *Deposit Submission:* The buyer should pay the deposit amount D for its accepted order. The seller and the buyer will then receive a notification confirming the deposit submission. The deposit transaction must include the buyer's and the seller's addresses and the deposit amount. It must be recorded in the BC to prove that the buyer has paid the deposit amount for this particular order. Alternatively, if the seller does not receive the confirmation notification after a certain period, the corresponding order will be rejected automatically. The deposit is held in an escrow smart contract until the process is complete. The smart contract acts as a trusted intermediary, holding the funds until specific conditions are met. The specific structure of the data field of the *depositPayment* transaction TX_{deP} is:

$$data_{deP} = \mid \text{depositPayment} \mid (\text{orderID, } Adr_S, Adr_B, \text{Deposit}) \mid$$

6. *Service Provision:* Once online trading between both parties has been completed, the following step is the service provision by the seller. This step is an off-chain process, which takes place in the real world. The results of this process will be verified and validated in a decentralized way using the verification module.

7. *Service Verification:* Upon completion of the ordered service, the verification module within the nodes linked to the trading parties must receive ample evidence from the buyer, the seller, and an external controller. Consequently, the module can generate a certificate indicating whether the service has been correctly provided. It's essential to note that we presume this certificate is immune to tampering.

   The verification module must submit this certificate to the BC for on-chain verification and storage. The presentation of a valid certificate followed by the buyer's acceptance of the delivered product will then trigger the reputation sub-module, which will evaluate the interaction and update the reputation scores of both parties (positive update). It also triggers the guarantee sub-module, which releases the funds deposited for the benefit of the seller. On the other hand, the absence of a certificate is a clear indicator of inappropriate behavior on the part of the seller and will result in a significant loss of reputation (negative update). The specific structure of the data field of *validateOrder* transaction denoted by TX_{vaO} generated by the verification module is as follows:

$$data_{vaO} = \mid \text{validateOrder} \mid (\text{orderID, deliveryCertificate}) \mid$$

8. *Deposit Release:* At this point, we have three possible scenarios:

   (a) The seller did not deliver the service correctly (no certificate is presented for the on-chain verification); the guarantee sub-module will release the deposit amount to the buyer.

   (b) The service is correctly delivered, but the buyer refuses to take it; in this case, the deposit will be divided into two parts according to the pre-established agreement (release D-G for the buyer and G for the seller).

   (c) The seller has correctly delivered the service (a valid certificate is presented) and the buyer accepts it; the deposit amount will be transferred to the seller.

   The specific structure of the *depositUnlocking* transaction TX_{deU} data field is:

$$data_{deU} = \mid \text{depositUnlock} \mid (\text{orderID, deliveryCertificate}) \mid$$

3.6 Lightweight Reputation-based Consensus

In this section we discuss the consensus mechanism employed by nodes in GuRuMarket. This mechanism is crucial for the network to reach an agreement on the state of

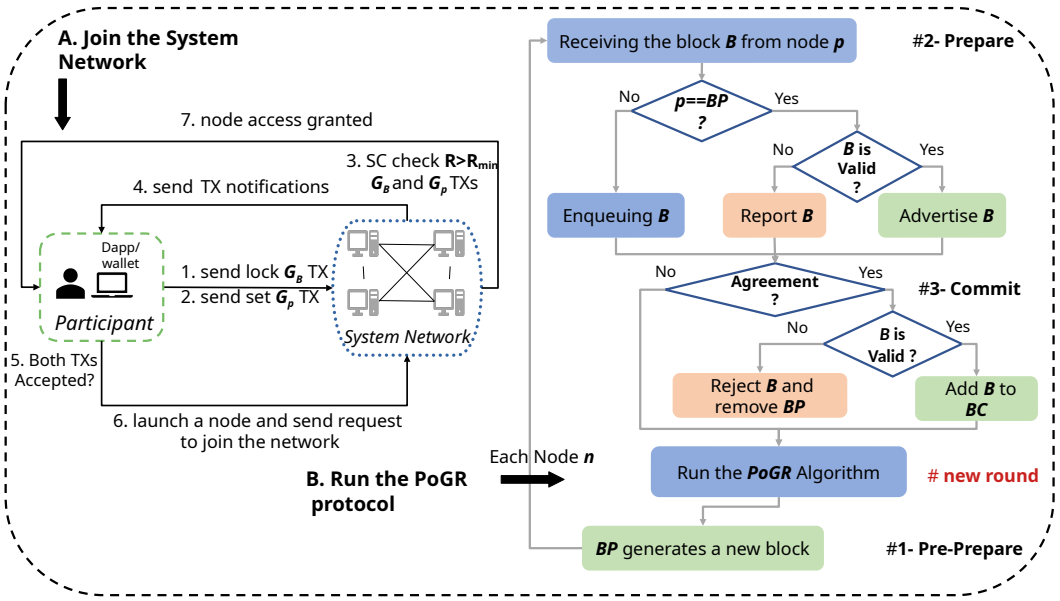


Figure 3.4: PoGR consensus protocol

the ledger, particularly in selecting the block for addition, ensuring a consistent view of the ledger. To function optimally, this protocol must ensure safety, validity, and liveness defined in Chapter 2. Additionally, maintaining fairness and cost-effectiveness is essential for achieving decentralization and enhancing scalability. To achieve this, we define two additional properties for our reputation-based consensus.

- *Score consistency:* At the same index/round, any two honest nodes must have the same score for any other node.

- *Fairness:* (Partial) No single entity or group should have excessive advantage or control over the update process.

To ensure the above properties, we present *Proof-of-Guarantee&Reputation (PoGR)* a lightweight consensus mechanism that incorporates a scoring method to enable Block Producer (BP) selection. Figure 3.4 depicts the complete PoGR protocol. PoGR reduces the computations required by running nodes, thereby lowering overall consensus cost. In particular, by leveraging the reputation scores of participants, PoGR achieves reputation consistency at a minimal cost. The scoring technique addresses the update monopolization problem, by improving the fairness of the network. The PoGR consensus leverages GuRuMarket's utilization of guarantee and reputation management, making it perfectly tailored to our platform. However, this does not preclude its applicability in different contexts with alternative reputation models [9, 31].

3.6.1 Join The System Network

A participant who wants to join the system network must run a node that executes the PoGR protocol and thus attempts to create blocks according to this protocol. To do this, the participants have to submit two transactions, TX_{G_B} to lock a deposit $G_B \in [G_{min}, G_{max}]$ as a guarantee balance and TX_{G_p} to specify the effort or risk they are willing to take $G_p \in [G_{min}, G_B]$ as a proportion to G_B to get the updating right. Both transactions TX_{G_B} and TX_{G_p} are sent to the blockchain network for verification and validation (Step 1-4 in Figure 3.4). The participant will not be able to run the consensus instance and create updates unless the following two conditions are satisfied: both TX_{G_B} and TX_{G_p} transactions are valid, and his/her reputation is higher than the consensus threshold R_{min} (Step 5-7 Figure 3.4). Each node running the PoGR protocol locally maintains a list of all operating nodes and their scores. The list is updated after each consensus round to keep the scores up-to-date.

3.6.2 Run The PoGR Protocol

The proposed protocol uses a scoring technique to refresh a list of candidate $listBP$ for producing a block for a given round. In particular, each node updates the list of candidates $listBP$ at the start of the round before selecting a leader BP using the following formula:

$$S_n = [(\alpha.G + \beta.E + \gamma.R).\tau]_n \ ; \ \alpha + \beta + \gamma = 1 \tag{3.7}$$

where, S_n is the score of the node n, $G = G_B/G_{max}$ is the proportion to the maximum amount the participant wants to lock as a guarantee balance, $E = G_p/G_B$ is the proportion to the balance that represents the Effort/Risk taken by the participant, R is the reputation score of the participant who manages the node. Finally, $\tau_n = F(N_{gb})$ is the available generation rate initially equal to 1. It decreases exponentially with the number of blocks N_{gb} generated during a block cycle B_c, which refers to the period in blocks needed for a block producer to recover 100% of its producing right. For example, with $B_c = 5$ and a node n that generates a new block at 10^{th} index and is its first block in the last five blockss. Node n must wait at least five rounds to recover 100% of its available generation rate τ_n (until 15^{th} index). In other words, τ_n will only be reset to 1 if n does not generate any of the next five blocks. Node n in this scenario will have $\tau_n = 1/exp(2, 1) = 1/2$ (with a $base = 2$ and $N_{gb} = 1$, since 10^{th} was its first block). The value τ_n will continue to decrease exponentially with the number of blocks N_{gb} generated by n during this period.

The protocol consists of three phases: score refresh, block generation, and block validation.

1. **Scores Refresh:** At the beginning of each index i, each node n refreshes the scores of all nodes in the network using $S_n = [(\alpha.G + \beta.E + \gamma.R).\tau]_n \ ; \ \alpha + \beta + \gamma = 1$ (Run the proposed Algo 1). Once all the nodes' scores have been updated, the Algo selects the node with the highest score as the next block producer BP.

Algorithm 1: Scores Refresh & Leader Selection.

Data: B_c, $listNodes$, G_{min} , G_{max} , CL
Result: selection of the next BP

```
1  begin
2      BP ← receivedBlock.getBP()
3      if ! receivedBlock.isValid then
4          Remove <BP, S_BP > from listBP
5      else
6          S_BP = computeScore(BP)
7          Call updateBPList(BP,S_BP)
8          foreach node n in listNodes do
9              if exists a TX of n in receivedBlock
10             or n has a Block in the last B_c indexes
                then
11                 S_n = computeScore(n)
12                 Call updateBPList(n,S_n)
13     Set next BP the node with the
14     highest score
15 Function updateBPList(n, S_n):
16     if n ∉ listBP then
17         Insert < n, S_n > in listBP
18     else
19         Update <n, S_n > in listBP
20 Function computeScore(n):
21     Read ( R, G_p, and G_B ) of n from CL
22     if R < R_min or G_p ∉ [G_min G_B] then
23         Remove < n, S_n > from listBP
24     else
25         N_gb ← 0, i ← 0
26             ▷ get the current block Header
27         Header ← CL.blockHeader
28         while i < B_c and ! Header.isNil do
29             if Header.getBP() == n.adr then
30                 N_gb = N_gb + 1
31             Header ← CL.getPrvHeader()
32             i ← i + 1
33         τ_n = 1/ exp(base, N_gb)
34         S_n = [(α.G_B/G_max + β.G_p/G_B + γ.R).τ]_n
35     return S_n
```

2. **Block Generation:** (a.k.a. Proposal or **Pre-prepare** Phase) The selected leader BP ($pk_n = pk_{BP}$) proceeds by generating a block $\mathcal{B}$, signing it, and triggering a broadcast protocol. The other node waits for the $\mathcal{B}$ block.
3. **Block Validation:** In two steps: (i) **Prepare Phase:** each node n, upon receiving the block $\mathcal{B}$ from the node p with address pk_p, first checks the validity of the block header (the signature matches $pk_p == pk_{BP}$), then the validity of entire block $\mathcal{B}$. If $\mathcal{B}$ is valid, then the node propagates it, with a positive vote (advertise $\mathcal{B}$). Otherwise, it reports the block $\mathcal{B}$. (ii) **Commit Phase:** If the nodes agree on the validity of $\mathcal{B}$, they commit to $\mathcal{B}$ and move on to the next index. Otherwise, the block $\mathcal{B}$ is rejected and the leader BP is automatically removed from the process.

3.6.3 GuRu-based Committee Selection

In PoGR, we employ a dynamic committee selection based on guarantee and reputation to foster fairness and decentralization. The composition of this verification committee is periodically refreshed every k rounds (k is defined by the consortium). At the end of each period, the current leader runs Algo.2 to select V new nodes for the next period. The leader begins the election process by generating a random number r_1 and its corresponding proof π using a Verifiable Random Function (VRF) [76]. The seed used in the function is derived from the previous block's hash to resist manipulation. The leader

then iteratively hashes r_1 $V-1$ times to produce a set of V random numbers, denoted as $\mathcal{R} = \{r_1, ..., r_2\}$. The probability of a node being elected is proportional to its weighted score. The scores of all nodes are represented as L indexed units, $S_{spread} = \{s_i, ..., s_L\}$. For each r_i, randomly select a unit, s_{Idx}, from S_{spread}. The node owning s_{Idx} is then elected to the committee. This will guarantee that nodes with higher weighted scores are more likely to be chosen.

Algorithm 2: GuRu-based Committee Selection.

Data: Indexed score units $S_{spread} = \{s_i, ..., s_L\}$, number of nodes V, previous block $\mathcal{B}$.
Result: selection of V nodes for the next period

```
begin
    Generate seed from the previous block: seed ← hash(B);
    Compute VRF random number and proof: < r_1, π > ← VRF_sk(seed);
    Produce V − 1 additional random numbers by hashing r_1, V − 1 times to get
      R = {r_1, ..., r_V};
    foreach r_i ∈ R do
        Compute Idx = r_i mod L;
        Append the node n_i with score unit at Idx to listVerif;
    return listVerif, π ;
```

In the PoGR protocol, nodes who successfully validate blocks are compensated for their efforts. The system accumulates transaction fees and distributes them proportionally among validators. The parameters G_{min} and G_{max} define the minimum and maximum reward thresholds, which can be adjusted over time. If a block producer generates an invalid block, a portion of its deposited guarantee, denoted by G_p ($G_p \geq G_{min}$), is forfeited. This amount G_p, is then distributed to other validators as compensation. Nodes with a remaining guarantee (G_B) less than G_{min} or a reputation score less than the current threshold R_{min} are automatically and temporarily excluded from further participation in the process. Note that, the permanent expulsion of nodes and the adjustment of R_{min}, G_{min}, and G_{max} is done by the consortium through majority voting.

3.7 Security Analysis

In this section, we first discuss the security properties of our consensus. Then detail the potential threats in GuRuMarket and show how it can withstand these attacks.

3.7.1 Consensus Properties

- **Lemma 1** *(Safety). If the broadcast protocol satisfies agreement, the PoGR satisfies safety.*
  Proof. Let us assume for the sake of contradiction that PoGR does not satisfy safety. That is, there exists an index i where two honest nodes commit different blocks $\mathcal{B}$ and $\mathcal{B}'$. Given the scoring mechanism, for each honest node and Byzantine node, the score of the honest node is not less than that of the Byzantine node. Since there are more than half of honest nodes in the network, they control more than half of the voting power. Under this setting, the broadcast protocol satisfies the agreement.

Therefore, if they commit to $\mathcal{B}$ and $\mathcal{B}'$ for index i, then $\mathcal{B} = \mathcal{B}'$, which contradicts our assumption. Thus, to break the safety of PoGR, an attacker must corrupt more than half of the voting power.

- **Lemma 2** *(Liveness). If the broadcast protocol satisfies termination, then the PoGR satisfies liveness.*
  Proof. For the sake of contradiction, suppose that PoGR does not satisfy liveness. That is, there exists a transaction TX that is received by an honest node at index i but is not committed in every honest node's local ledger. The leader BP is either honest or Byzantine. If the BP is honest, it will propose a block containing this transaction, and since the broadcast protocol satisfies termination, more than half of the honest voting power will commit to this block. If BP is Byzantine, it will either 1) withhold the block, or 2) propose multiple conflicting blocks. In both cases and upon termination, nodes will commit an empty block and move to the next index. Thus, to break the liveness of PoGR, one must corrupt more than half of the voting power or break the broadcast protocol termination.

- **Lemma 3** *(Score Consistency). If PoGR satisfies agreement and termination, then when a new block is added, any two honest nodes must have the same score for any other node.*
  Proof. Assume, for the sake of contradiction, the existence of two nodes n_1 and n_2 such that the score of a node n_3, S_3 on n_1 is not equal to S'_3 that on n_2 when the block is added (at the end of the block epoch). Since PoGR satisfies agreement and termination, in each index, any two honest nodes commit the same block proposed by the leader if the block is not empty. Thus, they have the same view of the ledger. This means the same score for every other node, which contradicts our assumption. Therefore, to break the score consistency of PoGR, an attacker must break the agreement or termination of the protocol.

- **Lemma 4** *(Partial Fairness). If the broadcast protocol satisfies agreement and termination, then the PoGR satisfies partial fairness.*
  Proof. For the sake of contradiction, suppose that PoGR does not satisfy partial fairness. That is, there exists a small set of nodes that exclusively control the update process (*i.e.*the same leaders are elected at multiple consecutive indexes). Given the scoring method used in PoGR, which involves three different parameters (E, G, and R) in conjunction with the available generation rate. Since more than half of the nodes are honest, then honest nodes control more than half of the voting power. In this setting, the broadcast protocol satisfies agreement and termination. Therefore, honest nodes update the available generation rate and the three factors correctly for all nodes at each index. This means that a new leader is elected after each index for at least B_c rounds, which contradicts our assumption. Thus to control the update process, an attacker must break the agreement or termination of the protocol.

While the score consistency property is easily achieved thanks to protocol agreement and termination and by using reputation scores maintained via smart contracts, the

fairness property is partially achieved depending on factors such as the block cycle and the number of nodes in the network. To improve such partial fairness, the scoring formula (3.7) used to evaluate node candidates must satisfy the following two conditions:

- **Reputation should be prioritized over the guarantee** ($\gamma \geq \alpha, \beta$) for two main reasons. Firstly, reputation explicitly indicates the level of trust the system places in the participant. Secondly, while participants control their guarantee balance and can decide the portion they wish to use, the reputation score is entirely governed by the system. In other words, participants have no influence over the calculation of their reputation.
- **The effort (E) should be prioritized over the guarantee balance (G)**. Formula (3.7) must adhere to this condition because, in PoGR, a higher effort ($G_p \approxeq G_B$) implies a greater risk of leaving the consensus. Consider two participants, P_1 and P_2, who have the same reputation R but have not yet generated a block. Participant P_1 has a low guarantee balance ($G_{B_1} \approxeq G_{min}$), requiring a high effort ($E_1 \approxeq 1$). In contrast, P_2 has a very high guarantee balance ($G_{B_2} \approxeq G_{max}$) but prefers to take a lower risk, resulting in $E_2 = G_{p_2}/G_{B_2} \leq \frac{G_{B_2}-G_{min}}{G_{B_2}} = \frac{G_{max}-G_{min}}{G_{max}}$. Therefore, P_1 should receive at least the same score as P_2 because if P_1 produces a malicious block, their node will be automatically removed, as their remaining guarantee balance (G_{B_1}) will fall below G_{min}.

$$
\begin{aligned}
S_{P_1} \geq S_{P_2} &\Rightarrow \alpha.G_1 + \beta.E_1 \geq \alpha.G_2 + \beta.E_2 \\
&\Rightarrow \alpha.\frac{G_{min}}{G_{max}} + \beta.1 \geq \alpha.1 + \beta.\frac{G_{max} - G_{min}}{G_{max}} \\
&\Rightarrow \alpha.\frac{G_{min}}{G_{max}} \geq \alpha - \beta.\frac{G_{min}}{G_{max}} \Rightarrow \frac{G_{min}}{G_{max}} \geq \frac{\alpha}{\alpha+\beta} \\
&\Rightarrow \frac{G_{min}}{G_{max}} \geq \frac{\alpha}{\alpha+\beta}
\end{aligned} \tag{3.8}
$$

The designed mechanism aims to ensure the reliability of blockchain updates without requiring participants to spend more money, as implicitly guaranteed by condition (3.8). It incentivizes participants to take on greater risks rather than make larger financial investments. Since a malicious update with higher risk results in node expulsion, this approach is likely to eliminate all malicious nodes, thereby enhancing the overall efficiency and reliability of the system.

3.7.2 Common Attacks

- **Sybils attack:** GuRuMarket participants cannot create multiple accounts; only authentic users are allowed to join the system. The organizational (or consortium) admin accounts oversee system access on-chain via smart contracts. This protocol is established on-chain using contract code, with exclusive authorization granted to admin accounts for adding or removing participants from the system.

- **Double spending attack:** We have two cases according to the attacker role:
  Case-1: The attacker is a seller. → The seller must prove the deliverance of the accepted order. If he/she does not have the required quantity or tries to sell products he/she does not have at all, he/she will end up losing the deposit and taking a huge reputational hit.
  Case-2: The attacker is a buyer. → The system network is responsible for verifying all transactions, and therefore only one TX will be accepted and stored in the DL, *i.e.* the first TX included in a valid block. As a result, the corresponding seller can check its validity and start the service delivery, and there will be no delivery for the invalid deposit payments. It is important to note that in GuRuMarket, TXs reaches finality within a block. Thanks to the agreement phase, the absence of forks in GuRuMarket eliminates any risk of transaction removal.

- **Whitewashing attack:** Participants are registered with the system via the consortium. They can only join the platform if the organizations (administrators) authorize them. This is done through a majority vote.

- **Bad-collusion attack:** A trader's reputation computation depends mainly on the collected proofs that allow the certificate generation. Therefore, the system can resist this attack by giving more importance to objective trust. Furthermore, the primary source of this attack, feedback, is replaced in our model by an automatic evaluation using subjective trust logic.

- **Malicious updates:** In PoGR design, selected block producers cannot create malicious updates. A newly generated block is received and checked by all committee nodes. If the nodes agree on the invalidity of this block, all nodes in the system will immediately remove the block producer from the list of validators. Furthermore, the participant who runs this node will face punishment and lose his deposit (the amount fixed by G_p).

3.8 Evaluation and Results

In this section, we first present the business model formulation of the proposed platform in the context of Hyperledger Besu[2] followed by the experimental setup. Next, we discuss the evaluation results of the platform in terms of the effectiveness of the trust and reputation model, the fairness of PoGR, the overall system scalability, and performance. We conclude with an analytical comparison of our proposal with some existing solutions using several metrics.

3.8.1 Business Model

We implement the proposed platform framework on Hyperledger Besu which is an open-source Ethereum client developed under the Apache 2.0 license and written in

[2] https://besu.hyperledger.org

Table 3.1: Evaluation environment.

Machine	#
Intel® Core^{TM} i7-6820HQ CPU @ 2.70GHz × 8 ; 16GiB	3
Intel® Core^{TM} i7-8700 CPU @ 3.2GHz × 12 ; 32GiB	1
Intel® Core^{TM} i7-4710MQ CPU @ 2.50GHz × 8 ; 16GiB	1
Intel® Core^{TM} i7-4800MQ CPU @ 2.70GHz × 8 ; 32GiB	1
Intel® Core^{TM} i7-6500U CPU @ 2.50GHz × 4 ; 16GiB	1
Intel® Core^{TM} i7-6700HQ CPU @ 2.60GHz × 8 ; 16GiB	1

Java. Besu includes a command line interface and a JSON-RPC API to run, maintain, debug, and monitor nodes in an Ethereum network. The API can be used via RPC over HTTP or WebSockets. The API supports typical Ethereum features such as Smart Contracts and Decentralized application development (DApp). For the evaluation, as described in section 3.5 we define a trading logic that includes:

- *Participants:* Traders (Sellers and Buyers) and Admins of the organization.
- *Assets:* A data structure that can represent any real-world service or product.
- *Smart Contracts:* There are three types of smart contracts used to develop the business model: Access Control Smart Contract (on-chain permissioning[3]); it provides the functionality of adding and removing participants and nodes by admins. Trading Smart Contract which handles all the trading operations. Reputation and Guarantee Smart Contract, managing trust calculations, and reputation updates, as well as handling deposit submissions and withdrawals.

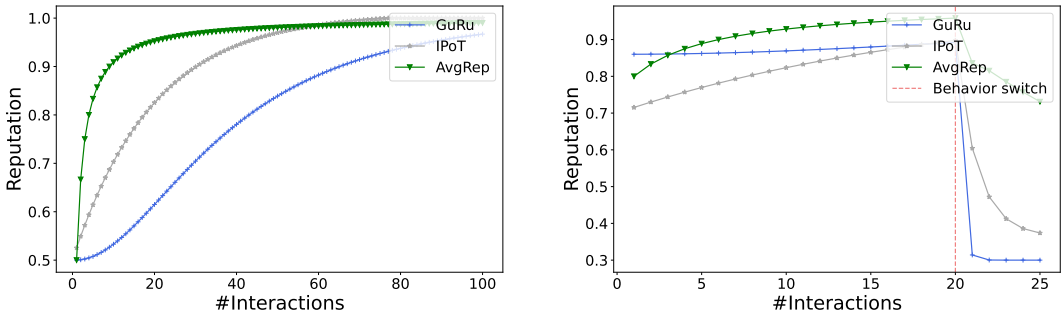
3.8.2 Experimental Setup

The deployment of the overall system platform and the evaluation tests are carried out on a cluster of eight PCs whose characteristics are given in Table 3.1. We develop the smart contract within GuRuMarket using Solidity. We then use Hyperledger Caliper[4] to run the evaluation experiments. Additionally, we analyze the fairness of our consensus protocol based on the logs of the running nodes.

3.8.3 Reputation Model Effectiveness

We begin the discussion with results that demonstrate the effectiveness of our trust and reputation model. Figure 3.5a shows the ideal reputation growth of a participant in our system ($PoGR$) compared to the referenced one ($IpoT$) [2] which uses the same logic to compute reputation (interactions) and provides better results than the traditional models including the aggregated average reputations model ($AvgRep$). We observe that the reputation score R increases as the number of interactions increases in both models. However, we note that the growth of R in our model is slower than in the other.

[3] https://github.com/ConsenSys/permissioning-smart-contracts
[4] https://github.com/hyperledger/caliper-benchmarks



(a) Ideal reputation growth of a normal participant (b) Reputation changes of a malicious participant

Figure 3.5: Effectiveness of the Trust and Reputation Model

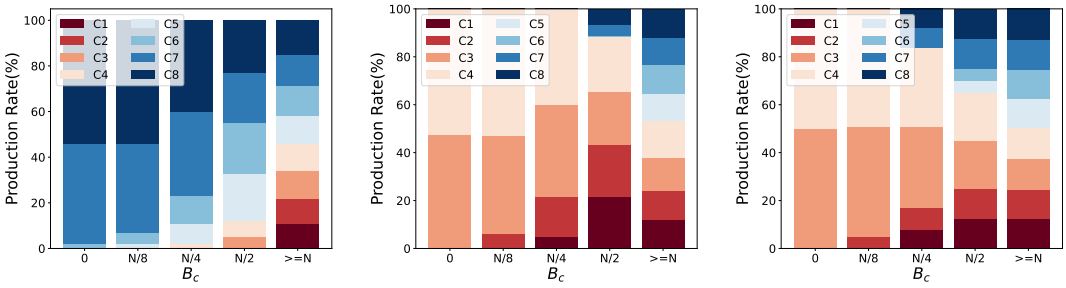
Therefore, the participant in our system needs to perform more interactions to reach the maximum score. The reason behind this is that, in our reputation model, we select the update formula based on the trust value of the last interaction. The model gives less relevance to T_i if its value is above the critical trust line T_{min} because it reflects the expected behavior. On the other hand, if the trust value is below the threshold, it will be accorded considerable importance compared to the old reputation value R_{old} as it indicates unsuitable behavior. Figure 3.5b shows the response of both systems to improper behaviors. To trick the system, a malicious participant will continue to behave correctly for a while and then try to do something wrong. As the figure shows, after detecting an anomaly, the reputation score decreases rapidly in both systems. However, what we notice is that the score drops faster in ($PoGR$). This is because our system has a high sensitivity to improper behaviors. Overall, participants in our system need more time and interactions to achieve a high reputation, and as soon as they misbehave, they will receive a reputation hit that will lower their score below the critical line. As a result, other participants will consider them untrustworthy, and it will be difficult for them to regain their reputation.

Table 3.2: Hyperparameter's Configuration.

Parameter	**Value**
# Blocks N_b	200
# Nodes N	$[8-32]$
R_{init}	0.5
R_{min}	0.7
G_{min}	25
G_{max}	100
α	1/4
β	3/8
γ	1/2

Table 3.3: Fairness evaluation scenarios.

Scen.	**Setup**
1^{st}	$R \in [0.7-1]$ and $E = 1, \forall c \in [c1, c2, \ldots, c8]$
2^{nd}	$R \in [0.7-1]$, $\forall c \in [c1, c2, \ldots, c8]$ $E = 1, \forall c \in [c1, c2, c3, c4]$ $E = 0.75, \forall c \in [c5, c6, c7, c8]$
3^{rd}	$R \in [0.85-1]$, $\forall c \in [c1, c2, c3, c4]$ $R \in [0.7-0.85], \forall c \in [c5, c6, c7, c8]$ $E = 1, \forall c \in [c1, c2, \ldots, c8]$



(a) Scenario 1. Impact of $\mathcal{T}$ (b) Scenario 2. Impact of $\mathbf{E}$ (c) Scenario 3. Impact of $\mathbf{R}$

Figure 3.6: Impact of the consensus parameters on the block production rate.

3.8.4 PoGR Fairness

Having demonstrated the effectiveness of our reputation model, we now empirically assess the fairness of the PoGR consensus. PoGR uses a scoring method to reduce computation and ensure fairness among running nodes. Through our evaluation here, we aim to show the scoring formula parameters' impact on the block producers' selection. Therefore, showing the impact of each metric on the block production rate at each node. We conduct several experiments with different configuration scenarios to evaluate the proposed approach. We define two possible node classes based on their reputation score $R \in \{High : [0.7-0.85], VeryHigh : [0.85-1]\}$ and four classes based on the guarantee balance $G_B \in \{low : [25-50], medium : [50-70], high : [70-85], veryhigh : [85-100]\}$. We run three different scenarios several times to show the impact of each parameter separately.

- The 1^{st} scenario involves node classes with uniform reputations ($[0.7-1.0]$) taking full risks, differing only in G_B. This basic setup aims to demonstrate the impact of the available generation rate τ.
- In the 2^{nd} scenario, classes are split into two groups: $[c1-c4]$ with high risk and low-medium budget G_B, and $[c5-c8]$ with partial risk and high-veryhigh budget G_B, illustrating the impact of E.
- The 3^{rd} scenario explores the impact of R. Similar to the previous scenario, node classes are divided into two groups based on their reputation: $[c1-c4]$ with Very High reputation and low-medium G_B, and $[c5-c8]$ with High reputation and high-veryhigh G_B.

The evaluation parameters are summarized in Table 3.2 and the configuration of each scenario in Table 3.3. The average results obtained for the 1^{st}, 2^{nd}, and 3^{rd} scenarios are shown in Figures 3.6a, 3.6b, and 3.6c, respectively.

Figure 3.6a shows the production rates achieved for each class after running the 1^{st} scenario. The results highlight the influence of the block cycle (B_c) on the block generation rate. When a small B_c is set, meaning rapid restoration of full production right ($\tau = 1$),

only nodes with vary high G_B values (c_7 and c_8) can generate blocks. Conversely, in the case of $B_c \geq N$, all consensus nodes show almost uniform block generation rates. This is due to the PoGR selection approach, which depends mainly on the available generation rate (τ). More precisely, it depends on the time required for nodes to fully recover their initial τ. A longer B_c implies an extended recovery period for nodes that have previously generated blocks, thereby increasing the likelihood of other nodes generating the next blocks.

Figures 3.6b and 3.6c illustrate a similar impact of the block cycle (B_c) as observed in the 1^{st} scenario. However, the results of the 2^{nd} scenario show that nodes with both high G_B and low effort (E), denoted as c_5, c_6, c_7, c_8, are unable to produce updates when a small B_c is used. Only nodes that have taken more risk can produce blocks, emphasizing the importance of E over G. Furthermore, the results of the 3^{rd} scenario show that even with modest balances (G_B), nodes with higher reputations (c_1, c_2, c_3, c_4) are more likely to generate updates, emphasizing the importance of R over G.

In summary, the PoGR logic we propose depends on the choice of B_c. A short B_c configuration favors a competitive environment, where nodes are required to improve their scores to earn the right to update. Conversely, a larger B_c creates a fair and non-competitive network in which each node patiently waits its turn to produce. Furthermore, even within a competitive setup, PoGR ensures a good degree of fairness by favoring the most reputable and risk-taking nodes (those with high R and E).

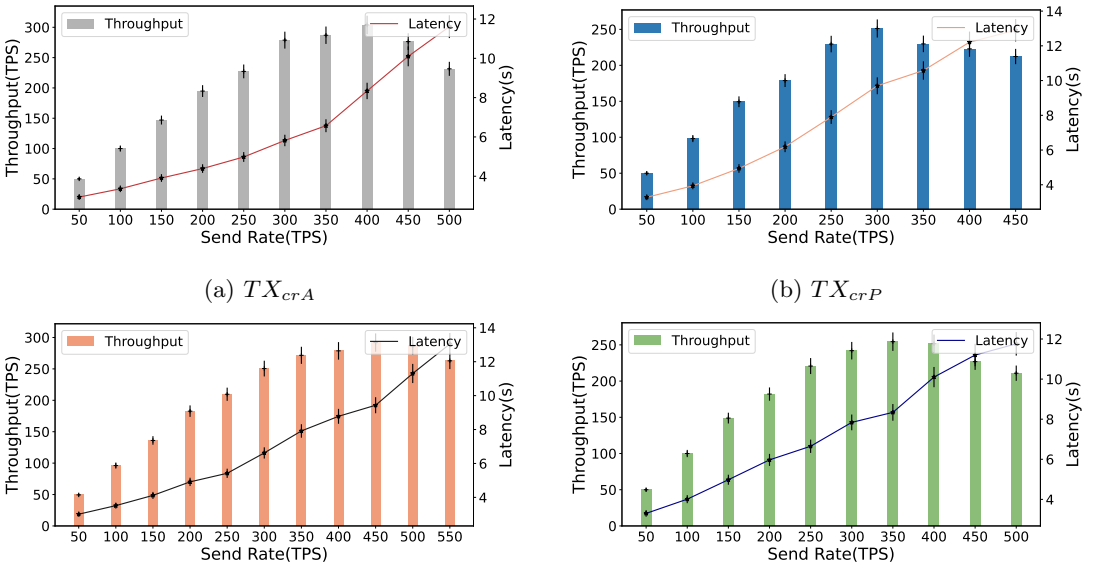
3.8.5 Performance and Scalability

After presenting the performances of PoGR, let us now discuss the results quantifying the performance of our entire system for relevant benchmarks using Hyperledger Caliper. We consider three metrics for GuRuMarket performance evaluation as described below:

- *Throughput:* the number of successful transactions per second (TPS).
- *Latency:* the time interval in seconds, between transaction submission and its completion.
- *Scalability:* the changes in throughput and latency when altering a configuration parameter, such as network size or node configuration.

3.8.5.1 Throughput and Latency

We start with a qualitative evaluation of GuRuMarket's throughput and latency. For this purpose, several experiments have been performed using Caliper. By setting the same network configuration and changing the transaction sending rate (50-550 TPS), we obtained the results shown in Figure 3.7. As mentioned in Section 3.3, each new participant must register with the organization to get access to the system. Each participant is uniquely identified by his/her address/public key. The generated transaction TX_{crA} involves the creation of a new account that initializes the participant's reputation score and trading parameters. Figure 3.7a shows the throughput and latency of TX_{crA}. The throughput increases as the transaction sending rate increases. It reaches its maximum



(a) TX_{crA}

(b) TX_{crP}

(c) TX_{crO}, TX_{acO}, and TX_{deP}

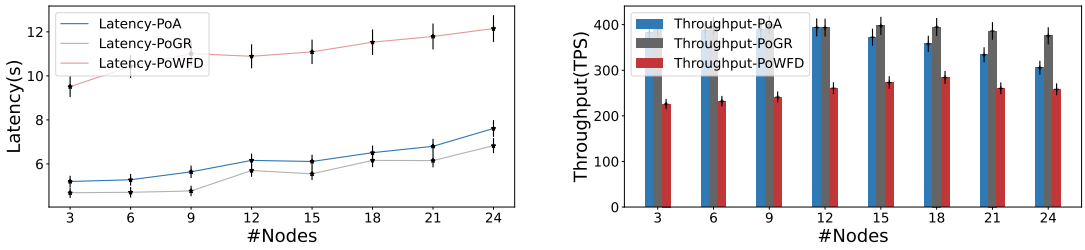
(d) TX_{upR} (T_i and R_{new})

Figure 3.7: Latency and Throughput of GuRuMarket.

value of $295TPS$ at a send rate of $400TPS$ and then starts to drop. Such a drop signals that the system is overloaded. For the TX_{crA} latency, it stays around the block period ($< 10s$) as long as the system is not congested, and increases after that.

Within GuRuMarket, each product is registered using TX_{crP} and traded using TX_{crO}, TX_{acO} and TX_{deP}. Figure 3.7b and 3.7c show the throughput and latency for creating and trading a product respectively. Similar to TX_{crA}, the throughput of both transaction types increases with the transaction send rate until the system reaches its limit. However, TX_{crP} has a lower maximum throughput ($251TPS$) compared to the average throughput of TX_{crO}, TX_{acO} and TX_{deP} ($391TPS$) because it stores a new data object, which is a more complex write transaction. This is in contrast to trading transactions, which involve simple transfer transactions and property changes.

The final set of evaluations covers trust and reputation computation. Figure 3.7d shows throughput and latency for TX_{upR}, including computations for T_i and R_{new} (presented in section 3.3.4). In the case of TX_{upR}, the latency remains relatively stable until the system approaches its limit, after which it increases significantly. The maximum throughput achieved by GuRuMarket for TX_{upR} is nearly identical to that for TX_{crP}, which is to be expected given the computational complexity involved in calculating T_i and R_{new}. Overall, the results of this study are consistent with previous research [77] emphasizing the performance dependency on the type of workload used.



(a) Latency comparison

(b) Throughput comparison

Figure 3.8: Latency and Throughput comparison between Ethach(PoW) and clique(PoA) and PoGR.

3.8.5.2 Scalability

Increasing the size of the network may be a feasible approach to improve the performance of some P2P systems. However, in the context of blockchains, additional factors such as block propagation time and consensus costs may have a direct impact on the system's scalability. In this section, we present scalability evaluation results obtained from several comparison tests between our proposed protocol (PoGR), the standard PoA protocol (Clique), and the Ethach (Ethereum's PoW). In our experiments, we collected several data points, each corresponding to the average of several runs with a specific network configuration. Each run consisted of 3000-5000 transactions. To fairly compare the protocols, we applied the same settings for both PoGR and PoA consensus, *i.e.*, the same block period (5s), node configuration, block size (gas limit), and workload configuration. We only varied the network size at a time while other parameters were set to the same values. For the PoW settings, we used a low fixed difficulty to adjust the block frequency and get an average block time between 5s. The other parameters are the same as for PoGR and PoA.

From Figure 3.8, we can see the gap between the consensus protocols PoGR, PoA (Clique), and PoW (fixed difficulty). PoGR and PoA outperform PoW in terms of both latency and throughput. Furthermore, when adding nodes to the network, the results show that the PoW's throughput increases at the beginning but starts to decrease once the peak is reached. Contrary to the PoGR's throughput, which remains stable as the block producer selection in our protocol does not depend on the number of participating nodes. The slight overhead is related to the network's communication and consensus costs. Furthermore, compared to the standard PoA protocol, no significant performance difference has been observed for small networks ($N < 12$). This is due to the experimental setup where nodes in both protocols seal blocks in the same fixed period. However, when the network size exceeds a certain threshold ($N > 12$), the throughput of Clique starts to decrease while we continue to get higher throughput with PoGR. This is justified by the fact that in Clique, the probability of a fork increases with the number of validators. Therefore, the system will take extra time to resolve this fork, which generates time overhead.

3.9 Analytical Comparison

We end our evaluation by presenting an analytical comparison of GuRuMarket with existing literature in five crucial dimensions.

- *System Structure Complexity:* This attribute represents the extent of additional components required for a given solution. If the solution requires nothing more than the blockchain itself, or only the devices that can be easily provided by its application scenario, then its complexity is **Low**. Otherwise, if the solution requires a complex computing module or additional systems, its complexity is marked as **High**.

- *Trust Computational Complexity:* Trust and reputation computation varies among these references. The complexity of the trust computation is marked as **High** if the proposed solution requires a large amount of computation and data to evaluate the participants' behavior. Otherwise, the complexity of trust computation is marked as **Low**. The label **Mid** represents the intermediate case between the two extremes.

- *Consensus Complexity:* All the propositions mentioned above aim to reduce the cost of traditional consensus by introducing trust management. We since have only two labels for this dimension: **Mid** and **Low**. A consensus protocol requiring only lightweight computation or comparison is marked as **Low**. Otherwise, if the consensus still needs several negotiations or voting phases to decide who can finally update the ledger, then it is marked as **Mid**.

- *Scenario Scalability:* Scenario scalability refers to whether we can use the solution in other use-case scenarios. A proposition focusing on consensus improvement will get a **Yes** for its scenario scalability. Otherwise, a proposed solution for a dedicated use case will receive a **No**.

- *Real-World Correlation:* Solutions that address the entire process, including offline interactions, get a **Yes**. Otherwise, if the solution only aims to secure the recording of transactions, it gets a **No**.

The comparison results are outlined in Table 3.4.

Compared to various trust-based solutions, it is clear that GuruMarket's design stands out for its reduced complexity in terms of node structure, trust assessment, and consensus. Furthermore, our systems showcase superior scalability in different scenarios and exhibit enhanced adaptability to real-world use cases.

3.10 Conclusion

In this chapter, we presented the GuRuMarket framework, a Blockchain-based reputation system for real-world marketplaces. The proposed solution introduces trust management into both application and consensus layers to provide a blockchain system suitable for such real-world applications.

Table 3.4: Comparison of blockchain-based trust and reputation solutions.

Reference	Use-Case	System Structure Complexity	Trust Computation Complexity	Consensus Complexity	Scenario Scalability	Real-World Correlation
[20]	Supply Chains	High	Low	Mid	No	Yes
[10]	IoT	High	Low	Mid	No	No
[17]	Blockchains	Low	Mid	Mid	Yes	No
[18]	Blockchains	Low	High	Mid	Yes	No
[19]	IIoT	Low	Mid	Mid	Yes	No
[21]	Edge Computing	High	High	Mid	Yes	No
[2]	Crowdsourcing	Low	Mid	Mid	Yes	No
[70]	Blockchains	Low	High	Mid	Yes	No
[61]	E-Cheque	Low	Mid	Mid	Yes	No
[62]	Energy Trading	Low	Low	Mid	No	No
[63]	E-Commerce	High	Mid	Mid	Yes	No
[78]	Blockchains	Low	High	Mid	Yes	No
[79]	IoT	Low	Mid	Mid	Yes	No
Ours	E-Commerce	Low	Low	Low	Yes	Yes

Our primary focus in this first chapter was on constructing a robust and effective reputation model tailored for real-world marketplaces. To achieve this, we propose a BRS framework designed to foster accountability and trust by enabling effective on-chain reputation management. However, it is crucial to acknowledge that on-chain reputation management introduces additional computational overhead due to interaction evaluation and reputation update. To reduce its impact on the performance of GuRuMarket, we leverage the proposed model to design a lightweight reputation-based consensus called *Proof-of-Guarentee&Reputation (PoGR)*. PoGR improves Layer-1 scalability and prevents updating monopolization (*i.e.*, wealth centralization) by employing a block producer selection based on guarantee and reputation. We performed the necessary security analysis of the proposed framework to show its capability to withstand multiple attacks. We implemented and deployed a prototype of GuRuMarket using Hyperledger Besu and other Web3 tools. Accordingly, several test scenarios were performed to evaluate the effectiveness of the proposed trust model, the fairness of the PoGR consensus, and the performance of the entire system. The evaluation results demonstrate GuRuMarket's effectiveness, fairness, and scalability.

The main points of this first chapter are summarized as follows:

- A Blockchain-powered marketplace that incorporates guarantee and reputation mechanisms at both the application and consensus layers, promoting credibility and accountability.
- An efficient reputation model that evaluates the trustworthiness of each participant in the system. It examines each interaction that occurs within the system and calculates a weighted score for the users involved in this interaction by combining two different trust evaluation methods (objective and subjective).

- An incentive mechanism based on guarantee and reputation to control participants' behavior. In GuRuMarket, positive behavior is rewarded by an improved reputation. Conversely, misbehavior results in a significant loss of reputation and forfeiture of deposits.
- A lightweight consensus called *Proof-of-Guarantee-and-Reputation (PoGR)* that requires less computation than existing protocols and can ensure fairness among nodes through a scoring formula that uses guarantee and reputation ratings.
- A qualitative security analysis of the proposed framework demonstrating its ability to counter common blockchain and reputation attacks.
- A proof of concept for the proposed framework, using emerging technologies, enabling a more meaningful assessment of GuRuMarket's capabilities.

Users in reputation systems may be reluctant to interact with each other and provide feedback for fear of possible retaliation [1]. The use of an automated, feedback-free reputation model in GuRuMarket partially solves this problem. Through automated scoring, we build an effective yet privacy-friendly reputation model. However this is not enough, what if our application needs feedback? How can we enable the users to provide this feedback secretly? Additionally, what about the issue of linking reputation scores to a single public key, which remains a concern regarding potential tracking in our BRS?

In the next chapter, we will explore existing solutions dedicated to addressing the above privacy concerns in decentralized reputation systems. More precisely, we first delve into cryptographic techniques integrated with blockchain technology for constructing privacy-preserving reputation systems. Next, we introduce a decentralized anonymous reputation system, leveraging the capabilities of blockchain and other cryptographic building blocks.

Chapter 4

Privacy and Anonymity in Blockchain-based Reputation Systems

Contents

In chapter 3, we presented GuRuMarket, a blockchain-based real-world marketplace that addresses the challenge of effective and L1 scalable on-chain reputation management. This chapter complements our solution by addressing privacy preservation issues. Indeed, without privacy safeguards, some participants might be reluctant to contribute to our BRS for fear of retaliation and tracking.

This chapter proposes a new privacy-preserving blockchain-based reputation system (PPBRS) called "DARS". DARS is a decentralized anonymous reputation system that

promotes trust in blockchain-based real-world applications. The proposed system allows users to use different pseudonyms during interactions, thus preserving their anonymity. To prevent reputation attacks in the system, all pseudonyms of a given user are cryptographically linked to the same access token, allowing honest users to maintain their reputation and preventing malicious ones from starting over.

4.1 Introduction

As discussed in the previous chapters, blockchain-based reputation systems (BRS) are a recent and important development in decentralized trust and reputation management. The decentralization, transparency, and immutability brought by blockchain are clearly what we always hoped for to build effective trustless reputation systems. Despite these promising attributes, existing BRS face a critical challenge in countering common reputation attacks, including whitewashing, self-promotion, and bad-collision attacks. In our first contribution, we presented a BRS that relies on reputation scores tied to the same address or public key to mitigate these attacks. However, this approach poses a notable limitation to the widespread adoption of BRS in general, as it fosters concerns about potential retaliation, leading to a reluctance among users to interact and provide feedback.

The primary goal of reputation systems is to hold users accountable for their behavior [1]. To achieve this, traditional centralized systems consider the server a semi-honest entity. That is, it honestly executes the specified protocol but maintains a degree of curiosity about user privacy [80]. In a real-world deployment, however, the central server may become malicious due to malware infections or internal attacks. Once the server is compromised, ensuring user privacy and aggregation correctness becomes uncertain. While cryptographic techniques prove useful in addressing certain security concerns such as identity management and access control associated with centralized and hybrid frameworks [8, 57], they cannot solve the inherent single point of failure that remains an unavoidable problem for such systems. In addition, reputation systems require the collection and mining of certain sensitive data (*e.g.*, feedback, delivery verification, location, etc.) to evaluate interactions and display reputation scores, raising privacy concerns about whether this information is handled properly. Furthermore, as mentioned above, users in these systems may be reluctant to interact with each other and provide feedback for fear of potential retaliation [1]. The use of automated, feedback-free reputation models is a potential solution to these problems [9]. Through automated evaluation, we can build effective yet privacy-preserving reputation models. However, the issue of linking reputation scores or tokens to a single master key remains a concern regarding potential tracking in current BRS.

A recent approach to address privacy issues involves the development of decentralized privacy-preserving reputation models, allowing users to interact and share feedback confidentially. Privacy-preserving reputation systems provide participants with the tools to share feedback anonymously or pseudonymously while still maintaining the integrity and usefulness of the reputation information. Using cryptographic techniques with decen-

tralized systems such as blockchain [11, 22] can help reputation systems guarantee both privacy preservation and efficiency. However, existing solutions present some security problems, as they fall short of being entirely decentralized since they depend either on a centralized entity or a group of trusted peers to handle identities, credentials, and security parameters. Additionally, despite the use of blockchain technologies in numerous research efforts [23–25] to develop decentralized and privacy-centric reputation systems, the proposed solutions do not address its cost-effectiveness and scalability. Furthermore, the challenges associated with the implementation of real-world blockchain-based reputation frameworks have not undergone thorough examination. In particular, existing solutions fail to adequately tackle the challenge of effectively and privately incorporating real-world data into a blockchain for reputation management. This is because blockchains deal primarily with information that is native to the network.

To overcome all the above issues, we present in this chapter, a PPBRS entitled "DARS: Decentralized Anonymous Reputation System" [32]. To achieve anonymity, we build our system on top of two separate ledgers decoupling identity management from business activities. To prevent Sybil attacks without compromising user privacy, we propose using Decentralized Oracle Networks not only to automate smart contract execution but also to import credentials from external systems. DARS users can generate/use an unlimited number of pseudonyms on the blockchain to protect their digital identity and ensure continued anonymity. The main challenge in this setting is to guarantee robust reputation management while maintaining anonymity through pseudonyms. To achieve this, we use zkSNARKs for set membership on both identity and business ledgers. In particular, the proposed framework relies on two collision-resistant hash-based Merkle trees (UCTree and RCTree) and zkSNARK proofs to maintain robust and anonymous reputation management. This design is suitable for all trust-based applications, such as decentralized marketplaces and crowdsourcing platforms.

The remainder of this paper is organized as follows: Section 4.2 describes related work. Section 4.3 presents the proposed framework. Section 4.4 details the construction of the proposed decentralized anonymous reputation system. The security analysis is presented in 4.5. The performance evaluation is discussed in Section 4.6. Finally, We conclude the chapter in Section 4.8.

4.2 Related Work

Privacy-preserving reputation systems, initially proposed in the mid-2000s, have seen continuous evolution to adapt to emerging application areas. Recent developments include applications in Social IoT [81], Industrial IoT-enabled retail marketing [22], and Crowdsourcing [57]. The integration of blockchain technology has further advanced research in this field, resulting in privacy-preserving reputation systems with novel properties such as trustlessness, transparency, and immutability.

Privacy-driven reputation systems aim to make it extremely difficult for an attacker to establish a link between operations like submitting feedback and the user's true identity [1]. However, maintaining an effective and robust reputation management without

linking it to a unique identity is not straightforward. On one hand, achieving accountability requires that the reputation score mirrors the user's behavior, necessitating some form of linkage to a unique ID. Conversely, privacy preservation, especially anonymity, seeks to detach a user's identity from their activities.

In an attempt to address anonymity issues in reputation systems, Liu et al. [22] propose an anonymous reputation system for retail marketing in the industrial IoT environment. The system uses smart contracts on a PoS-enabled blockchain system to provide transparency and public verifiability under the malicious adversarial model. To ensure anonymity, the solution employs a randomizable signature scheme as well as non-interactive zero-knowledge proofs. However, the proposed framework employs a centralized entity called IDentity Management (IDM), which is responsible for managing identities, credentials, and security parameters. The IDM has a powerful role in the framework, which represents a central point of failure and a security threat.

Zhao et al. [11] propose a privacy-preserving reputation system that leverages blockchain technology in the resource-constrained environment of mobile crowdsensing. Global reputation scores are updated by a smart contract based on the average of all feedback. The proposed approach employs additive secret sharing and a delegation set to provide privacy in a dynamic environment. However, similar to [22] the authors consider a semi-honest model in which two entities, called Task Distribution Center (TDC) and Key Distribution Center (KDC), are considered semi-honest and fully trusted, respectively.

BPRF [82] is a blockchain-based privacy-preserving reputation framework for participatory sensing systems. It uses smart contracts to manage the reputation scores of participants based on their sensing data and corresponding feedback. The smart contract and blockchain enable transparency and public auditability of reputation scores. BPRF employs a group signature scheme and a partially blind signature algorithm to protect user information. However, BRBF operates under the assumption of an honest-but-curious model, where the server (*i.e.* centralized entity) is expected to consistently adhere to the protocol but may attempt to gain additional information from the protocol transcript beyond the intended scope of shared information.

To achieve greater transparency, Schaub et al. [24] proposed a fully decentralized reputation system on top of a public blockchain with a blind signature to guarantee consumer anonymity in an e-commerce platform. However, their study does not provide any mathematical formulation for the reputation model. In addition, the performance of the proposed protocol was not tested either.

Soska et al. [25] proposed an anonymous reputation system based on a ring signature, which results in a linear overhead in generating the anonymous review proof. However, while the system provides robust privacy and anonymity for buyers, it provides only partial protection for sellers. In addition, the designed protocol allows a customer to link her transaction or reveal her true identity, thereby reducing the anonymity set of all other customers who bought the item.

While the aforementioned studies on PPBRS and others [23,83,84] have put considerable effort into investigating the use of blockchain in building robust privacy-preserving reputation systems, the proposed solutions have not fully explored its potential. In

addition, the cost-effectiveness and scalability issues associated with privacy-preserving on-chain reputation management have not been well investigated. Furthermore, the implementation challenges of real-world blockchain-based reputation systems have not been thoroughly explored. In particular, almost all these frameworks fail to adequately tackle the challenge of securely and privately incorporating real-world data into a blockchain for robust reputation management.

To address these concerns, our second contribution introduces a privacy-preserving BRS named "DARS". DARS is a fully decentralized anonymous reputation framework applicable to various blockchain-based real-world scenarios. To achieve anonymity, DARS allows users to use different pseudonyms during interactions. Consequently, DARS affords an equivalent level of privacy to all parties involved (*e.g.*, buyers and sellers). However, maintaining a consistent and robust reputation across these pseudonyms is a significant challenge. The flexibility of using multiple pseudonyms comes at a cost, as it exposes the system to various reputation attacks such as whitewashing and self-promotion. To prevent such attacks in DARS, all of a user's pseudonyms are cryptographically linked to the same access token, allowing honest users to maintain their reputations and preventing malicious ones from starting over. In addition, to address the centralization problem in current solutions and achieve end-to-end decentralization, we propose an architecture with two different ledgers and a decentralized Oracle network. The first ledger is responsible for controlling access to the second ledger while the Oracle network employs an Oracle protocol [85] to securely and privately bring off-chain data needed for reputation management on-chain. Together with the use of pseudonyms, the Oracle network enables smart contract automation for transparent reputation updating.

4.3 DARS Framework

To build our Decentralized Anonymous Reputation System we use two separate ledgers: an Authentication Management Ledger (AML) and a Business Management Ledger (BML). This is done to decouple personal information (physical identities) from business operations. In addition, DARS incorporates a Decentralized Oracle Network (DON) to automate the execution of smart contracts and import credentials from external systems in a privacy-preserving way.

4.3.1 Overview

To guarantee their anonymity, DARS users must follow the protocol described in Figure 4.1. The DARS approach involves several steps detailed in Section 4.4, and summarized as follows:

1. Alice first registers with AML in a privacy-preserving manner.
2. Alice receives master and context-based credentials from the AML committee.
3. Then, Alice posts an anonymous access token using these credentials.
4. DON triggers ISC, on the BML to update UCTree (minting Alice's token)

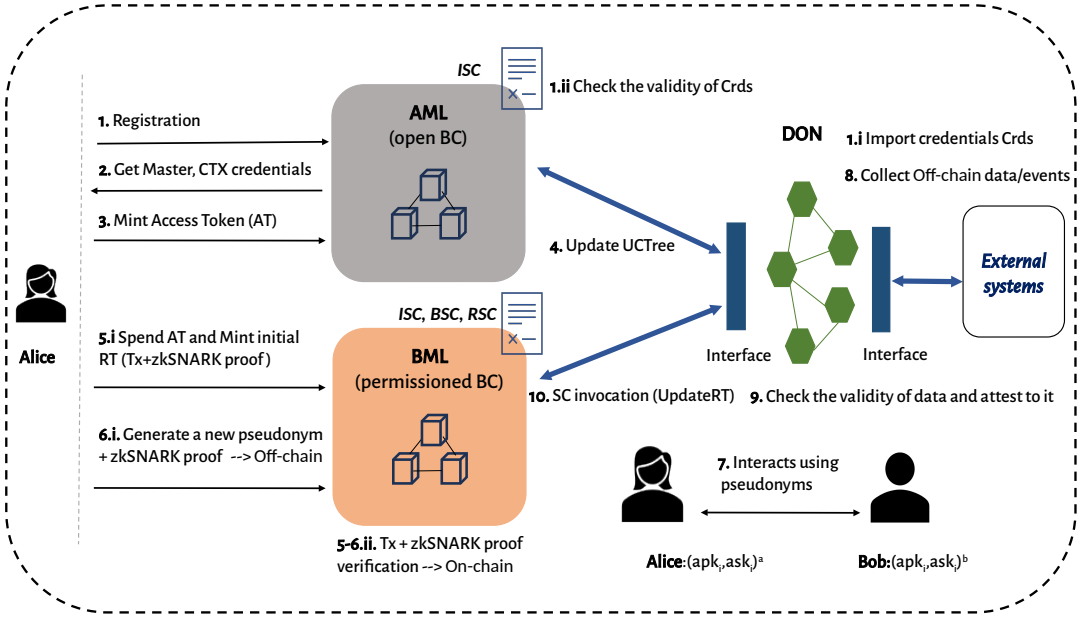


Figure 4.1: DARS Framework

5. Next, Alice spends her Access Token (AT) and mints her initial Reputation Token (RT) on the BML by generating a spendAT TX with a zkSNARK proof. The spendAT TX is sent to the BML network for verification and accepted only if the zkSNARK proof is valid.
6. Subsequently, Alice can interact with an existing user Bob using her RT with a new pseudonym on the BML. Alice must spend her old RT and mint a new one each time she wants to use a new pseudonym. The spendRT TX is sent to the BML network for verification and accepted only if the zkSNARK proof is valid.
7. Using the generated pseudonyms and their respective reputation tokens, Alice and Bob can initiate an interaction.
8. The DON collects the data provided by Alice and Bob and other actors (*e.g.*, a controller) to prove that the off-chain part of their interaction took place.
9. The DON checks the validity of the collected Data by corresponding measures, then attests to it.
10. Finally, the overall reputations of Alice and Bob are automatically updated by triggering the reputation smart contract (RSC), *i.e.*, The DON triggers the RSC using Alice and Bob's pseudonyms.

4.3.2 Threats Model

Having presented an overview of our system model, we will now introduce the adversarial model and explore the security features of the proposed framework intended to build an end-to-end decentralized anonymous reputation system.

In DARS, users can generate as many pseudonyms as they like, but all of them must be bound to the same long-term master key to prevent Sybil attacks. This allows a user to have a unique reputation but use different pseudonyms when interacting with other users. Therefore, to prevent Sybil attacks that may undermine the reputation system, we assume the existence of a decentralized entity such as CanDID [73] to verify identities and guarantee the uniqueness of users.

4.3.2.1 Adversarial Model

For our adversarial model, we borrow the assumptions of [73]. Therefore, an adversary is able to statically and actively corrupt up to t of the n nodes in the committee, for $t < n/3$. In addition, the adversary can corrupt any number of external entities, such as users and applications.

4.3.2.2 Security Properties

Under the above security assumptions, we summarize the security properties and goals of DARS.

- *Sybil resistance:* A user cannot have any credentials other than his own.
- *Unforgeability:* An adversary cannot forge the credentials of honest users or impersonate them.
- *User privacy*: It is infeasible for an adversary to ascertain a user's attributes through the examination of issued identification information, the analysis of transaction data during interactions with other users, or the observation of the ongoing evaluation of interactions.
- *Reputation binding:* The user's reputation is unique and stored publicly in the blockchain. Although users can generate as many pseudonyms as they wish, all are cryptographically linked to the same access token.
- *Forward Reputation binding:* No user should be able to mint/use a reputation token with a reputation score higher than that linked to his/her most recent token.

4.3.3 DARS Architecture

This section describes our proposed Decentralized Anonymous Reputation System (DARS) architecture. In the following, we detail its main components.

4.3.3.1 Access Management Ledger (AML)

AML is an *open* blockchain that manages data about Decentralized IDentifiers (DIDs). DIDs are publicly identifiable endpoints, such as documents, wallets, smart contracts, etc. We adapt the approach of CanDID [73] to uniquely identify legitimate users. The system relies on a PKI-like infrastructure [86] to support the use of DIDs. Each user manages a master public/private key pair (M_{pk}, M_{sk}). The PKI infrastructure then stores the correspondence between the DIDs and the public key. For issuing credentials, we use a permissioned model to select a set of nodes that act as a committee. Let CS be the committee set, which consists of n nodes, $\{CS_i\}_{i=1}^{n}$. The nodes in the committee jointly store a secret key sk_{cs}, which is used to issue credentials. The corresponding public key pk_{cs} is used to verify credentials. Any party (*e.g.*, applications, validators) can act as a credential verifier.

AML aims to convert commonly used legacy data to Sybil-resistant privacy-preserving decentralized credentials. This goal is achieved in two steps. First, AML converts a set of claims referred to as "pre-credentials" to a master credential $M_{cred} = (M_{pk}, M_{sk})$ with a privacy-preserving deduplication protocol [73]. Master credentials are Sybil-resistant in that each user can only get one credential, but they are not intended to be used in interactions. Rather, AML allows users to create context-based credentials by linking application-specific attributes (attested to by pre-credentials) to the master credential. Thus, the second step involves the generation of such contextual credentials. Within AML, each application defines a distinct context denoted as ctx (*e.g.*, ctx = online trading). For Alice to obtain a credential on context ctx, she must submit her master credential to the committee, along with a set of required claims specified by ctx (*e.g.*, age over 18). The committee proceeds to validate the claims and issue a credential for ctx through a process akin to that used for issuing the master credential.

4.3.3.2 Business Management Ledger (BML)

BML is a *permissioned* blockchain that implements the overall business logic. Access to the BML is provided using contextual (application-based) credentials provided by the AML after successful registration. Therefore, within our design, multiple BMLs with various business logic can be incorporated with the AML. For instance, we can have a BML for Crowdsourcing and another for E-commerce. Users on the BML can generate as many pseudonyms as they want (ideally a new pseudonym for each new interaction) to protect their digital identity. To enable this while maintaining effective reputation management on pseudonyms, and above all to prevent reputation attacks, users need to provide verifiable proof in their business transactions.

In addition to the business logic, BML also implements the reputation model of DARS, which is responsible for evaluating the interaction between users, and updating their reputation scores. It is worth mentioning here that the reputation model itself has to protect the user's privacy. This means that to evaluate the interaction, the only parameters that can be used are those that do not reveal any information about the user's identity. For example, we cannot use reputation models that rely on the history of interactions be-

tween the users concerned. Therefore, the only parameter that reflects past interactions in DARS is the current reputation score. Nevertheless, other parameters can be used to objectively and accurately evaluate the interaction, such as cost and execution time in crowdsourcing scenarios, or transaction amount and proof of delivery in e-commerce applications. Overall, the above and other context-based metrics can be combined with user feedback to build an effective and privacy-preserving reputation model.

4.3.3.3 Decentralized Oracle Network (DON)

Not long ago all blockchains had a common problem which was the lack of external connectivity. This problem is known as *"The Oracle Problem"*, which refers to the issue of securely integrating real-world data into a blockchain, as blockchains primarily deal with information that is native to the network. Specifically, smart contracts that run "On-chain" cannot process external "Off-chain" data and events to provide the user with functionality that needs to be realized outside the blockchain. Consequently, solving the Oracle problem is crucial to guarantee the accuracy and reliability of the off-chain data we want to integrate into the blockchain.

Over the past years, various decentralized oracle protocols have emerged to address this limitation. Among them, two protocols, DECO [85] and Town Crier [87], have been developed specifically to enable Oracle nodes to securely fetch data from off-chain systems while safeguarding user privacy and data confidentiality. DECO uses cryptographic primitives to achieve its integrity and confidentiality properties. Town Crier, on the other hand, relies on the use of a Trusted Execution Environment (TEE), which functions as a black box to execute applications in a tamper-proof and confidential manner. We prefer to use DECO over Town Crier in DARS, as it offers trustless capabilities. Hence, a single node in our DON can export data from a private session with a web server or the AML to all nodes in the DON. Therefore, the entire DON can attest to the authenticity of the data and trigger the smart contracts on the BML.

4.4 Anonymous Reputation

In this section, we describe the mathematical and cryptographical aspects of DARS. DARS uses a set of cryptographic building blocks on top of the Access Management Ledger (AML) and Business Management Ledger (BML), to guarantee anonymous and effective reputation management. In the following, we first, introduce the cryptographic primitives used to build DARS. Then, we discuss, the entire construction in detail.

4.4.1 Cryptographic Building Blocks

The main cryptographic building blocks upon which the DARS system is built are the following.

4.4.1.1 Commitment Scheme

A commitment scheme is a cryptographic protocol that allows a party, referred to as the committer, to commit to a chosen value without revealing it, while still being able to prove its validity later on [88]. It is designed to fulfill two crucial security properties:

1. *Hiding Property:* Given $\mathsf{COMM}(\mathsf{x})$, it should be computationally infeasible to determine the original value x.
2. *Binding Property:* It should be computationally infeasible to find two distinct values x_1 and x_2 such that $\mathsf{COMM}(\mathsf{x}_1) = \mathsf{COMM}(\mathsf{x}_2)$.

4.4.1.2 zkSNARKs

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zkSNARKs) are an advanced variant of Zero-Knowledge Proofs (ZKPs). More precisely, a zkSNARK scheme is a Non-Interactive Zero-Knowledge (NIZK) scheme [89], wherein the proof itself is a self-contained data block that can be verified without requiring any interaction from the prover [90, 91].

A zkSNARK construction consists of three algorithms (Gen, Prov, Verif) defined as follows:

- The key generator algorithm takes Gen a secret parameter λ and a program C, and generates two publicly available keys: a proving key pk, and a verification key vk ($(pv, vk) = \mathsf{Gen}(\lambda, C)$). These keys are public parameters that need to be generated only once for a given program C.
- The proving algorithm Prov takes as input the proving key pk, a public input t, and a private witness w. The algorithm generates a proof $\pi = \mathsf{Prov}(pk, t, w)$ that the prover knows a witness w and that the witness satisfies the program C.
- The verification algorithm computes $\mathsf{Verif}(vk, t, \pi)$ which returns true if the proof is correct, and false otherwise. Thus, this function returns true if the prover knows a witness w satisfying C.

4.4.1.3 zkSNARKS for Set Membership via Merkle Trees

The set membership problem via Merkle trees involves proving that an element belongs to a set using the Merkle tree data structure. More formally, given a set S containing n elements and a Merkle tree constructed from the hash values of these elements, the set membership problem is to prove that a specific element x belongs to the set S without revealing any other elements in S [92]. Merkle trees alone do not provide ZK property. To achieve this property, we need to combine Merkle trees with additional techniques such as zkSNARK or other cryptographic primitives [93].

In DARS, we address the zkSet membership problem using a Merkle tree structure over commitments to ensure privacy-preserving access control on the BML. This involves constructing and verifying membership zkSNARKs proofs through Merkle trees,

providing a solution to the set problem with logarithmic complexity to the set's size. This approach maintains both privacy and integrity. An illustration of the Collision-Resistant Hash(CRH)-based Merkle tree over a list of access commitments is shown in Figure 4.2

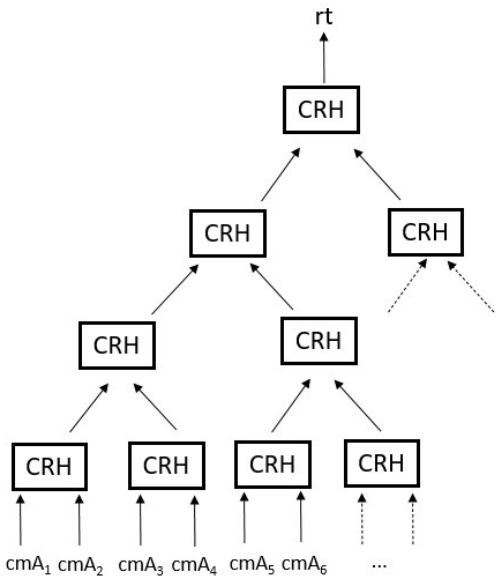


Figure 4.2: CRH-based Merkle tree over a list of access commitments UCTree

The zkSNARK proofs used in DARS are constructed as follows :

- *Setting up the circuit.* The first step involves, (i) *Circuit Representation* representing the logical operations and constraints of the computation or statement being proven as a mathematical circuit. (ii) *Arithmetic Circuit to QAP (Quadratic Arithmetic Programs)*, the arithmetic circuit is then converted into a system of quadratic equations. This transformation simplifies the representation and allows for more efficient processing. (iii) *QAP to R1CS (Rank-1 Constraint System)*, the QAP is further converted into a Rank-1 Constraint System, a set of linear equations that can be efficiently evaluated [94].
- *Public Parameters Generation.* This step involves generating the proving and verification keys. It uses the circuit to generate the public parameters/keys needed for proof generation and verification.
- *Proving.* This step uses the proving key, the circuit, and the input values to generate a proof that the computation is correct. This step is performed off-chain.
- *Verifying.* This step uses the public key, circuit, and proof to verify that the calculation is correct. This step is performed on-chain using smart contracts [90, 91].

4.4.2 On-chain Anonymous Reputation Management

Now that we have described the primitives we intend to use to develop our decentralized anonymous reputation system, we will describe its construction in detail. We build the DARS framework in four main phases.

4.4.2.1 Registration

The registration process takes place between a user u and the AML committee. The AML relies on a Decentralized Oracle Network (DON) to ensure uniqueness while issuing master credentials M_{cred} for any valid user u. As in [73], we make use of DON to import identities from existing systems. For example, Alice can use her credentials on her Social Security Administration (SSA) account to generate a credential certifying her Social Security Number (SSN). The DON uses the DECO protocol [85] to provide privacy for user data. DECO is a three-party protocol involving a prover denoted as P, a verifier denoted as V, and a Transport Layer Security (TLS) server denoted as S. The protocol enables P to persuade V that a data item, which may be private to P, obtained from S, meets a specified predicate. DECO relies on multiparty computation (MPC) to protect the confidentiality and authenticity of the data, and on zero-knowledge proofs (ZKPs) to prove that a predicate is satisfied.

Once the identity of the user u is verified by the committee nodes following the DECO protocol, the corresponding user will be able to post his/her set of claims on AML and get access on BML using context-based credentials. To obtain a new credential for the ctx context (*e.g.* "a trading or crowdsourcing activity"), u must submit to the committee $(pk_{ctx}, M_{cred}, C^u_{ctx})$, a new identifier to be used in the ctx context, master credentials and a set of pre-credentials with the claims C^u_{ctx} required by ctx. The committee upholds a set of $Granted_{ctx}$ identifiers denoting those that have already obtained a credential within this specific context. If M_{pk^u} of M_{cred} is not part of this set, a credential is issued. Finally, (M_{pk^u}, pk^u_{ctx}) is added to $Granted_{ctx}$. For more details on issuing context-based credentials, see [73].

In our construction, master credentials are purposely excluded from any interactions with the BML to prevent linking them to the user's real identity. On the other hand, contextual credentials are used exclusively on the AML and remain cryptographically hidden on the BML to separate identity management from business operations. To guarantee these properties, the committee nodes maintain a CRH-based Merkle tree with root rt called $UCTree$, which contains all the user access commitments. When a user u is successfully registered with the committee nodes he/she must provide a commitment to his/her credentials. The user will use this commitment to access the BML without revealing any information that could be linked to his/her identity. To do that the user proceeds as follows: u generates a key pair (pk, sk); Next samples a random a and computes $cm_u = \mathsf{COMM}_a(sk^u_{ctx})$, then computes $cm_A := \mathsf{COMM}_b(cm_u||pk)$ for a secret b, and defines $AT := (sk^u_{ctx}, a, b, cm_A)$. A corresponding mint access token transaction, $TX_{AM} := (pk, cm_u, cm_A)$, is added to the AML (accepted only if pk^u_{ctx} is known to the committee). The $UCTree$ is then updated with a new leaf (cm_A). We use

DON to synchronize any changes made to the $UCTree$ on the AML with the BML. This construction allows the user to prove to BML validators that he/she has valid credentials on AML efficiently and anonymously, *i.e.*, the time and space complexity is logarithmic to the size of $UCTree$, and pk^u_{ctx} remains cryptographically hidden in cm_u.

4.4.2.2 Mint Initial Reputation Token

The second phase in our construction is the minting of a reputation token that is cryptographically linked to the user's contextual credentials. This phase aims to hide the user's digital identity pk^u_{ctx} while ensuring robust reputation management through the utilization of zkSNARK proofs and a commitment scheme.

To interact with other users and post transactions on the BML, the user must spend their access tokens and mint their initial reputation token RT_{init}. This is equivalent to adding a new leaf to the $RCTree$ (similar to $UCTree$) containing a commitment to its initial reputation score. To provide targets for new tokens, we use addresses: each user u generates an address key pair (apk, ask), the address public key and address private key, respectively. The tokens of u contain the value apk and can only be spent with the knowledge of ask. A key pair (apk, ask) is sampled by choosing a random seed ask and setting $apk := \mathsf{PRF}_{ask}(0)$ using a Pseudo-Random Function (PRF).

To achieve the property of forward reputation binding the user must sample a random serial number S_n for each new reputation token. S_n is then released when using the token. This is realized as follows, u first generates a new key pair (apk, ask), then samples a random s and computes $S_n := \mathsf{PRF}_{ask}(s)$, and commits to the tuple (apk, R_{init}, s) in two steps: $cm_P := \mathsf{COMM}_r(apk||s)$ for a random r, and then $cm_R := \mathsf{COMM}_{r'}(R_{init}||cm_P)$ for another random r'. The outcomes comprise: (i) a reputation token $RT_{init} := (apk, R_{init}, s, r, r', cm_R)$ and (ii) an initial RT mint transaction $TX_{RM} := (R_{init}, cm_P, r', cm_R)$. However, this alone does not fulfill the criteria for the transaction to gain acceptance on the BML. The user must provide a zkSNARK proof π_{AS} of the NP statement ***"I know a secret*** b ***such that*** $\mathsf{COMM}_b(cm_u||pk)$ ***appears as a leaf in a CRH-based Merkle tree*** $UCTree$ ***whose root is*** rt***"***.

The previous prerequisite permits access to the BML exclusively for authorized users. With this in mind, we edit the TX_{RM} transaction to $TX_{RM} := (R_{init}, cm_P, r', cm_R, \pi_{AS})$ which is submitted to the BML. The TX_{RM} is accepted if and only if the π_{AS} and cm_R are valid. Because of commitment nesting, anyone can verify that cm_R in TX_{RM} is a commitment of a token of value R_{init} (by checking that $\mathsf{COMM}_{r'}(R_{init}||cm_P)$ equals cm_R), but is unable to identify the owner through the knowledge of apk or the serial number S_n (derived from s). Finally, user anonymity is achieved because the proof π_{AS} is zero-knowledge: while cm_u and pk are revealed, no information about b is revealed, and finding which of the many commitments in $UCTree$ corresponds to TX_{RM} is equivalent to inverting $f(b) := \mathsf{COMM}_b(X)$, which is assumed to be infeasible [93].

4.4.2.3 Reputation Token Spend/Use

So far, user u has minted his initial reputation token RT_{init}. Therefore, u can interact with any other user v on the BML by submitting transactions. Within DARS, users' reputation scores are tied to their most recent reputation commitment cm_R. As a result, for a user u to engage with other users, they must reveal the nested commitment to display their reputation score. Since only the user possessing the secret r' can unveil it, there's no susceptibility to forgery. In addition, TX_{RM} is submitted using a pseudonym different from pk^u_{ctx}, and since u can generate numerous pseudonyms (ideally a new pseudonym apk^{new} for each new interaction), the likelihood of revealing one's true identity is effectively eliminated.

Spend A Reputation Token. Users within the BML can spend their reputation token by submitting a reputation spending transaction, denoted as TX_{RS}. TX_{RS} allows them to create a new token of identical value to the current one. Consider a scenario where a user u possesses a pair of address keys (apk^{old}, ask^{old}), wishes to consume its current/old token $RT^{old} := (apk^{old}, R^{old}, s^{old}, r^{old}, r'^{old}, cm_R^{old})$ and produce a new one RT^{new}, targeted at the public address key apk^{new}. The user u proceeds as follows, (i) u samples serial number randomness s^{new}; (ii) u computes $cm_P^{new} := \mathsf{COMM}_r^{new}(apk^{new}||s^{new})$ for a random r^{new}; and then computes (iii) $cm_R^{new} := \mathsf{COMM}_{r'^{new}}(R^{new}||cm_P^{new})$ for a random r'^{new}. This yields the token RT^{new}.
$RT^{new} := (apk^{new}, R^{new}, s^{new}, r^{new}, r'^{new}, cm_R^{new})$. Next, u generates a zkSNARK proof π_{RS} for the following NP statement:

"Given the** $RCTree$ **root** rt_R**, serial number** S_n^{old}**, and token commitment** cm_R^{new}**, I know a token** RT^{old}**,** RT^{new}**, and address secret key** ask^{old} **such that:

(i) The tokens are well-formed: $cm_P^{old} := \mathsf{COMM}_{r^{old}}(apk^{old}||s^{old})$
and $cm_R^{old} := \mathsf{COMM}_{r'^{old}}(R^{old}||cm_P^{old})$ for cm_R^{old} and similarly for cm_R^{new}.

(ii) The secret key matches the public key: $apk^{old} = \mathsf{PRF}_{ask^{old}}(0)$.

(iii) The serial number is calculated correctly: $S_n^{old} = \mathsf{PRF}_{ask^{old}}(s^{old})$.

(iv) The commitment cm_R^{old} appears as a leaf in $RCTree$ whose root is rt_R.

(v) The reputation values are equal $R^{new} = R^{old}$."

A resulting spend transaction $TX_{RS} := (rt_R, S_n^{old}, cm_R^{new}, \pi_{RS})$ is sent to the BML. TX_{RS} gets rejected if S_n^{old} appears in a prior transaction. Thus, the user is forced to use his/her most recent reputation token for each new interaction.

Use A Reputation Token. To use the old token RT^{old} without minting a new one (using the same pseudonym without forward anonymity), u must submit a reputation use transaction TX_{RU}. The user u first generates a new key pair (apk^{new}, ask^{new}). Then submit TX_{RU} which contains the following zkSNARK proof:

"Given the** $RCTree$ **root** rt_R **and serial number** S_n^{old}**, I know a token** RT^{old}**, and address secret key** ask^{old} **such that:

(i) The old token RT^{old} is well-formed: $cm_P^{old} := \mathsf{COMM}_{r^{old}}(apk^{old}||s^{old})$ and $cm_R^{old} := \mathsf{COMM}_{r'^{old}}(R^{old}||cm_P^{old})$ for cm_R^{old}.

(ii) The secret key matches the public key: $apk^{old} = \mathsf{PRF}_{ask^{old}}(0)$.

(iii) The serial number is calculated correctly: $S_n^{old} = \mathsf{PRF}_{ask^{old}}(s^{old})$.

(iv) The commitment cm_R^{old} appears as a leaf in $RCTree$ whose root is rt_R."

The resulting use transaction $TX_{RU} := (rt_R, S_n^{old}, R^{old}, \pi_{RU})$ will reveal the current reputation R^{old} and the corresponding serial number S_n^{old} and associate them with apk^{new}. If the proof π_{RU} is valid, the (apk^{new}, R^{old}) is whitelisted. This means that u will be able to interact using his new pseudonym coupled with his current reputation.

Now, if, after some interactions, u wants to use a different pseudonym apk^{rec}, u must mint a new token using apk^{new} and the most recent reputation R^{rec}. Thus, u must proceed as follows, (i) u samples serial number randomness s^{rec}; (ii) u computes $cm_P^{rec} := \mathsf{COMM}_r^{rec}(apk^{rec}||s^{rec})$ for a random r^{rec}; and then (iii) $cm_R^{rec} := \mathsf{COMM}_{r'^{rec}}(R^{rec}||cm_P^{rec})$ for a random r'^{rec}. This results in a token $RT^{rec} := (apk^{rec}, R^{rec}, s^{rec}, r^{rec}, r'^{rec}, cm_R^{rec})$ and a mint transaction $TX_{RM} := (R_{rec}, cm_P^{rec}, r'^{rec}, cm_R^{rec}, \pi_{RM})$. TX_{RM} is accepted if and only if cm_R^{rec} is computed correctly. When TX_{RM} passes, the address public key apk^{new} (not apk^{rec}) is removed from the list of authorized addresses. The user cannot use it again and will have to spend the freshly minted token RT^{rec} for further interactions.

4.4.2.4 Reputation Update

This process is automated using the DON and smart contracts. We use an Oracle network to collect the off-chain data needed to evaluate the interaction, and then trigger the reputation module implemented using smart contracts to update the reputation scores of the users involved in the interaction using their pseudonyms.

The update process takes place once the interaction is over. For a more comprehensive explanation, let us consider a scenario within a marketplace. Let us suppose that user u wants to introduce a new product into the system. In this case, u is faced with two choices: use the existing reputation token or mint a new one, as detailed earlier. Then, u can simply add the new product to the system by posting the corresponding transaction $TX_{newProd} := (prodID, price, CID, ...)$, where CID is the content identifier of the product description on IPFS. Once the product is listed for sale, when another user v intends to purchase this item from u, v has also the option to either utilize his/her existing reputation token or spend it to generate a new one, retaining the same reputation score and ensuring ongoing anonymity. Additionally, v is required to provide a zkSNARK proof demonstrating the absence of a shared secret key with u. This condition is necessary for our construction, as it prevents self-promotion attacks. To do this, v computes $H_i^v := H(prodID||sk_{ctx}^v)$. Then produces a zkSNARK proof π_i^v for the following NP statement:

"Given the product identifier $prodID$***, I know*** sk_{ctx}^v, cm_u***, and*** AT ***such that,*** H_i^v ***is computed correctly and shares the same*** sk_{ctx}^v ***with*** AT***"***.

The proof is then sent to the BML as part of the new order transaction $TX_{newOrd} := (ordID, info, H_i^v, \pi_i)$. Like v, if u chooses to approve v's order, u is required to compute the interaction hash $H_i^u := H(prodID||sk_{ctx}^u)$ using its own sk_{ctx}^u and provide the corresponding zkSNARK proof. The proof is then sent to BML as part of the order acceptance transaction $TX_{accOrd} := (ordID, info, H_i^u, \pi_i^u)$. The transaction is rejected if H_i^u and H_i^v are identical or if the proof π_i^u is invalid.

Once the off-chain interaction is over, users u and v must transmit the data needed to evaluate the interaction. In our design, we use DON to collect data from external systems, verify it, and calculate the required values of all the metrics used in the reputation model. Then, the Reputation Smart Contract (RSC) is triggered to perform the evaluation and update the global reputation scores (*i.e.*, submitting TX_{upRep}). Both u and v have shown their last reputation using their pseudonyms apk_i^u and apk_i^v, respectively. Consequently, the RSC will update the reputation scores of u and v automatically and transparently using the revealed information. It is important to note that the interaction evaluation itself should preserve user privacy. This means that our reputation model does not use or reveal any details about the users' identities involved in the interaction. To guarantee this, we propose the following formula to automatically evaluate the interaction:

$$\begin{cases} T_i = \mathcal{P}\ [\omega_p + \omega_t F_t + \omega_a F_a] \\ \omega_p, \omega_t, \omega_a \in [0,1]\ ;\ \omega_p + \omega_t + \omega_a = 1 \end{cases} \tag{4.1}$$

where T_i is the value of the interaction, $\mathcal{P}$ is a Boolean which refers to the presence of the proof "1" or not "0", *i.e.*, whether the interaction outside the chain has taken place. This could be proof of delivering a "product" or completing a "task". ω_p is the weight of the proof itself. ω_t and ω_a are the weights of the time t and the amount a of the interaction, respectively. F_t and F_a are the functions that normalize t and a, respectively (F_a, F_t $\in [0,1]$). The value and timing metrics are implemented to thwart coordinated attacks. These attacks occur when users collude to boost each other's reputation by engaging in multiple low-cost (*i.e.*, micro) interactions within a brief timeframe. The formula can be extended with additional contextual factors (feedback, data quality, etc.), provided they do not reveal any information about the user's digital or physical identity. The global reputation update is performed using the following threshold-based formula [9]:

$$R_{new} = \begin{cases} (1-\mathcal{W}_f)R_{old} + \mathcal{W}_f T_i & ;T_i \geq T_{min} \\ \mathcal{W}_f R_{old} + (1-\mathcal{W}_f)T_i & ;T_i < T_{min} \end{cases} \tag{4.2}$$

where R_{old} is the old reputation, T_i is the value of the interaction, T_{min} is the trust threshold, and $\mathcal{W}_f \in [0,1]$ is a weighting function that gives more or less relevance to T_i, depending on the role played by the user in the interaction, *e.g.*, "seller" or "buyer", and the value of the interaction itself (*i.e.*, positive or negative interaction).

4.5 Security Analysis

In this section, we examine the key security risks and the measures implemented by DARS to counter these threats. While DARS primarily safeguards interactions within the BML, it is crucial to acknowledge that network activities related to transaction posting and retrieval of real-world data may still expose certain identifying details, such as IP addresses. Consequently, users should employ an anonymous network for the secure utilization of DARS. Using The Onion Router (Tor) [95] can be a practical way to achieve this protection.

- *Sybil Attacks:* Involve creating multiple pseudonymous to manipulate the reputation system. In DARS, each legitimate user is granted only one valid credential for each specific context. Specifically, the AML committee maintains a set of $Granted_{ctx}$ identifiers, representing those that have already received a credential within that context. If M_{pk^u} in M_{cred} is not part of this set, a credential is granted; otherwise, no additional credential is issued for that user. This design ensures that a user cannot generate and utilize more than one valid access token per context. Consequently, DARS effectively guards against Sybil attacks.

- *Unforgeability:* Identity Theft is mitigated in the AML subsystem as users' keys remain in their wallets. These keys are utilized only for signing challenges during the protocol as part of credential verification. Consequently, the assurance of unforgeability within this subsystem is a direct consequence of the overall unforgeability of signatures.

- *User Privacy:* Regarding the privacy of credential issuance, it is important to note that generating a pre-credential for a claim within the Oracle protocol does not disclose any information about the user. Furthermore, given the commitment's hiding property and the privacy guarantees provided by the Multi-Party Computation (MPC) evaluation, there is no opportunity for an attacker to gain knowledge about the user during the issuance process. In addition, since no personal information is used when evaluating interaction within BML, there is no risk of de-anonymization or leakage of information about interacting parties.

- *Reputation Binding:* Our DARS is based on the forgery-proof nature of the cryptographic signatures used to create contextual credentials and submit access tokens. This ensures that the reputation score remains cryptographically linked to the original user or entity.

- *Forward Reputation Binding:* DARS satisfies this property if the signature scheme prevents forgery, the commitment scheme maintains the hiding and binding properties, and the zkSNARK scheme ensures soundness and ZK properties. These combined properties help ensure that the new reputation score is reliable, private (if not shown), and consistently linked to the entity (access token) it represents.

4.6 Evaluation and Results

In this section, we first describe the environment used to evaluate the proposed solution, followed by the experimental setup. Next, we examine the evaluation results of the proposed framework in terms of off-chain and on-chain performances.

4.6.1 Experimental Setup

We carried out the benchmarks on our local BC platform. The platform is a cluster of two HPE ProLiant XL225n Gen10 Plus servers dedicated to the experimentation and evaluation of blockchain solutions. Each server is equipped with two AMD EPYC 7713 64-Core 2GHz processors and 2x256 GB RAM.

To evaluate the proposed solution, we developed a proof of concept for the Decentralized Anonymous Reputation System (DARS). The circuits employed in DARS are implemented using the circom programming language and the circomlib library[1]. We utilized the snarkjs library[2] to compile the circuits and perform the powers of tau ceremony for the trusted setup [96]. Additionally, we developed the smart contracts of DARS using the Solidity programming language[3] and established a local network consisting of twelve validators using Hyperledger Besu[4] as BC client with Proof of Authority (PoA) as consensus protocol. We utilized Web3js library[5] for developing the client side and deploying the system's smart contracts. Lastly, for conducting benchmarking tests, we utilized Hyperledger Caliper [6].

4.6.2 Performance Evaluation

Three metrics are considered for DARS performance evaluation:

- *Time overhead:* refers to the processing time for the proving and verification operations. This time is measured off-chain for the proving operation; or from when a specific transaction that contains a zkSNARK proof is received at the smart contract (on-chain) for verification until the appropriate response is sent back to the prover.
- *Throughput:* is the number of successful transactions per second (TPS).
- *Latency:* refers to the time difference in seconds between the submission and completion of a transaction.

4.6.2.1 Time Overhead

To evaluate the time overhead of our circuits we employ two distinct proving systems, namely zkSNARK Groth16 and PlonK. Groth16 is a circuit-specific preprocessing

[1] https://github.com/iden3/circomlib
[2] https://github.com/iden3/snarkjs
[3] https://docs.soliditylang.org
[4] https://besu.hyperledger.org
[5] https://web3js.readthedocs.io
[6] https://github.com/hyperledger/caliper-benchmarks

general-purpose zkSNARK construction that has become a standard choice in various BC projects [90]. This popularity is owed to its proofs' constant size and efficient verifier time. However, Groth16 necessitates a circuit-specific trusted setup during its preprocessing phase, which could be considered a drawback. On the other hand, PlonK represents a universal preprocessing general-purpose zkSNARK construction [91]. It features an updatable preprocessing phase and boasts a short and constant verification time. Nevertheless, PlonK proofs tend to be larger and take more time to generate compared to Groth16.

Table 4.1 and 4.2 present the timing and memory-related measurements for the Groth16 and Plonk zk-proof components, namely π_{AS}, π_{RS}, and π_i. The proof π_{AS} serves to verify valid credentials on AML efficiently, π_{RS} verifies the validity of RT, and π_i attests the validity of the interaction, preventing self-promotion attacks (see Section 4.4.2).

Table 4.1: Time overhead measurements for the zkSNARK proofs generation and verification using the Groth16 proving system.

TX type	**Proving(ms)**	**Verification(ms)**	**Overall Time(ms)**	**Call Data size**
spendAT (π_{AS})	2400	730	3130	705B
spendRT (π_{RS})	2900	950	3850	705B
newOrd (π_i)	480	640	1120	705B

Table 4.2: Time overhead measurements for zkSNARKS proofs generation and verification using the PlonK proving system.

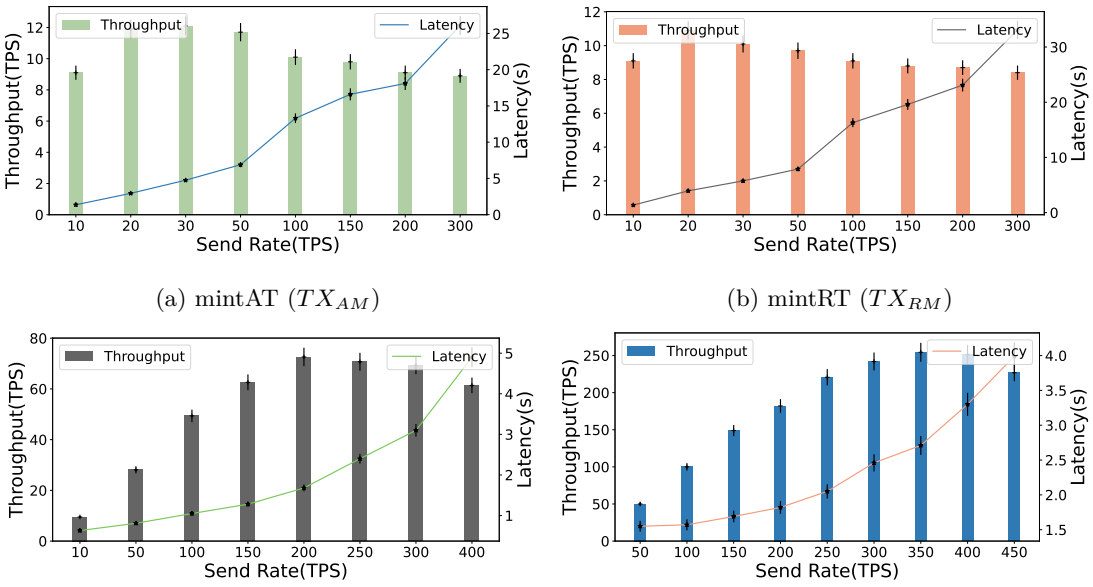
TX type	**Proving(ms)**	**Verification(ms)**	**Overall Time(ms)**	**Call Data size**
spendAT (π_{AS})	67000	760	67760	1750B
spendRT (π_{RS})	79500	935	80435	1750B
newOrd (π_i)	3400	670	4070	1750B

The results show that proofs generation using the Groth16 scheme takes only 100s to 1000s milliseconds (480-2900ms), while it takes relatively longer using the PlonK system (3.4-80s). Compared to verification with Groth16, no significant difference is observed for the proof verification using PlonK.

4.6.2.2 Throughput and Latency

Now, we present a qualitative evaluation of DARS performance based on a series of experiments conducted using Caliper. The experiments involved changing the TX sending rate (ranging from 10 to 450 TPS) using a consistent network configuration for the four main operations performed within our system. The results are illustrated in Figure 4.3.

As the TX send rate increases for each operation, the throughput increases accordingly. For the updateRT TX, it reaches a peak of 255 TPS at a send rate of 350 TPS and then starts to degrade, which indicates that the system is congested. On the other hand, the system latency for the same TX remains relatively low (less than $3s$) as long as the



(a) mintAT (TX_{AM})

(b) mintRT (TX_{RM})

(c) spendRT (TX_{RS})

(d) updateRT (TX_{upRep})

Figure 4.3: Latency and Throughput of DARS

system is not overloaded. The remaining operations exhibit similar behavior, but their performance is comparatively lower. In particular, the mintAT and mintRT operations are the most computationally intensive due to the significant amount of computation required to insert the cm_A and cm_R commitments into the $UCTree$ and $RCTree$ structures, respectively. This increased computational complexity directly translates into higher workloads for these operations.

It is important to note that currently all transactions are sent directly to the mainchain without using any scaling solution. However, except for token minting transactions, DARS performance is quite practical for small to medium-sized marketplaces and crowdsourcing platforms.

4.7 Analytical Comparison

In this section, we provide a detailed comparison of DARS with relevant blockchain-based privacy-preserving/anonymous reputation systems. Table 4.3 outlines the main characteristics of each PPBRS. It presents the system architecture along with the attributes of its ratings and reputation scores.

- The architecture of a reputation system describes the methods used to collect ratings and calculate or update reputation scores. Such architectures can fall into three categories: centralized, decentralized, or hybrid.

- Rating and Reputation properties include the range or domain of their values; The

visibility of reputation, which can be either local or global; Reputation durability, which can be either persistent or non-persistent; Finally, aggregation methods that include various models such as sum, mean, and weighted mean.

Table 4.4 enumerates the security aspects of relevant anonymity-oriented PPBRSs. These properties include:

- *Threats Model:* This describes the types of threats the system is designed to withstand. Threat models can be categorized as Malicious or Semi-Honest. Systems that use a central server are typically classified as Semi-Honest, although some may adopt a Malicious model for other entities.

- *Collusion Resistance:* This property indicates the system's resistance to collusion between entities. Collusion resistance can be classified as partial or strong, depending on the system model. In a strongly collusion-resistant system, a security breach requires more than t of n entities to collude. Otherwise, the system is considered to have partial collusion resistance.

- *Reputation Binding:* The system can be either pseudonym-bound or identity-bound. In pseudonym-bound, changing or creating a new pseudonym means losing reputation, while in identity-bound, the system ties reputation to a unique id. Thus, in identity-bound, entities can maintain their reputation.

- *Trust Model:* This refers to the trust framework used by the system to achieve its security properties. Trust models may rely on a trusted third party (TTP *e.g.*, a server) or employ a trustless approach (*e.g.*, a committee of nodes with arbitrary K or a chosen K).

  Table 4.5 compares the privacy properties achieved in DARS with other anonymity-oriented PPBRSs. These properties include:

  - *Multiple Pseudonyms:* The system allows users to use multiple pseudonyms in their interactions/transactions.
  - *User-Pseudonym Unlinkability:* means that a user's true identity cannot be linked to any pseudonym they use in the system.
  - *Pseudonym-Pseudonym Unlinkability:* means that two different pseudonyms belonging to the same user cannot be linked. The adversary cannot determine whether two given pseudonyms belong to the same user.
  - *Rater Anonymity:* A user can rate another user without revealing their true identity.
  - *Ratee Anonymity:* A user can receive a review without revealing his or her true identity.
  - *Accountability:* A user's pseudonym can be linked to her real identity only if they commits a predefined adversarial act.

Table 4.3: Blockchain-based anonymous reputation solutions: Principles.

Reference	Architecture	Range	Visibility	Durability	Aggregation
[24]	Decentralized	$\mathbb{R}$	Global	No	Open
[97]	Centralized	$\mathbb{R}$, $[0,1]$	Global	Yes	Polynomial
[82]	Hybrid	$\mathbb{R}$	Global	No	Sum
[11]	Centralized	$[0,1]$	Global	Yes	Mean
[22]	Decentralized	$\mathbb{N}$	Global	Yes	Sum
[23]	Decentralized	$\mathbb{Z}$	Global	Yes	Sum
DARS	Decentralized	$[0,1]$	Global	Yes	Weighted Mean

Overall, thanks to the use of zkSNARK proofs on top of two separate ledgers, DARS is the first end-to-end decentralized PPBRS that fully leverages the potential of blockchain in real-world scenarios. DARS achieves strong collusion resistance in a trustless setup, as the evaluation and aggregation processes are fully automated through smart contracts and decentralized oracles.

By integrating AML, DARS can achieve privacy and accountability. In particular, users in our system can use multiple pseudonyms in their interactions. Thanks to the use of zkSNARKs for set membership over commitments, these pseudonyms cannot be linked to the user's true identity or to each other. Thus, DARS achieves both ID-pseudo and pseudo-pseudo unlinkability. In addition, since the rater and the ratee in our design can use a new pseudonym for each interaction, their anonymity is also protected. Furthermore, users' true identities are kept private as long as they are not acting maliciously. This is done by following the CANDID approach, where a set of committee members run a MPC program (for periodic review) to agree on revealing the identity of users acting maliciously or not [73]. In addition, The AML committee can identify credentials associated with users who should be prevented from using the system, such as appearing on a sanctions list, for further action such as blacklisting.

4.8 Conclusion

This chapter discusses concerns regarding privacy preservation and anonymity in blockchain-based reputation systems. More precisely, it proposes a decentralized anonymous reputation system that leverages blockchain and zkSNARKs. The system is built on top of two separate ledgers to decouple identity management from business activities. It makes use of Decentralized Oracle Networks not only to automate smart contracts execution as it is traditionally used but also to import credentials from external systems to prevent sybil attacks without compromising user privacy. DARS users can generate/use as many pseudonyms as they wish on the blockchain to protect their digital identity and guarantee continued anonymity. The proposed framework relies on two Collision-Resistant Hash-based Merkle trees, UCTree and RCTree, over a list of access and reputation commitments, respectively, to guarantee anonymity while main-

Table 4.4: Blockchain-based anonymous reputation solutions: Security Aspects.

Reference	Threats Model	Collusion Resistance	Reputation Binding	Trust Model	Building Blocks
[24]	Malicious	Strong	Pseudo	Trustless	Blind signatures
[97]	Semi-Honest, Malicious	Partial	Id	TTP	Merkle trees, digital certificates
[82]	Semi-Honest, Malicious	Partial	Id	TTP	Group signatures, blind signatures, smart contracts
[11]	Semi-Honest	Partial	Id	TTP	Additive secret sharing, secure multiparty computation
[22]	Malicious	Strong	Id	TTP	PS signature, bulletproof system, NIZK proofs, smart contracts
[23]	Semi-Honest, Malicious	Strong	Id	TTP, Trustless	Pedersen commitments, zkSNARK proofs
DARS	Malicious	Strong	Pseudo	Trustless	Hash commitments, zkSNARKs proofs, smart contracts

Table 4.5: Blockchain-based anonymous reputation solutions: Privacy Aspects.

Reference	Multiple Pseudos	ID-Pseudo Unlinkability	Pseudo-Pseudo Unlinkability	Rater Anonymity	Ratee Anonymity	Accountability
[24]	Yes	Yes	Yes	Yes	No	No
[97]	No	Yes	No	No	Yes	No
[82]	No	Yes	No	Yes	No	Yes
[11]	No	No	No	No	No	Yes
[22]	No	Yes	No	Yes	No	Yes
[23]	Yes	Yes	Yes	Yes	Yes	No
DARS	Yes	Yes	Yes	Yes	Yes	Yes

taining effective reputation management. We also designed a general reputation model that achieves security and privacy properties. Our design applies to all trust-based applications, such as decentralized marketplaces and crowdsourcing platforms. We have implemented and tested a prototype of the proposed framework. The results of this evaluation demonstrate the feasibility and effectiveness of DARS.

To sum up, the main points presented in this chapter are:

- An end-to-end decentralized framework incorporating two distinct ledgers to separate identity management from business activities and a decentralized oracle network to automate smart contracts execution and import credentials from existing systems to prevent Sybil attacks.
- The use of zkSNARK proofs for Set Membership over commitments, allowing DARS users to gain the ability to generate and use numerous pseudonyms to

safeguard their digital identity and ensure anonymity.

- An effective reputation model that achieves all the security and privacy properties of our formal model.
- A proof-of-concept for the proposed framework, leveraging emerging technologies and cryptographic tools is developed. This allows for a more meaningful assessment of DARS's capabilities.

Improving scalability and performance was not our primary goal in this chapter. However, we acknowledge that the performance of DARS needs to be improved for better practicality. Ongoing research in the fusion of ZKPs and blockchain shows promising potential. In particular, advances in L2 scaling solutions such as zkRollups[a] offer hope for significant improvement [27]. Consequently, our next focus will be on improving the scalability of privacy-preserving BRSs. Specifically, we will explore L2 scaling techniques for concurrent on-chain reputation and business management. Hence, a novel L2-based design aiming to improve the scalability of privacy-preserving reputation systems while maintaining almost the same level of security as L1 will be presented.

[a] https://docs.zksync.io/userdocs/intro/

Chapter 5

Leveraging zkRollups to Scale Privacy-Preserving Blockchain-based Reputation Systems

Contents

In the previous two chapters we presented GuRuChain a BRS and DARS a privacy-preserving extension to our BRS. We acknowledge that the scalability and performance of both systems need to be improved. Therefore, this chapter proposes an extension of these frameworks to go beyond their scalability limitation. In particular, it introduces a new decentralized framework that improves the performance of BRS and allows business

transactions and reputation updates to be managed concurrently within the same main-chain. To do so, we leverage zkRollups as a Layer-2 (L2) scaling solution to improve performance and reduce gas costs in crowdsourcing as a common application of BRSs. In order to evaluate its efficiency and demonstrate the increase in performance, we develop a complete proof of concept for the proposed framework.

5.1 Introduction

Current Blockchain-based reputation solutions for trust-related real-world applications fail to tackle the challenge of ensuring both efficiency and privacy without compromising the scalability of the blockchain. Developing an effective, transparent, and privacy-preserving reputation model necessitates on-chain implementation using smart contracts [1]. However, managing task evaluation and reputation updates alongside business transactions such as crowdsourcing tasks on-chain substantially strains system scalability and performance. In Chapter 3, we introduced PoGR, a lightweight consensus scheme to improve the scalability and performance of our BRS GuRuMarket. Layer-1 scaling techniques enable the blockchain to process a large number of transactions per second (*i.e.*, high throughput). Although solutions like sharding can significantly increase throughput, they do so at the expense of security or decentralization (key features of blockchain) [27]. We argue that L1 scaling techniques are essential as transaction finality depends on the state of the L1 blockchain. However, relying solely on these techniques is insufficient for secure large-scale deployment. Therefore, to support a large ecosystem of millions of users to broaden the adoption of BRS, one needs to rival the performance of current centralized applications.

As a result of the growth of the Internet and the prevalence of mobile devices, crowdsourcing become increasingly popular, and the platforms facilitating this approach have experienced a significant surge in usage and recognition. The term crowdsourcing was first introduced by Jeff in 2006 [98]. It refers to a collaborative approach that delegates tasks, problems, or ideas to a broad collective set of contributors. Crowdsourcing platforms, like E-commerce platforms, often rely heavily on reputation systems to maintain quality control, ensure trust among participants, and incentivize desirable behavior. Requesters in these systems can use reputation scores to filter and select workers for their tasks. They may prefer to assign tasks to workers with higher reputation scores, which often indicate reliability and competence [4].

Most operating crowdsourcing platforms, such as Fiverr[1], are centralized, which raises concerns about security and transparency. This concentration also raises privacy concerns as user information and activities may be more susceptible to breaches or misuse. Security becomes a pressing issue due to the vulnerability of a central point of access, potentially exposing the platform to various risks and threats. Moreover, the lack of transparency in decision-making or data handling within such centralized platforms can lead to ambiguity, eroding users' trust and understanding of how their information is

[1] https://www.fiverr.com/

managed and utilized. In reputation-centric crowdsourcing systems, transparency gains even greater importance as users need to know how their reputation scores are maintained. More precisely, they seek the ability to track and verify updates to their scores at any given moment [2, 40].

In the last few years, numerous efforts have arisen to leverage blockchain technology in addressing these issues [7, 40, 57]. The decentralization, transparency, and efficiency brought by blockchain are clearly what we need to build effective trustless reputation systems for crowdsourcing or any real-world application. However, as discussed previously in Chapter 3, transparent and effective blockchain-based reputation management requires the reputation model to be implemented on-chain using smart contracts to enhance trust and achieve accountability [9, 40]. In this situation, the blockchain network is required to handle additional transactions involving task evaluation and global reputation updating. This added workload significantly impacts the system's scalability and performance, leading to heightened gas costs, prolonged processing times, and increased time overhead [77]. Therefore, addressing these challenges becomes imperative to implement this solution in real-world situations. In other words, to support a large ecosystem of millions of users to broaden the adoption of BRS, one needs to match the performance of current centralized applications. Recently, zkRollups, an L2 scaling technique for EVM[2]-based blockchains, has seen great success due to its ability to achieve scalability while maintaining a high level of security thanks to zero-knowledge proofs [27].

The critical inherent limitations of BRS mentioned above have rarely been addressed in the literature when incorporated into crowdsourcing. To further leverage the superiority of combining BRS and crowdsourcing, this chapter introduces "RollupTheCrowd", a novel blockchain-based platform that leverages zkRollups to improve the scalability of BRS systems for crowdsourcing. We employ zkRollups as an L2 scaling solution for our blockchain-based reputation system primarily due to their ability to significantly enhance transaction throughput and reduce costs while maintaining a high level of security. By bundling multiple transactions into a single batch, zkRollups reduce the amount of data that needs to be processed and stored on-chain. Most importantly, zkRollups thanks to zero-knowledge proofs maintain the same level of security as the underlying blockchain. To our knowledge, we are the first to use this emerging L2 scaling approach to improve the throughput and reduce the cost of on-chain reputation management.

The remainder of this chapter is organized as follows. The rollup preliminaries are presented in section 5.2. Section 5.3 presents related work. Section 5.4 presents the proposed framework and describes the designed crowdsourcing logic. Section 5.5 describes the proposed reputation model in detail. Section 5.6 presents the performance evaluation. Finally, Section 5.7 concludes the chapter.

[2]Ethereum Virtual Machine

5.2 Rollups Preliminaries

Rollups are an L2 scaling solution that provides a means to streamline transaction validation, reducing resource and time overhead by minimizing the amount of data each node must process. This optimization is achieved through a secondary layer network involving actors who handle transactions off the primary chain. Subsequently, the transaction data is consolidated into batches and sent onto the L1 blockchain. Two types of Rollups exist Optimistic and Zero-Knowledge (zk) Rollups. In Optimistic rollups, an L2 entity called the aggregator or operator, executes transactions off-chain and publishes only transaction data and the new state root on-chain. L1 and L2 nodes in this type of rollup optimistically assume that the computation executed by this entity is valid. ZkRollups, on the other hand, the operator must compute a zero-knowledge proof for the execution done on L2 for each batch of transactions sent to L1. The proof is called a "validity proof" and is constructed using Non-Interactive Zero-Knowledge (NIZK) proving systems such as zkSNARKs [89]. Calculating the proofs is complex, but checking them on the mainchain is fast [27].

In summary, both optimistic and zkRollups aim to improve performance by moving computation from L1 to L2. However, zkRollups guarantee a higher level of security thanks to the validity proofs in place. In particular, zkSNARKs proofs provide cryptographic guarantees that transactions are executed correctly on L2. Transactions are not finalized unless the proof sent to an L1 smart contract is valid. This allows zkRollups to maintain the same level of security as the underlying blockchain while enabling greater scalability and cost efficiency.

5.3 Related Work

In the previous chapters, we reviewed existing initiatives that have contributed to the decentralization of reputation systems using blockchain. In this section, we look specifically at those designed for crowdsourcing platforms. We will focus on key aspects such as decentralization, reputation management, user privacy, and scalability.

zkCrowd [57] presents a hybrid blockchain crowdsourcing platform with two ledgers. It combines Delegated-Proof-of-Stake (DPoS) and Practical Byzantine Fault Tolerance (PBFT). These consensus protocols are chosen for their good performance, but the dual use of these protocols introduces complexity into the design of the hybrid blockchain without sufficient feasibility analysis. CHChain [99] framework also proposes a hybrid structure and uses a Reputation-based PBFT consensus scheme to improve system throughput. However, the feedback-based reputation model is vulnerable to bad collusion attacks, raising concerns about the security of the whole system. RC-CHAIN [100] focuses on vehicular data sharing, using a consortium blockchain. However, it introduces centralization through Roadside Units (RSUs), acting as intermediaries. In [83, 84], supervised blockchain architectures are adopted for mobile crowdsourcing (sensing), introducing centralized entities such as Key Distribution Center (KDC) and Task Distribution Center (TDC).

In [101], a decentralized reputation system for e-commerce stores content on IPFS, addressing content volume concerns without explicitly delving into performance and scalability challenges. The proposed reputation model groups evaluations and considers transaction magnitude, interaction time, and historical reputation scores which lead to linkability and privacy exposure. RBT [102] tailors reputation assessment based on individual roles, raising re-entry attack concerns. ExCrowd [103] addresses challenges associated with new users with an exploration approach through linear regression and decision tree algorithms. These algorithms can be computationally intensive and lack transparency and flexibility. In [83], a reputation model focuses on data reliability for the crowdsensing use case, noting potential issues with storing a high volume of sensed data. NF-CROWD [104] introduces a novel scaling solution for crowdsourcing, proposing optimized one-to-many (N:1) and many-to-one (1:N) transactions to reduce fees. However, the proposed approach does not discuss performance improvement and only presents gas cost improvements.

Ensuring good scalability is fundamental to the design of a blockchain-based reputation solution. However, some previous studies [83, 101, 103] tend to ignore the performance challenges associated with on-chain reputation management. Meanwhile, alternative methods employing rapid consensus protocols to scale the system [57, 102] or resorting to centralization for enhanced performance [83, 99] have proven ineffective and, at times, insufficiently secure. We believe that L1 scaling solutions are essential, but cannot offer a complete solution to the problem. Congestion on layer 1 still occurs no matter how fast the blockchain protocol is. Solana[3], which is often considered one of the fastest blockchains, has recently been affected by this problem.

In summary, finding the best tradeoff between reputation effectiveness, privacy preservation, and scalability is vital for building robust blockchain-based crowdsourcing solutions. The existing studies present several limitations that prevent their widespread application. Therefore, to overcome these issues, this chapter introduces "RollupTheCrowd", a scalable, privacy-preserving, and fully decentralized reputation-based crowdsourcing framework.

5.4 RollupTheCrowd Framework

After exploring related studies and the challenges identified in current solutions, in this section, we will present our framework. We begin by describing the complete architecture of the system and then detail all the components of the proposed solution.

5.4.1 System Architecture

Figure 5.1 shows the proposed architecture for RollupTheCrowd. It has four components.

[3] https://cryptopotato.com/solana-explains-reasons-behind-the-recent-network-slowdown/

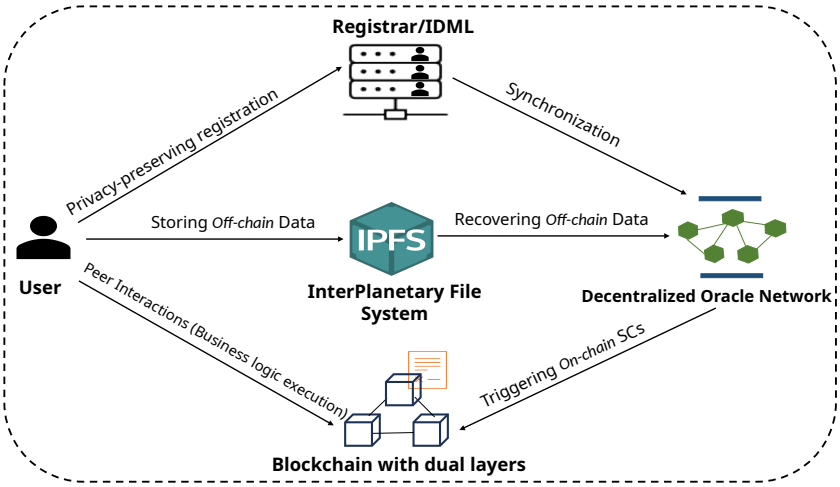


Figure 5.1: High-Level System Architecture

5.4.1.1 Main Ledger with Dual Layers

The main ledger is the central element of our crowdsourcing platform, featuring a dual-layer structure as shown in Figure 5.2. The first layer operates as a traditional permissioned blockchain network, employing a Proof of Authority (PoA) consensus, ensuring the integrity of the mainchain (L1). The second layer employs a zkRollups solution to enhance scalability. Instead of processing each transaction on the mainchain, a batch of transactions is processed and validated off-chain (on L2), by an aggregator. It then publishes the new state root, compressed transaction data, and proof of validity on the mainchain. This proof of validity ensures the computation made to execute the transactions was correct. zkRollups inherent security from L1 and upholds privacy by design, which makes them the perfect solution for our crowdsourcing system. This design not only reduces congestion and lowers fees, but also ensures transparency and maintains the decentralized nature of the system. At the same time, the security guarantees of the mainchain are preserved through cryptographic proofs.

5.4.1.2 A Decentralized Registrar

To complement the capabilities of the main ledger, we introduce a registrar ledger, which is responsible for identity management in our decentralized ecosystem. This ledger serves as an identity management system (*e.g.*, CanDID [73]). It provides a secure environment where users can assert their identities using zero-knowledge proofs. In RolluptheCrowd, the role of registrar can be taken over by a group such as the CanDID committee that guarantees identity uniqueness in a decentralized and privacy-preserving manner.

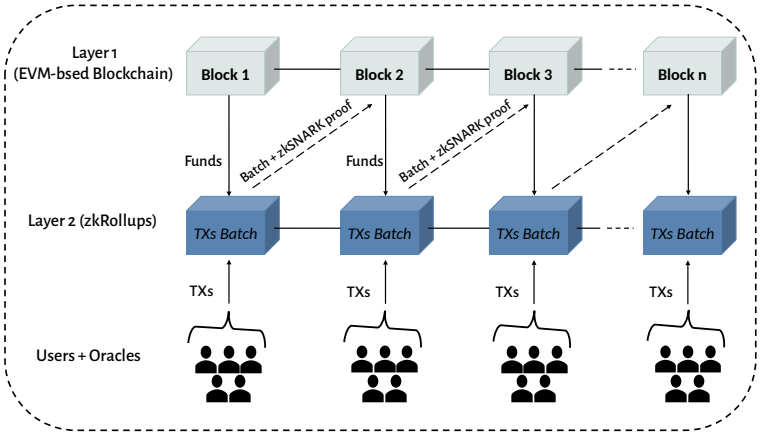


Figure 5.2: Dual Layers Workflow in RollupTheCrowd.

5.4.1.3 Dual Layers with IPFS Integration

The two layers in our design are coupled with an InterPlanetary File System (IPFS) [75], providing an efficient solution to the storage issues inherent in traditional blockchain-based crowdsourcing systems. By offloading substantial data off-chain to IPFS, the transaction times and costs within RollupTheCrowd are optimized.

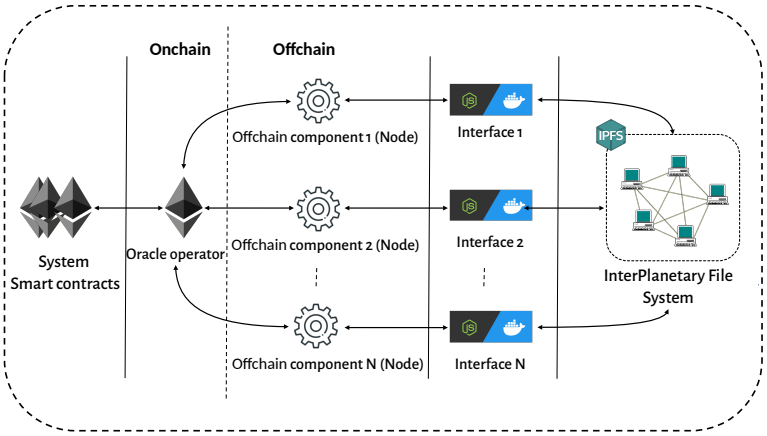


Figure 5.3: Decentralized Oracles in RollupTheCrowd.

5.4.1.4 Interoperability and Synchronization

A key aspect of our system architecture is the seamless interaction between its components. Smart contracts deployed on-chain can be triggered by authorized entities to execute operations. This can be achieved using decentralized oracles, ensuring that the data sent to our blockchain is accurate. The inclusion of IPFS and the off-chain storage of business logic data requires a reliable mechanism for data synchronization

and retrieval. This is where decentralized oracles excel, ensuring that data from the IPFS can be efficiently used on the main blockchain without compromising security or decentralization. Figure 5.3 explains how these decentralized oracles serve as a bridge between the on-chain and off-chain sides. The interfaces allow the integration of custom computations and specialized APIs. They are services that the core of the Oracle node communicates through its API with a simple JSON specification [26].

5.4.2 RollupTheCrowd Modules

Below, we list the functional modules we developed using smart contracts and deployed on the system network.

5.4.2.1 Oracle Operator Module

An Oracle operator module is a smart contract within a blockchain ecosystem that acts as an intermediary or bridge between the blockchain and external data sources. Its primary purpose is to fetch, verify, and provide off-chain data to on-chain smart contracts, enabling them to interact with real-world information.

5.4.2.2 Access Management Module

It is a smart contract that manages access control and permissions within our Decentralized application (DApp). It is a common approach used to control who can perform certain actions or access specific functionalities within the application. The primary objective of this is to ensure that only authorized users or addresses are allowed to execute specific operations or access sensitive data *e.g.*, only Oracles can trigger the $calculateNewRep$ function.

5.4.2.3 Business logic Module

This is the smart contract that implements crowdsourcing operations within our system. It enables users to create, submit, and complete tasks in a decentralized manner. It implements the complete business logic behind the crowdsourcing scenario. Details are presented in the next section.

5.4.2.4 Reputation Module

This module is the component responsible for managing reputation scores in the system. It implements our proposed privacy-preserving reputation model that we detail in Section 5.5.

It is important to point out that with the integration of this reputation module into the framework, we have the option of employing reputation-centric consensus [2,9]. This alternative offers better scalability and fairness than PoA and PoS respectively.

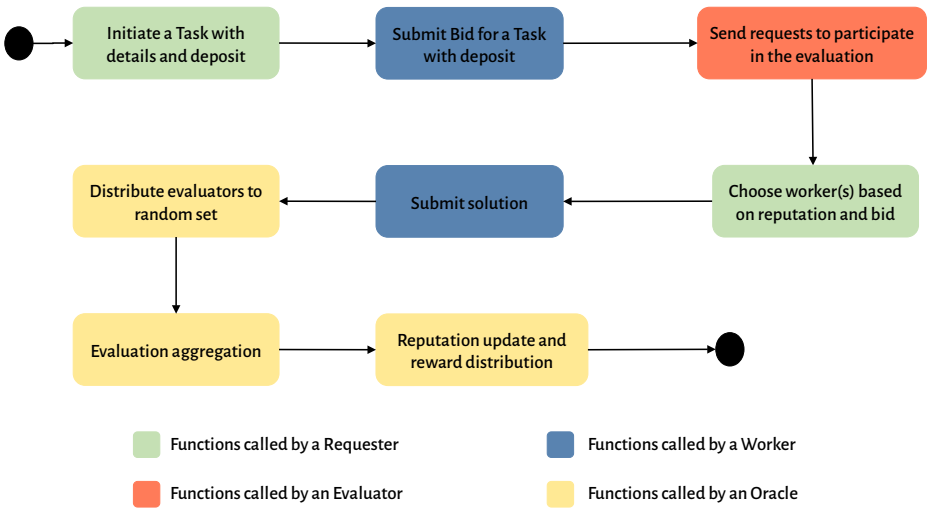


Figure 5.4: The crowdsourcing workflow within RollupTheCrowd.

5.4.3 Reputation-based Business Logic

The entire business logic of RollupTheCrowd is implemented using smart contracts (SCs). We employ three primary entities in our crowdsourcing process: *Requesters*, *Workers*, and *Evaluators*. SCs in RollupTheCrowd are designed to provide transparency and accountability among these entities by managing both reputation and crowdsourcing tasks on-chain.

Figure 5.4 depicts the crowdsourcing workflow within our framework, illustrating SCs functions called by each entity. Deposits are locked into a SC as collateral, and any misconduct will result in penalties. Evaluators are selected before Workers, with Workers unaware of the Evaluators' identities. To ensure fair evaluations, Evaluators are randomly assigned to evaluate tasks. This setup encourages timely and high-quality contributions from Workers while maintaining the integrity of evaluations by Evaluators. Below, we detail the core functionalities implemented through smart contracts code.

5.4.3.1 Create Task

Algorithm 3 presents the *createTask* function. The function initially verifies whether the caller is a registered user; if affirmative, it proceeds to store only the necessary data on-chain, as all other details have already been submitted off-chain to IPFS by the requester via a dedicated interface. Additionally, the function updates the amount to be used later in the reputation model.

Algorithm 3: Create Task

```
1 T: New Task (each task T has: CID, amount)        ▷ CID is the content identifier on IPFS
2 begin
3     Assert(isUser(msg.sender) = true)
4     tasks[T.CID].requester ← msg.sender
5     tasks[T.CID].CID ← T.CID
6     tasks[T.CID].amount ← T.amount
7     updateMinMaxAmounts(T.amount)        ▷ update A_min and A_max of our reputation model
```

5.4.3.2 Submit Solution

Algorithm 4 implements the *submitSolution* function, triggered by the worker upon task completion. This function first verifies if the bidder submitting the solution is indeed the assigned worker, ensuring that workers can only submit solutions within their bids. Subsequently, it checks whether the bid has been accepted by the requester, allowing only accepted workers to submit their solutions. Finally, it stores the Content Identifier (CID) of the submitted solution to IPFS.

Algorithm 4: Submit solution

```
1 T: New Task (each task T has: CID, amount, bid progress, and bid submitter)
2 B: Task bid (each bid B has: CID)
3 S: Task Submission (each submission S has: CID)
4 begin
5     Assert(tasks[T.CID].bidSubmitter[B.CID] = msg.sender)
6     Assert(tasks[T.CID].bidProgress[B.CID].answer = true)
7     tasks[T.CID].bidProgress[B.CID].submission ← S.CID
```

5.4.3.3 Distribute Evaluators to Random Sets

Algorithm 5 implements how we distribute evaluators into random sets to achieve randomness in the evaluation process, each set is responsible for one submission. The function is called only by the oracle when there are enough evaluators for the task.

5.4.3.4 Calculate New Reputation

Upon completion of the evaluation of a specific task using corresponding measures detailed in section 5.5, Evaluators transmit their local ratings to the Oracle. The Oracle network checks the validity of these ratings and then calculates the average scores and submits the result on-chain through the invocation of the *calculateNewRep* function. This function computes the new reputation using the proposed reputation model, considering the task type, and then initiates the *updateReputation* function.

After describing the architecture of RollupTheCrowd and its main components, we will now delve into the mathematical details of the proposed reputation model. Next

Algorithm 5: Distribute Evaluators To Random Sets

```
1  T: New Task (each task T has: CID, bids , and evaluators)
2  B: Task bid (each bid B has: CID)
3  S: Task Submission (each submission S has: CID) R: A random number generated by DON
     using a VRF.
4  begin
5      evaluators ← tasks[T.CID].evaluators
6      numSets ← tasks[T.CID].bids.length
7      for i ∈ [0, 1, ..., evaluators.length] do
8          n ← i + (R)mod(evaluator.length − i))
9          permutation(evaluators[n], evaluators[i])
10     for i ∈ [0, 1, ..., evaluators.length] do
11         tasks[T.CID].evaluatorSets[i mod numSets] ← evaluators[i]
```

section presents our reputation model.

5.5 Reputation Modeling

We propose a reputation model that can be adapted to various crowdsourcing situations including solving complex problems, collecting data, conducting research, or harnessing collective intelligence. Those diverse scenarios can be categorized from our perspective into two principal categories: problem-solving and knowledge acquisition. In problem-solving tasks, crowdsourcing initiatives focus on human-level intelligence or expertise to perform. Participants bring reasoning, problem-solving, decision-making, and learning, among other cognitive abilities that are characteristic of human intelligence. Examples of such scenarios include crowdsourcing platforms dedicated to innovation, where individuals contribute creative solutions for product development or process improvement. On the other hand, knowledge acquisition situations in crowdsourcing aim to gather a broad range of data from a diverse group of individuals or machines. This may involve the crowdsensing use case.

5.5.1 Task Evaluation

Recognizing the two types of crowdsourcing situations allows us to design targeted evaluation strategies and approaches that meet the specific needs and goals of each scenario. We define a common metric for all crowdsourcing scenarios, namely value rating. We first present this common metric and then the metrics specific to each situation.

5.5.1.1 Common Metric: Value Rating

It is not expensive for a malicious requester to submit multiple tasks with low costs to a selected worker to unfairly post low/high ratings. Therefore, to alleviate the problem

of unfair ratings, the rating of a task should be related to the task amount A_t. Thus, the value rating V_r is computed using the following function:

$$V_r = f(A_t) = \frac{A_t - A_{min}}{A_{max} - A_{min}} \tag{5.1}$$

5.5.1.2 Problem-Solving Tasks Metrics

Problem-solving tasks refer to cognitive tasks that require human-level intelligence or expertise to perform. These tasks typically involve reasoning, problem-solving, decision-making, and learning, among other cognitive abilities that are characteristic of human intelligence. Examples of human intelligence tasks include natural language understanding, logical reasoning, creativity, and social intelligence. Evaluating these tasks involves subjective judgments, which can vary from one evaluator to another. For instance, tasks that require creativity may elicit multiple valid solutions, leading to diverse ideas and approaches among individuals. To minimize conflicts in evaluation, we introduce objectivity and establish clear criteria during the task posting phase. By providing explicit guidelines and specifications upfront, we strive to facilitate a more structured and consistent evaluation process. This helps to ensure that evaluators have a standardized framework to assess tasks and reduce discrepancies. Within each Problem-solving task, the assessment of user submission is influenced by various factors. We define two factors to assess the Worker's submission: Effort Rating and Contextual Rating.

- *Effort Rating (E_r):* to bring more objectivity to feedback submission, we gauge the user effort on the task by considering the two following parameters:

  1. *Task completeness:* $C_t \in [0, 1]$ designates the degree of completion or realization of a task or project. It is a measure of progress toward the task goal and can be computed using a defined checklist by the requester.
  2. *Task Quality:* $Q_t \in [0, 1]$ refers to the level of expertise or efficiency in performing a specific task. It can be calculated using a set of rubric rules defined by the requester. Rubric rules are criteria or guidelines used to evaluate the quality of an assignment. For example, for a logo design task, quality evaluation using rubric rules may include creativity and originality, relevance to brand identity, technical execution, and aesthetic appeal. Each of these metrics can be rated on a scale of one to ten.

  We give requesters the freedom to determine the weighting of completeness and quality, enabling them to specify their preferences in advance. Therefore, the effort rating is computed as follows:

$$E_r = f(C_t, Q_t) = \alpha C_t + \beta Q_t \; ; \; \alpha + \beta = 1 \tag{5.2}$$

- *Contextual Rating (C_r):* Worker submissions can be evaluated taking into account additional validity aspects, which may differ depending on the use case. For example,

in programming contexts, considerations may relate to the success of test cases or the cleanliness of the code, enabling a more in-depth and personalized assessment to measure the quality and effectiveness of the submission.

The overall rating of the problem-solving task $T_r = f(V_r, E_r, C_r)$, which is a linear combination between the three metrics. The weighting of each metric is determined by either the group or the platform operator.

5.5.1.3 Knowledge Acquisition Task Metrics

In this type of task, the focus is on collecting data. The gathered knowledge can be evaluated by its reliability and can be provided by IoT devices (temperature, pressure, etc.) or humans (Location, pictures, surveys). There are many existing methods for evaluating data reliability. Inspired by the method proposed in [83], we develop a new evaluation method that enables accurate estimation of knowledge acquisition based on the reliability of acquired data.

- *Data Distortion Rating (D_r):* Data distortion represents the difference between observation and truth. We use the deviation of sensed data V_i from V_a to denote the degree of data distortion. V_a is the final aggregation result held by the decentralized oracle, which is considered truth data. We calculate the squared difference between V_i and V_a, then the result is normalized as the deviation of V_i from V_a.

$$d_i = (\frac{V_i - V_a}{b_U - b_L})^2 \tag{5.3}$$

  b_L and b_U represent the lower and upper bounds of the sensed data range, respectively. They are used to normalize the deviation d_i (*i.e.*, $d_i \in [0, 1]$). We calculate the data distortion metric as follows:

$$D_r = 1 - d_i \tag{5.4}$$

- *Contextual Rating (C_r):* In addition to the data distortion rating, other contextual factors could be taken into account toward evaluating the reliability of data. For instance, the specific sensing task is strict in location and time, which means that the sensing data from the expected location might be more reliable than that from a remote location.

Similar to the problem-solving task the overall score is $T_r = f(V_r, D_r, C_r)$ the linear combination of three ratings V_r, D_r, and C_r.

5.5.2 Reputation Update

In our reputation calculation process, we use the task evaluation method outlined in the previous section, along with past behavior. While developing our model, we carefully considered privacy concerns. Therefore, the only data the model requires is the users' current reputation scores. In RollupTheCrowd, each new user is assigned an initial

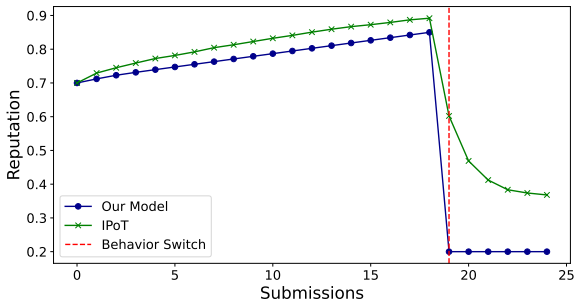


Figure 5.5: Reputation Model Effectiveness Compared To [2].

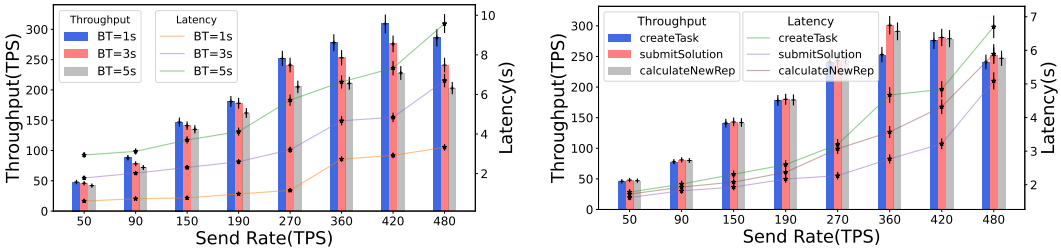
reputation score R_{init}. This value can be the average of the reputation scores of all existing users or another value fixed by the system operators.

Reputation is the perception that users have of an individual in the system, and past behavior is a key factor in calculating reputation. The reputation update process differs depending on whether the behavior exhibited is considered good or bad. We define good work when the task value exceeds a certain threshold $T_{min} \geq R_{init}$, which we consider the critical line of trust [4, 57]. Otherwise, the work is categorized as bad. When updating for good behavior, the model gives more weight to the old reputation score, resulting in reputation growth that matches the expected positive behavior. Conversely, in the case of bad behavior, the emphasis is shifted to the current task score, imposing a stronger punishment as a consequence [9]. The update formula is as follows:

$$R_{new} = \begin{cases} (1 - \mathcal{W}_f)R_{old} + \mathcal{W}_f T_r & ; T_r \geq T_{min} \\ \mathcal{W}_f R_{old} + (1 - \mathcal{W}_f)T_r & ; T_r < T_{min} \end{cases} \tag{5.5}$$

where, T_r is the task rating, R_{old} refers to the old reputation, and R_{new} is new reputation. $\mathcal{W}_f \in [0, 1]$ is a weighting function that gives more or less relevance to T_r, depending on the worker's behavior. It uses S the number of submissions done by the worker, $\mathcal{W}_f(T_r, S) = \kappa T_r \frac{1-e^{-\lambda.S}}{1+e^{-\lambda.S}}$, which leads the model to become progressively stricter as the worker engages in more tasks.

The ability to respond quickly to unexpected actions is an essential feature of an effective reputation model. To prove that RollupTheCrowd possesses this characteristic, we compared our model with the model employed in IPoT [2], which uses a similar approach based on the evaluation of crowdsourcing interactions. Figure 5.5 shows the variation in updates in response to positive behavior, following the user's interactions up to interaction 25, when his/her behavior becomes negative. As soon as a negative action is taken, the reputation score declines rapidly in both systems. However, the score decreases faster in our model. This difference reflects our system's increased ability to react to inappropriate behavior.



(a) L1 Throughput and Latency of CreateTask TX under different Block Times 1s, 3s, and 5s.

(b) L1 Throughput and Latency under three different workloads for a Block Time = 3s.

Figure 5.6: RollupTheCrowd L1 Throughput and Latency.

5.6 Evaluation and Results

The main objective of RollupTheCrowd's development is to build a scalable and decentralized system capable of efficiently managing reputation and crowdsourcing tasks simultaneously. To validate the feasibility, scalability, and effectiveness of our proposed framework, we developed a proof of concept. This section details the experimental setup, followed by a brief overview of the technologies utilized during development and experimentation. Next, the key metric used for evaluation and the results demonstrating our system's performance for relevant benchmarks are described.

5.6.1 Experimental Setup

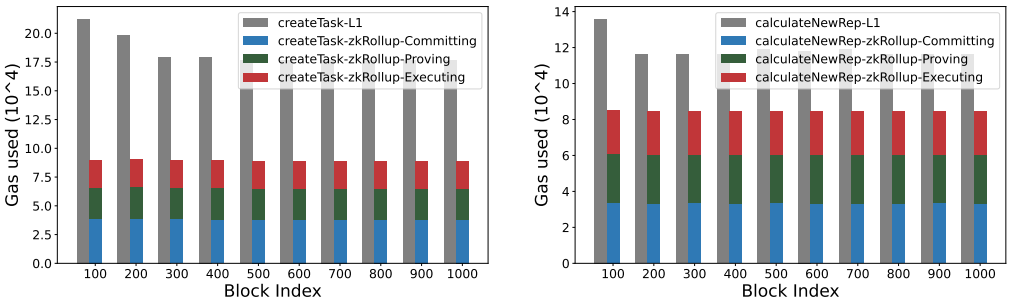
The deployment of the proposed crowdsourcing platform and performance tests are carried out on a cluster of two servers HPE ProLiant XL225n Gen10 Plus dedicated to the experimentation and evaluation of BC solutions. Each server is equipped with two AMD EPYC 7713 64-Core 2GHz processors and 2x256 GB RAM.

We implemented our smart contracts using Solidity and used Geth[4] to run our EVM-based blockchain network. For the Oracle integration, we used Chainlink[5] nodes with customized external adapters within Docker-containerized Node.js servers [26]. For the L2 scaling solution, we used zkSync[6] Rollups. To demonstrate the efficiency and scalability of the proposed solutions, we conducted extensive benchmarks using Hyperledger Caliper[7].

5.6.2 Performance Evaluation

Our experimental methodology involves running tests on separate modules for fine-grained evaluation. Each module is triggered for executing several workloads. We submit a predefined amount of transactions (TXs) using different sending rates. We aim to

[4] https://geth.ethereum.org/
[5] https://chain.link/
[6] https://zksync.io/
[7] https://github.com/hyperledger/caliper-benchmarks



(a) Create Task TX

(b) Calculate New Reputation TX

Figure 5.7: Gas Consumption: A Comparison Between L2 and Non-L2 Implementation.

provide a qualitative evaluation of the business logic and reputation functions. To do this, our performance evaluation focuses on three key metrics:

- *Throughput:* refers to the number of successful transactions per second (TPS).
- *Latency:* refers to the time difference in seconds between the submission and completion of a transaction.
- *Gas:* is a unit that measures the computational work required to perform operations and is influenced by the complexity of the operation, the computational steps involved, and the amount of data processed.

The results below concern the evaluation of a complete crowdsourcing problem-solving scenario, from task creation to reputation updating.

5.6.2.1 L1 Throughput and Latency

We begin our analysis with the performance of the mainchain. Figure 5.6a illustrates the throughput and latency of the heaviest function *createTask* in our design for different block periods (1s, 3s, 5s). Latency increases as the block period increases, which is obvious. However, even with a block duration of 5s, our approach of submitting data off-chain and storing only essential data on-chain proved to be very efficient. The system achieves its best throughput of 310 TPS with a send rate of 420 TPS. By changing the workload type, Figure 5.6b compares *submitSolution* to *createTask* and *calculateNewRep*, the former has better throughput and latency as it requires relatively less computation on-chain.

5.6.2.2 L2 vs L1 Performance

Before discussing the comparison results, it is important to show the workflow of a TX on zkSync L2. Figure 5.8 depicts the various stages a TX goes through before it is finalized. When a TX is sent to a zkSync node, it is first included in the TXs mempool, and marked as pending. The operator slices the transaction mempool into blocks called

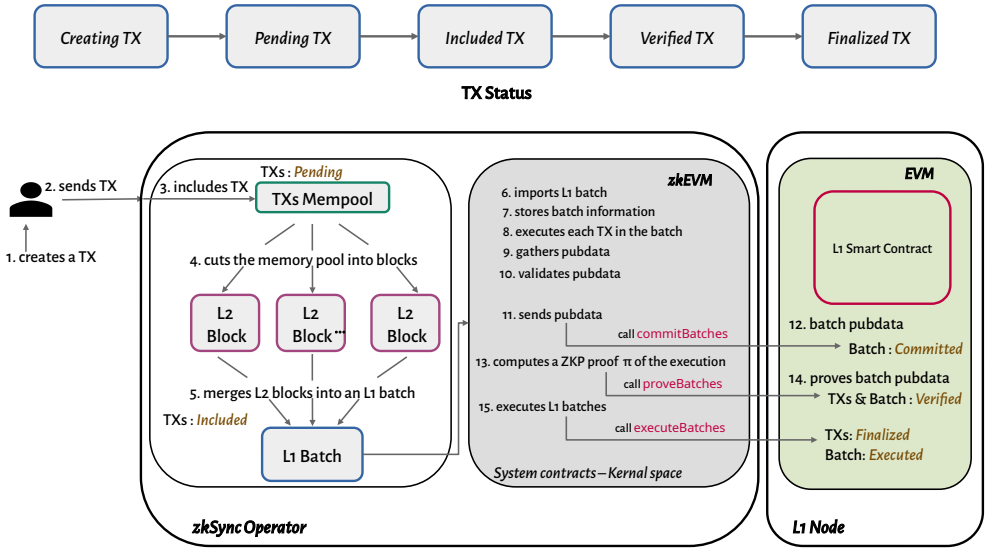


Figure 5.8: The workflow of a TX on a zkSync L2.

L2 blocks. This provides a fast software validation for user experience, as wallets show validation to the user as soon as the transaction has been included into an L2 block. The TX status is set to "included" once the operator has put TX into an L2 block. Multiple L2 blocks can be merged into one L1 batch containing all transactions. The L1 batch is then made available to the *Bootloader* L2 contract. This smart contract is unique because it is the entry point to the zkEVM. The *Bootloader* starts with a call to the *SystemContext* system contract, setting multiple context variables such as the L1 batch timestamp, its index, and the hash of the previous L1 batch. Once variables are set, it executes the transactions. The execution of transactions in zkEVM generates public data. This data called *pubdata*, is stored on L1 and can be used to reconstruct the complete state of the zkSync VM. To reduce costs, the storage of this data is optimized by compression on the L2 before it is sent to the L1 blockchain.

Once all the L1 batch *pubdata* has been compressed, it is committed to the *L1Messenger* system contract. This contract verifies that the *pubdata* is consistent and was compressed properly. The valid *pubdata* is stored on zkSync's L1 smart contract using the zkEVM-specific opcode *to-l1*. This step is called "L1 batch commitment" and is done by calling the *commitBatches* function on the L1. Once the L1 batch data is stored on L1, the L1 batch is set to the status committed. Note that TXs in the batch are still included. Then, to prove the correct execution of the transactions, the operator uses specific ZK circuits. In particular, it produces a ZK witness each time an L1 batch is processed. This witness allows the operator (*i.e.*, the prover) to prove the correct computation of the batch. When the batch is committed to zkSync's L1 smart contract, the witness can be provided to the contract. The L1 smart contract then delegates verification to a specialized smart contract (*i.e.*, the verifier). This verification step called "L1 batch proving" is done by calling the *proveBatches* function on the L1 smart

contract. Once the *pubdata* is verified on the L1, the batch and all its transactions are set to the "verified" status. The final step of the workflow is the execution of the batch. This means that the state obtained after the L1 batch must become the official state of the L2 in the L1 contract and all TXs become "finalized". This final step called "L1 batch execution", is done by calling the *executeBatches* function in the L1 smart contract.

Now we discuss the results of the L2 versus L1 evaluation. Figure 5.7 shows the gas consumption associated with the *createTask* and *calculateNewRep* functions in two different implementations of our solutions. The transaction batching process in zkSync Rollup has three steps on L1: commit, prove, and execute, during which batches are committed, proven, and executed on L1. Each step incurs gas consumption, with the total gas being the sum of the gas expended in these three stages. The results clearly show that even when sending a single TX, the gas consumption significantly decreases when passing through L2. Furthermore, these results also demonstrate that as the transaction complexity changes when calling the *calculateNewRep* function, the gas cost does not change much and remains below that of the L1 execution.

Table 5.1: Time overhead (s) for different functions

Function calls (TX)	1	5	10	20	50	100
CreateTask	0.13	0.84	1.28	2.18	4.85	10.35
SubmitSolution	0.13	0.81	1.28	2.26	4.61	9.9
CalculateNewRep	0.13	0.83	1.29	2.19	5.09	9.94

Table 5.2: Gas consumption of createTask: L1 vs L2

Calls	**Dual layer (L2)**				**Single Layer (L1)**
—	commit	prove	execute	Total	Total
1	38828	27260	23964	**90052**	**212615**
2	33964	27272	23964	**85200**	**396636**
5	33348	27284	23964	**84596**	**896100**
10	34348	27272	23952	**85572**	**1792080**
15	34324	27272	23964	**85560**	**2646120**
20	35396	27284	23964	**86644**	**3966360**
25	68744	29904	26584	**125232**	**4412700**

Table 5.1 presents the latency for the *createTask*, *submitSolution*, and *calculateNewRep* functions. The results indicate that concurrent computation for multiple transactions does not take more than a few seconds. We must mention that within this duration, the transactions are included in L2 blocks and are not yet finalized.

Table 5.2 illustrates the gas cost dynamics associated with multiple function calls in two different scenarios. With an L1 implementation, the gas cost increases linearly with

the number of calls. Since all calls have almost the same gas cost, the resulting overall cost is close to the cost of a single call multiplied by the number of calls. Using L2 (zkRollup), on the other hand, allows the gas cost to remain stable for up to 20 function calls. This stable cost indicates the aggregation of up to 20 transactions into a single batch, proving the effectiveness of zkRollups' batching. Upon exceeding 20 function calls, a doubling of commit gas costs occurs, indicating the submission of a new batch to L1. Compared to costs obtained in the L1 scenario, there is a significant reduction of about 20X, demonstrating the consistent benefit of batching with zkRollups. Finally, we estimate that RollupTheCrowd can reach an effective throughput of 6000 TPS (L1-Throughput x Batch-Size = 300 x 20 TPS) with this configuration.

5.7 Conclusion

In this chapter, we propose a solution to the problem of managing reputation and business operations concurrently within the same blockchain. This is to cope with BRSs' high demand in terms of throughput. To achieve this, we design and develop a fully decentralized framework that improves the scalability and performance of BRSs using zkRollups. The proposed framework RollupTheCrowd is a novel blockchain-based crowdsourcing platform with a privacy-preserving reputation model and an L2 scaling solution. Using zkRollups enhances the entire system's scalability and enables simultaneous management of reputation and crowdsourcing operations. The proposed framework incorporates an effective, yet privacy-friendly reputation model. The designed model evaluates the trustworthiness of participants based on their crowdsourcing interactions. To reduce the load on our blockchain, we introduce an off-chain storage solution, improving the overall performance of RollupTheCrowd. The proof-of-concept we have provided supports the feasibility of our framework and the obtained results affirm its scalability and efficiency.

To sum up, our contribution makes the following points:

- A Blockchain-powered decentralized platform to manage the entire reputation-based crowdsourcing process.
- RollupTheCrowd leverages zkRollups (L2) to boost scalability by alleviating the burden on the mainchain (L1).
- A privacy-preserving reputation model adaptable to diverse crowdsourcing scenarios.
- The proposed solution is supported by a concrete proof of concept implemented using emerging technologies. The experimental evaluations validate the efficiency and scalability of our framework.

Looking at the set of contributions we have made in the previous chapters, one question that may arise is that of the extensibility of such frameworks. Are our BRSs adaptable to other uses beyond those for which they have already been used?

In the next chapter, we will explore the extensibility of the concepts developed in the previous chapters to emerging applications. Specifically, we will propose a solution to the problem of evaluating the credibility of Large Language Models (LLMs) using a blockchain-based reputation system. The system aims to ensure an effective and transparent evaluation of LLMs' behavior when answering open-ended questions.

Chapter 6

Extensibility and Adaptability of Blockchain-based Reputation Management

Contents

In the previous chapters, we proposed a complimentary set of contributions aiming at enabling effective, privacy-preserving, and scalable blockchain-based reputation management. The designed solutions aim to foster trust and accountability through transparent yet privacy-preserving reputation management. Our main goal in this chapter is to explore the extensibility of the work proposed in these chapters and its adaptability to new applications. Therefore, in this chapter, we propose a novel blockchain-based reputation

framework tailored to innovative usage, namely the evaluation of large language models (LLMs).

6.1 Introduction

LLMs have received a great deal of attention in the last few years due to their surprising capabilities in managing a wide range of Natural Language Processing (NLP) tasks including information retrieval, language understanding, generation, and reasoning [105, 106]. Despite their impressive capabilities, LLMs such as GPT-3, Llama, and Vicuna [107–109] exhibit certain challenges that compromise their efficacy. One prominent issue is the manifestation of biases and fairness concerns. LLMs often inherit biases present in their training data, reflecting societal prejudices and stereotypes [110]. Consequently, these models can produce outputs that perpetuate or even exacerbate existing social inequalities. Another limitation arises from the models' difficulty in grasping common sense and contextual understanding. LLMs may struggle to interpret nuances in language, leading to responses that appear nonsensical or detached from real-world knowledge [111]. These behaviors encompass hallucinations, evident in the generation of text that invents or imagines information lacking a factual or coherent basis [28]. LLMs may also display unreliable reasoning [29], characterized by a lack of consistent or dependable logical abilities. There is also a risk of harmful content generation [30], where LLMs may produce offensive and inappropriate material. Overall, these behaviors can significantly deviate from the expected or desired output, undermining the credibility of LLMs and posing challenges to their widespread adoption. They also present hurdles to the utilization of LLMs in critical contexts such as medical diagnostics, legal advice, or sensitive information processing, where accuracy and reliability are essential.

One key way to assess the behavior of LLMs and measure their reliability involves soliciting input from users. Individuals can highlight issues they encounter while engaging with AI-generated content [112]. However, this method has two notable drawbacks. First, collecting user feedback is costly as it requires analyzing and categorizing the gathered information. Second, human feedback lacks real-time capabilities as users might not offer immediate responses. This delay hinders prompt evaluation given the absence of instant responses from humans. Therefore, to reduce reliance on human involvement, an alternative strategy consists of employing automatic evaluation methods. These techniques leverage automated feedback [30, 106] or language models [113, 114] to evaluate LLMs' performance cost-effectively. Despite the efficient processing of language data generated by LLMs, the automatic evaluation metrics they rely on may not perfectly align with human preferences or perceptions, thereby introducing certain limitations. These assessments may fail to capture nuances or qualitative aspects that are crucial for understanding how users perceive the content generated by LLMs [115]. The current human and automatic evaluation-based methods face many challenges linked to the lack of transparency and decentralization, as they operate within centralized frameworks. Entities wishing to use LLMs for specific tasks have to choose between trusting centralized third-party evaluations and independent testing, which is a costly process that

depends on the availability of code and data. Moreover, most of the recent studies concentrate on either human feedback or automated evaluation [30, 112, 116, 117], missing the opportunity to capture human preferences and enhance scalability while reducing costs.

Sharing and evaluating LLMs is an interesting innovative use to study within the context of blockchains. The necessity for employing a BRS for this specific issue arises from the lack of transparency and verifiability in existing frameworks (*e.g.*, TrustLLM [110]) and leaderboards (*e.g.*, Chatbot Arena[1]) designed for evaluating LLM capabilities. Therefore, the goal of this chapter is to demonstrate the methodology for conducting an effective and transparent assessment of LLMs utilizing a BRS. This BRS must effectively showcase the nuanced distinctions between various LLMs. This is not straightforward, given that the evaluation involves probabilistic models capable of generating differing responses for the same query. The framework must also possess scalability features to accommodate substantial demands, as the recent surge in published models necessitates high throughput for interacting with them. Hence, LLMChain, the framework we propose in this chapter will elucidate how LLMs can be shared and evaluated atop a blockchain network. By building such a BRS, we aim to instill trust via transparent and efficient evaluation processes. Blockchain - known for its resistance to tampering - can be used to track and manage the reputation of various LLMs via smart contracts. LLMChain's primary objective is to help users find the most reliable LLM that aligns with their specific needs and preferences. Consequently, it empowers individuals to access language models shared by LLM providers and actively participate in the evaluation process. Additionally, it provides LLM developers with valuable insights, enabling them to enhance and optimize their models by incorporating human feedback.

The remaining organization of this chapter is as follows: First, the related literature is presented in Section 6.2. Section 6.3 presents the proposed LLMChain framework. Section 6.4 details the proposed reputation model. Section 6.5 discusses the evaluation results. Finally, Section 6.6 concludes the chapter.

6.2 Related work

The goal of this chapter is to explore the extensibility of the work proposed in the previous chapters and its adaptability to emerging usages. One such usage is the evaluation of language models. Thus, in this section, we discuss their related work and some of the existing blockchain-based reputation frameworks and then present these as a potential solution to existing LLM evaluation approaches.

6.2.1 LLMs Evaluation

To assess the credibility and capabilities of LLMs, several studies have introduced diverse evaluation methods, including pairwise comparison, single-answer grading, or reference-guided grading, employing another LLM as an evaluator [106, 116]. These

[1] https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

methodologies offer advantages in scalability and interoperability. Nevertheless, it comes with notable limitations:

- *Position Bias*, where the evaluator tends to favor the initial model;
- *Verbosity Bias*, where the evaluator prefers longer responses over shorter ones; and
- *Self-Enhancement/Promotion Bias*, where the judging model prioritizes its own text or that generated from a similar model.

Moreover, evaluating a LLM using another LLM appears paradoxical since the evaluator itself is subject to evaluation. On the other hand, alignment-based methods are used to make large-scale alignment research more accessible like OpenAssistant Conversations [118], which is a corpus of conversations that resemble interactions with assistants, created and annotated by humans. Nonetheless, alignment-based methods face some scalability challenges and annotation expenses. In Core-GPT [119] and [120], authors focus on assessing the credibility of LLMs. Core-GPT proposes an approach that combines open-access scientific literature with LLMs to improve their reliability and trustworthiness. However, its methodology's scope is limited to two LLMs, "GPT3.5" and "GPT-4", failing to illuminate the credibility gap between open-source and commercial models. In contrast, the approach proposed in [120] introduces an automated workflow designed to manage an increased number of requests/responses, facilitating the assessment of the credibility of multiple LLMs. In G-Eval [117], which is a framework that leverages large language models, used a Chain-of-Thoughts (CoT) and a form-filling paradigm to evaluate the quality of Natural Language Generation (NLG) outputs. G-Eval experimentation involves two generation tasks: text summarization and dialogue generation. However, the methodology is limited to only two LLMs which are "GPT3.5" and "GPT-4".

When delineating the prevailing approaches employed to assess the credibility of LLMs, typical challenges become apparent. These approaches lack transparency and decentralization as they all operate within centralized frameworks. To determine the most credible LLM for a specific context, individuals are faced with two alternatives: either relying on centralized evaluations or carrying out tests independently. Additionally, the majority of current studies focus on either human feedback or automated evaluation separately, missing an opportunity to capture human preferences effectively while enhancing scalability and reducing costs.

6.2.2 Blockchain-based Reputation Systems

The inherent decentralized and tamper-proof nature of blockchain technology provides essential attributes for effective reputation management. Several blockchain-based reputation frameworks have been proposed in recent years, some of which are briefly presented hereafter. TrustChain [20], which aims to foster trust in IoT-supported Supply Chains. This solution constitutes a service platform operating on a permissioned blockchain network. GuRuChain [9] incorporates guarantee and reputation at application and consensus layers to foster accountability and trust. In ValidatorRep [58], the

authors propose a verification scheme that utilizes blockchain with trust management to foster accountability within crowdsourcing systems. In, Reputable [39], the authors propose a decentralized reputation system for assessing service providers' activity within a blockchain-based ecosystem. The proposed solution integrates a centralized oracle to perform off-chain computations and triggers on-chain smart contracts, impeding the system from achieving complete decentralization. In Trustd [60], the authors propose an ecosystem powered by blockchain and collective signatures, designed to support content creators in garnering community backing for their content. To provide transparency and foster trust, all these proposed solutions and frameworks use smart contracts and usually other Web3 tools like IPFS to implement their reputation models.

To address the lack of transparency and trust challenges in existing LLM evaluation approaches, this chapter proposes a novel BRS named "LLMChain". LLMChain is a decentralized framework for evaluating LLMs on open-ended question answering. The proposed approach leverages blockchain to build a robust and transparent reputation system. It merges human evaluation feedback with automated embedding-based metrics to assess LLMs responses effectively. To our understanding, this work marks the first exploration of the fusion of human and automated evaluations within a decentralized setting. It also marks the first BRS that enables the sharing and evaluation of language models.

6.3 LLMChain Framework

In this section, we introduce LLMChain, a Blockchain-powered reputation system for LLM's evaluation. In particular, the proposed framework aims to foster trust in LLMs by amalgamating human feedback and automated evaluations. LLMChain can be seen as a decentralized reputation-based store that allows sharing and evaluating LLMs. It serves a dual role by addressing the needs of users seeking reliable AI assistance, as well as assisting LLMs developers in enhancing the performance and reliability of their model. In what follows, we first describe the underlying architecture of LLMChain, then detail the evaluation process of LLMs within this framework.

6.3.1 LLMChain Architecture

The proposed LLMChain framework is composed of multiple entities distributed over four main layers as depicted in Figure 6.1.

6.3.1.1 User Layer

This layer is composed of individual participants. Each participant has at least one end device to interact with the system. Users with different areas of expertise can join the system to use shared, open-access LLMs and provide feedback after engaging with any of the models. This allows users not only to gain insights into the most suitable LLM for their specific domains but also to actively participate in the evaluation process by testing these models and sharing their feedback.

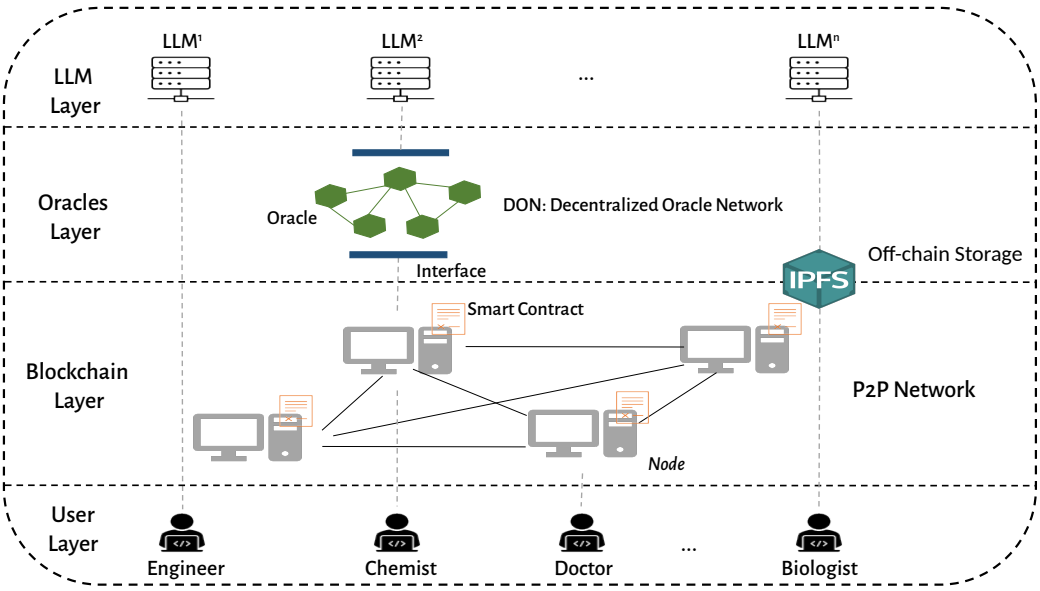


Figure 6.1: LLMChain Architecture

6.3.1.2 Blockchain Layer

This layer functions as a permissioned blockchain, comprising nodes initiated by LLM providers and/or developers. As a condition of joining the network, an entity must develop and share at least one LLM. LLMChain network employs a consensus mechanism to uphold a uniform copy of the ledger. We advocate for a reputation-based consensus, as the one we presented in Chapter 3, leveraging an existing reputation model within the system [9]. Compared to traditional consensus protocols, reputation-based consensus offers scalability and enhanced fairness. To further improve the scalability and accessibility of our decentralized application, we use an InterPlanetary File System (IPFS) as an off-chain storage system. The core business logic of LLMChain is securely executed via smart contracts deployed over the network and accessed through the submission of transactions. LLM providers benefit from joining the network by gaining full access to LLMChain and consequently, all the evaluations occurring within the system. This access allows them to accumulate extensive information, aiding in the enhancement and rectification of their models.

6.3.1.3 Oracle Layer

This layer consists of Oracle nodes that merge on-chain code with off-chain infrastructure, creating a sophisticated distributed application (DApp). Oracle nodes connect LLMChain smart contracts with real-world data, as these contracts are typically isolated from external information. Oracles serve as bridges that fetch and relay external data to smart contracts, enabling them to make informed and automated decisions (*i.e.*, reputation updates) based on real-world events. Within LLMChain, the Oracle network

intercepts responses from models, conducts off-chain automatic evaluations, and subsequently triggers on-chain smart contracts to update the overall score of the targeted model. All of that is achieved in a decentralized and trustless way by running an Oracle protocol [26].

6.3.1.4 LLM Layer

This layer consists of language models that are administered locally by LLM providers and/or developers. For users who wish to utilize these models for inference tasks, developers need to maintain ongoing access to their shared models. The Oracle network conducts regular checks on the connectivity of these shared models. Any model that goes offline automatically gets removed from the list of running models.

6.3.2 LLMs' Evaluation Process

Human evaluation of LLMs involves the participation of human experts or users to assess the quality, coherence, and overall appropriateness of the generated content. These metrics attempt to capture subjective aspects that automated metrics may miss [115]. However, evaluating generated responses through human feedback presents challenges because it relies on the willingness of users to provide genuine and immediate feedback. To better address these, we investigate automated methods that allow LLMChain to evolve even in the absence of human feedback.

Unlike centralized frameworks where the evaluation is implemented by a third party, we define end-to-end decentralized evaluation protocols. The proposed protocols are implemented in the LLMChain architecture using smart contracts. The evaluation process consists of three main phases.

6.3.2.1 Registration

To obtain their credentials, including public (address) key and private key, Users and Developers must register on LLMChain through the Identity Smart Contract (ISC). The registration process can be done in a decentralized, privacy-preserving, and Sybil-resistant way using an IDentity Management Ledger (IDML) [73].

6.3.2.2 LLM Sharing

LLM developers can add a new model to LLMChain via Reputation Smart Contract (RSC) by calling the *addModel* function. This creates a new model:

$$\text{LLM}= \{CID_{llm}, Owner, R_0^a, R_0^h, R_0\}$$

Owner is the developer's public key. The initial human score R_0^h, automatic score R_0^a, and weighted reputation R_0 for the model are calculated as the average values across all existing models in the system. CID_{llm}, the Content Identifier, is the hash of the model's details published on IPFS. To ensure the security of LLMChain's smart contract functionalities, we implement role-based access control to manage permissions. This is

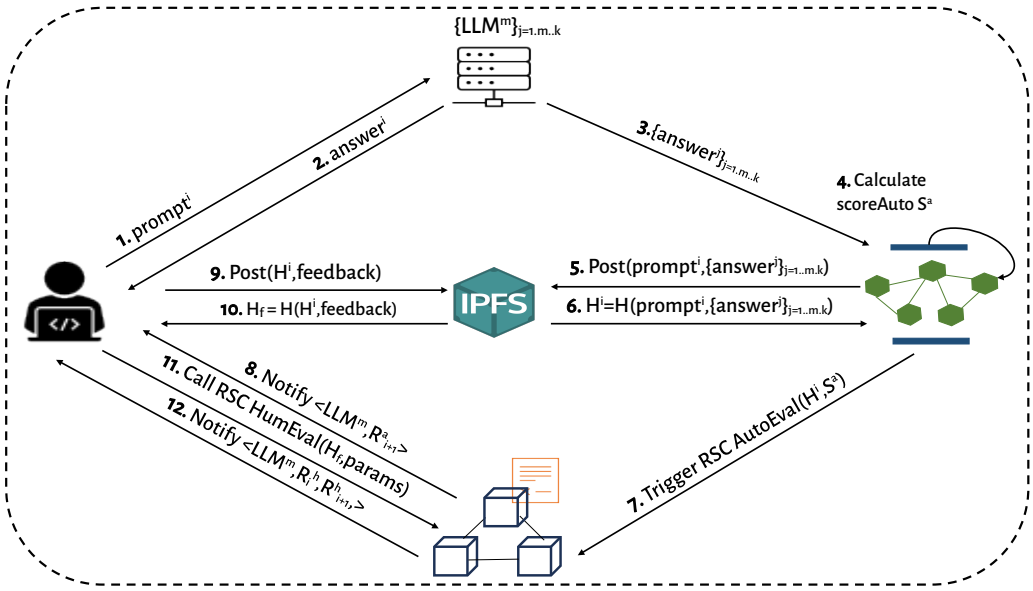


Figure 6.2: LLM Evaluation Workflow in LLMChain

realized through the Access Control Smart Contract (ACSC). ACSC restricts calling functions by role, for example, it restricts the ability to share models on LLMChain to developers only.

6.3.2.3 LLM Evaluation

The comprehensive process, spanning from prompt submission to updating the global reputation for the chosen model is illustrated in Figure6.2. It begins with the user formulating a request intended for a specific LLM_m, directly transmitted to the model via a dedicated interface (API). Subsequently, the response from LLM_m is relayed back to the user.

To perform **Automatic evaluation**, the Oracle intercepts both the request and the response. Then, it dispatches identical prompts to other k models $\{LLM_j\}_{j=1,\ldots,k}$, to use their answers as comparative references. Next, it calculates the automatic score for LLM_m (using the model that will be described in Section 6.4.2.1). It then stores the prompt and the corresponding answers off-chain using IPFS. In return, it gets H^i, which is the content identifier of the published data. Finally, it triggers the RSC to update the overall automatic score of LLM_m by calling the *autoEval* function. Upon receiving the answer, users can opt for direct **Human evaluation** by calling the *humEval* function or seek alternative candidate responses to gauge the quality of LLM_m's answer *i.e.*, using the shared hash H^i, they can retrieve all k answers from IPFS. Once this operation has been completed, the overall weighted reputation score is updated by calling the *updateReputation* function.

Further details on the automatic and human evaluation procedures are presented in the

next section.

6.4 Reputation Modeling

Human evaluation entails the participation of human experts or users to assess the quality, coherence, and overall adequacy of generated answers. These metrics seek to encompass subjective aspects that automated metrics may overlook [115]. Nevertheless, evaluating generated answers through human feedback poses challenges as it relies on users' willingness to offer genuine and immediate feedback. To better address these, we investigate automatic methods, enabling LLMChain to evolve even in the absence of human feedback. In this section, we introduce our reputation model that blends human and automated evaluations. This approach aims to leverage the efficiency and scalability of automated methods while upholding strong alignment through human feedback.

6.4.1 Reputation Formulation

We model the reputation of a LLM as a tuple denoted by $REP = \{R^a, R^h, R\}$. Our approach involves assigning an initial reputation, noted $REP_0 = \{R_0^a, R_0^h, R_0\}$, to each new LLM. The values of R_0^a, R_0^h, and R_0 are derived from the average scores of all LLMs in the system.

The REP tuple undergoes updates after each interaction i, following two stages:

1. *Interaction Evaluation*, which involves computing three scores for the targeted LLM: an automatic score S^a, a human score S^h, and a weighted combination S^θ between both scores; the three scores are respectively weighted with ω^a, ω^h, and ω^θ.

2. *Global Scores Updating*, where each global score R_i in REP is updated using a specific function securely implemented in the RSC contract. For each $(R_i, S_{calc}, \omega) \in \{(R^a, S^a, \omega^a), (R^h, S^h, \omega^h), (R, S^\theta, \omega^\theta)\}$,

$$\begin{array}{rccc} \mathcal{U}: & [0,1] \times [0,1] \times [0,1] & \longrightarrow & [0,1] \\ & (R_i,\ S_{calc},\ \omega) & \longrightarrow & R_{i+1} \end{array} \tag{6.1}$$

6.4.2 Interaction Evaluation

In the following, we present the mathematical details of the two approaches we use to evaluate the behavior of LLMs.

6.4.2.1 Automatic Evaluation

Several studies have demonstrated that embedding-based metrics can effectively match human judgments by considering semantic relevance [121, 122]. However, their effectiveness is influenced by the quality of the underlying embedding. Consequently, when developing LLMChain, we emphasized a modular framework to retain flexibility in updating the automatic evaluation technique at any time. The metrics we explore to use for

our Automatic evaluation require a minimum of one reference to compute the score S^a (cf. Section 6.5.1.3). Hence, we propose to use k references, denoted as $\{ref^j\}_{j=1...k}$ to evaluate the answer of the targeted model for better precision. The k references are the answers that the decentralized Oracle gets from the top k models within the context of the prompt. The final score of the answer from the model LLM is computed as follows:

$$S^a = \frac{1}{k} \sum_{j=1}^{k} scoreAuto(answer,\ ref^j) \tag{6.2}$$

We assess the quality of the automatic evaluation using a weighting function $\omega^a \in [0, 1]$. Its outcome varies depending on the average reputation of the models used as references (*i.e.*, the better the reputation the higher importance is given). Once this is done, the Oracle triggers the *autoEval* function in RSC to update the overall automatic score of the LLM_m using the model described in Section 6.4.3.

6.4.2.2 Human Evaluation

While it is straightforward to carry out an automated evaluation by measuring the distance/similarity between generated answers, it is less easy to gather information about trust, user satisfaction, completeness, and usefulness of a generated text. Inspired by [115] and [123], our approach involves employing a multi-item scale questionnaire for efficient and scalable human evaluation. Our focus encompasses two types of dimensions (constructs) essential for users to assess text generated by LLM accurately:

Answer's Constructs: are the metrics that allow the evaluation of the quality of a single answer/response (*i.e.*, calculate S^h). To do so, we employ three metrics.

- *Reliability*, denoted as A_r, evaluates the trustworthiness of the provided answer.
- *Completeness*, denoted as A_c, measures the comprehensiveness or completeness of the answer.
- *Utility*, denoted as A_u, determines the usefulness of the answer.

The human score of an answer is a linear combination of the three metrics:

$$S^h = [\alpha_a A_r + \beta_a A_c + \gamma_a A_u]\ ;\ \alpha_a + \beta_a + \gamma_a = 1 \tag{6.3}$$

User Constructs: encompass parameters that signify a user's proficiency and ability in evaluating the generated text, showcasing the quality of their assessment and its influence on the overall human score (*i.e.*, calculate ω^h). To do so, we define four metrics.

- *Duration*, denoted as D, measures the time interval in minutes between the last two evaluations.

- *Familiarity*, denoted as F, gauges the user's familiarity with the response context.
- *LLM Trust*, denoted as T, assesses the user's belief in the expertise of the targeted LLM.
- *Uncertainty*, denoted as U, captures the user's degree of uncertainty regarding the evaluation.

The weight of the human evaluation is given by:

$$\omega^h = \mathcal{W}^h \mathcal{F}_D \tag{6.4}$$

where,

$$\mathcal{W}^h = [\alpha_u F + \beta_u T + \gamma_u (1 - U)] \; ; \; \alpha_u + \beta_u + \gamma_u = 1$$

and $\mathcal{F}_D$ is a hyperbolic tangent function that normalizes D ($\mathcal{F}_D \in [0, 1]$):

$$\mathcal{F}_D = tanh_\lambda(D) = \frac{1 - e^{-\lambda . D}}{1 + e^{-\lambda . D}}$$

$\mathcal{F}_D$ is implemented in a way that thwarts potential abuse. It reduces the impact of successive evaluations performed within a short period, thereby protecting the LLM's overall reputation and reinforcing the model's effectiveness. Furthermore, the positive correlation with the other metrics (*i.e.*, F, T, and $1 - U$) leads to two important considerations: (i) ratings from users less familiar with the context carry less weight in updating the model's overall human reputation; (ii) ratings from users with minimal trust or with higher uncertainty have less impact on updates compared to those with lower uncertainty and higher trust in the overall expertise of LLMs.

6.4.3 Reputation Update

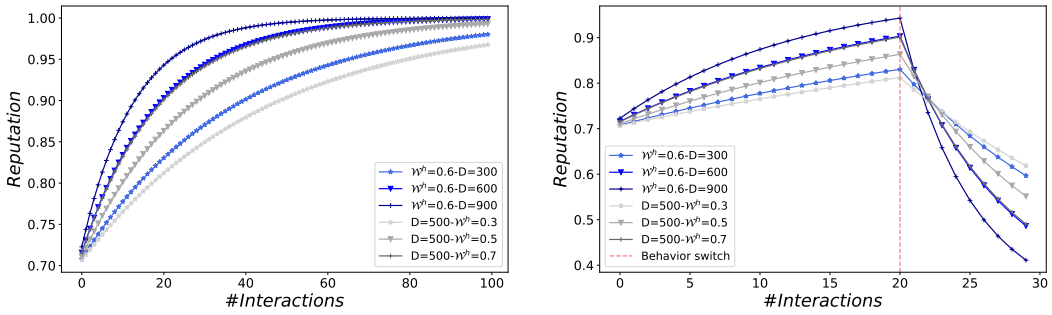
In LLMChain, we employ three types of updates. The overall automatic reputation R^a update occurs after each interaction to keep tracking the LLM behavior, while changes in R^h and R only occur if the interaction includes a human evaluation. These updates depend on the outcome of the automatic evaluation S^a, the human evaluation S^h, or the weighted evaluation S^θ. We use θ, a configurable weighting factor, to give more emphasis to the human evaluation when calculating S^θ and ω^θ, as follows:

$$\begin{cases} S^\theta = \theta S^h + (1 - \theta) S^a \\ \omega^\theta = \theta \omega^h + (1 - \theta) \omega^a \end{cases} \tag{6.5}$$

The updating formula $\mathcal{U}_{\psi,\xi} : (R_i, \; S_{calc}, \; \omega) \longrightarrow R_{i+1}$ for the three scores R^h, R^a, and R is thus defined as follows:

$\forall (R_i, S_{calc}, \omega) \in \{(R^a, S^a, \omega^a), (R^h, S^h, \omega^h), (R, S^\theta, \omega^\theta)\}$,

$$R_{i+1} = \begin{cases} (1 - \psi\omega) R_i + \psi\omega S_{calc} \; ; & S_{calc} \geq T_{min} \\ (1 - \xi\omega) R_i + \xi\omega S_{calc} \; ; & S_{calc} < T_{min} \end{cases} \tag{6.6}$$



(a) Reputation growth following successive accurate answers

(b) Reputation changes after successive incorrect answers

Figure 6.3: The Effectiveness of LLMChain's Reputation model under different $\mathcal{W}^h$ and D.

where R_i and T_{min} are the current reputations and trust thresholds (*i.e.*, before the interaction i), respectively. Here, the threshold T_{min} is defined as the average of LLM reputations $\overline{R_i}$.

By employing two distinct formulas for R_{i+1} (Equation 6.6) for the update process using a trust threshold T_{min}, we separate expected good behavior from unexpected bad behavior (no/bad response, hallucination, harmful content, etc. [4, 7]). Consequently, we can put more weight (*i.e.*, $\xi > \psi$) on the newly calculated score S_{calc} in the case of an incorrect response. Moreover, the integration of the weighting function ω into both equations establishes a direct relationship between the quality of the evaluation and its impact on the update of the overall reputation. For instance, for a R^h update, the greater the user's familiarity, certainty, and trust in the LLM expertise, the more significant their evaluation's impact becomes. Moreover, the use of D allows the system to mitigate consecutive inaccurate ratings that may be intended to enhance or damage LLM's reputation. We note that this metric is reset at regular intervals (*e.g.*, every 24 hours), preventing users who abstain from evaluations for a long time from exploiting the model.

Figure 6.3 demonstrates the impacts of the metrics D and $\mathcal{W}^h$ on the overall reputation updates. It demonstrates the shifts in reputation between a skilled model consistently providing accurate responses and a less competent one that produces consecutive incorrect answers after delivering multiple correct ones. Both positive and negative updates have a direct correlation with D and $\mathcal{W}^h$. This suggests that the longer the time interval between the last two evaluations, the more significant impact the user's latest evaluation has. Likewise, increased levels of familiarity, trust, and certainty contribute to a more substantial impact.

6.5 Evaluation and Results

Having introduced the architecture of LLMChain and defined the evaluation process in our framework, this section discusses experiments that demonstrate its feasibility, effectiveness, and scalability. We first present the experimental setup. Next, we discuss the evaluation results of the proposed framework in terms of reputation model effectiveness and overall system performance and scalability.

6.5.1 Experimental Setup

6.5.1.1 Environment

We conducted the experimental tests on two separate clusters:

- A GPU cluster for hosting the LLM part of the system; it comprises two servers, one featuring an NVIDIA RTX A6000 GPU card and the other equipped with an NVIDIA GeForce RTX 2080 Ti card.
- A CPU cluster dedicated to running the blockchain network; it consists of two HPE ProLiant XL225n Gen10 Plus servers specifically allocated for experimenting with blockchain solutions, both are powered by two AMD EPYC 7713 64-Core processors and 2x256 GB RAM.

6.5.1.2 Datasets

We consider three datasets to evaluate LLMChain:

- *MTBench*[2] is a recent dataset extensively utilized in evaluating LLMs [106]. MT-Bench consists of 3.3K expert-level pairwise human preferences for answers generated by six models ("Llama-13B", "Alpaca-13B", "Vicuna-13B", "GPT-3.5", "Claud-v1", and "GPT-4") across 80 questions.
- *GooAQ*[3] is a large-scale dataset with a variety of answer types. This dataset comprises more than 5M questions and 3M answers sourced from Google [124].
- *LLMGooAQ.*[4] We prepare this comprehensive database, covering 100k questions and answers in 20 different fields/contexts. We randomly sample 100K tuples from the GooAQ dataset and perform inference using seven LLMs ("Alpaca-13b", "Llama-2-13b", "Chatglm-6b", "Fastchat-t5-3b", "Koala-13b", "Vicuna-7b", "Vicuna-13b").

6.5.1.3 Automatic Metrics

To pinpoint the optimal technique for our context, we conduct rigorous benchmarks among various embedding-based metrics that achieved SoTA performance.

[2] https://huggingface.co/spaces/lmsys/mt-bench
[3] https://huggingface.co/datasets/gooaq
[4] https://github.com/mohaminemed/LLMGooAQ/

- *BERTScore* [113] is an automatic evaluation metric for text generation. It evaluates the similarity between tokens in a candidate sentence and those in a reference sentence. Unlike N-Gram methods relying on exact matches like BLEU Score [125] and ROUGE Score [126], BERTscore relies on contextual embeddings to gauge token similarity. The approach employs cosine similarity to measure the likeness between a reference token x_i and a candidate token $\hat{x}_i$. The total score involves comparing each token in x with tokens in $\hat{x}$ to calculate recall, and each token in $\hat{x}$ with tokens in x to determine precision. To maximize the similarity score, a greedy matching technique is employed, wherein each token is paired with the most similar token from the other sentence. Precision and recall are combined to derive an F1 score.

- *BARTScore* [121] is an automated evaluation method that frames the evaluation of generated text as a text generation problem, utilizing pre-trained sequence-to-sequence models. The fundamental concept revolves around the notion that models trained to convert generated text into or from a reference output or the source text will yield higher scores for superior generated text. This concept is implemented using BART, a pre-trained model based on an encoder-decoder architecture. The metric BARTScore offers various adaptable variants that can be applied in an unsupervised manner to evaluate text from multiple perspectives, such as informativeness, fluency, or factuality.

- *DISCOScore* [122] is a parametrized discourse metric, which uses BERT to model discourse coherence from different perspectives, through the lens of readers' focus, driven by Centering theory. DISCOScore offers two variations: FocusDiff and SentGraph, differing in their treatment of focus. This approach models the frequency and semantic relevance of focus and then compares the disparities between the hypothesis and the reference. It utilizes two adjacency matrices to represent coherence based on focus. In FocusDiff (DSFocus), the matrix represents relationships between foci and tokens, indicating focus frequency. Meanwhile, in SentGraph (DSSent), the matrix showcases the interdependence between sentences based on shared foci and sentence proximity.

Table 6.1: Hyperparameter's Configuration.

Parameter	Value
ψ	1/3
ξ	2/3
λ	10^{-3}
$\alpha_a = \beta_a = \gamma_a$	1/3
$\alpha_u = \beta_u = \gamma_u$	1/3
θ	2/3

Table 6.2: Metrics performance on the MTBench dataset.

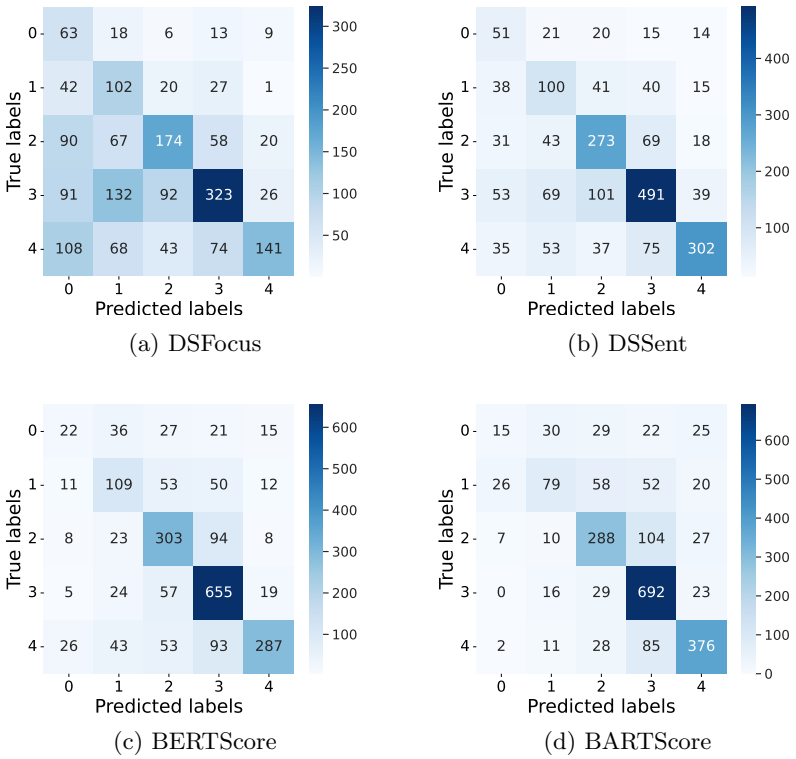
Metric	Accuracy	Kendall's Correlation
DSFocus	0.44414	-0.60
DSSent	0.59540	0.60
BertScore	0.66991	0.60
BartScore	**0.70594**	**0.80**

6.5.2 Reputation Model Effectiveness

In the following, we first conduct an experimental comparison of the automatic metrics described in Section 6.5.1.3. Next, we perform two additional experiments aiming to evaluate the efficiency of both the automatic and human models. The values of the configurable parameters used in these experiments are summarized in Table. 6.1.

6.5.2.1 Automatic Benchmark

Determining the most fitting metric for evaluating LLM-generated answers analytically is not straightforward. That is why we embarked on a benchmark experiment to pinpoint the best technique. This experiment assesses the metrics commonly used in automatically evaluating NLP tasks. Our goal is to identify the one that best aligns with human judgments. To achieve this, we conduct an experiment that involves computing automatic scores on MTBench answers using four different metrics. These scores automatically determine the winner between two different LLMs for each question. Figure 6.4 demonstrates the correlation between human-selected winners (true) and automatic winners (predicted).



(a) DSFocus (b) DSSent

(c) BERTScore (d) BARTScore

Figure 6.4: Human Winners vs Automatic Winners on the MTBench dataset. Labels are denoted as: {“0:Llama-13B”, “1:Alpaca-13B”, “2:Vicuna-13B”, “3:GPT-3.5”, “4:Claud-v1”}.

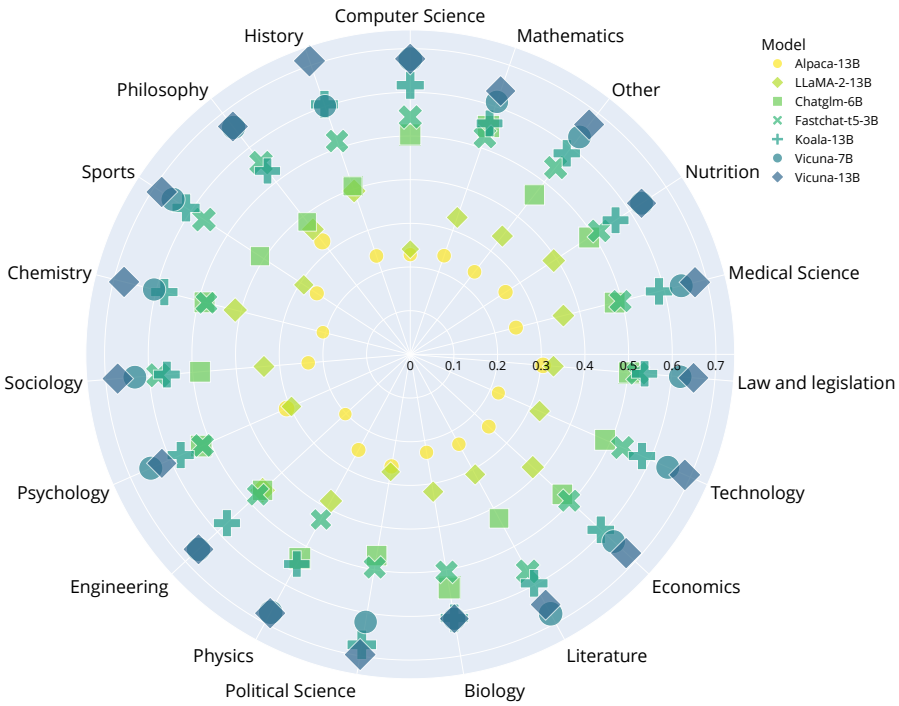


Figure 6.5: BARTScore-based Contextual Win-Rates on LLMGooAQ.

The matrices show nearly diagonal patterns, indicating good correlations, yet variations in accuracy exist among the metrics. For instance, the DISCOScore DSSent variant boasts an accuracy of 59%, surpassing that of the DSFocus variant (44%). BARTScore, on the other hand, demonstrates superior accuracy, with 71% of predicted winners matching actual human winners, compared with 67% for BERTScore. Table 6.2 illustrates Kendall's Tau correlation of these four metrics. We can see that BARTScore can significantly outperform all other techniques by offering a superior correlation of 80% with human judgments. Thus, we will use BARTScore in the following experiments.

6.5.2.2 BART Evaluation

To effectively evaluate the automatic model, we utilize BARTScore to conduct a pairwise comparison between the seven models using GooAQ's answers as benchmarks. Subsequently, we calculate the win rates for each model per context. The experimental results, showcased in Figure 6.5, highlight "Vicuna-13b" as the best model outperforming others in nearly 90% of the contexts. Moreover, the obtained win rates align with earlier research [106], affirming that the BARTScore metric exhibits a stronger correlation with human judgments.

Now, to assess the efficacy of leveraging the best model's answers within specific con-

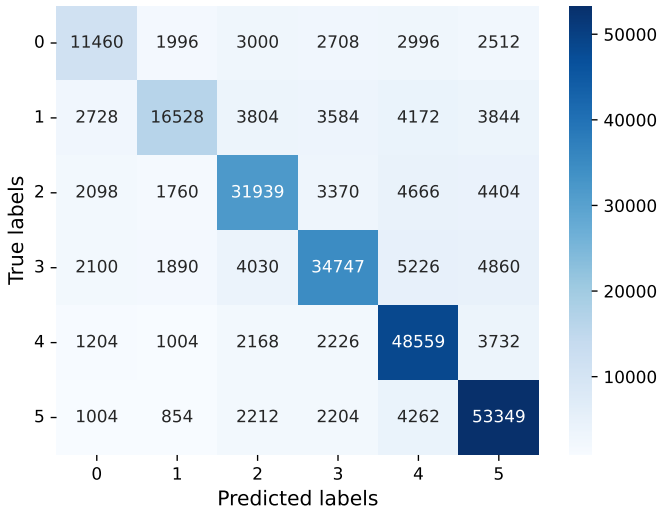


Figure 6.6: Ground-Truth Answers vs Vicuna-13B Answers as References for BARTScore-based Pairwise-comparison on the LLMGooAQ dataset. Labels are denoted as: {0: "Alpaca-13b", 1: "Llama-2-13b", 2: "Chatglm-6b", 3: "Fastchat-t5-3b", 4: "Koala-13b", 5: "Vicuna-7b"}.

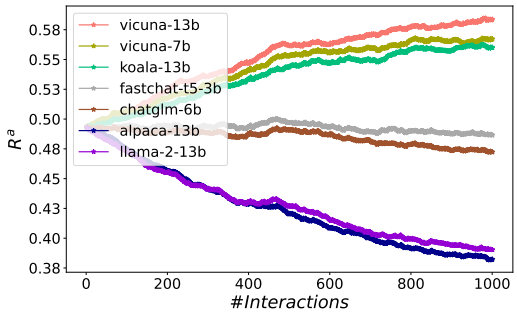
texts, we conduct a subsequent test using the answers from "Vicuna-13b" as references. Figure 6.6 presents the confusion matrix comparing the winners (true) computed using GooAQ answers with those (predicted) computed using "Vicuna-13b" answers. The results are compelling, revealing robust accuracy (70%) between the two cases.

It is important to note that, according to current benchmarks [106, 110] (Chatbot Arena[5] and TrustLLM[6] leaderboards), "Vicuna-13b" is a well-ranked open source model, but it is not the best. Despite this, the results obtained using its answers as references are convincing.

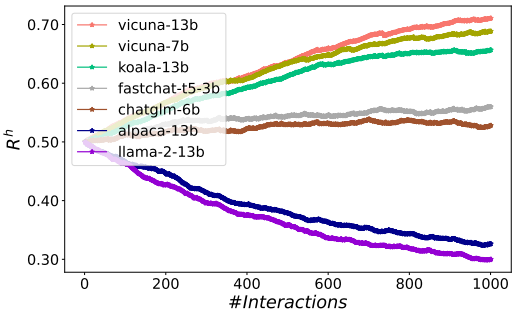
6.5.2.3 Reputation Evaluation

The third experiment involves employing the proposed models and monitoring changes in reputations in a real scenario. To do this, we use our prepared dataset with automatic scores computed using BARTScore. Given the high cost of obtaining human judgments, we employ GPT-4 as an expert for human evaluation. GPT-4 is recognized as the leading model in current benchmarks [106, 110, 119]. In this experiment, GPT-4 serves as a human expert, responding to a questionnaire that enables the calculation of metrics (*i.e.*, F, T, U, A_r, A_c, and A_u) used in the human model (presented in Section 6.4.2.2). Figure 6.7 illustrates the variations in R^a, R^h, and R for the seven LLMs in our dataset. Despite the disparities between the R^a and R^h scores, a consistent pattern emerges, with

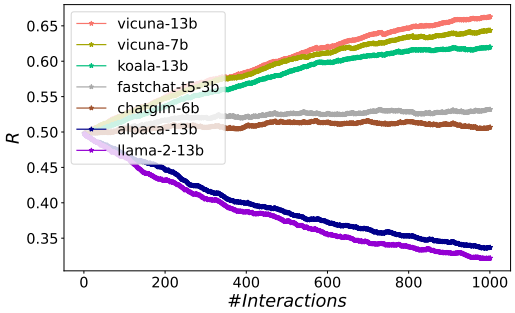
[5] https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard
[6] https://trustllmbenchmark.github.io/TrustLLM-Website/leaderboard.html



(a) BARTScore-based Automatic scores



(b) GPT-4-generated Human scores



(c) Weighted Reputation scores

Figure 6.7: Changes in R^a, R^h, and R of seven LLMs using LLMGooAQ.

scores for good models such as "Koala-13b", "Vicuna-7b", and "Vicuna-13b" steadily increasing, while scores for less effective models such as "Alpaca-13b" and "Llama-2-13b" continually decrease. Moreover, with an increasing number of evaluations, the distinctions between closely ranked models become more pronounced. This demonstrates the effectiveness of our models, showcasing their ability to discern even subtle differences between close LLMs like "Chatglm-6b" and "Fastchat-t5-3b".

6.5.3 Blockchain Performance

We follow the same methodology as in the previous chapters. The main functions in LLMChain are invoked by sending several workloads. We submit a predefined number of transactions (TXs) using a workload controller. In the following, we first present the business model and then discuss the evaluation results.

6.5.3.1 Business Model

We implement the proposed platform framework on a blockchain network powered by Hyperledger Besu[7], an open-source Ethereum client. Our evaluation approach includes:

[7] https://besu.hyperledger.org

- *Participants:* Users with different expertise and Admins of the organization or the consortium operating the system.
- *Assets:* we define a data structure that represents the model on-chain.
- *Smart Contracts:* Three types of smart contracts are used to develop the business model: Identity Smart Contract (ISC), Access Control Smart Contract (ACSC), and Reputation Smart Contract (RSC). ISC implements the registration process, ACSC employs a role-based access control to manage the permissions when calling RSC functions, *e.g.*, only Oracles can trigger the *autoEval* function. RSC implements the four main functions, *addModel*, *autoEval*, *humEval*, and *updateReputation*.

We develop the smart contracts of LLMChain using the Solidity programming language[8] and established a local network consisting of sixteen validators using Hyperledger Besu with Proof of Authority (PoA) as consensus protocol. We utilize Web3js library[9] for developing the client side and deploying the system's smart contracts.

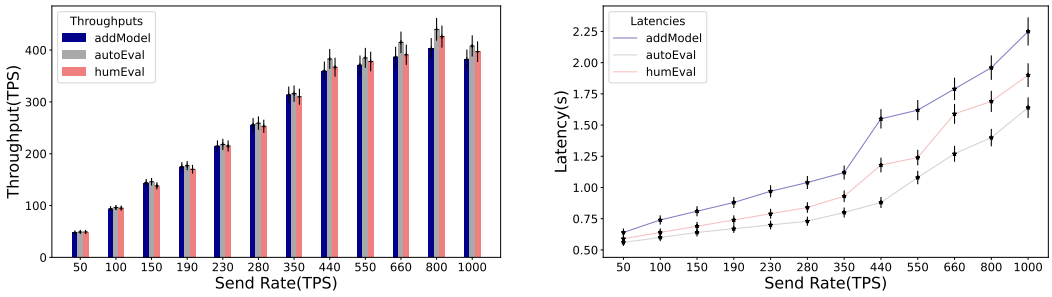
6.5.3.2 Performance Evaluation

To conduct tests, we utilized Hyperledger Caliper[10], a benchmarking tool for BC-powered systems. The experiments involve changing the TX sending rate (ranging from 50 to 1000 TPS) using a consistent network configuration for the main operations performed within our system. As a result, two metrics are measured:

- *Throughput:* is the number of successful transactions per second (TPS).
- *Latency:* refers to the time difference in seconds between the submission and completion of a transaction.

The Figure 6.8 illustrates the throughput and latency values for each function under different sending rates. At the beginning, the pattern is evident: throughput and latency increase as the TX send rate increases. With lower sending rates (<350 TPS), there is no significant difference in throughput between the three transactions. However, nearing system capacity, distinctions emerge. The lightest function, *autoEval*, achieves a peak throughput of 440 TPS, surpassing *humEval* at 426 TPS, and the heaviest function, *addModel*, managing 403 TPS, primarily due to the initialization and storage of model information on-chain. This also explains the comparatively higher latency of *addModel* compared with the other functions. Nevertheless, leveraging storage scaling via IPFS, LLMChain achieves an average throughput close to 420 TPS, comfortably meeting the specific demands of our use case.

[8] https://docs.soliditylang.org
[9] https://web3js.readthedocs.io
[10] https://github.com/hyperledger/caliper-benchmarks



(a) Throughput under different send rates

(b) Latency under different send rates

Figure 6.8: Throughput and Latency of LLMChain.

6.6 Conclusion

In this chapter, we explore the adaptability of the concepts proposed in previous chapters to emerging applications such as LLM evaluation. The lack of transparency and verifiability in current frameworks and leaderboards designed to show LLM capabilities underscores the need for new paradigms. To ensure the trustworthiness of their results, these paradigms must provide a fair degree of decentralization, transparency, and verifiability. To this end, we present LLMChain, a novel blockchain-based framework specifically designed to share and assess LLMs in a decentralized and transparent manner. LLMChain addresses trust concerns associated with flawed behavior, such as hallucinations and unreliable reasoning of LLMs, by employing a context-driven reputation system. To do so, we first study existing techniques for evaluating language models, which include two categories: human-based evaluation and automatic evaluation. Then, we propose a reputation model powered by blockchain to provide a transparent and verifiable evaluation of LLMs' capabilities in answering open-ended questions.

Our efforts involve designing and implementing a reputation model that evaluates user satisfaction and trust in every interaction involving an LLM. This model amalgamates human feedback with automatic evaluation to assign contextual reputation scores that accurately mirror LLM behavior. Consequently, the system aids users and entities in pinpointing the most credible LLM for their requirements while offering LLM providers valuable insights to refine and enhance their models. This research marks the first initiative to introduce a distributed framework dedicated to LLMs evaluation. Through extensive experiments and benchmarks, we demonstrate the effectiveness of both human and automatic evaluations in LLMChain. Moreover, the benchmarks conducted on our blockchain affirm LLMChain's efficiency and scalability, validating its practical applicability in real-world scenarios.

In summary, the contributions of this chapter are:

- A new reputation model is proposed to evaluate user satisfaction and to deter-

mine the level of trust associated with every interaction involving a language model, through a comprehensive yet scalable evaluation of LLM's answers with both human feedback and automatic evaluation.

- A Blockchain-powered fully decentralized framework is introduced to share and evaluate LLMs through the designed reputation-based model.
- The preparation of a comprehensive dataset encompassing diverse questions and answers across various domains and contexts. This dataset consists of over 100k questions pulled from the large-scale GooAQ dataset and their corresponding answers obtained by performing inference on seven open-source LLMs.
- Extensive experimental evaluation scenarios are presented to demonstrate both the effectiveness of the proposed reputation model and the scalability of LLM-Chain.

This chapter marks the end of our efforts in this thesis to improve the effectiveness, privacy preservation, and scalability of blockchain-based reputation systems. While we recognize that our research represents only the first step toward broader adoption, we assert that our contributions significantly advance the field. In the next and final chapter, we summarize our accomplishments and outline our vision for future research efforts.

Chapter 7

General Conclusion

Contents

This thesis offers several significant contributions aimed at enhancing the effectiveness, privacy preservation, and scalability of blockchain-based reputation systems. As decentralized applications (DApps) increasingly permeate societal and real-world contexts, concerns regarding the security, privacy preservation, and resilience of reputation-driven DApps have become paramount. In this thesis, we introduce multiple frameworks that demonstrate promising advancements in performance and privacy preservation. First, to cope with the growing workload on the main blockchain, we propose and implement innovative mechanisms utilizing emerging techniques to demonstrate their feasibility and good performance. second, our investigation into various architectures that decouple identity management from business operations reveals straightforward measures applicable across diverse scenarios. The deployment of these frameworks not only enhances privacy preservation and the overall performance of blockchain-based reputation systems but also promotes the broader integration of reputation-driven DApps into everyday life, thereby propelling the realization of a trustworthy Web3. Although acknowledging that the research presented herein constitutes a preliminary step toward the widespread adoption of blockchain-based reputation systems, we contend that our contributions significantly advance the current state of the art.

In the remainder of this chapter, we summarize the accomplishments of our goals and objectives presented in Chapter 1. We then discuss our future research directions.

7.1 Objective Outcomes

The primary objective of this thesis was to improve the effectiveness, privacy preservation, and scalability of blockchain-based reputation systems (BRS) to foster their broader adoption in real-world applications. Our research's first objective was to enhance BRS's

effectiveness and Layer-1 scalability. To this end, we introduced GuRuMarket [31] “Initially GuRuChain" [9].

GuRuMarket: is a blockchain-based reputation system for real-world marketplaces that can be adapted to other scenarios such as crowdsourcing and supply chains. In this first contribution, we first explored the intersection between blockchain and decentralized reputation management and showed why traditional blockchains are unable to support effective on-chain reputation management due to their additional workload (computational overhead). Then, to enhance GuRuMarket effectiveness, we developed a secure trading logic supported by a robust guarantee and reputation-based incentive mechanism. Next, to reduce blockchain congestion we leveraged the developed reputation model, proposing a novel consensus called *Proof-of-Guarantee&Reputation (PoGR)*, and demonstrating the potential of using reputation scores and their impact on the overall scalability and performance. Unlike PoS-based blockchains, PoGR ensures a high degree of decentralization and fairness by preventing wealth centralization using reputation and guarantee. Finally, GuRuMarket enables effective and transparent on-chain reputation management and fosters accountability and trust among traders when exchanging real-world services via a robust incentive mechanism. Our experiments on a running local blockchain network demonstrate the feasibility, effectiveness, and scalability of GuRuMarket.

The second part of our research objectives was enhancing privacy preservation in BRSs. Thus, we proposed the DARS framework [32].

DARS: To mitigate privacy concerns and fear of potential retaliation in BRSs while maintaining their effectiveness, we presented a decentralized blockchain-based anonymous reputation system. In our privacy-preserving BRS, users can use different pseudonyms when interacting with each other allowing them to hide their digital identities. In DARS design, all pseudonyms of a specific user, yet, are cryptographically linked to the same access token, allowing honest users to maintain their reputation and preventing malicious ones from starting over [32]. This is achieved through the use of zkSNARK proofs for set membership via Merkle trees over commitments. We extended our framework with an efficient reputation model that respects all the security and privacy properties of our formal model. We built the proposed framework using emerging technologies (*e.g.*, Hyperledger Besu) and cryptographic tools (*e.g.*, Circom and Snarkjs libraries). The results of the evaluation of the proposed DARS in terms of time overhead (zkSNARK proofs generation and verification), throughput, and latency demonstrate its feasibility and effectiveness.

The third part of our research goal was to improve the performance of blockchain-based reputation systems via Layer-2 scaling techniques. Thus, we proposed RollupTheCrowd [33].

RollupTheCrowd: To improve blockchain-based reputation system scalability while protecting user privacy, we proposed RollupTheCrowd, a novel blockchain-powered crowdsourcing framework that leverages zkRollups to reduce gas costs and improve perfor-

mance [33]. This allows for concurrent crowdsourcing tasks and reputation update management. RollupTheCrowd as a BRS introduces an effective and privacy-preserving reputation model that gauges workers' trustworthiness by assessing their crowdsourcing interactions. To alleviate the load on the main chain, we employed an off-chain storage scheme, optimizing RollupTheCrowd's performance. To prove the feasibility of the proposed framework, we developed a proof-of-concept implementation using cutting-edge tools. Utilizing smart contracts and zero-knowledge proofs, our Rollup layer achieved a significant 20x reduction in gas consumption without compromising the systems' privacy and security. Overall, our results demonstrate the effectiveness and scalability of RollupTheCrowd, validating its potential for real-world application scenarios.

The fourth and final part of our research goal was to show the extensibility and adaptability of the frameworks developed in the three previous parts and their applicability to new emerging usages. Thus, we proposed LLMChain [34].

LLMChain: To effectively and transparently evaluate the credibility of LLMs, we proposed LLMChain, a decentralized blockchain-based reputation system that combines automatic evaluation with human feedback to assign contextual reputation scores that accurately reflect LLM's behavior [34]. LLMChain not only helps users and entities to identify the most trustworthy LLM for their specific needs but also provides LLM's developers with valuable information to refine and improve their models. To our knowledge, this represents the first work to introduce a blockchain-based distributed framework specifically designed for assessing LLMs. The evaluation results obtained from several experiments demonstrate the effectiveness of the proposed reputation model and the scalability of LLMChain. In particular, the results show the ability of our model to detect even subtle differences between close LLMs such as "Chatglm-6b" and "Fastchat-t5-3b". Moreover, these also demonstrate that LLMChain can achieve an average throughput of nearly 420 TPS, meeting the specific demands of our use case.

7.2 Future Work

In this thesis, we have presented various contributions aimed at improving the effectiveness, privacy, and scalability of blockchain-based reputation systems to support reputation and business workloads. While the full replacement of the current version of the Web2-based reputation system with a Web3 one is a work in progress, our contributions have provided a foundation upon which to build. The work achieved in this thesis allows us to identify several new avenues for future research directions. A high-level overview of these avenues is described below.

7.2.1 Unpredictability

The fairness, cost-effectiveness, and decentralization of PoGR are noteworthy, but further measures such as the unpredictability of the block producer selection can be added to the scheme to enhance its safety. The unpredictability or secret leader selection

concept ensures that an adversary lacking control over the leader, cannot discern which node will be elected [127]. This aims to protect leaders from Denial of Service (DoS) or bribery attacks, enhancing the blockchain protocol's security.

Algorand employs a decentralized Byzantine Agreement protocol, integrating Pure Proof of Stake (Pure PoS) and Verifiable Random Function (VRF) mechanisms to randomly designate a leader for proposing a block within a given round. Upon a block proposal, a committee of voters is selected to validate the proposal. If a supermajority of votes originates from honest participants, the block becomes certified [16]. Committees consist of pseudorandomly chosen accounts, with their voting power determined by their stake online. Essentially, every token undergoes a VRF execution, with accounts possessing higher stakes being more likely to obtain committee membership and thus have more voting influence. Although using randomly selected committees maintains protocol performance and promotes network inclusiveness, fairness remains a concern. Hence, future work will explore VRF enhancements aimed at bolstering the security of PoGR by achieving unpredictability without compromising fairness.

7.2.2 Leaderless Consensus

A recent study on leaderless consensus has demonstrated better scalability [43]. The work proposed the Set Byzantine Consensus, enabling nodes to reach an agreement on multiple blocks at the next available index. This approach harnesses the resources of all nodes, eliminating the bottleneck associated with relying on a single leader. The outcome resulted in scalability within a Wide Area Network (WAN) environment, as performance scales proportionally with the network size. Given its emphasis on network collaboration over competition, we consider leaderless consensus as a suitable alternative for PoGR to scale even more GuRuMarket and enhance any BRS performance. Therefore, in future research, we will examine this approach as a Layer-1 solution to improve GuRuMarket's performance scalability.

7.2.3 Adaptability and Extensibility

We are confident that the design of our system can be flexibly adapted to a multitude of use cases, thereby improving their operational security and effectiveness. In prospective research endeavors, there is an opportunity to tailor the proposed frameworks to alternative applications such as decentralized federated learning and IoT-enabled systems, such as those found in supply chains and energy trading. For example, establishing resilient and transparent on-chain reputation management systems could facilitate the selection of training nodes in federated learning environments operating within decentralized frameworks. Working on this problem also involves taking into account the scalability challenges of blockchain, particularly for solutions designed for cross-device federated learning. Because this approach extends the concept of federated learning by enabling a large number of devices, such as smartphones, tablets, and IoT devices, to participate in the training process (*i.e.*, which generates a larger workload).

7.2.4 Untruthful Evaluation in RollupTheCrowd

While the evaluators in Rollupthecrowd are randomly assigned to evaluate tasks to avoid collusion attacks, we recognize the need to design an incentive mechanism to encourage effort among these entities to avoid untruthful and lazy evaluations. Although they are rewarded by the requester for their efforts, guaranteeing that the evaluators will behave honestly is not discussed in chapter 5 of this thesis. Thus, future research directions may include the design and implementation of a game-theoretic incentive mechanism to encourage the correct evaluation of crowdsourcing tasks by the designed set of evaluators.

7.2.5 Limitations and lessons learned from the thesis journey

Although we have presented various contributions to enhance the effectiveness, privacy preservation, and scalability of blockchain-based reputation systems, we have yet to introduce a unified system that integrates all these contributions and evaluates their collective potential. Specifically, we introduced GuRuMarket to improve BRS effectiveness and layer-1 scalability, followed by the proposal of DARS as a decentralized anonymous reputation system to support privacy preservation. We further introduced RollupTheCrowd to improve scalability and performance using zkRollups as a Layer-2 technique. However, we did not incrementally implement all of these frameworks introduced in this thesis to build a system encompassing all our contributions. Thus, future research directions may include combining the contributions in this thesis to produce a single overarching system.

The evaluations of GuRuMarket, DARS, RollupTheCrowd, and LLMChain were restricted to our evaluation platform. Considering the geographical distribution when evaluating blockchain is important. Potential future research can focus on evaluating the proposed work at a much larger scale (*e.g.*, using BlockZoom [50] or AWS instances[1]). Additional aspects for evaluation in future research could encompass: (1) assessing a broader range of DApp workloads, and (2) employing smaller instances (*i.e.*, machines) to gauge GuRuChain's performance on resource-limited devices like Internet-of-Things devices.

[1] https://aws.amazon.com/ec2/

Glossary

Blockchain-based Reputation Systems Leverage blockchain technology to establish, manage, and validate reputation scores for individuals or entities.

Blockchain Consensus The mechanism by which a decentralized network of nodes agrees on the validity and ordering of transactions.

DApps Decentralized Applications are software applications that run on a blockchain, inheriting its decentralization.

Node Blockchain nodes are PCs, servers, or other hardware devices connected through a P2P network. Each node on the network runs software to participate in blockchain protocols.

PoS Proof of Stake is a consensus mechanism that selects validators based on the amount of cryptocurrency they hold and are willing to "stake" as collateral.

Reputation-based Blockchain Systems Blockchain networks that incorporate reputation management to assess and validate the trustworthiness and reliability of participants/nodes within the network.

Reputation System A system implementing mechanisms and models used to assess the trustworthiness, reliability, and credibility of entities in various contexts.

Participant A blockchain participant is an entity that plays an active role in the functioning and maintenance of the blockchain network. Participants are directly involved in the consensus process, but they can also act as users by posting transactions to the network.

Smart Contract A piece of code that is stored and executed "deterministically" on a blockchain.

Sybil Attacks a single entity creates multiple fake identities or pseudonyms to gain control or influence over a network, typically in a decentralized system.

User blockchain users, refer to individuals or entities that interact with the blockchain to use its services and functionalities without necessarily being involved in its maintenance. When an entity can operate a blockchain node and join the network, it becomes a participant.

Web3 The vision of a decentralized and user-centric Web built on blockchain technology through DApps.

Wealth centralization A phenomenon in PoS blockchains that refers to the concentration of cryptocurrency wealth among a small number of participants or entities within the network.

Zero-Knowledge A cryptographic technique used to prove the validity of a claim (*i.e.*, statement) without revealing details about the claim.

List of Acronyms

ACSC Access Control Smart Contract

AML Access Management Ledger

BA Byzantine Agreement

BC Blockchain

BFT Byzantine Fault Tolerance

BML Business Management Ledger

BRS Blockchain-based Reputation System

BP Block Producer

CID Content Identifier

DApps Decentralized Applications

DARS Decentralized Anonymous Reputation System

DLT Distributed Ledger Technologies

DON Decentralized Oracle Network

EVM Ethereum Virtual Machine

GuRuChain Guarantee & Reputation based Blockchain

IDML IDentity Management Ledger

IPFS InterPlanetary File System

ISC Identity Smart Contract

LLMs Large Language Models

MPC Multi-Party Computation

PBFT Practical Byzantine Fault Tolerance

PoGR Proof-of-Guarantee&Reputation

PoX Proof-of-X (*e.g.*, Work, Stake, Authority, Trust, etc)

PPBRS Privacy-Preserving Blockchain-based Reputation Systems

PRF Pseudo-Random Function

RBS Reputation-based Blockchain System

RPC Remote Procedure Calls

SMR State Machine Replication

UTXO Unspent Transaction Output

zkSNARKs: Zero-Knowledge Succinct Non-Interactive Argument of Knowledge

Bibliography

[1] O. Hasan, L. Brunie, and E. Bertino, "Privacy-preserving reputation systems based on blockchain and other cryptographic building blocks: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–37, 2022.

See pages 5, 9, 10, 12, 13, 20, 21, 30, 33, 36, 37, 39, 75, 78, 79 and 102.

[2] X. Zhu, Y. Li, L. Fang, and P. Chen, "An Improved Proof-of-Trust Consensus Algorithm for Credible Crowdsourcing Blockchain Services," *IEEE Access*, vol. 8, pp. 102 177–102 187, 2020.

See pages 5, 30, 44, 45, 47, 67, 74, 103, 108 and 114.

[3] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, dec 2000.

See pages 9, 20 and 44.

[4] Y. Zhang and M. van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *In The 31st Proceedings IEEE International Conference on Computer Communications, INFOCOM, 2012, Orlando, FL, USA*, 2012, pp. 2140–2148.

See pages 9, 102, 114 and 132.

[5] Y. Wang and K.-J. Lin, "Reputation-oriented trustworthy computing in e-commerce environments," *IEEE Internet Computing*, vol. 12, no. 4, pp. 55–59, 2008.

See page 9.

[6] X. Xu, J. Gu, H. Yan, W. Liu, L. Qi, and X. Zhou, "Reputation-aware supplier assessment for blockchain-enabled supply chain in industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 4, pp. 5485–5494, 2022.

See page 9.

[7] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-Based Distributed Trust and Reputation Management Systems: A Survey," *IEEE Access*, vol. 8, pp. 21 127–21 151, 2020.

See pages 9, 10, 20, 21, 22, 30, 35, 39, 103 and 132.

[8] Y. Zhang, R. H. Deng, D. Zheng, J. Li, P. Wu, and J. Cao, "Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5099–5108, 2019.

See pages 9 and 78.

[9] M. A. Bouchiha, Y. Ghamri-Doudane, M. Rabah, and R. Champagnat, "GuRuChain: Guarantee and Reputation-based Blockchain Service Trading Platform," in *2023 IFIP Networking Conference (IFIP Networking), Barcelona, Spain*, 2023, pp. 1–9.

See pages 10, 12, 15, 20, 45, 54, 55, 56, 60, 78, 92, 103, 108, 114, 124, 126 and 144.

[10] V. Dedeoglu, R. Jurdak, G. D. Putra, A. Dorri, and S. S. Kanhere, "A Trust Architecture for Blockchain in IoT," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Houston, Texas, USA*, ser. MobiQuitous '19. Association for Computing Machinery, 2020, pp. 190–199.

See pages 10, 12, 44, 47 and 74.

[11] K. Zhao, S. Tang, B. Zhao, and Y. Wu, "Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing," *IEEE Access*, vol. 7, pp. 74 694–74 710, 2019.

See pages 10, 12, 79, 80, 98 and 99.

[12] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable Byzantine fault-tolerant services," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 59–74, 2005.

See page 10.

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

See pages 10, 24, 25 and 27.

[14] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

See pages 10, 22, 23, 24, 25, 27 and 53.

[15] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22 328–22 370, 2019.

See page 12.

[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles, Shanghai, China*, 2017, pp. 51–68.

See pages 12, 23, 25, 26 and 146.

[17] M. Huang, R. Han, Z. Du, Y. Fu, and L. Liu, "Reputation-based state machine replication," in *2022 IEEE 21st International Symposium on Network Computing and Applications (NCA), Virtual*, vol. 21, 2022, pp. 225–234.

See pages 12, 30, 31, 44, 48 and 74.

[18] T. Do, T. Nguyen, and H. Pham, "Delegated proof of reputation: A novel blockchain consensus," in *the Proceedings of the 1st International Electronics Communication Conference, Okinawa, Japan*, 2019, pp. 90–98.

See pages 12, 30, 44, 45, 47 and 74.

[19] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, and M. K. Khan, "PoRX: A reputation incentive scheme for blockchain consensus of IIoT," *Future generation computer systems*, vol. 102, pp. 140–151, 2020.

See pages 12, 44, 48 and 74.

[20] S. Malik, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "TrustChain: Trust Management in Blockchain and IoT Supported Supply Chains," *2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA*, pp. 184–193, 2019.

See pages 12, 30, 44, 46, 74 and 124.

[21] U. Jayasinghe, G. M. Lee, Á. MacDermott, W. S. Rhee *et al.*, "TrustChain: A privacy preserving blockchain with edge computing," *Wireless Communications and Mobile Computing*, vol. 2019, jan 2019.

See pages 12, 20, 44, 46 and 74.

[22] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3527–3537, 2019.

See pages 12, 36, 79, 80, 98 and 99.

[23] T. Dimitriou, "Decentralized Reputation," in *In the Proceedings of the 11th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 119–130.

See pages 13, 30, 79, 80, 98 and 99.

[24] A. Schaub, R. Bazin, O. Hasan, and L. Brunie, "A trustless privacy-preserving reputation system," in *ICT Systems Security and Privacy Protection: 31st IFIP TC 11 International Conference, Ghent, Belgium.* Springer, 2016, pp. 398–411.

See pages 13, 79, 80, 98 and 99.

[25] K. Soska, A. Kwon, N. Christin, and S. Devadas, "Beaver: A decentralized anonymous marketplace with secure reputation," *Cryptology ePrint Archive*, no. 2016/464, 2016. [Online]. Available: https://eprint.iacr.org/2016/464.pdf

See pages 13, 79 and 80.

[26] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs, White paper*, vol. 1, 2021. [Online]. Available: https://research.chain.link/whitepaper-v2.pdf

See pages 13, 53, 108, 115 and 127.

[27] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain Scaling Using Rollups: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, 2022.

See pages 13, 30, 100, 102, 103 and 104.

[28] J. Li, X. Cheng, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "HaluEval: A large-scale hallucination evaluation benchmark for large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore*, 2023, pp. 6449–6464.

See pages 14 and 122.

[29] O. Golovneva, M. Chen, S. Poff, M. Corredor, L. Zettlemoyer, M. Fazel-Zarandi, and A. Celikyilmaz, "ROSCOE: A suite of metrics for scoring step-by-step reasoning," in *The 11th International Conference on Learning Representations, (ICLR), Kigali, Rwanda*. OpenReview.net, 2023.

See pages 14 and 122.

[30] L. Pan, M. Saxon, W. Xu, D. Nathani, X. Wang, and W. Y. Wang, "Automatically Correcting Large Language Models: Surveying the landscape of diverse self-correction strategies," 2023. [Online]. Available: https://arxiv.org/abs/2308.03188

See pages 14, 122 and 123.

[31] M. A. Bouchiha, Y. Ghamri-Doudane, M. Rabah, R. Champagnat, and G. Bailly-Maitre, "GuRuMarket: Fostering Trust in Blockchain-Enabled Real-World Marketplaces via Guarantee and Reputation," *IEEE Transactions on Services Computing - Submitted*, 2024.

See pages 15, 60 and 144.

[32] M. A. Bouchiha, Y. Ghamri-Doudane, M. Rabah, and R. Champagnat, "DARS: Empowering Trust in Blockchain-Based Real-World Applications with a Decentralized Anonymous Reputation System," in *Advanced Information Networking and Applications*. Kitakyushu, Japan: Springer Nature Switzerland, 2024, pp. 48–61.

See pages 15, 79 and 144.

[33] A. M. R. Bendada, M. A. Bouchiha, Y. Ghamri-Doudane, and M. Rabah, "RollupTheCrowd: Leveraging ZkRollups for a Scalable and Privacy-Preserving Reputation-based Crowdsourcing Platform," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC) Osaka, Japan*. IEEE, 2024.

See pages 15, 144 and 145.

[34] M. A. Bouchiha, Q. Telnoff, S. Bakkali, R. Champagnat, M. Rabah, M. Coustaty, and Y. Ghamri-Doudane, "LLMChain: Blockchain-based Reputation System for Sharing and Evaluating Large Language Models," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC) Osaka, Japan*. IEEE, 2024.

See pages 16 and 145.

[35] C. Moorman, R. Deshpande, and G. Zaltman, "Factors affecting trust in market research relationships," *Journal of marketing*, vol. 57, no. 1, pp. 81–101, 1993.

See page 20.

[36] J. L. Johns, "A concept analysis of trust," *Journal of advanced nursing*, vol. 24, no. 1, pp. 76–83, 1996.

See page 20.

[37] R. C. Mayer, J. H. Davis, and F. D. Schoorman, "An integrative model of organizational trust," *Academy of management review*, vol. 20, no. 3, pp. 709–734, 1995.

See page 20.

[38] D. Gambetta, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 13, no. 2000, pp. 213–237, 2000.

See page 20.

[39] J. Arshad, M. A. Azad, A. Prince, J. Ali, and T. G. Papaioannou, "REPUTABLE–A Decentralized Reputation System for Blockchain-Based Ecosystems," *IEEE Access*, vol. 10, pp. 79 948–79 961, 2022.

See pages 20, 46 and 125.

[40] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "CrowdBC: A Blockchain-Based Decentralized Framework for Crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251–1266, 2019.

See pages 20 and 103.

[41] S. Malik, N. Gupta, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "TradeChain: Decoupling Traceability and Identity in Blockchain-enabled Supply Chains," *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1141–1152, 2021.

See pages 20, 44 and 50.

[42] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," in *Concurrency: the works of Leslie Lamport*, 2019, pp. 203–226.

See pages 22 and 26.

[43] T. Crain, C. Natoli, and V. Gramoli, "Red belly: A secure, fair and scalable open blockchain," in *Proceedings 2021 IEEE Symposium on Security and Privacy*. IEEE, 2021, pp. 466–483.

See pages 23, 24, 26 and 146.

[44] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the 13th EuroSys conference, Porto, Portugal*, 2018, pp. 1–15.

See page 25.

[45] J. M. Chase, "Quorum white paper." 2019. [Online]. Available: https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf

See page 25.

[46] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, "Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism," *IEEE Access*, vol. 7, pp. 118 541–118 555, 2019.

See pages 25, 29 and 47.

[47] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 179–196.

See page 26.

[48] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and Applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria*. Springer, 2015, pp. 281–310.

See page 26.

[49] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

See page 27.

[50] W. M. Shbair, M. Steichen, J. François, and R. State, "BlockZoom: Large-Scale Blockchain Testbed," in *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency - Demo*. Seoul, South Korea: IEEE, May 2019, pp. 5–6.

See pages 29 and 147.

[51] A. Hafid, A. S. Hafid, and M. Samih, "Scaling Blockchains: A Comprehensive Survey," *IEEE access*, vol. 8, pp. 125 244–125 262, 2020.

See page 29.

[52] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-Based Decentralized Trust Management in Vehicular Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019.

See page 30.

[53] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 429–445, 2018.

See pages 30, 31, 44 and 47.

[54] P. Resnick and R. Zeckhauser, "Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system," in *The Economics of the Internet and E-commerce*. Emerald Group Publishing Limited, 2002, pp. 127–157.

See page 36.

[55] E. Pavlov, J. S. Rosenschein, and Z. Topol, "Supporting privacy in decentralized additive reputation systems," in *Proceedings of 2nd International Conference on Trust Management, Oxford, UK*. Springer, 2004, pp. 108–119.

See page 36.

[56] M. Kinateder and S. Pearson, "A privacy-enhanced peer-to-peer reputation system," in *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies, Prague, Czech Republic*. Springer, 2003, pp. 206–215.

See page 36.

[57] S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li, "zkCrowd: A Hybrid Blockchain-Based Crowdsourcing Platform," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4196–4205, 2020.

See pages 36, 78, 79, 103, 104, 105 and 114.

[58] R. Lai and G. Zhao, "ValidatorRep: Blockchain-based Trust Management for Ensuring Accountability in Crowdsourcing," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Virtual*. IEEE, 2022, pp. 716–725.

See pages 46 and 124.

[59] S. Qi, Y. Li, W. Wei, Q. Li, K. Qiao, and Y. Qi, "Truth: A blockchain-aided secure reputation system with genuine feedbacks," *IEEE Transactions on Engineering Management*, 2022.

See page 46.

[60] Z. Jaroucheh, M. Alissa, W. J. Buchanan, and X. Liu, "TRUSTD: Combat fake content using blockchain and collective signature technologies," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Virtual*, 2020, pp. 1235–1240.

See pages 47 and 125.

[61] N. Singh, T. Kumar, and M. Vardhan, "Blockchain-based e-cheque clearing framework with trust based consensus mechanism," *Cluster Computing*, vol. 24, pp. 851–865, 2021.

See pages 47 and 74.

[62] W. Cai, W. Jiang, K. Xie, Y. Zhu, Y. Liu, and T. Shen, "Dynamic reputation–based consensus mechanism: Real-time transactions for energy blockchain," *International Journal of Distributed Sensor Networks*, vol. 16, p. 1550147720907335, Mars 2020.

See pages 47 and 74.

[63] Y. Sun, R. Zhang, R. Xue, Q. Su, and P. Li, "A Reputation Based Hybrid Consensus for E-Commerce Blockchain," in *Proceedings of the 27th International Conference, Held as Part of the Services Conference Federation, SCF, Honolulu, HI, USA, 2020.* Springer-Verlag, 2020, p. 1–16.

See pages 47 and 74.

[64] A. Hammoud, R. Mizouni, H. Otrok, S. Singh, A. Mourad, and Z. Dziong, "A blockchain-based hedonic game scheme for reputable fog federations," *IEEE Transactions on Services Computing*, 2023.

See page 47.

[65] M. Li, L. Zhu, Z. Zhang, C. Lal, M. Conti, and M. Alazab, "Anonymous and verifiable reputation system for e-commerce platforms based on blockchain," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4434–4449, 2021.

See page 47.

[66] A. Josang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Conference Proceedings of the 29th Australasian Computer Science Conference, ACSW, Tasmania, Australia.* Australian Computer Society, 2006, pp. 85–94.

See pages 47 and 54.

[67] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2906–2920, 2019.

See pages 47 and 54.

[68] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, "Proof of Reputation: A reputation-based consensus protocol for blockchain based systems," in *Proceedings of the 1st International Electronics Communication Conference, Okinawa, Japan*, 2019, pp. 131–138.

See page 48.

[69] M. T. de Oliveira, L. H. Reis, D. S. Medeiros, R. C. Carrano, S. D. Olabarriaga, and D. M. Mattos, "Blockchain reputation-based consensus: A scalable and resilient mechanism for distributed mistrusting applications," *Computer Networks*, vol. 179, p. 107367, 2020.

See page 48.

[70] L. Kleinrock, R. Ostrovsky, and V. Zikas, "Proof-of-reputation blockchain with nakamoto fallback," in *Proceedings of 21st International Conference on Cryptology, Bangalore, India, December 13–16, 2020*. Springer, 2020, pp. 16–38.

See pages 48 and 74.

[71] C. Tang, L. Wu, G. Wen, and Z. Zheng, "Incentivizing honest mining in blockchain networks: a reputation approach," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 117–121, 2019.

See page 48.

[72] M. Nojoumian, A. Golchubian, L. Njilla, K. Kwiat, and C. Kamhoua, "Incentivizing blockchain miners to avoid dishonest mining strategies by a reputation-based paradigm," in *Intelligent Computing: Proceedings of the 2018 Computing Conference, London, UK*. Springer, 2019, pp. 1118–1134.

See page 48.

[73] D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller, "CanDID: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability," in *Proceedings 2021 IEEE Symposium on Security and Privacy, Virtual*. IEEE, 2021, pp. 1348–1366.

See pages 50, 83, 84, 88, 98, 106 and 127.

[74] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 31–42.

See page 50.

[75] J. Benet, "IPFS-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

See pages 58 and 107.

[76] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

See page 62.

[77] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proceedings of the 40th international conference on software engineering: software engineering in practice, Gothenburg, Sweden*, 2018, pp. 134–143.

See pages 71 and 103.

[78] A. Bugday, A. Ozsoy, S. M. Öztaner, and H. Sever, "Creating consensus group using online learning based reputation in blockchain networks," *Pervasive and Mobile Computing*, vol. 59, p. 101056, 2019.

See page 74.

[79] J. Feng, X. Zhao, G. Lu, and F. Zhao, "PoTN: a novel blockchain consensus protocol with proof-of-trust negotiation in distributed IoT networks," in *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things, London, UK*, 2019, pp. 32–37.

See page 74.

[80] Y. Zheng, H. Duan, and C. Wang, "Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2475–2489, 2018.

See page 78.

[81] M. A. Azad, S. Bag, F. Hao, and A. Shalaginov, "Decentralized self-enforcing trust management system for social internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2690–2703, 2020.

See page 79.

[82] H. J. Jo and W. Choi, "BPRF: Blockchain-based privacy-preserving reputation framework for participatory sensing systems," *Plos one*, vol. 14, no. 12, p. e0225688, 2019.

See pages 80, 98 and 99.

[83] K. Zhao, S. Tang, B. Zhao, and Y. Wu, "Dynamic and Privacy-Preserving Reputation Management for Blockchain-Based Mobile Crowdsensing," *IEEE Access*, vol. 7, pp. 74 694–74 710, 2019.

See pages 80, 104, 105 and 113.

[84] J. Chen, W. Liang, L. Xiao, C. Yang, R. Zhang, Z. Gui, and A. Poniszewska-Marańda, "PrivBCS: a privacy-preserving and efficient crowdsourcing system with fine-grained worker selection based on blockchain," *Connection Science*, vol. 35, no. 1, p. 2202837, 2023.

See pages 80 and 104.

[85] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual*, 2020, pp. 1919–1938.

See pages 81, 85 and 88.

[86] D. R. Kuhn, V. Hu, W. T. Polk, and S.-J. H. Chang, *Sp 800-32. Introduction to public key technology and the federal PKI infrastructure.* National Institute of Standards & Technology, 2001.

See page 84.

[87] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 270–282.

See page 85.

[88] R. Gennaro, "Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks," in *Annual International Cryptology Conference, Santa Barbara, California, USA*. Springer, 2004, pp. 220–236.

See page 86.

[89] M. Abe and S. Fehr, "Perfect NIZK with adaptive soundness," in *Theory of Cryptography Conference, Amsterdam, Netherlands.* Springer, 2007, pp. 118–136.

See pages 86 and 104.

[90] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria.* Springer, 2016, pp. 305–326.

See pages 86, 87 and 95.

[91] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge," *Cryptology ePrint Archive*, 2019.

See pages 86, 87 and 95.

[92] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: efficient, succinct, modular," *Designs, Codes and Cryptography*, vol. 91, no. 11, pp. 3457–3525, 2023.

See page 86.

[93] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proceedings 2014 IEEE symposium on security and privacy, San Jose, California*. IEEE, 2014, pp. 459–474.

See pages 86 and 89.

[94] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic Span Programs and succinct NIZKs without PCPs," in *Proceedings 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece*. Springer, 2013, pp. 626–645.

See page 87.

[95] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *USENIX security symposium*, vol. 4, 2004, pp. 303–320.

See page 93.

[96] V. Nikolaenko, S. Ragsdale, J. Bonneau, and D. Boneh, "Powers-of-Tau to the people: Decentralizing setup ceremonies," *Cryptology ePrint Archive*, 2022. [Online]. Available: https://eprint.iacr.org/2022/1592

See page 94.

[97] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu, "A privacy-preserving trust model based on blockchain for VANETs," *IEEE Access*, vol. 6, pp. 45 655–45 664, 2018.

See pages 98 and 99.

[98] J. Howe, "The rise of crowdsourcing," *Wired*, vol. 14, 01 2006.

See page 102.

[99] W. Tong, X. Dong, Y. Shen, Y. Zhang, X. Jiang, and W. Tian, "CHChain: Secure and parallel crowdsourcing driven by hybrid blockchain," *Future Generation Computer Systems*, vol. 131, pp. 279–291, 2022.

See pages 104 and 105.

[100] L. Sun, Q. Yang, X. Chen, and Z. Chen, "RC-Chain: Reputation-based crowdsourcing blockchain for vehicular networks," *Journal of network and computer applications*, vol. 176, no. 102956, 2021.

See page 104.

[101] Z. Zhou, M. Wang, C.-N. Yang, Z. Fu, S. Xin, and Q. M. J. Wu, "Blockchain-based decentralized reputation system in e-commerce environment," *Future Generation Computer Systems*, vol. 124, 06 2021.

See page 105.

[102] T. Wang, J. Guo, S. Ai, and J. Cao, "RBT: A distributed reputation system for blockchain-based peer-to-peer energy trading with fairness consideration," *Applied Energy*, vol. 295, p. 117056, 2021.

See page 105.

[103] S. L. Kodjiku, Y. Fang, T. Han, K. O. Asamoah, E. S. E. B. Aggrey, C. Sey, E. Aidoo, V. N. Ejianya, and X. Wang, "ExCrowd: A Blockchain Framework for Exploration-Based Crowdsourcing," *Applied Sciences*, vol. 12, no. 13, 2022.

See page 105.

[104] C. Li, B. Palanisamy, R. Xu, J. Wang, and J. Liu, "NF-Crowd: Nearly-free blockchain-based crowdsourcing," in *Proceedings of 2020 International Symposium on Reliable Distributed Systems, Shanghai, China.* IEEE, 2020, pp. 41–50.

See page 105.

[105] C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang, "Is ChatGPT a general-purpose natural language processing task solver?" in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing.* Singapore: Association for Computational Linguistics, 2023, pp. 1339–1384.

See page 122.

[106] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge with MT-bench and chatbot arena," in *37th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

See pages 122, 123, 133, 136 and 137.

[107] R. Dale, "GPT-3: What's it good for?" *Natural Language Engineering*, vol. 27, no. 1, pp. 113–118, 2021.

See page 122.

[108] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

See page 122.

[109] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

See page 122.

[110] L. Sun, Y. Huang, H. Wang, S. Wu, Q. Zhang, C. Gao *et al.*, "TrustLLM: Trustworthiness in large language models," *arXiv preprint arXiv:2401.05561*, 2024.

See pages 122, 123 and 137.

[111] C. Wang, X. Liu, Y. Yue, X. Tang, T. Zhang, C. Jiayang, Y. Yao, W. Gao, X. Hu, Z. Qi *et al.*, "Survey on factuality in large language models: Knowledge, retrieval and domain-specificity," *arXiv preprint arXiv:2310.07521*, 2023.

See page 122.

[112] P. Fernandes, A. Madaan, E. Liu, A. Farinhas, P. H. Martins *et al.*, "Bridging the Gap: A Survey on Integrating (Human) Feedback for Natural Language Generation," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1643–1668, 12 2023.

See pages 122 and 123.

[113] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," in *8th International Conference on Learning Representations, (ICLR), Addis Ababa, Ethiopia*, 2020.

See pages 122 and 134.

[114] J. Belouadi and S. Eger, "UScore: An effective approach to fully unsupervised evaluation metrics for machine translation," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 358–374.

See page 122.

[115] H. Schuff, L. Vanderlyn, H. Adel, and N. T. Vu, "How to do human evaluation: A brief introduction to user studies in NLP," *Natural Language Engineering*, vol. 29, no. 5, p. 1199–1222, 2023.

See pages 122, 127, 129 and 130.

[116] C.-H. Chiang and H.-y. Lee, "Can large language models be an alternative to human evaluations?" in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Toronto, Canada: Association

for Computational Linguistics, 2023, pp. 15 607–15 631. [Online]. Available: https://aclanthology.org/2023.acl-long.870

See page 123.

[117] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: NLG evaluation using gpt-4 with better human alignment," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, 2023, pp. 2511–2522.

See pages 123 and 124.

[118] A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam *et al.*, "OpenAssistant Conversations – Democratizing Large Language Model Alignment," *arXiv eprint arXiv:2304.07327*, 2023.

See page 124.

[119] D. Pride, M. Cancellieri, and P. Knoth, "CORE-GPT: Combining Open Access research and large language models for credible, trustworthy question answering," in *International Conference on Theory and Practice of Digital Libraries, Zadar, Croatia*. Springer, 2023, pp. 146–159.

See pages 124 and 137.

[120] W. Ye, M. Ou, T. Li, X. Ma, Y. Yanggong, S. Wu, J. Fu, G. Chen, J. Zhao *et al.*, "Assessing hidden risks of llms: An empirical study on robustness, consistency, and credibility," *arXiv preprint arXiv:2305.10235*, 2023.

See page 124.

[121] W. Yuan, G. Neubig, and P. Liu, "BARTScore: Evaluating Generated Text as Text Generation," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 27 263–27 277.

See pages 129 and 134.

[122] W. Zhao, M. Strube, and S. Eger, "DiscoScore: Evaluating text generation with BERT and discourse coherence," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 3865–3883.

See pages 129 and 134.

[123] M. Körber, "Theoretical considerations and development of a questionnaire to measure trust in automation," in *Proceedings of the 20th Congress of the International Ergonomics Association, Florence, Italy*. Springer, 2019, pp. 13–30.

See page 130.

[124] D. Khashabi, A. Ng, T. Khot, A. Sabharwal, H. Hajishirzi, and C. Callison-Burch, "GooAQ: Open question answering with diverse answer types," *arXiv preprint arXiv:2104.08727*, 2021.

See page 133.

[125] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318.

See page 134.

[126] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

See page 134.

[127] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco, "Single secret leader election," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 12–24.

See page 146.

Laboratoire Informatique, Image, Interaction
Faculté des Sciences & Technologie
Avenue Michel Crépeau

17042 LA ROCHELLE CEDEX 1