WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:
Raj Jain, Chair
Elisa Bertino
Shantanu Chakrabartty
Roger Chamberlain
Ning Zhang

A Collaborative Knowledge-Based Security Risk Assessments Solution using Blockchains

by

Tara Thaer Salman

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2021
St. Louis, Missouri

© 2021, Tara Thaer Salman

Table of Contents

List of Figures viii
List of Tables x
Acknowledgments xi
Abstract xv
Chapter 1: Introduction 1
1.1. The Visionary Blockchain-based Security Risk Assessment 3
1.2. Dissertation Objective and Contributions 4
1.3. Dissertation Organization 7
Chapter 2: Blockchain Background and Standard Security Risk Assessment 9
2.1. Blockchain Technology 9
2.1.1. Blockchain Architecture 9
2.1.2. Main Characteristics of Blockchains 10
2.1.3. Consensus Algorithms in Blockchains 11
2.1.4. Permissioned and Permissionless Blockchains 15
2.2. State-of-the-Art Security Risk Assessment Solutions 17
2.2.1. Fully Centralized Approaches 17
2.2.2. Distributed Assessment Single Party Aggregation Approaches 18
2.2.3. Hierarchal Approaches 18
2.2.4. Distributed Approaches 19
2.3. Challenges with Discussed Risk Assessment Solutions 19
2.3.1. Centralization 19
2.3.2. Inefficient Distribution 20
2.4. Blockchain-based Security Risk Assessment 20
2.4.1. Blockchains Help Resolve the Challenges 20
2.4.2. Existing Blockchain-based Security Risk Assessment 20
2.5. Problems with Existing Blockchain-based Security Risk Assessment Solutions 21
2.5.1. Storage without Knowledge 21
2.5.2. Reputation Scheme 22
2.5.3. Practical Use-case and Proper Analysis 22
2.5.4. Proper Evaluation Metric 23
2.6. How This Dissertation Resolve These Challenges 23
Chapter 3: Knowledge-based Blockchains 25

3.1. Introduction and Related Works 25
3.2. Motivation and Decision-Making Requirements 28
3.2.1. Transactions Types 28
3.2.2. Achieving the Knowledge Summary 29
3.3. Contributions 30
3.4. Proposed Knowledge-based Blockchains 31
3.4.1. Overview of the Solution 31
3.4.2. Proposed Metrics for Knowledge-based Blockchains 33
3.4.3. Chain Design in Knowledge-based Blockchains 36
3.4.4. Challenges in Achieving Knowledge Summary 37
3.5. Security Analysis of Knowledge-Based Blockchains 38
3.5.1. Resiliency to Malicious or Bad Agents 38
3.5.2. DDoS Protection 41
3.5.3. Fraud Mitigation 41
3.6. Performance Evaluation 42
3.6.1. Application Scenario and The Summary Function 42
3.6.2. Experiment Setup 43
3.6.3. UNSW-NB15 Dataset 44
3.6.4. Evaluation Metrics 44
3.6.5. Experimental Results 45
3.7. Summary 49
Chapter 4: A Reputation Scheme on Top of Knowledge-based Blockchains 52
4.1. Introduction and Related Works 52
4.2. Motivation 55
4.3. Contributions 56
4.4. Reputation Scheme Requirements 57
4.4.1. Continuity and Boundedness 57
4.4.2. A Reputation Starts at 0.5 58
4.4.3. Time-Dependency 58
4.4.4. Configurable Increase and Decrease 58
4.4.5. Slower Reputation Recovery 59
4.4.6. Proportional Update 59
4.4.7. Security Configurable Damage 60
4.5. Proposed Reputation Scheme 60
4.5.1. EWA as a Reputation Function 61
4.5.2. Extensions to Meeting the Requirements 61

4.5.3. RPMC-EWA Meets the Requirements 64
4.6. RPMC-EWA in This Dissertation Research 67
4.6.1. RPMC-EWA for Decision Making Process 67
4.6.2. RPMC-EWA Use for Malicious Node Detection 68
4.6.3. Reputation as an Agent Feedback 69
4.6.4. RPMC-EWA Use for Mining 69
4.7. Performance Evaluation 70
4.7.1. Beta Reputation Algorithm 70
4.7.2. RPMC-EWA Algorithm Setting 71
4.7.3. Attack Strategies 71
4.7.4. Malicious Node Detection Results 72
4.7.5. Effect on Knowledge Summary 74
4.8. Summary 77
Chapter 5: Performance Analysis of Quorum 79
5.1. Introduction and Related Works 79
5.2. Motivation 81
5.3. Contributions 82
5.4. Blockchain Performance Metrics 83
5.4.1. Transaction Delay 83
5.4.2. Throughput 84
5.4.3. Transaction Success Probability 85
5.4.4. Resource Consumptions 86
5.4.5. Invalid Blocks Probability 86
5.4.6. Query invalid/attack probability 87
5.4.7. The Ratio of Forked Blocks 87
5.5. Blockchain Parameters and Factors 87
5.5.1. Blockchain System Parameters 88
5.5.2. Blockchain Workload Parameters 90
5.6. Blockbench and Quorum 90
5.7. Performance Analysis 93
5.7.1. Chosen Metrics 94
5.7.2. Smallbank Workload 94
5.7.3. Chosen Factors, and Their Levels 95
5.7.4. Effect of Transaction Rates on the Performance 96
5.7.5. Effect of the Number of Nodes on the Performance 99
5.8. Summary 103

Chapter 6: CrowdFAB: A Framework for Crowd-Forecasting Applications using Blockchains 105
6.1. Introduction and Related Works 105
6.2. Motivation 108
6.3. Contributions 109
6.4. CrowdFAB Framework Modeling 110
6.4.1. System Components 111
6.4.2. CrowdFAB System Architecture 112
6.4.3. System and Security Assumptions 113
6.4.4. Threat Assumptions 115
6.4.5. Security Goals 118
6.5. CrowdFAB Specification and Implementation 119
6.5.1. CrowdFAB Data Flow 119
6.5.2. Summary Function in Knowledge-based Blockchains 121
6.5.3. RPMC-EWA Reputation Scheme 121
6.5.4. CrowdFAB Algorithm 123
6.5.5. CrowdFAB Implementation 126
6.6. Performance and Security Analyses 128
6.6.1. Analysis of CrowdFAB Overhead 128
6.6.2. Agents' reputation Analysis 132
6.6.3. Security Analysis of CrowdFAB 134
6.7. Summary 138
Chapter 7: Safety Score as an Evaluation Metric for Security Risk Assessment Applications .. 141
7.1. Introduction and Related Works 141
7.2. Motivation 143
7.3. Contributions 144
7.4. Conventional Evaluation Metrics 145
7.4.1. ML Prediction Outcomes 145
7.4.2. Accuracy 145
7.4.3. Weighted Accuracy 146
7.4.4. AUC 146
7.4.5. Precision, Recall, and F1 Score 147
7.5. The Challenge and The Proposed Metric 147
7.5.1. An Illustrative Example 147
7.5.2. The Performance in Relation to Cost 149
7.5.3. Proposed Safety Score Metric 149

7.5.4. Weight Assignment Procedure 150
7.5.5. Extending the Illustrative Example 151
7.5.6. Metrics Virtualization with Different FN Rates 152
7.6. Case Study 1: General Breach Intrusion Detection 153
7.6.1. Datasets Used 153
7.6.2. ML Models and Weights Setup 154
7.6.3. Results and Analysis 155
7.7. Case Study 2: Injection Breaches in HIoT 156
7.7.1. Healthcare Networks and Their Requirements 156
7.7.2. Healthcare Testbed and Dataset 157
7.7.3. Results and Analysis 157
7.8. Summary 159
Chapter 8: Malware Detection as a Security Risk Assessment Case Study 161
8.1. Introduction and Related Work 161
8.2. Motivation and Contributions 163
8.3. Malware Detection Application Setup 164
8.3.1. Application Scenario 164
8.3.2. Agents' Setup 165
8.3.3. CrowdFAB Setup 167
8.4. Performance Evaluation 168
8.4.1. Baseline Algorithms 168
8.4.2. Metrics for Evaluation 168
8.4.3. Results and Analysis 169
8.4.4. Discussion 171
8.5. Summary 172
Chapter 9: Application Domains in and Beyond This Dissertation 174
9.1. Security Risk Assessment Applications 174
9.1.1. Cyber Threats Information and Knowledge Exchange 174
9.1.2. Node Admission and Access Control 176
9.1.3. Fraud Identification and Spam Filtering 177
9.1.4. Content Sharing and Trust in Vehicular Networks 179
9.1.5. Anti-Money Laundering 180
9.2. Other Risk Assessment and Decision-Making Applications 181
9.2.1. Aggregation in Wireless Sensor Networks 181
9.2.2. Stock Market Predictions 182
9.2.3. Disease Outbreak Prediction 183

9.2.4. Recommendation Systems 184
9.2.5. Reinforcement Learning Applications 184
9.3. Summary Functions Recommendations 185
9.4. Summary 188
Chapter 10: Conclusion and Future Work 189
10.1. What Have We Achieved in This Dissertation? 190
10.2. Future Work 192

List of Figures

Figure 2.1 Blockchain Process 10

Figure 3.1: A comparison between traditional blockchains and knowledge-based blockchains processes 32

Figure 3.2: Knowledge-based blockchains architecture 36

Figure 3.3: Accuracy results from the first experimental settings 46

Figure 3.4: FAR results from the first experimental settings 47

Figure 3.5 UND results from the first experimental settings 48

Figure 3.6:Accuracy results from the second experimental settings 49

Figure 4.1 RPMC-EWA meets the slow reputation recovery requirement ……67

Figure 4.2 Reputation change in the case of random flipping strategy after 50 correct prior decisions. 74

Figure 4.3 knowledge summary with and without malicious node detection 76

Figure 4.4 The change in agents' reputation after it turns malicious 77

Figure 5.1 Different definitions of transaction's write delays ……84

Figure 5.2 Blockbench evaluation benchmark 91

Figure 5.3 Effect of the number of nodes on the throughput 97

Figure 5.4 Effect of the number of nodes on the delay 97

Figure 5.5 Effect of the number of nodes on the success rate 98

Figure 5.6 Effect of the number of nodes on the throughput 100

Figure 5.7 Effect of the number of nodes on the delay 101

Figure 5.8 Effect of the number of nodes on the delay 102

Figure 6.1 CrowdFAB architecture ……113

Figure 6.2 CrowdFAB data flow 120

Figure 6.3 Screenshot of request and response transactions 127

Figure 6.4 Screenshots of block with and without response transactions..... 127

Figure 6.5 CrowdFAB overhead analysis on Quorum platform 131

Figure 6.6 With α =0.05, the reputation of a competent agent increases slowly 133

Figure 6.7 With γ=0.7, the reputation decreases rapidly on an FN mistake (at response index=3) 133

Figure 6.8 With β=0.3, the reputation decreases moderately on an FP mistake (at response index=7) 134

Figure 7.1 The change in metrics as the percentage of FN increases ……. 153

Figure 8.1 A workflow for blockchain-based malware detection for mobile applications …..165

List of Tables

Table 2.1: Key characteristics of blockchains 11

Table 2.2 Comparison of different consensus algorithms 13

Table 2.3 Blockchain covered platforms with their types, consensus algorithm, advantages, and disadvantages 16

Table 3.1 UNSW dataset statics (used version) 44

Table 4.1 Results of RPMC-EWA and Beta algorithms performance for malicious node detection 73

Table 5.1 Parameters and levels for our experimental design 95

Table 6.1 Notations used in this chapter..... 111

Table 6.2 Parameters and Factors for the Experimental Design 129

Table 7.1 Illustrative example evaluation with traditional metrics 148

Table 7.2 Extended illustrative example evaluation 151

Table 7.3 Dataset statistics of UNSW-NB15 DoS attack 154

Table 7.4 Statistics and results for UNSW-NB15 DoS dataset 155

Table 7.5 Statistics of HIoT dataset 157

Table 7.6 Statistics and results for HIoT dataset 158

Table 8.1 Performance of CrowdFAB malware detection with different summary functions ... 170

Table 9. 1 Summary functions used in this work 187

Acknowledgments

During my time at WashU, I have received enormous help and support from many people, and this dissertation would not have been possible without all of them. Words can't express my gratitude to many of them. My sincerest apologies to those who I have left out unintentionally.

First and foremost, I would like to express my deepest gratitude to my advisor Prof. Raj Jain who I consider my dearest academic and research father. Even though I came to WashU with some research background, working with him has drastically changed how I see and conduct research. He has taught me every aspect of state-of-the-art research, from technical skills to big picture and research taste. He has believed in me and helped me get through many difficulties adjusting to the US academic life. I have learned from him much more than just research. All of which would be invaluable to me in my future career and life.

I am also incredibly thankful to the rest of the dissertation committee for their valuable recommendations and making the time to be part of this dissertation. Prof. Elisa Bertino has always inspired me with her guide, research ideas, and generous help and recommendation. Having her on the committee has been such a great honor. Prof. Shantanu Chakrabartty has provided excellent support and encouragement as part of this dissertation committee. Prof. Roger Chamberlain, with whom I have taken my first ever class at WashU, has been a great source of inspiration all through my research and the job search process. Prof. Ning Zhang has impressed me enormously with his research and contributions to the cybersecurity programs and events at WashU. He also has been a great help throughout my research journey over the last three years.

Being part of the computer science and engineering department has been a tremendous fortune for me. I'd like to mention a few people with whom I've had interacted and who have left an indelible

impression on me. Prof. Roch Guerin with whom I did a rotation and had significantly learned from his insightful discussion. Prof. Jeremy Buhler, Prof. Ben Moseley (now at CMU), and Prof. Sanmay Das (now at GMU), with whom I have taken classes, and I have learned a lot from them. Prof. William Richard, an examiner for my DSS, has made excellent recommendations to enhance my presentation skills. Dr. Marion Neumann and Dr. Hila Ben Abraham, whom I enjoyed working with over the years.

The CSE Department and Graduate School staff have been extremely accommodating and helpful. Myrna Harbison, Monét Demming, Kelli Eckman, Sharon Matlock, Lia Garofolo, Cheryl Newman, and Andrew Swafford are a few names that come to mind. They all have a way of making even the most hopeless situations seem manageable.

I feel extremely lucky to have worked with many talented and hard-working members at Prof. Jain's lab. They have provided me with valuable insights through discussions, presentation trials, coauthoring, programming, and polishing my unpolished work. First, I would like to thank Dr. Deval Bhamare, a remote postdoc in the lab who helped me put together my first two Ph.D. conference publications. A special thanks to one of the former members, Prof. Lav Gupta (now at UMSL), who I have known since my first day at WashU and has been a great help in research ideas, Ph.D. requirements, coauthoring, and many other aspects throughout the years. I would like to also thank Ali Gubaish for being such a supportive friend, a programming problem solver, and a great listener throughout the Ph.D. years. Prof. Marcio A. Teixeira and Maede Zolanwari, my coauthors, who have helped a lot in research work in the past few years. Lav, Maede, and Ali have also been great lab and office mates, and I have enjoyed every minute of our discussions. Another thank you goes to Zebo Yang, who has joined the lab during COVID time, and so he has been a great virtual friend and collaborator. Another thank you goes to Modhi Bin Kulib, a previous

master student at the lab, who I immensely enjoyed supervising her master project on blockchains. A final WashU thanks goes to all the friends during my time here, Dina Elready, Yehan Ma, Haoran Li, Shali Jaing, and many others who are not listed for all the great time we had together.

Thanks to the funding agencies, NSF, QNRF, and Cisco, for supporting the research. All the research work presented in this dissertation have been partially or fully funded by the following grants: QNRF grants NPRP11S-0109-180242, NPRP-10-0125-170250, NPRP8-634-1-131 from the Qatar National Research Fund (a member of Qatar Foundation) and National Science Foundation grants NSF CNS-1718929 and NSF CNS-1547380.

An extended thanks go to two of my former supervisors, Prof. M. Samaka and Prof. T. Elfouly, who have always believed in me and endorsed my Ph.D. application. Without them, I would never have the courage to start or even apply for a doctoral degree.

I reserve the most heartfelt thanks for my parents, who have led me to follow my academic dream and who have been constantly by my side (virtually) and supporting me unconditionally. A similar thank you goes to Dr. Judy Gainer, who I have lived with over the past six years and consider as my US mother. Without them listening to my endless complaints about the challenges of being a Ph.D. Scholar, this work would not have seen the light of day.

To my aunt, Noor (who I consider as a second mother), my three brothers, Salman, Mohamed, and Mustafa, my sisters-in-law, and my three nieces, thank you for being a part of my Ph.D. life and always providing me with the support and joy I need to release all the Ph.D. stress.

Tara Salman

Washington University in St. Louis

May 2021

Dedicated to my family.

ABSTRACT OF THE DISSERTATION

A Collaborative Knowledge-Based Security Risk Assessments Solution using Blockchains

by

Tara Thaer Salman

Doctor of Philosophy in Computer Science and Engineering

Washington University in St. Louis, 2021

Professor Raj Jain, Chair

Artificial intelligence and machine learning have recently gained wide adaptation in building intelligent yet simple and proactive security risk assessment solutions. Intrusion identification, malware detection, and threat intelligence are examples of security risk assessment applications that have been revolutionized with these breakthrough technologies. With the increased risk and severity of cyber-attacks and the distributed nature of modern threats and vulnerabilities, it becomes critical to pose a distributed intelligent assessment solution that evaluates security risks collaboratively. Blockchain, as a decade-old successful distributed ledger technology, has the potential to build such collaborative solutions. However, in order to be used for such solutions, the technology needs to be extended so that it can intelligently process the stored information and achieve a collective decision about security risks or threats that might target a system.

The research done in this dissertation aims to redesign the blockchains and build a collaborative, distributed, intelligent, and hostile solution that can be used for security risk assessment purposes. In particular, the research focus can be divided into two stages: (1) extending blockchains for decision-making and risk assessments applications, where knowledge should be made out of decisions, risks, or any information stored on the blockchain; (2) applying the proposed extensions to security risks assessment applications such as malware detection and threat intelligence.

Knowledge-based blockchains are introduced to extend blockchains from simple storage systems to processing and knowledge-based systems in Stage 1. The newly proposed concept processes and summarizes stored information within the blockchain process. Thus, it extends the blockchain's applications to decision-making and risk assessments where multiple decision-makers predict a future outcome, and the system needs to collaboratively assess the outcomes and take actions accordingly. In addition, knowledge-based blockchains are topped with a proper reputation scheme that assigns reputation to decision-makers based on their performances. CrowdFAB, a novel framework for Crowd-Forecasting Applications using Blockchains, is introduced to combine the two contributions and build a crowd-forecasting system to underlie any decision-making applications.

For security risk assessment applications, a new evaluation metric is first proposed to evaluate blockchain-based and non-blockchain-based security risk assessment solutions. The metric mainly focuses on evaluating any security assessment solution by the cost associated with the solution's predictions or assessments. Then, a novel blockchain-based solution for malware detection in mobile applications is formulated and evaluated as a security risk assessment case study. Empirical evaluations indicate a secure framework's feasibility, a sustainable close-to-optimal performance that outperforms earlier works, and a reasonable overhead made by the proposed system. Finally, to extend the research's applicability beyond the scope of this dissertation, a discussion of the possible security-oriented and non-security applications that can utilize the proposed concepts is presented.

Chapter 1:Introduction

Modern networking applications, including healthcare, financial, and cyber-physical systems, are massively distributed. To support these applications, multiple cloud providers, internet service providers, and content providers need to collaborate with the application providers. This collaboration results in globally distributed systems that include various entities that do not necessarily trust each other. With such diverse and multi-trust systems, a centralized third-party is needed to manage the resources between different entities. However, a centralized third-party is susceptible to hacking and may itself be a source of insecurity.

Blockchain, as cutting-edge technology, has the potential to distribute resource management in these multi-trust systems. Blockchains are secured, shared, and distributed ledgers that facilitate the process of recording, managing, and tracking resources without the need for a centralized authority. It started with Bitcoin, a popular digital payment system [1]. With the success of Bitcoin, a new trend has started in using blockchains for other massively distributed applications. Blockchains can provide distributed, scalable, and secure infrastructures that are of interest to many application providers. As a result, blockchain applications currently span financial applications [1], healthcare [2], Internet of Things (IoT) [3], cloud storage [4], supply chains [5], among others. Generally speaking, blockchain technology has proven its potential in many massively distributed applications that currently require a trusted third party. Many high-tech companies, including IBM, Microsoft, and Google, are working to provide Blockchain as a Service (BAAS) [6]. These services offer a smooth deployment of blockchain-based applications to facilitate the transition from centralized to distributed management.

On the other hand, security for such distributed applications is a significant challenge. The number of cybersecurity breaches has drastically increased over the past few years. It has become a major day-to-day struggle for governments, businesses, and individuals, especially with the increasing use of the distributed applications to support distant and online workers. For example, in 2020, security breaches have increased by 11% since 2018 and 67% since 2014. That is, breaches have more than doubled over six years. According to IBM, the average cost of data and cybersecurity breaches has exceeded $3.86 million as of 2020. Most of these breaches come from ransomware, malware, Distributed Denial of Service (DDoS), and other networking attacks. Further, the average time to identify a breach is 207 days [7].

One of the challenges in standard security solutions is that they are centralized or inefficiently distributed. This especially applies to security risk assessment, such as malware or intrusion detection, where an assessment of security threats or events is required. Current approaches either assess the security locally at each entity or mitigate the assessment to an authority. Local assessments do not have the full view of the system and have no access to others' assessments. Meanwhile, authorities are centralized, cost-inefficient, time-consuming, and require all parties to trust them. A decentralized approach can distribute the risk assessment among application entities, resulting in a better awareness of security at an application-level and enabling faster innovation and detecting the risk [8].

Given the above, there is a need for an alternative intelligent, secured, distributed, and efficient assessment solution to evaluate security risks in massively distributed applications.

1.1. The Visionary Blockchain-based Security Risk Assessment

The role of artificial intelligence (AI) in providing intelligent security assessment solutions is undeniable nowadays. AI can automate attack detection and reduce human error that is a significant cause of cybersecurity breaches [9]. Our own work prior to this dissertation research has been focusing on building AI-based intrusion detection systems in healthcare [10], industrial IoT [11], and multi-cloud applications [12].

Through these works, the research has identified that one major challenge and an active area of research in AI-based security solutions is the lack of security data to build efficient detection models. To reduce the severity of this challenge, a solution can intelligently combine predictions from multiple (thousands) of AI-based local models designed for the same objective. However, to combine these predictions, one would need an aggregator, which leads the centralization, high service cost, single point of failure, and trust issues discussed earlier.

The success of blockchains in resolving a similar challenge in other applications has proven their potentials for this application. They can help build better security assessment approaches that are distributed, scalable, time and cost-efficient, and span many domains that may not trust each other. Multiple security assessors (agents) can share their assessments about a risk to the blockchains, and the system can reach a collaborative assessment or decision about that risk. For example, tracking and detecting malicious applications or network flow in mobile platforms and networks can be solved by blockchains. Thousands of distributed malware detection agents individually evaluate if an application is malicious or not, and the system will reach a collaborative response about the application [13]. Similarly, various nodes can vote if an active node is misbehaving or if an account or an identity is fake, which would help detect malicious nodes in distributed networks.

However, current blockchains need several extensions to be used for such assessments. First, blockchains are currently used for storage and simple validations only. *A transformation from standard storage blockchains to knowledge-based blockchains is required to make collaborative and efficiently distributed assessments suitable for security applications.* Second, a way to distinguish between multiple agents who participate in the assessment and detect malicious agents is necessary for a quality assessment. Thus, *a reputation scheme that provides reputations to agents based on their performance is needed for such applications*. Further, due to the importance and time criticality of these assessments, it is critical to keep them secure and keep the overhead low for any proposed solution. *The success of blockchain-based security assessment solutions requires an analysis of the system performance and security that guarantees the security of the collaborative assessment with provable minimal overhead*.

From another research perspective, our earlier research in AI-based security solutions has identified another challenge for AI-enabled security solutions. Namely, the lack a good evaluation metric that represents the cost that machine learning (ML) or an AI model makes for security applications. As these models would underlie the above security risk assessment applications, the visionary blockchain-based solution would lack a good evaluation metric. Therefore, *a new cost-representative evaluation metric is required to evaluate any blockchain-based AI-based solution for security risk assessment applications.*

1.2. Dissertation Objective and Contributions

As discussed in the previous subsection and as will be detailed in Chapter 2, blockchain technology needs several extensions to build the visionary blockchain-based security risk assessment. The aim of the research in this dissertation is to redesign the blockchains to achieve the extension task first.

Then, it applies the extended blockchain framework to security risk assessment, thereby creating distributed security solutions suitable for next-generation distributed applications.

Our objective is to design an intelligent, collaborative, distributed, and hostile node resilient solution that can be used to build efficient security assessment solutions suitable for next-generation distributed applications. This will be done by utilizing state-of-the-art technologies, including blockchains, machine learning, reputation scheme and crowdsourcing models.

The significant contributions of the dissertation are as follow:

1- The blockchain technology is first extended from storage-based systems to knowledge-based systems. Such extension is critical for risk assessment solutions as it allows blockchains to make collaborative decisions within their processes, thus, getting rid of the centralization problem and third-party trust. This extension results in a new era of blockchains that combines the technology with AI to build collaborative and intelligent decision-making systems. This new extension is referred to as ***knowledge-based blockchains*.** The work for this contribution has been published in [14, 15].

2- A set of requirements is defined for any scheme that assigns reputations to participants/agents in security risk assessment applications. Then ***RPMC-EWA***, a reputation scheme that tops knowledge-based blockchains to provide reputations for agents based on their performance, is proposed to meet these requirements. The proposed reputation scheme is used to manage agents' contributions to decision-making and eliminate/disregard malicious agents who try to manipulate the system's performance. The work done to achieve such reputation scheme has been published in [16].

3- ***CrowdFAB***, a framework for Crowd-Forecasting Applications using Blockchains, is conceptualized. CrowdFAB combines the previous two contributions to build a blockchain-based solution designed explicitly for forecasting applications. Its novelty lies in providing an efficient, low overhead, and provenly secure underlying platform for crowd-forecasting applications. Even though this research specifically targets security risk assessment crowd-forecasting applications, it is anticipated that CrowdFAB will grow not only for these applications but also as a distributed and secure alternative for general crowd-forecasting applications. The work in here is under preparation for publication in [17].

4- CrowdFAB is implemented on an existing blockchain-based platform and analyzed in terms of overhead and security. It is shown that the system achieves its tasks securely and efficiently with a reasonable overhead.

5- Moving on to security assessment applications, the research first shows that traditional evaluation metrics that evaluate AI-enabled solutions fail to represent the actual performance in terms of cost. A novel and adjustable metric dedicated is proposed and analyzed for several applications with different requirements. It is shown that the proposed metric, named ***Safety score***, represents the performance of the actual models in terms of costs or system effects. This makes the metric easy to explain to network administrators who care about security cost the most. Safety score has been published in [18].

6- The proposed approaches are validated by building a passive malware detection in mobile applications as security case-study. Using ML models learned from previously published datasets, multiple malware agents that reflect real-world scenarios are simulated. The system is ten empirically evaluated in terms of several performance metrics. Results show

that using the proposed approaches can enhance the prediction performance over earlier works and result in a close to optimal performance. The work in here is under preparation for publication in [17].

1.3. Dissertation Organization

The rest of the dissertation has the following structure:

- Chapter 2 provides the necessary background for blockchain technology. In addition, it will review the state-of-the-art security risk assessment solutions, their challenges, how blockchains can resolve these challenges, and how this dissertation will advance blockchain-based solutions.
- Chapter 3 introduces knowledge-based blockchains paradigm as a method developed to extend blockchains from storage-based systems to knowledge-based systems. The paradigm resolves the first storage without knowledge challenge and introduce a new generation of blockchain applications dedicated to decision-making and security risk assessment applications.
- Chapter 4 discusses RPMC-EWA as a new reputation scheme that assigns weights to agents based on their performance. Further, it shows the use of this scheme for risk assessment applications and knowledge-based blockchains.
- Chapter 5 analyzes the performance of Quorum as a chosen permissioned blockchain platform that is used in the implementation of the proposed approaches. The objective here is to verify the suitability of the platform for risk assessment applications.
- Chapter 6 integrates knowledge-based blockchains with the reputation scheme to propose CrowdFAB as a novel framework undelaying crowd-forecasting blockchain-based system

for risk assessment applications. Then, CrowdFAB is analyzed in terms of overhead and security features.

- Chapter 7 brings out the issue of proper metrics to analyze the performance of a security risk assessment solution and introduces a Safety score new metric that resolves these issues.
- Chapter 8 develops and analyzes a blockchain-based AI-powered malware detection system as a case-study for security risk assessment applications.
- Chapter 9 presents some other security and non-security risk assessment applications that can use this work for decision-making applications.
- Chapter 10 summarizes the work and highlights some future work.

Chapter 2:Blockchain Background and Standard Security Risk Assessment

In this chapter, we first provide a background on blockchain technology for a general computer science reader. Then, some state-of-the-art approaches for security risk assessments, their challenges, and how blockchains can resolve these challenges are discussed. We detail the blockchain-based strategies to resolve these traditional challenges, their own problems, and the possible blockchain extensions to meet the requirements for distributed decision-making and security risk assessment applications.

2.1. Blockchain Technology

Since its start in 2008, blockchain technology has evolved as a promising secure, shared, and distributed ledger technology that facilitates the process of recording and tracking resources without the need for a centralized trusted authority. Resources can be tangible (e.g., money, houses, cars, lands) or intangible (e.g., copyrights, digital documents, and intellectual property rights). In general, anything that has value can be tracked on a blockchain network to reduce its security risks and save the cost of security monitoring for all involved. The following provides some of the necessary background to understand the technology and its later extensions for a general computer science audience. Most of the discussion here have been published in [19].

2.1.1. Blockchain Architecture

A blockchain consists of two main components: a chained database and a distributed network of nodes. The database is a set of time-ordered transactions that are signed and recorded. Several transactions are verified and bundled together to form a block based on predefined techniques

known as consensus algorithms (also named mining algorithms). Each block has a header that includes a hash value of its predecessor. Thus, all transactions are linked together in a chained database. The database is maintained by many globally distributed nodes that form the blockchain network. Although all nodes can access them, the blocks and transactions cannot be altered or denied once committed to the chain [1].

As shown in Figure 2.1, one can divide the nodes' roles in a blockchain system into three categories: users, miners, and validators. Users are clients of the systems that submit transactions as fundamental interactions in the system. These are not necessarily blockchain nodes, i.e., they do not necessarily hold the chained database and interact with the system using predefined application programming interfaces (API). Miners validate transactions and form blocks. Validators verify the generated block's correctness and add it to the chain if it is valid.

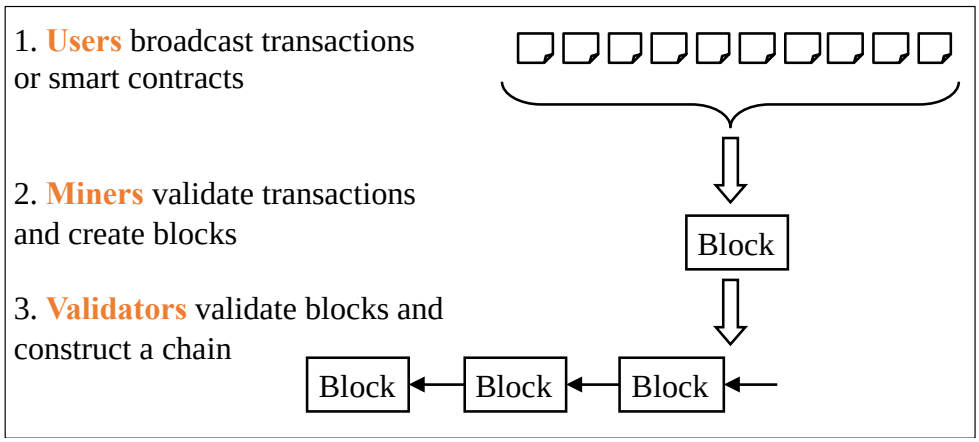


Figure 2.1 Blockchain Process

2.1.2. Main Characteristics of Blockchains

This novel architecture of blockchains yields many appealing characteristics, including distributed management, decentralized consensus, trustless partners, provable security, immutability, and

non-repudiation guarantees. Table 2.1 briefly summarizes these properties and the problems they try to solve.

Table 2.1: Key characteristics of blockchains

Characteristic	Problem to be solved	Blockchains' solution
Distributed Nature	Current applications are distributed by nature, therefore, they require distributed control and security mechanisms. Most of the existing practical security solutions are centralized and inefficient for these applications.	The blockchains are distributed by nature. Thus, blockchain-based solutions can be implemented in a distributed fashion
Decentralized Consensus	Centralized decisions by one controller can make the controller a single point of failure.	Blockchain decisions are achieved by decentralized consensus, majority votes, and nodes agreement.
Trustless System	Security provided by third parties can impose security and privacy risks if the party is compromised	Blockchain technology imposes a trust of majority, which is impenetrable to compromise unless attackers have control over the entire system.
Cryptographic Security	Algorithms for security should prove that they are challenging to break.	The blockchains use elliptic curve cryptography that is difficult to break. Further, the decentralized consensus make it even more challenging to break.
Non-repudiation Guarantee	Users can deny their interactions in the system	The blockchains use signatures of transactions and blocks in addition to permanent databases such that transactions cannot be denied later.

2.1.3. Consensus Algorithms in Blockchains

In blockchains, a miner creates blocks that will be attached to the database by what is refer to as consensus algorithms. In some blockchain applications, such as in Bitcoin, the miner who creates

the first valid block is rewarded. This reward is given by the system and is generally in terms of money for financial applications. Consensus is one of the critical concepts in blockchain technology. It allows nodes to create blocks that will be validated by others as well. If the new block is found as valid, it is attached to the blockchain database.

Several approaches exist to decide which miner wins, including proof of work (PoW) [20], proof of Stake (PoS) [21], Proof of Space (PoSpace) [22], proof of Importance [23], Measure of Trust (MoT) [24], minimum block hash [25], and Practical Byzantine Fault Tolerance (PBFT) [26]. For a reader with some blockchain knowledge, Table 2.2 summarizes the main difference between these algorithms and their use in current blockchains [27]. We summarize the major consensus approaches for a general computer science reader in the following.

- **Proof of Work:** PoW is the consensus algorithm used in Bitcoin and many other blockchain platforms. It requires the miners to solve a hard-mathematical puzzle that is changed frequently and has been agreed upon by all the miners. Once a node validates the transactions and solves the puzzle, the block is submitted to the blockchain network. Other miners (validators) validate the block to make sure that the submitter is not falsifying. Once it is agreed among the validators that the block is legit, it will be added to the blockchain, and the miner may be rewarded. The agreement here is based on a majority consensus. Thus, it is difficult to fake unless the attackers compromise more than 50 percent of the miners. The problem with this approach is that high computational power is wasted in solving the mathematical puzzle [20].

Table 2.2 Comparison of different consensus algorithms

Mining approach	Resources needed	Randomness	Implementations	Reward miner?
PoW	High computation power	No randomness	Bitcoin	Yes
PoS	Wealth or stake	Randomized blockchain selection	Ethereum	No
PoSpace	High memory	No randomness	Permacoin	Yes
PoI	Node significance	No randomness	NEM	Yes
MoT	Trustworthiness	No randomness	Not implemented	Yes (trust)
Minimum block hash	None	Randomized blockchain selection	Bitcoin extension	Yes
PBFT	None	No randomness	Hyperledger	No

- **Proof of Stake:** Unlike PoW, PoS does not require the miners to solve a computationally expensive mathematical puzzle. Instead, the miner is chosen in a pseudo-random way. The chance of a node being selected to create the new block depends on the node's wealth or "stake." In other words, the more money a node has, the higher chance it will have to mine a block. The native version of PoS does not award the miner; however, the extended versions award and punish the creators based on their performance. Selection based on the wealthiest account may result in a single account handling all the creations; hence, it may lead to an unfair distribution or even centralization. Therefore, a randomized node selection and a coin age-based selection have been proposed. In the coin age-based method, users who have not created any block for the past 30 days are considered for mining [21].

- **Proof of Space:** PoSpace is similar to PoW, except that the puzzle requires storage capabilities instead of computation. A miner proves its ability to create a new block by allocating the needed storage space to perform mining. In other words, instead of having a high computational capability, the miner needs to have a high storage capacity. Several theoretical and practical implementations of PoSpace have been released, however, the required high memory space is a challenge similar to the computation challenge of PoW [22].
- **Proof of Importance:** PoI is a consensus algorithm that calculates the significance of an individual node based on the transaction amount and the balance of that node. It assigns a priority with a hash calculation to the more significant nodes. Further, the node with the highest priority is chosen for the next block creation [23].
- **The measure of Trust:** Another way to perform mining is to use dynamic trust measurements and select the node with the highest trust level as the miner [24]. The trustworthiness is based on the nodes' behaviors; therefore, good behaving nodes that follow the protocols are rewarded. More specifically, the trustworthiness could be formulated as the expected value of the node's behavior in the future. Thus, the trustworthiness is approximated using the history of good and bad actions that the node has taken so far. The MoT approach could be subject to malicious attacks if a specific node plans to increase its trustworthiness for several iterations to attack the network later. Several mechanisms had been proposed to handle such attacks [24].
- **Minimum Block Hash:** In [25]. a consensus algorithm is proposed where the miner is chosen randomly and not based on its resources. The system selects the miners based on a generated minimum hash value across the entire network. Thus, the selection of the next miner is randomized, and the probability of selecting the same miner is low. This approach was

implemented on a modified Bitcoin network, and it was shown to offer energy savings for mining. However, it has not been adopted by the Bitcoin community.

- **Practical Byzantine Fault Tolerance:** Unlike others, PBFT [26] is a consensus approach that does not include any type of resources but utilizes the blockchain consensus based on the Byzantine fault tolerance approach. In this approach, first, a leader is selected and agreed upon among the nodes. The leader decides on the transactions' validation and publishes a block to all other nodes in the network. A transaction is committed to a new block only if two-thirds of the nodes verify its correctness. The leader changes frequently; therefore, the approach is not considered as centralized. PBFT is faster than other methods; however, it suffers from scalability issues due to the resulting communication overhead.

2.1.4. Permissioned and Permissionless Blockchains

Current blockchain systems are of two types: permissioned or permissionless. As the names indicate, nodes in permissioned blockchains, such as Quorum and Corda, need prior approval to join the network or interact with the system. On the contrary, any node with a valid public key (e.g., a proper wallet) can join the network in permissionless blockchains such as Bitcoin and Ethereum. Table 2.3 summarizes some of the major permissioned and permissionless open-source blockchain platforms along with their advantages and disadvantages.

Permissioned blockchains enjoy better transaction privacy as only authorized nodes can join the networks and hold the database. They also have better resiliency against known blockchain attacks, such as Sybil attacks, where an attacker joins the network with many pseudonymous identities [28]. Generally, they have better performance in terms of transactions' delay and scalability.

However, since permissioned blockchains regulate the nodes that can join the network, they suffer censorship and are generally regarded as less anonymous.

Table 2.3 Blockchain covered platforms with their types, consensus algorithm, advantages, and disadvantages

Platform	**Blockchain type**	**Consensus algorithm**	**Advantages**	**Disadvantages**
Bitcoin	Permissionless	PoW	Scalable in terms of number of nodes and users; Currently most popular	Computationally expensive; Time consuming
Etherum	Permissionless	PoW (current), PoS (planned)	More secure; scalable; for PoS, no extra computational power is required.	Requires stake or computations to be selected for mining; throughput is lower than other permissioned blockchains.
Fabric	Permissioned	Kafka [29]	No extra computations; expected high throughput; famous for industrial use.	Arguably not fully decentralized; Does not Support Solidity contracts (a common smart contract language).
Quorum	Permissioned	IBFT, Raft [30]	No extra computations; expected high throughput; multiple consensus algorithms offered; offers private transactions.	Scalability concerns; not suitable for cryptocurrency; higher network overhead.
Parity	Permissioned	Proof of Authority (PoA)	Expected high throughput; scalable.	Centralized assigning authorities; not resilient to malicious authorities.

All the blockchain extensions proposed in this work apply to both permissioned and permissionless blockchains. We will focus our discussion in the evaluation and the application part on

permissioned blockchains as the extensions will be implemented on top of Quorum, a permissioned blockchain platform.

It should be noted that when we refer to blockchains in the context of the dissertation, we mean both permissioned and permissionless. Whenever a distinction is necessary, we will state the type of blockchain, i.e., either permissioned blockchains or permissionless blockchains.

2.2. State-of-the-Art Security Risk Assessment Solutions

Security risk assessments have been an active area of research over the past two decades. The proposed solutions range from simple rule-based firewall and antivirus to sophisticated intelligent and adaptive AI-based solutions, all of which are used by industry and enhanced by academia. We focus here on AI-based approaches for malware and intrusion detection as examples of state-of-the-art security assessment solutions used throughout this dissertation. One can divide recent approaches in this area into fully centralized, distributed assessment but single party aggregation, hierarchal, and fully distributed. In the following, we briefly discuss each of these approaches and provide examples of earlier works in that direction.

2.2.1. Fully Centralized Approaches

These approaches utilize a single-party to assess the security of the system by one trusted party. A basic example of that is any intrusion detection system (IDS), where all network traffic would be forwarded to a single party that decides if a network file is malicious or not. The state-of-the-art techniques here are AI-based techniques that intelligently and adaptively assess the security risk and detect if a file or a network flow is malware or an intrusion.

Example works in this area for general networks, smart homes, and vehicular network systems are [31, 32, 33]. Our works in AI-based IDS have been applied to multi-cloud applications, industrial

IoT, and healthcare systems [11, 12, 10]. All these works forward network traffic to a single-point IDS model pre-trained on application-specific network data. This model detects if a network flow is malicious or not and reacts accordingly.

2.2.2. Distributed Assessment Single Party Aggregation Approaches

Distributed assessment single party aggregation approaches assess the systems' security by multiple agents, but a centralized aggregator makes the final decision. VirusTotal is a good example of as a malware detection system that combines results from various scanners and presents them to the client [34]. The agents involved can use their pre-determined rules, AI-based models, or any assessment strategy to determine if the file is malicious or not.

Most of works in this category use crowdsourcing models that distribute the tasks, where a trusted aggregator makes the final decision. For example, Yang et al. leveraged crowdsourcing for malicious user node detection in large-scale social networks [35]. Raff et al. proposed a crowdsourcing approach for malware detection from log files [13]. Burgura et al. proposed Crowdraid, an android malware detection system that utilizes crowdsourcing, for collecting malicious applications from several agents [36].

2.2.3. Hierarchal Approaches

Hierarchal approaches have assessments in a hierarchy where global decisions are made at the top of the hierarchy. Local decisions are forwarded up in the hierarchy whenever a decision is needed. In a wireless sensor setting, where the network is divided into clusters with cluster-heads, local cluster-heads have their IDS models and forward their predictions up to a global model. If an intrusion is detected at a local cluster head, then the cluster head will take simple actions such as modifying the route or dropping the flow. However, the global IDS is the party that decides on the

final decision regarding if a flow is malicious or not, and the whole system will take global action accordingly.

An example of such systems is the hierarchical-based IDS proposed in [37] for wireless sensor networks, as described above. Ahmim et al. proposed a multi-stage hierarchical prediction model for general networks [38]. In their proposed scheme, the local models classify flow as benign or intrusion while the global model classifies the type of the attack and takes actions accordingly. Similarly, Schueller et al. proposed a hierarchical IDS for general networks where the local models detect intrusions in the networks and forward the flow to the global model when an intrusion is detected [39].

2.2.4. Distributed Approaches

Distributed approaches disperse the assessment process and achieve global consensus without the need for centralization. These approaches divide assessors into clusters that reach local decisions first, while global decisions are achieved after some iterations to negotiate the final decision. Examples of such systems include those proposed for security decision-making in wireless sensor networks and IoT [40, 41].

2.3. Challenges with Discussed Risk Assessment Solutions

As discussed in Chapter 1, two of the significant challenges with current state-of-the-art security assessment is centralization and inefficient distribution. Here we detail these challenges for the techniques discussed in the previous subsection.

2.3.1. Centralization

A common challenge with fully centralized, distributed assessment single party aggregation, and hierarchal approaches is that they rely on a trusted controller to make decisions and act as a

monitor. These controllers have several drawbacks, including being a single point of failure, i.e., if this controller fails, the system fails. Furthermore, centralization makes security solutions inefficient due to the distributed nature of the original applications.

2.3.2. Inefficient Distribution

Distributed approaches are better than others in these systems. However, they mostly rely on local information and assume that agents are trustworthy and update their decisions to achieve a collaborative decision. In addition, many iterations may be involved to achieve a collaborative decision, which introduces delay and high communication overhead.

2.4. Blockchain-based Security Risk Assessment

Blockchains have several properties that can resolve the previously discussed challenges. In the following, we discuss how current blockchains can solve some of the challenges and highlight some of the existing work for blockchain-based security risk assessment.

2.4.1. Blockchains Help Resolve the Challenges

Blockchains do not have a single point of failure as the tracked assessments are duplicated in many blockchain nodes. They are not easy to hack, given that they are fully distributed and immutable. They are trustless in the sense that the blockchain nodes do not need to trust each other. Finally, the Bitcoin experience has shown that blockchains are scalable, at least in terms of the number of participants. Thus, blockchains can resolve the centralization or inefficient distribution challenges in addition to providing a fully-distributed and provably secure solutions [19].

2.4.2. Existing Blockchain-based Security Risk Assessment

Over the past three years, researchers investigated some attempts to use blockchains for risk assessment security applications. For example, the works in [42, 43, 44] discussed how the

blockchains could be used to build an intelligent IDS. The Telecooperation group studied blockchain-based IDS, where nodes exchange alerts depending on their role in the system [42].

The works in [45, 46] proposed to use blockchains for malware detections in general cloud malware and portable executable (PE) files. In addition, Gu et al. addressed the problem of detecting and eliminating malicious codes in malware files using blockchains [47]. We will further discuss the work for blockchain-based malware detections systems in Chapter 8.

It should be noted that these works use blockchains as a distributed database. The discussions are theoretical without practical consideration of how the alerts should be processed and how a final decision should be made.

2.5. Problems with Existing Blockchain-based Security Risk Assessment Solutions

In the pursuit of using the technology for security assessment applications, we have identified three main challenges, which significant parts of this dissertation will try to resolve by providing extensions to the technology. The fourth challenge is more related to AI-powered security assessment, that will underlie our blockchain-based solution. These challenges can be summarized as follows.

2.5.1. Storage without Knowledge

The first challenge, and the most important one, is that current blockchains are used as storage-based systems while security risk assessments (and decision making systems in general) need knowledge-based systems. Conventional blockchains communicate and store data being transmitted between two parties. For the works discussed in the previous subsection, the blockchain is used only to store assessments, but no collaborative decision or processing of these

assessments is included. This leaves it unclear how the stored information processing is done and how a joint assessment can be achieved.

Hence, the joint assessment is necessary to acquire knowledge or summaries from individual assessments in a fully decentralized fashion. Still, it cannot be achieved without a knowledge-based system that is able to process the stored information.

To efficiently support blockchain-based security risk assessment applications, there is a need to transfer blockchains from simple storage to processing and knowledge-based systems.

2.5.2. Reputation Scheme

A way to assess the goodness of an agent prediction or decision is necessary to provide a high-quality assessment. Agents should be assigned reputations based on their past performance. This reputation should be able to distinguish good agents from bad ones. Further, it should be an active part of the decision-making process when reaching the collaborative assessment. Bad agents should have low reputations so that they can be detected and marked as untrusted in the system.

Thus, a reputation scheme that assigns reputations to agents based on their performance is needed for blockchain-based decision-making and risk assessment applications.

2.5.3. Practical Use-case and Proper Analysis

As previous works are primarily theoretical, a practical use-case that specifically targets security risk assessment is needed to assess the blockchain usability for these applications. Requirements including secure and distributed assessment generation and low overhead of the system are necessary to be ensured.

An analysis of the system security that guarantees the security of the collaborative assessment with provable minimal overhead is required for the success of blockchain-based decision-making and security assessment solutions.

2.5.4. Proper Evaluation Metric

Existing security risk assessment solutions, both blockchain-based and non-blockchain-based, fail to efficiently evaluate the chosen approaches from the cost perspective. Most of these state-of-the-art works use ML and AI-empowered solutions for security risk detections. They base their evaluation on conventional evaluation metrics such as accuracy, false alarm rate, etc. However, to effectively represent a solution's cost in security applications, two requirements should be considered. First, as missing an attack (or even a risk) costs much more than missing a benign, a mistake made in detecting such an attack should not be treated as a mistake made in detecting a benign. Second, also from a cost perspective, correctly classifying a sample in a class (e.g., as an attack) should not be the same as a mistake made by misclassifying in the same class.

A new metric that satisfied the above two requirements should be introduced to evaluate both blockchains and non-blockchains security risk assessment solutions.

2.6. How This Dissertation Resolve These Challenges

We propose to advance existing blockchain-based risk assessments in the following ways: the introduction of knowledge-based blockchains, a novel reputation scheme, a new metric for security risk assessment, proper implementation and analysis of the overhead and security features, and a practical case study.

Our ***knowledge-based blockchains paradigm*** is designed to extend the blockchains from simple storage systems to processing and knowledge-based systems. Chapter 3 details this contribution much further and analyzes the proposed approach for simple intrusion detection.

A reputation scheme, called ***RPMC-EWA***, is introduced to favor competent behaving assessors and detect misbehaving ones. We discuss seven requirements for a good reputation scheme and propose a new algorithm in Chapter 4.

The first two contributions are then combined to introduce a novel blockchain-based crowd-forecasting system named ***CrowdFAB***. CrowdFAB is implemented on Quorum and analyzed both practically and theoretically in terms of overhead and security features. This is to be further discussed in Chapter 6.

The new cost-representative evaluation metric, named ***Safety score***, is developed to incorporate the cost associated with model predictions within its evaluation. We show how conventional ML-based evaluation metrics fail to represent the cost and propose the new metric in Chapter 7.

Finally, the ***blockchain-based malware detection*** use case presented in this work is still unique of a kind as no earlier work has targeted blockchain-based malicious mobile application detection. The details will be discussed in Chapter 8. We also discuss other applications of the proposed solutions that go even beyond this dissertation in Chapter 9.

Chapter 3: Knowledge-based Blockchains

The previous chapter has discussed several challenges in state-of-the-art blockchain-based security assessment solutions. This chapter discusses the work we have carried to resolve the first challenge, i.e., the storage without knowledge challenge, and transform blockchains from storage-based systems to knowledge-based systems. This part of the research introduces a new generation of blockchain applications dedicated to intelligent, collaborative, and distributed decision-making and risk assessment solutions. Most of the discussion in this chapter is on the extension stage of the research, thus, covering general decision-making and general risk assessment applications. Security risk assessments are a sub-field of these applications with the exact requirements.

3.1. Introduction and Related Works

In collaborative decision-making or risk assessment systems, multiple agents assess an event, and one summarized solution should be achieved. Applications here can be thought of as intelligent forecasting applications, where various agents respond with predictions about the possible future outcome of an event. These responses are combined to collaboratively assess the events and take actions accordingly.

An example application that falls under this category is stock market prediction, where multiple agents predict whether the stock would rise or fall and the value by which the market will change. The individual prediction can be based on historical data, recent political news, or the predictor's geographic location. In any case, different agents give their predictions, and the system achieves

its collaborative prediction about the market. This prediction helps inquiry nodes make knowledgeable decisions about buying, selling, or holding their shares in that specific market.

For the rest of this dissertation, we refer to these agents, assessors, or predictors as **"agents".**

Decisions, predictions, or assessments will be referred to as **"individual decisions"** or simply **"decisions**".

The collaborative decision or assessment will be called "**knowledge summary"**.

Risk assessment and decision-making in multi-trust domain applications can be considered as another example of these systems. Collaborative decisions need to be made continuously about resource distribution, security, and several other application aspects. For instance, in content sharing in a vehicular network setting, evaluating whether a base-station or a car is trusted to share the content with can be seen as distributed decision-making applications. Several agents share their individual decisions on the trustworthiness of the base-station or the car. These agents give a differing opinion based on different evaluation algorithms that may utilize machine learning, predetermined rules, or any other assessment strategy. In any case, the system should return a collaborative trustworthiness value to the requester and recommend whether to share the content with the evaluated base station or car. Thus, a knowledge summary made from the submitted decisions is essential even when the agents have differing "decisions" for a single object's trustworthiness.

To achieve this knowledge summary, current solutions use semi-distributed decision-making platforms such as Adaptive Decision Making Broker (ADMB) [48]. These platforms are geographically distributed and managed. However, critical decisions are made at a centralized global controller (also known as a broker) trusted by all system entities. As emphasized earlier,

the centralized controller introduces a single point of failure, which is vulnerable to many security risks.

Other researchers have proposed fully distributed architectures to resolve the problem. For example, Han et al. introduce a peer-to-peer recommendation system where decisions are made without any centralized controller [49]. While their method does not require a centralized controller, such solutions are generally insecure against malicious nodes. False information/decisions could propagate quickly in the system, and the knowledge summary can be manipulated.

As discussed earlier, blockchains have the potential to efficiently and securely resolve the centralization challenge. Thus, it comes as no surprise that earlier researchers approached this problem using blockchain-based solutions. The works discussed earlier in Chapter 2 are examples of applying blockchains to security risk assessments and decision-making. Other works have targeted content sharing in vehicular network settings using blockchains. For instance, Kang et al. exploited blockchains and smart contracts along with proper reputation schemes to resolve the trust, security, and privacy problem in data storage and sharing in vehicular edge networks [50]. Lu et al. proposed a blockchain-based architecture for data sharing and an AI-based algorithm to select with which node to share the content [51].

Blockchain-based voting systems constitute another research area that is closely related to decision-making applications. Such systems are typically multi-trust decision-making applications, as voters, candidates, and vote casters do not necessarily trust each other. Blockchain technology can help track the votes and possibly cast them at the end of the process. McCorry et al. discussed the implementation of their proposed blockchain-based voting approach in the

Ethereum blockchain platform and evaluate the system by the time and cost analysis [52]. Further, there exist many trials that have applied blockchain-based voting systems to elections. For example, in November 2018, Thailand held a primary election to elect a new political party leader using ZCoin, an extended Ethereum blockchain platform that provides additional privacy.

3.2. Motivation and Decision-Making Requirements

While the previously discussed blockchain-based works have the potential to resolve the decision-making problems, they still use the blockchain as a storage system. Thus, the challenge of storage rather than knowledge still exists, and the processing of decisions to achieve the knowledge summary is unclear and probably not practical.

To achieve a proper solution, the system needs to meet two requirements. First, as decision-making systems have two types of interactions, namely, requests and the agents' decisions, there should be two different types of transactions in blockchain systems. More importantly, there should be a way in which the individual decisions can be processed (i.e., summarized) when forming the blocks, and the knowledge summary about a request should be made within the blockchain process. In the following, we detail these two requirements that motivate us to propose an extension for blockchain systems.

3.2.1. Transactions Types

As there are two types of interactions in any decision-making system, requests and responses, transactions for these applications should be of two kinds: **request transactions and response transactions**. Request transactions include the event to be assessed (i.e., the request); response transactions should have the agents' individual decisions.

Storing decisions or risks in response transactions can be challenging with the current blockchain platforms. For most risk assessment and decision-making applications, the decisions or risks are probabilistic. For example, a base station is 50% trusted or there is a 70% chance that a stock is rising tomorrow. However, current blockchain systems store deterministic information, e.g., Alice sent x coins to Bob, or Alice signed a contract with Bob to pay 10 coins for 1 kg of x. Earlier works handled this challenge by storing only the decision without probabilities, e.g., a base station is trusted. However, this results in a loss of information, especially when making the collaborative knowledge summary. Thus, there is a need to store probabilistic information within the blockchain transaction.

While both of the previous two challenges, i.e., the two types of transactions and the probabilistic information storage, can be solved using smart contracts, we believe that smart contracts make the decisions challenging to access and process later on when making the knowledge summary. Thus, introducing two types of transactions and resolving the probabilistic information challenge at a transaction level are better approaches compared to smart contracts usage. However, as smart contracts can be eventually accessed, this is a minimal challenge compared to the knowledge summary requirement discussed next.

3.2.2. Achieving the Knowledge Summary

Current blockchains only store information. The processing of transaction data when the blocks are created is yet to be offered using the technology. In order to achieve the knowledge summary required for decision-making applications, the requester needs to do the processing offline whenever a decision is required. To do so, the requester will need to query/read all response transactions that have been submitted for the targeted request. This requires going back to the chain

and querying all blocks, which is tremendously inefficient and computationally expensive, especially when a large number of decisions have been made. These issues limit the practicality of the earlier solutions and motivate us to propose an alternative approach.

The idea of knowledge-based blockchains is to process the response transactions and calculate the knowledge-summary when generating the blocks. That is, as response transactions are added to the chain, the miner should summarize these transactions, calculate the knowledge summary, and add it to the new block header. By doing so, we will require less computation as the summary needs only the currently available transactions and the headers of earlier blocks. Further, the processing can be done online when the decision is made, resulting in a faster and more efficient solution.

3.3. Contributions

This part of the research will make the following contributions:

1. Knowledge-based blockchains concept is first introduced as a paradigm that transforms blockchains from simple storage systems to processing and decision-making systems, thereby resolving the storage without knowledge challenge.
2. The security analysis of knowledge-based blockchains are then provided, proving that the system is resilient against malicious agents and protected against several attacks that may target traditional decision-making systems.
3. The performance of knowledge-based blockchains is analyzed for a simplified intrusion detection system use-case as a security risk assessment application. The objective of these analyses is to show that the collaborative security risk assessment approaches provided by the

knowledge-based blockchains perform better than any singular agent that participates in the decision.

The work presented in the rest of this chapter has been published in [14] and patented in [15].

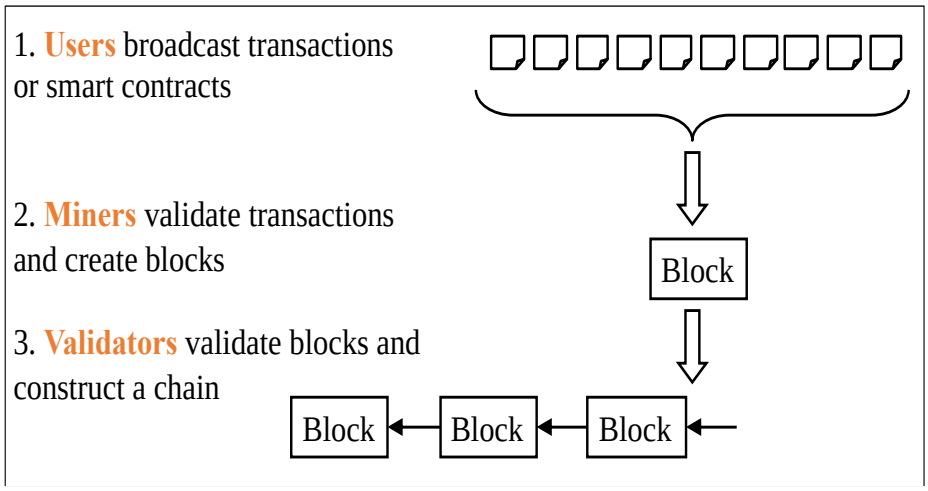
3.4. Proposed Knowledge-based Blockchains

This subsection provides an overview and the detailed specification of knowledge-based blockchains that extend blockchains to reflect on the above requirements for the decision-making systems.

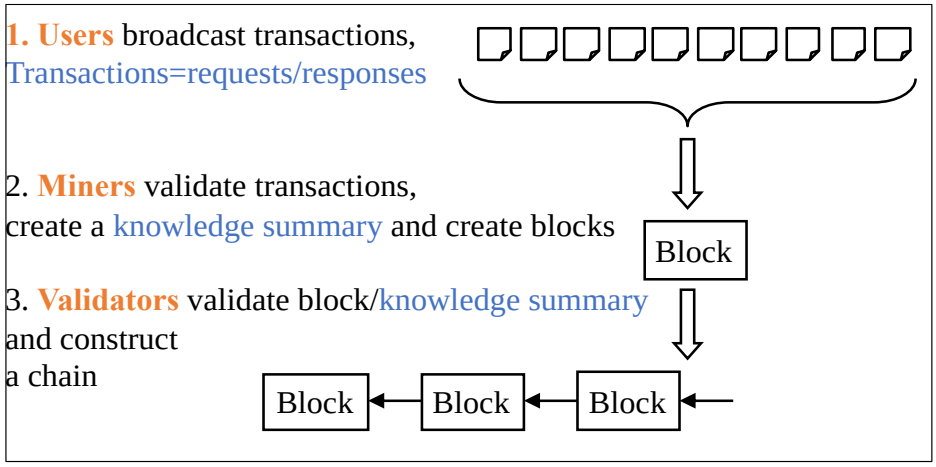
3.4.1. Overview of the Solution

As interpreted from previous discussions, the proposed knowledge-based blockchains paradigm is simple yet has the potential to transform blockchains from storage engines to knowledge-based engines. In the following, we summarize the main processes of knowledge-based blockchains compared to the blockchain processes (see Figure 3.1).

1. The users in knowledge-based blockchains are allowed to broadcast the two types of transactions: ***request transactions*** and ***response transactions***. Request transactions are conventional transactions that include user requests. Response transactions include decisions along with specific probabilities that reflect the risk involved in that decision. Thus, response transactions are extended conventional transactions that need additional variables to reflect decision probabilities/uncertainties.



a. Traditional blockchains process



b. Knowledge-based blockchains process

Figure 3.1: A comparison between traditional blockchains and knowledge-based blockchains processes

2. The miners validate the transactions by verifying signatures as in the traditional blockchains. More importantly, they combine several related responses into a knowledge summary and include this summary in the block header. For example, a knowledge summary can be the mode of several response transactions that state if a specific file has malware or not. This mode, along with the modes of responses for other files, is the knowledge summary of the block, which will be stored in the block header. In this way,

the knowledge summary is generated within the blockchain process (i.e., the blockchain makes the "knowledge" required for forecasting applications out of stored forecasts) and can be accessed by other nodes in the system.

3. The validators validate the blocks and construct the chain. They also verify that the generated knowledge summary is correctly summarizing the included response transactions.

In knowledge-based blockchains, *the role of users has been replaced by two new roles: Requesters and Agents*. This is done just to distinguish requests from responses. However, in terms of nodes' involvement in a blockchain system, requesters and agents are the same as users. Any node with a valid public key can be a requester or an agent in permissionless blockchains, while the roles can be prespecified in permissioned blockchains.

For the rest of this dissertation, unless otherwise specified, we refer to various blockchain nodes as: ***requesters, agents, miners, validators***. Note that these are roles, and a node can serve multiple roles.

3.4.2. Proposed Metrics for Knowledge-based Blockchains

To achieve the above knowledge-based blockchains process, response transactions, blocks, and block generation should be modified to reflect extensions.

Response transactions are introduced and have a decision variable reflecting the agent's local decision about a particular event. This variable can be probabilistic, where an agent returns a probabilistic decision value between [0,1], or deterministic, where an agent returns 0 or 1 with confidence. Different agents may inspect the same event and may produce differing decisions. For example, in an intrusion detection application, various agents decide if a flow with certain features

is malicious or not and these decisions/responses can be probabilistic. These are to be stored in response transactions.

Then, miners form a block summarizing the decisions from multiple response transactions and create a knowledge summary about the inspected event. They would summarize agents' decisions for each event that was included in the block. This summary represents an interpretation of the system about that event. Hence, this is the current knowledge summary of the system for that event and can be formulated as follows:

$$Summary(event_i) = G(P_j(event_i))$$

Where $event_i$ is the inspected event, j is a participating agent, $P_j(event_i))$ is an individual probabilistic decision made by agent j, and G is a function that summarizes all the decisions made by agents for the targeted event. We refer to G as **"the summary function."**

We allow G to be an appropriate function with the following properties:

1- G should ***not be easy to manipulate*** such that one (or just a few) agents can change the knowledge summary interpretation. Examples of a lousy G would be the "minimum" or "maximum" function of all assessments involved. In such a case, a malicious node can use a 0 or 1 decision to manipulate the consensus. This requirement secures the system against bad summary function attacks.

2- G should be **incremental**. It should be possible for miners to compute a new knowledge summary given the previous ones and new responses. This helps in the assessments for the same event, which may arrive at different times, and therefore, will be cast in different blocks. In this case, when forming the new knowledge summary, G should use the

summary in a previous block (the last block that had this event) and the recent response transactions only. In other words, a miner does need to reread the whole chain to calculate the new summary. An example of a lousy *G* is the standard "mode" function, as it requires all assessments to be available each time the consensus is calculated.

We allow any function that satisfies these conditions to be used as a summary function. It can be as simple as taking the mean or the mode, or as complex as a moving average, a weighted average, or the result of a sophisticated machine learning algorithm. A more elegant summary can also be taken from social science literature, e.g., the social choice theory [53].

Further, the summary function may return a vector instead of a single value knowledge summary. For example, one may summarize using a vector of mean, standard deviation, confidence interval, skewness, kurtosis, and higher-order moments and other probability measures as its elements [54]. It can also include the number of agents that participated in achieving the summary.

As a matter of fact, the summary function is application dependent which calls for flexibility in choosing a function appropriate for the application. Later on, in Chapter 9 and toward the end of this dissertation, we will put forward some recommendations on the summary functions to use and verify these recommendations with several examples.

In this way, all transactions, blocks, and block generation have been modified to reflect the needs for decision-making systems and the process of knowledge-based blockchains. Further, the knowledge summary of events can be presented in each block for fast recovery of any prior decision required.

3.4.3. Chain Design in Knowledge-based Blockchains

To simplify the knowledge-based blockchain design, consider the malware detection example again, as illustrated in Figure 3.2. A request transaction consists of the file to be inspected and some features that can help the agents make their decisions whether the file is malicious on not. When an agent receives a file, it should be inspected, and a response transaction will be generated. Response transactions, which are the main difference here, consists of the *file ID* to be inspected, the *agent ID*, and the agent's confidence in the file is malicious.

Blocks are formed of transactions and include the knowledge summary of the decisions received until that block. As can be expected, the knowledge summary is stored in the block header to allow other nodes to query it without going through the response transaction. Then, the chain is formed from multiple blocks, as is done in current blockchains.

It should be noted that multiple files can be inspected in the same block. The block may have summaries of zero, one, or more files, depending on the transactions available at that time.

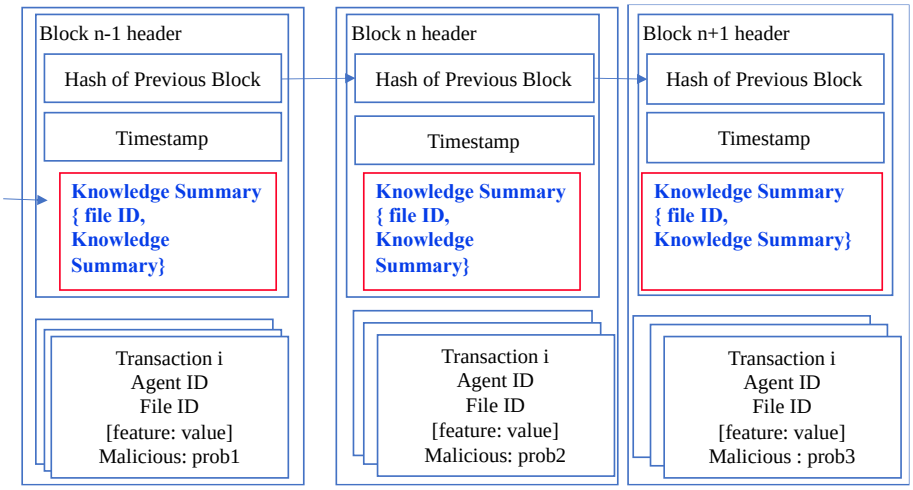


Figure 3.2: Knowledge-based blockchains architecture

3.4.4. Challenges in Achieving Knowledge Summary

One of the challenges in achieving the knowledge summary is that the transactions for the same event may arrive at different miners at different times. This results in these transactions being distributed over multiple blocks. Therefore, updating the knowledge summary and reaching a finalized summary are challenging.

To resolve the first challenge, i.e., updating the knowledge summary, we have made a condition on the summary function to be incremental. An incremental function allows for easy updates of the summary. Given the prior knowledge summary of an event and the new response transaction for the same event, the new knowledge summary can be easily computed using the summary of the latest transactions and previous knowledge summary. In addition, as knowledge summaries are stored in the block header, they can be easily accessed by miners whenever needed.

The second challenge, reaching a final knowledge summary, is harder to resolve as agents could be sending their transactions for a specific event at different times. For this work, we limit the number of blocks (i.e., limit the time) in which the agents should respond to an event, and the final summary has to be made after that. This can be achieved by allowing the requester to specify a deadline at which no future decision should be made, and hence the final knowledge summary is achieved. Future decisions for the same event are neglected after that deadline. If the requester does not provide the deadline, we limit the decisions to a certain number of blocks (e.g., 100 blocks) after the request has been made. In this case, the final knowledge summary is made after that number of blocks. For example, if the request transaction is committed in block 50 and assuming that this number of blocks limit is 100, all response transactions should be made between

block 50 and block 150. The final knowledge summary is made at block 150 or earlier. Again, any future transaction for the same request will be dropped by miners.

At a later point in this dissertation, we implement and verify knowledge-based blockchains on an existing blockchain platform. For this implementation, we have used the mean/weighted mean as an incremental summary function and we have limited the number of blocks to 50. In this way, both of these challenges have been resolved.

However, it should be noted that decision on how to resolve the challenge for real-world cases is application-dependent, but we have only verified that the above solutions work.

3.5. Security Analysis of Knowledge-Based Blockchains

Knowledge-based blockchains do not add to the security of traditional blockchains as they only extend their applications. At a later point in this dissertation, we will detail the security of the generated knowledge summary. However, the following will discuss the added security features that knowledge-based blockchains have compared to traditional decision-making solutions. These security features include resiliency to malicious agents, Distributed Denial of Service (DDoS) protection, and fraud mitigation.

3.5.1. Resiliency to Malicious or Bad Agents

Knowledge-based blockchains are resilient against malicious agents who try to take control over the knowledge summary. In a centralized solution, compromising the trusted centralized agent leads to the decision process failure. For example, in stock market predictions, if one centralized agent provides the predictions, then controlling that agent will control the decision. Similarly, in a semi-distributed management architecture, compromising the centralized controller or broker will

break the whole system. In contrast, with an appropriate summary function that satisfies the discussed properties, knowledge-based blockchains are hard to break and are resilient against malicious agents. Lemma 3.1 sketches the proof of this resiliency by an example.

Lemma 3.1: With a suitable summary function that satisfies the required properties, knowledge-based blockchains can be resilient against malicious agents that try to manipulate the knowledge summary such that the interpretation is changed. The constraint is that the number of malicious agents should be a minority here.

Note that manipulating the summary by a small fraction cannot be prevented. However, here we protect against drastically changing the knowledge summary such that the decision is flipped. The analysis here is based on a specific summary function (the arithmetic mean), while Chapter 6 will have a similar analysis for a generalized summary function.

Proof of Lemma 3.1: Consider the mean (first moment) of the individual agents' decisions as a summary function. That is

$$Summary(event_i) = \frac{1}{m}\sum P_j\ (event_i)$$

$event_i$ was inspected by an agent *j* which broadcasts a response transaction T_j as follows:

$$T_j\text{: \{ agentID: } j\text{, eventID: } i\text{, features: []}, P(event_i)\text{: 1\}}$$

Here, *m* is the number of response transactions included in the current block for the same event. Assuming that we have *m* agents forecasting the same event, response transactions $\{T_1, \ldots, T_m\}$ will be cast in the block, given that there are all seen by the miner.

As an example, consider the case in which all agents state that the event will happen with certainty and send a deterministic 1 as their decision. The miner will construct the block with $event_i$ summary as follows:

{EventID: i, #agents: m, mean:1, stdv: 0}

This indicates that the system concluded that $event_i$ will certainly happen with probability 1.

Now, let us assume that we have *n* malicious agents, where *n<m/2, e.g., n=0.2m*. This means that 20% of the agents are malicious and send malicious/wrong response transactions. That is, if agent g is malicious, the following transaction T_g will be sent:

T_g*: {agentID: g, eventID: i, features: [], P(*$event_i$*): 0}*

In the above transaction, we assume that $P(event_i)$ is manipulated to the fullest possible extent to have the most effect on the knowledge summary.

Now, the miner will have *m* agents participating in the knowledge summary: 80% of them are giving correct decisions "1" while 20% are giving wrong decisions "0". Consequently, the block summary:

{EventID: i, #agents: m, mean:0.8, stdv: non-zero value}

The *stdv* (standard deviation) here is dependent on how many agents are involved, but it is known to be a non-zero value.

This indicates that the blockchain has concluded that $event_i$ will happen with 80% probability instead of 100% in the no-adversary case. This is still a high probability value. Thus, the interpretation may not have changed if the application allows this level of uncertainty.

Assuming that the system has 1000 agents participating in the decision, this 20% will map to 200 faulty or malicious agents. It is difficult to compromise these many agents than compromising the one-party done in traditional centralized systems. ■

Again, the analysis in this chapter is proven by a simple example. A detailed proof will be provided in Chapter 6.

3.5.2. DDoS Protection

Knowledge-based blockchains can protect decision-making systems from DDoS attacks, which are considered the most threatening attacks nowadays [55]. In these attacks, a set of attackers targets the availability of the system by sending too many requests to be processed. In a centralized or semi-distributed architecture, this can be done by targeting the centralized controller.

However, in a blockchain design, the DDoS attack is made harder as there is no single point of vulnerability. DDoS attacks can cause the loss of billions of dollars in applications such as stock market predictions or other financial applications. Put another way, the protection provided by our approach can guarantee the availability of the system and prevent losses worth billions of dollars.

3.5.3. Fraud Mitigation

Knowledge-based blockchains can prevent fraudulent attacks, hence, they help applications keep the integrity and the correctness of current and prior information. A fraudulent attack tries to manipulate the individual decisions, that is, break the integrity of the decisions and prevent the

detection of such infringement. This is done by compromising and manipulating the data storage. Blockchains offer fraud mitigation guarantees since blocks and transactions in the chain are signed and replicated in many nodes. This leads to immutability in blockchain-based solutions, which makes it extremely difficult to alter or manipulate the decisions.

3.6. Performance Evaluation

To analyze knowledge-based blockchains compared to a single decision-making agent, we have performed an empirical study for an IDS application. In such an application, multiple IDS agents use a prespecified dataset to train ML models that evaluate if a specific network flow is malicious or not. Then the mean is used as a summary function to achieve a knowledge summary for the specific flow. This is compared to a singular IDS agent that has trained its models based on the same dataset.

In the following, we first discuss the application scenario and the summary function used. Then, we detail the experimental setup, the dataset used, the evaluation metrics, and the results from these evaluations. This serves as the first evaluation of the proposed knowledge-based blockchains for a security risk assessment application. Chapter 8 will present another application that is more usable and practical in a real-world setting.

3.6.1. Application Scenario and The Summary Function

An IDS system predicts if a network flow is malicious or benign and, in some cases, the certainty of such prediction. In a knowledge-based blockchains IDS scenario, a requester will request one or several flows to be inspected. This is to happen offline (or passively) after an attack has happened and the network administrator the source of the attack. The request will be broadcast in

request transactions where a transaction will have the flow ID, some features for agents to inspect the flow, and the traditional transaction parameters such as timestamp and hash-based signature.

Agents who receive such request transactions will evaluate the flow and make their own predictions on whether it is malicious or not. These agents will share the predictions in response transactions that are broadcast over the blockchain network, where a response transaction architecture follows the one discussed in Figure 3.2. For simplicity, we assume agents would make binary decisions as malicious flow ("1") or normal flow ("0").

We use the mean as a simplified summary function. Formally, given many returned responses in response transactions and considering that we have two possible choices per flow, e.g., malicious or normal, blocks will have a knowledge summary as follows:

$$G = P\ (Flow_i\ is\ malicious) = \frac{1}{m}\sum P_j\ (Flow_i\ is\ malicious)$$

Here m is the number of participating agents, $Flow_i$ is an inspected flow, and the j^{th} agent has submitted P_j as a decision on whether $Flow_i$ is malicious. P_j can be a result of a machine learning algorithm, a rule-based output, or a signature-based prediction. Note that P_j in this specific case is '0' or '1' but generally can be any probabilistic value can be used. This will be shown later in Chapter 8. Also, this is a simplified case that involves two decision options only but can be extended to any number of choices when applied to other applications.

3.6.2. Experiment Setup

To build knowledge-based blockchains, we simulated 1000 agents, one requester, five validators, and five miners to form the blockchain network. Agents make decisions that are submitted to the blockchain as response transactions. A positive decision ("1") indicates that the flow has been

classified as malicious (an attack is detected), while a negative decision ("0") implies normal traffic.

A mapping of the knowledge summary to malicious or normal flow is needed to evaluate the system. To do so, we use a simple mapping which states that a flow is malicious if more than 50% of the agents say so. That is, if the knowledge summary is more than 0.5, then the flow is declared malicious. Otherwise, the flow is normal.

3.6.3. UNSW-NB15 Dataset

We have used the UNSW-NB15 publicly available dataset, made possible by the University of New South Wales (UNSW) in 2015 [56]. The dataset is composed of nine types of attacks in addition to standard flows. The details of the dataset are out of the scope but can be read at [56, 57].

Since the target is just to test the knowledge-based blockchains concept, we decided to work with two types of attack detection, i.e., detecting DoS and Reconnaissance attacks. We excluded other attacks from the dataset and utilized only either DoS or reconnaissance and normal flows to build our models. Table 3.1 summarizes the statistics of our used dataset.

Table 3.1 UNSW dataset statics (used version)

Type of traffic	**Total number of samples**	**Percentage**
Benign	90191	86.58%
DoS attack	13982	13.42%
Reconnaissance	13987	13.42%

3.6.4. Evaluation Metrics

We consider accuracy, false alarm rate (FAR), undetection rate (UND) as evaluation metrics for this part. Later in Chapter 8, we will do the evaluation with more sophisticated algorithms.

However, our objective here is just to show the feasibility of the method and how knowledge-based blockchains (or collaborative decision making) perform compared to a single model.

Accuracy is the most frequently used metric for evaluation. It measures the degree of correctness of the predicted values to the overall number of samples. That is,

$$Accuracy\ \% = \frac{Correct\ Predictions}{Overall\ Samples} \times 100\%$$

FAR is another metric that measures the ratio of normal flows that are detected as malicious flows. That is, the ratio of false detected attacks which can be represented by:

$$FAR\ \% = \frac{FP}{FP+TN} \times 100\%$$

In contrast, UND is the metric that measures the ratio of malicious flows that are not detected by the model. That is:

$$UND\ \% = \frac{FN}{FN + TP} \times 100\%$$

Where false negatives (FN), false positives (FP), true negatives (TN), and true positives (TP). A high accuracy, low FAR, and low UND indicate better models.

3.6.5. Experimental Results

To test knowledge-based blockchains performance, two evaluation cases are used. In the first evaluation, a few machine learning models are trained with different algorithms and the same dataset. Namely, we use logistic regression (LR) [58], random forest (RF) [59], and decision tree (DT) [60] to build three learning models that detect only DoS attacks from the UNSW-NB15 dataset. The models are distributed randomly among the 1000 agents, who use them to decide the

received flows. The knowledge summary is made and is used to determine if the flow is malicious or not.

The results of the above evaluations are shown in Figure 3.3, Figure 3.4, and Figure 3.5, where the three ML models along with knowledge-based blockchains (KB in the figures) results are compared.

As shown in Figure 3.3., the RF model, the DT model, and knowledge-based blockchains have high accuracy, while the LR model has a relatively bad performance. Knowledge-based blockchain has the highest accuracy compared to other models, which indicates the feasibility of the proposed approach.

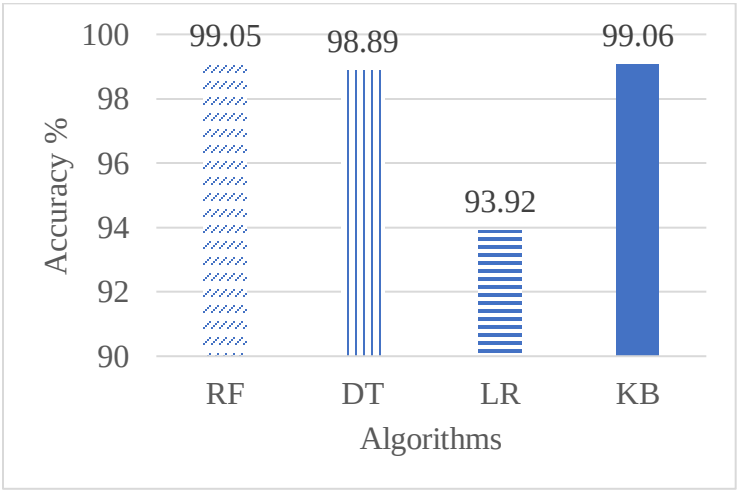


Figure 3.3: Accuracy results from the first experimental settings

However, accuracy is not enough to fully understand the model performance. This is specifically applied to the unbalanced dataset (such as the one used here), where the number of normal samples is much higher than the attack samples, and accuracy might be a misleading metric.

Thus, FAR is used after that, as shown in Figure 3.4. To recall, lower FAR values indicate better performance. Once again, knowledge-based blockchain outperforms others while RF and DT models perform relatively well. The LR model performs poorly compared to others. However, with the collaborative assessment in knowledge-based blockchains, the performance is not affected by these bad performing agents who used LR.

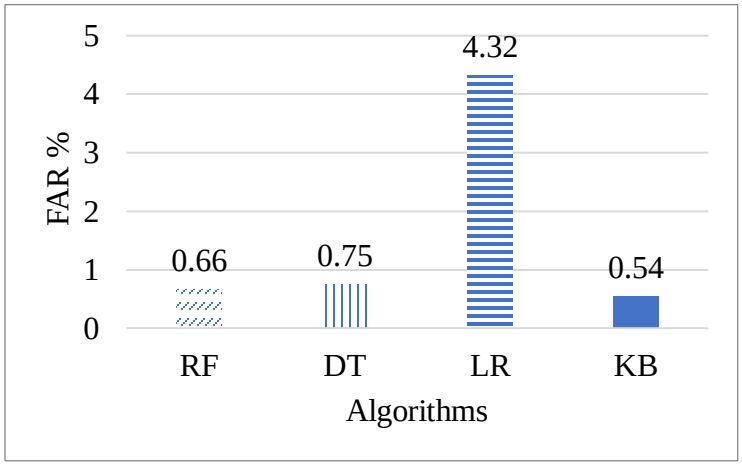


Figure 3.4: FAR results from the first experimental settings

For the UND rate, knowledge-based blockchains show comparable performance to the best performing machine learning models. As illustrated in Figure 3.5, the results show that knowledge-based blockchain had a slightly higher rate (i.e., slightly lower performance). This indicates that collaborative decision-making was not performing as good as the best in this specific case, but the performance is still very good. This degraded performance resulted from both RF, or DT, and LR making wrong predictions. In other words, if two-third of the clients wrongly predict a sample, then knowledge-based blockchains will predict poorly as well. LR performed much worse than others, and this shows that the proposed approach performs well and has the resiliency to poor predictors.

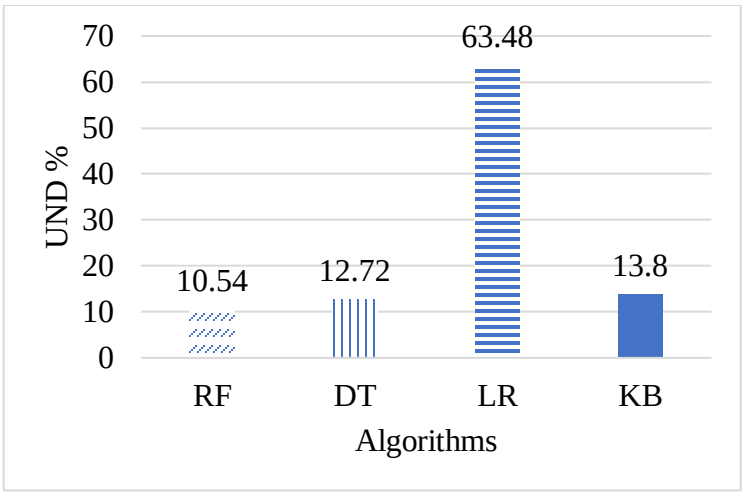


Figure 3.5 UND results from the first experimental settings

For the second evaluation, the training dataset is varied, and the same algorithm is used to build the learning models. Namely, we build models that use RF algorithm to detect DoS and Reconnaissance attacks. In this way, the training data differ for both models, while the test data will be the same. Like the first experiment, the two models have been distributed among the 1000 agents who predict the same testing sample flow. The blockchain achieves a knowledge summary about that flow.

To simplify, we will only present the accuracy results for this evaluation. As shown in Figure 3.6, the knowledge-based blockchains model performs the same as the DoS model, which does better than Reconnaissance (Recon the figure) model. This shows that knowledge-based blockchains will have a similar result to the best performing model even if only two models are involved.

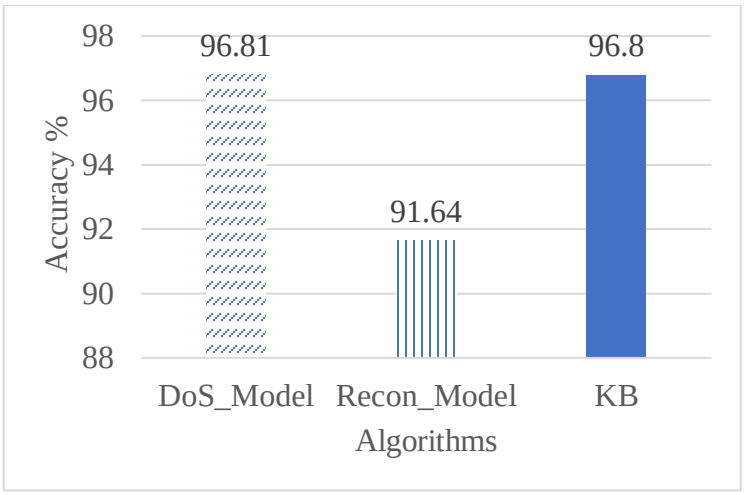


Figure 3.6:Accuracy results from the second experimental settings

These evaluations demonstrate the feasibility and performance of the proposed knowledge-based blockchains with simplified IDS evaluations. Results also show that the collaborative decision involved in knowledge-based blockchains has a performance that is comparable to the best performing machine-learning models used by agents.

It should be noted that with the summary function used here, knowledge-based blockchains will fail to give the correct prediction if the majority of agents (more than half) give wrong predictions. However, even with this drawback, the results are still good, considering the added security and distribution features provided by the proposed approach.

3.7. Summary

Blockchain technology has emerged as a secure, shared, distributed ledger technology with many promises for next-generation distributed applications. They have several potentials to be used for elegant decision-making systems, including security risk assessments in distributed applications. However, as emphasized in the earlier chapters, one of the most critical challenges with the

technology is storing information without making knowledge. An extension of the technology is needed to transform it from simple storage systems to processing and knowledge-based systems.

This chapter introduced knowledge-based blockchains as an extended blockchain paradigm that transforms the technology from storage systems to knowledge-based systems. We first discussed the need for knowledge-based blockchains and the motivation behind extending the current blockchain systems. Following that, we detailed the architecture and process needed for the proposed paradigm. We also analyzed the security of the proposed knowledge-based blockchains and how they are protected against malicious agents, DDoS attacks, and fraud attacks. We finally empirically evaluated the collaborative decision involved in knowledge-based blockchains in comparison with several single models for an IDS case study. Results, even though simplified, show comparable performance to the best performing model involved in the decision-making. This demonstrates the feasibility of knowledge-based blockchains in achieving accurate decisions.

Knowledge-based blockchains are still evolving and, when combined with other approaches, the concept is expected to be suitable for security risk assessment applications and generally decision-making systems. As will be further detailed starting from Chapter 6 and forward, knowledge-based blockchains have opened the road for a new generation of blockchains' applications that combine the technology with intelligent systems to achieve intelligent, collaborative, distributed, and hostile node resilient solution that can be used for efficient security assessment applications.

As a future work, one can testify knowledge-based blockchains for other applications and different summary functions. Later in this dissertation, specifically in Chapter 8 and Chapter 9, we will discuss some further applications and give recommendations on the proper summary function to use for different applications. Another future work can testify the concept on different type of

blockchains, including permissionless blockchains, and see how requirements would be different. Further, applying the knowledge concept to new distributed ledgers that does not include miners and blocks, e.g., IOTA system, is another future direction that one can take.

Chapter 4:A Reputation Scheme on Top of Knowledge-based Blockchains

As discussed in Chapter 2, state-of-the-art blockchain-based security assessment solutions have a few challenges, one of which is inefficient reputation schemes. In this chapter, we discuss the work we have carried out to resolve this challenge and put forward a reputation scheme that provides reputations to agents based on their performances. A scheme is initially developed to suit seven proposed requirements that should be meet in any reputation scheme. The proposed scheme is then evaluated and compared to a baseline using several adversarial strategies. Results show a sustainable performance, where the proposed scheme outperforms others in detecting malicious agents and achieving a better knowledge summary. Again, the work here is part of the extensions made to blockchains in the first stage of the research, thus, the discussion will target general decision-making and risk assessment applications. Security risk assessment can be considered as a subfield from those applications.

4.1. Introduction and Related Works

In any group decision-making, it is crucial that agents contributing to the collaborative decision, i.e., knowledge summary in the context of this dissertation, are distinguished based on their past performance. Some agents, who have had a good history of correct decisions in the past, should be given a higher weight. Others with a history of incorrect decisions may be given a low reputation. In other words, it is important to have a reputation scheme that assigns reputations to agents based on their performance.

These trust or reputation schemes currently play an important role in supporting distributed applications ranging from wireless sensor networks and e-commerce platforms to social networks. They provide a trustworthiness value (i.e., reputation) for participating sensors, users, or agents based on their past behaviors and interactions. A reputation can represent the accuracy of a sensor reading, the goodness of a product, the popularity of a user in social networks, etc. Besides, the reputation of an agent can define its contribution to the decision-making process and knowledge summary. For example, in a wireless sensor network setting, a sensor's reputation defines its contribution to an aggregated value sent to a base station. Such an aggregation process is similar to the generation of knowledge summary for decision-making applications targeted in this dissertation.

To calculate agents' trustworthiness or reputations, researchers use different functions ranging from a simple difference between the individual agent's value and the aggregated decision to more complicated functions. For example, Putra et al. used a Gompertz function to model the reputation's growth and calculate the new reputation value [61]. Similarly, Yu et al. utilized a sigmoid function to model the reputation based on past performance [62]. Iterative Filtering (IF) algorithms had also gained attention as an attractive option for trust and reputation functions [63, 64, 65, 66, 67, 68]. The main idea is to iteratively calculate the reputations as a discriminant function of several differences between the individual agent's decisions and the aggregated values or knowledge summaries. The discriminant function can be reciprocal, exponential, affine, or any other function [63, 68].

In general, IF algorithms showed better robustness to malicious or faulty agents compared to the simple averaging techniques. However, they require iterations, and several agents' decisions should be made at one time.

Another line of work worth mentioning is applying reputation schemes in the process of mining, i.e., within consensus algorithms, in blockchain systems. The main purpose here is to allow high reputation nodes to create blocks and gain rewards, replacing the originally resource-consuming choice of miners (i.e., the block creators). Gai et al. discussed a Proof of Reputation (PoR) consensus algorithm where the miner is chosen based on his reputation [69]. PoR is currently applied to the GoChain blockchain platform built on top of the Ethereum platform [70]. Another Proof of Reputations (PoRe) concept combined proof of work and proof of stake and had been applied to RPChain, a blockchain-based social network service [71].

Platform-wise implementation of such consensus algorithms included RepuCoin and RepChain. RepuCoin is a blockchain system that uses miners' reputation to define the mining power in the PoW algorithm [66]. RepChain is another blockchain reputation system that uses reputations to decide on the leader and validators in a Raft-like consensus algorithm [72]. Similar approaches have been applied in [73] and [74] with other consensus algorithms.

These blockchain works have all shown promising results in terms of security analysis and higher system throughput. Thus, reputation-based consensus algorithms can solve major transaction throughput and security issues found in other consensus algorithms.

4.2. Motivation

Earlier works in trust and reputation schemes in distributed decision-making systems put no requirements or a standardized reputation function/scheme to use when updating agents' reputations (or trustworthiness). Therefore, as could be seen, reputation functions ranged widely from simple average to sophisticated functions.

A set of reputation function requirements is needed, especially for risk assessment and decision-making applications targeted in this dissertation.

It should be noted that IF algorithms are considerably better than others in the literature and can be used as a standard. However, for this work, IF algorithms are challenging. First, *they require multiple single agent's decisions (for several events) to be present at the same time*. This requirement might not be realistic for risk assessment applications, as knowledge summary is made on-the-go and generally includes only one or two events at a time (i.e., one or two events would exist in a single block). To use IF algorithms for our decision-making purposes, a single decision should be considered in the discriminant function, which maps the algorithm back to a simple distance between the agent's decision and the knowledge summary. More importantly, *IF algorithms are not incremental and require all decisions to be present for the generation of knowledge summary*, which violates our earlier assumption on the summary function.

Above all, as knowledge-based blockchains concept is relatively new, the use of reputation in blockchain-supported decision-making applications has not been investigated. We are motivated to propose a reputation scheme to these systems for three reasons. First, we believe that a proper reputation scheme can *boost the assessment process* by giving the good agents higher contributions to the knowledge summary. In addition, agents can detect their wrong predictions and *enhance*

them using their assigned reputations. Finally, *malicious agents can be detected* and eliminated from the system if their reputations drop below a limit.

4.3. Contributions

This part of the research makes the following contributions:

1. A set of requirements that need to be meet for reputation schemes designed for decision-making systems is first propounded. These requirements include six general applications and one specific for security risk assessment applications. The objective is to define guidelines for other researchers on what is needed to be met in a reputation assignment function designed for decision-making applications.
2. A traditional reputation assignment function, namely exponential moving average (EWA), is extended to meet the discussed requirements. The function is set with configurable parameters such that it is configured to suit multiple domains with different requirements. This part formulates the proposed reputation function named as rated proportional multi-configurable EWA (RPMC-EWA)
3. The discussion of how the proposed scheme can be incorporated within blockchain systems and its use in knowledge-based blockchains discussed earlier in Chapter 3 is given.
4. The proposed scheme is analyzed to prove its efficiency compared to a baseline reputation assignment function commonly used in todays' decision-making platforms.

The work presented in this chapter has been published in [16].

4.4. Reputation Scheme Requirements

Some requirements are needed to be met to properly design a reputation scheme suitable for knowledge-based blockchains and risk assessment applications. In the following, we discuss six general application requirements and one security-specific requirement. These requirements will lay the foundation for the reputation scheme proposed next. Our basic premise is that the requirements should favor better-performing agents and provide fairness among similarly behaving agents. Further, they should be configurable to be used for different applications.

4.4.1. Continuity and Boundedness

A reputation function should be continuous such that it gives a different value for any change in an agent's performance. That is, for any correct or incorrect decision made by an agent, a suitable reputation value should be given. This requirement allows agents to have reputations at any point in the system. More importantly, it enables agents' reputation values to represent their performances at any time.

In addition, the function should be bounded between 0 and 1. That is, a good agent should eventually have a reputation of 1, while a malicious agent should have a reputation of 0. Such boundedness allows the reputation to be expressed as a percentage. For example, an agent with a 0.9 reputation is mostly, i.e., 90%, trusted while an agent with a reputation of 0.1 is not trusted, or only 10% trusted.

Boundedness helps to intuitively understand the reputation value based on the agent's performance. It also defines an expectation of how the agent will perform in the future. An agent with a 0.9 reputation is more likely to make correct decisions than an agent with a 0.1 reputation.

4.4.2. A Reputation Starts at 0.5

An agent that is just starting and has not contributed to the decision-making process should have a reputation of 0.5. This requirement makes the agent half trusted, giving it a 50% expectation to perform well in the future. It should be noted that this reputation value is lower than an excellent behaving agent but at the same time higher than a malicious or poorly performing agent.

4.4.3. Time-Dependency

The reputation function should be time-dependent, giving agents the chance to enhance their reputations as they make correct decisions. This feature indicates that an agent that has made wrong decisions recently should have a lower reputation than an agent that has made wrong decisions much earlier. Such a lower reputation is desirable as the second agent has enhanced its performance while the first has made it worse. Thus, the second is expected to perform better in the near future and should have a higher reputation.

As an example, consider two agents where the first agent made 98 correct decisions followed by 2 wrong decisions recently. The second agent made 2 wrong decisions followed by 98 correct decisions. The second agent has better recent performance and is expected to perform well. Intuitively, it should have a higher reputation.

4.4.4. Configurable Increase and Decrease

The increase in agents' reputations should be configurable to suit various application domains. Some applications, e.g., critical security assessment application decisions, require a slow increase in the reputations so that new agents are not trusted easily. Other applications, e.g., recommendation systems, require moderate or fast growth. Similarly, a decrease in reputation

should be configurable to suit different applications. Critical decision-making applications require a fast decrease to penalize misbehaving agents faster.

Further, it should be possible to have different increase and decrease factors to account for applications that require different configurations. For instance, critical applications require a slow increase with a rapid decrease.

As will be seen later, we use the reputation framework for malicious node detection, which requires a slow increase and a fast decrease. Nevertheless, it should be noted that configurable parameters are always desirable of any reputation scheme.

4.4.5. Slower Reputation Recovery

The increase in the reputation value after a wrong decision should be slower than a normal increase. That is, if two agents with a reputation of 0.5, where agent A is just starting and agent B had made several decisions (including bad ones), agent B should have a slower increase than agent A, which is just starting.

This stipulation is required to penalize malicious or bad behaving agents and prevent their fast reputation recovery. It also helps prevent attack strategies where a malicious agent gives wrong decisions randomly or using a specific pattern.

4.4.6. Proportional Update

When a node makes a wrong decision, its reputation decrease should be proportional to how good it has performed so far. To explain this, consider two agents, agent A and agent B, who have each made a wrong decision at the same time. Agent A previously made 1000 good decisions while agent B made only 100. This means that agent A made 1 in 1000 mistakes, and agent B made 1 in

100. Intuitively, agent A should have a higher reputation than agent B. The same thing applies to the increase in reputation recovery.

The proportional update feature helps to provide fairness and a higher reputation for the node that has been in the system longer.

4.4.7. Security Configurable Damage

In addition to the previously mentioned considerations, security risk assessment applications have one additional requirement. Namely, not all errors are equally damaging. Missing a security threat or a risk is much more damaging than missing a normal behavior. For example, in a malware detection system, missing a malicious file by classifying it as benign (false negative) is much more harmful than classifying a benign file as malicious (false positive). Similarly, any mistake made should penalize the agent more than a correct prediction.

Therefore, a false negative decision should penalize an agent's reputation more than a false positive decision. Further, a false positive should penalize the reputation more than any correct classification.

4.5. Proposed Reputation Scheme

The seven requirements discussed in the previous subsection should be reflected in any reputation scheme used. In the following, we propose a reputation formulation and prove that the proposed formulation meets the requirements.

4.5.1. EWA as a Reputation Function

The proposed reputation function starts with an exponentially weighted average (EWA) that is commonly used to calculate agents' reputations. Formally, $R_{i,t}$, the reputation of agent *i* at time *t*, may be calculated as follows:

$$R_{i,t} = \begin{cases} 0.5 & t = 0 \\ \alpha X + (1-\alpha) R_{i,t-1} & t \neq 0 \end{cases}$$

Where α is a configuration parameter between *0* and *1* and *X* is a value that is generally either *-1* or *1*, deciding whether to decrease or increase the new reputation. *X* can also be a fractional value defining the difference between an agent decision and the knowledge summary or how good the agent has been performing so far. For better notation, we will use R_t to represent $R_{i,t}$.

The parameter α affects the weight given to the agent's old performance. As α decreases, more weight is given to the prior performance, thus, the change in R_t value is minimal. Thus, for a function to have a slow increase or decrease, α should be small (generally less than 0.001).

4.5.2. Extensions to Meeting the Requirements

The EWA formulation above satisfies continuity, 0.5 reputation start, time dependency, and slow recovery reputation function requirements discussed before. We will prove these features later on in this chapter. However, we first need to extend EWA to meet the other requirements, namely boundedness (between 0 and 1), different configurable increase and decrease, proportional decrease properties, and security configurable damage. In the following, we discuss how to extend the function to meet all requirements.

Boundedness: The EWA formulation is [-1, 1] bounded while we require [0, 1] bound to represent a percentage. To satisfy our bounds, we need to change the EWA output from [-1, 1] to [0, 1]. To do so, the new EWA formulation can be extended as follows:

$$R_t = \frac{\begin{cases} 0 & t = 0 \\ \alpha X + (1-\alpha)(2R_{t-1} - 1) & t \neq 0 \end{cases}}{2} + 0.5$$

Note that *X* is generally either -1 or 1, and α < 1. It can be seen that with this translation, if R_{t-1} is in the interval [0, 1], R_t is also in the same interval [0, 1]. Values of R_t for three values of R_{t-1} are shown in the table below:

	R_t	
R_{t-1}	**X=-1**	**X=+1**
0	0	α
0.5	0.5(1- α)	0.5(1+ α)
1	1- α	1

Configurable Increase and Decrease: The EWA formula has one configuration parameter α to account for both the increase and decrease. However, as discussed earlier, some applications require different increase and decrease configurable parameters. For example, loss due to a wrong decision may be much higher than gain due to the correct decision. To satisfy such cases, we consider α as an increase parameter and introduce another parameter β for the decrease. The new multi-configurable EWA formula can be as follows:

$$R_t = \frac{\begin{cases} 0 & t=0 \\ \alpha X + (1-\alpha)(2R_{t-1} - 1) & t>0, \text{ correct decision} \\ \beta(-X) + (1-\beta)(2R_{t-1} - 1) & t>0, \text{ wrong decision} \end{cases}}{2} + 0.5$$

Proportional Update: As *X* generally has a value of either 1 or -1, i.e., decrease or increase, the first term in EWA will be the same for all agents. That is, if two agents have the same reputation

before making a wrong decision, their reputations will be the same regardless of their prior performance. However, as discussed earlier, proportional decreases and increases provide higher fairness to agents who have participated longer in the system. To achieve this property, we set *X* to the proportion of good or bad behavior of the agent up to this point. Formally:

$$R_t = \frac{\begin{cases} 0 & p=0, n=0, t=0 \\ \alpha\frac{p}{p+n}+(1-\alpha)(2R_{t-1}-1) & t>0, \text{correct decision} \\ \beta\frac{-n}{p+n}+(1-\beta)(2R_{t-1}-1) & t>0, \text{wrong decision} \end{cases}}{2}+0.5 \qquad \text{Equation (1)}$$

Where *p* is the number of correct decisions and *n* is the number of wrong decisions before time *t*. Note that here we consider correct or incorrect decisions (i.e., ones or zeros). However, this can be easily extended to fractional *p* and *n*, where the fraction is the difference between the agent's decision and the knowledge summary as an example.

Security Configuration: Similar to configurable increase and decrease, we can introduce a third configurable parameter to account for different types of wrong decisions made by an agent. β can be used when a false positive is made while new γ can be used when a false negative is predicated. Thus, the reputation can be rewritten as follow:

$$R_t = \frac{\begin{cases} 0 & p=0, n=0, t=0 \\ \alpha\frac{p}{p+n}+(1-\alpha)(2R_{t-1}-1) & \text{correct response} \\ \beta\frac{-n}{p+n}+(1-\beta)(2R_{t-1}-1) & \text{False Posative} \\ \gamma\frac{-n}{p+n}+(1-\gamma)(2R_{t-1}-1) & \text{False Negative} \end{cases}}{2}+0.5 \qquad \text{Equation (2)}$$

We refer to the last reputation formula as the Rated Proportional Multi-Configurable EWA (RPMC-EWA) formula. As will be seen, we propose using RPMC-EWA in calculating the reputation for each agent.

4.5.3. RPMC-EWA Meets the Requirements

The previous subsubsection discussions proved boundedness, configurable increase and decrease, proportional update properties, and the configurable security feature of RPMC-EWA. Next, we show that RPMC-EWA meets other requirements discussed earlier.

RPMC-EWA is a *continuous function* as any decision made will result in a slightly different reputation value. It is bounded between 0 and 1, as discussed earlier. It *starts at 0.5* following the first condition in the reputation formula shown in Equation 2. It has *a configurable increase and decrease* since both α and β are configurable. It follows a *proportional decrease* as $\frac{p}{p+n}$ and $\frac{-n}{p+n}$ were introduced to meet this requirement. It is also *security configurable,* as was introduced toward the end of Section 4.2.2.

Further, RPMC-EWA is *time-dependent* as an inherited feature of EWA (will be shown in Lemma 4.1). It also provides a *slower reputation recovery* (will be shown in Lemma 4.2). To simplify, Both Lemma 4.1 and Lemma 4.2 will use Equation 1 for the proof, but this can be easily extended to Equation 2.

Lemma 4.1: **RPMC-EWA is a time-dependent requirement function.**

This property is inherited from the EWA formula [75]. To show this, we consider two agents, A and B. Agent A made a mistake at time *t* while agent B made a mistake at *t-1*. The reputation at

$t-2$ is the same for both agents. If RPMC-EWA is time-dependent, agent A should have a lower reputation than agent B.

$$R_{A,t} = \beta \frac{\text{-n}}{\text{p+n}} + (1\text{-}\beta)\, R_{A,t-1} = \beta \frac{\text{-n}}{\text{p+n}} + (1\text{-}\beta)\, (\alpha \frac{\text{p}}{\text{p+n}} + (1\text{-}\alpha) R_{A,t-2})$$

$$R_{B,t} = \alpha \frac{\text{p}}{\text{p+n}} + (1\text{-}\alpha)\, R_{B,t-1} = \alpha \frac{\text{p}}{\text{p+n}} - (1\text{-}\alpha)\, (\beta \frac{\text{-n}}{\text{p+n}} + (1\text{-}\beta) R_{B,t-2})$$

$\frac{-n}{p+n}$ is almost equal to 0, and we denote it by $-\delta$. Similarly, $\frac{p}{p+n}$ is almost equal to 1, and we denote by Δ. Therefore,

$$R_{A,t} = \beta(-\delta) + (1\text{-}\beta)\, R_{A,t-1} = \beta(-\delta) + (1\text{-}\beta)\, (\alpha(\Delta) + (1\text{-}\alpha) R_{A,t-2})$$

$$= -\beta\delta + \alpha\Delta + (1\text{-}\alpha) R_{A,t-2} - \beta\alpha\Delta - \beta(1\text{-}\alpha) R_{A,t-2}$$

$$= -\beta\delta + \alpha\Delta - \alpha\beta\Delta + [(1\text{-}\alpha) - \beta(1\text{-}\alpha)] R_{A,t-2}$$

$$= -\beta\delta + \alpha\Delta - \alpha\beta\Delta + [(1\text{-}\beta) - \alpha(1\text{-}\beta)] R_{A,t-2}$$

$$R_{B,t} = \alpha(\Delta) + (1\text{-}\alpha)\, R_{B,t-1} = \alpha(\Delta) + (1\text{-}\alpha)\, (\beta(-\delta) + (1\text{-}\beta) R_{B,t-2})$$

$$= \alpha\Delta - \beta\delta + (1\text{-}\beta) R_{B,t-2} + \alpha\beta\delta - \alpha(1\text{-}\beta) R_{B,t-2} = \alpha\Delta - \beta\delta + \alpha\beta\delta + [(1\text{-}\beta) - \alpha(1\text{-}\beta)] R_{B,t-2}$$

The last term is the same for both agents and,

$$R_{A,t} - R_{B,t} = -\beta\delta + \alpha\Delta - \alpha\beta\Delta - \alpha\Delta + \beta\delta - \alpha\beta\delta =$$

$$= -\alpha\beta\Delta - \alpha\beta\delta = -\alpha\beta(\Delta + \delta)$$

As none of the left-sided parameters is above 0, the reputation of agent A is lower than the reputation of agent B. Thus, the agent who made the most recent mistake will have a lower reputation.

***Lemma 4.2*: RPMC-EWA meets the slower reputation recovery requirement**

To prove this, we look at the value of the term $\frac{p}{p+n}$ in Equation 1. For an agent that has just started and has made a correct decision, *n* will be 0. Thus, the above term will be 1, which is the maximum increase achieved. For an old agent who has committed mistakes earlier, $n \neq 0$. Thus, the term will be less than 1. The second case will have a slower increase.

Further, as *n* increases, the recovery will be even slower. Figure 4.1 illustrates this with a case scenario where agent A has just started, agent B made 10% wrong decisions earlier, agent C made 15% wrong decisions earlier, and agent D made 25% wrong decisions earlier. All agents are making good decisions, and we set α=0.005 and β=0.3. As can be seen, agent D has the slowest increase given that it had the highest number of mistakes. On the other hand, agent A who has not made any mistake has the fastest increase.

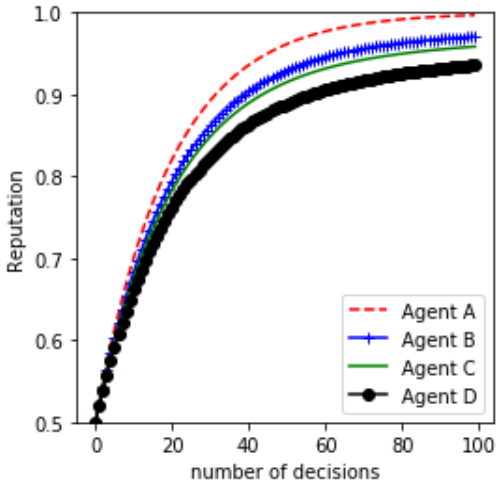


Figure 4.1 RPMC-EWA meets the slow reputation recovery requirement

4.6. RPMC-EWA in This Dissertation Research

RPMC-EWA can be used in this research to enhance the decision-making process, detect and eliminate malicious nodes, and act as feedback for agents to evaluate their performance. Further, with some extensions, it can support consensus algorithms, as was discussed in the earlier work. In the following, we briefly discuss each of these uses.

4.6.1. RPMC-EWA for Decision Making Process

In Chapter 3, we have used the simple arithmetic mean summary function to calculate the knowledge. Having introduced the concept of reputations in this chapter, a weighted mean summary function can be used with weights reflecting the agents' reputations. Such weights allow good agents to contribute more to the knowledge summary, given that these agents will have a higher probability of performing well in the future. In contrast, agents that had poor performance will contribute less to the knowledge summary. The following discussion will be further detailed later in Chapter 6 which combines knowledge-based blockchains with RPMC-EWA to propose a novel blockchain-based crowd forecasting model for risk assessment applications.

To achieve the above, RPMC-EWA can be used to calculate and validate the knowledge summary in knowledge-based blockchains. In the calculation process, the miner gets the last agents' reputations for all participating agents and calculates the new knowledge summary in the block. For validation, the validators also get these reputations for all agents and recalculate the proposed knowledge summary to ensure its correctness. The process guarantees that the miner is not controlling the decision-making process nor manipulating the reputations.

After the block is committed, all blockchain nodes (i.e., miners and validators) update agents' reputations for future blocks. The number of good and bad behavior for all agents must be stored at all nodes to calculate and update the reputation. However, it should be noted that these reputations are not stored in the chain and kept off-chain for block size limitation purposes. The details of this will be presented in Chapter 6.

4.6.2. RPMC-EWA Use for Malicious Node Detection

Another application of RPMC-EWA and reputation schemes, in general, is the malicious agents' detection in decision-making systems. This application is the focus of the performance analysis presented in this chapter. In such an application, agents with low reputations are considered malicious. As the starting agent has a reputation of 0.5, we consider an agent with a lower reputation as malicious or poorly performing. In some applications, such agents may be excluded from the decision process.

For a malicious node detection application, it is preferable to increase agents' reputations gradually as they make good decisions and decrease them rapidly as they make bad decisions. A slow increase is required so that newly joined agents are less favorable than agents with a long history

of good behavior. On the other hand, rapid decrease helps detect the malicious agents faster, as will be seen later in the analysis.

It should be noted that agents who commit only a few mistakes may also be detected as malicious. This is true in decision-making applications as these agents affect the summary. Also, if bad agents keep enhancing their performance, their reputations will eventually rise above 0.5. Hence, they will again be considered as non-malicious agents.

Before calculating the knowledge summary, miners should calculate the agents' reputations and exclude malicious agents from the decision process. However, if their transactions are valid, they will be added to the block. The addition acts as a record and allows all blockchain nodes to see other agents' performances. In addition, having these records helps prove agents' reputation recovery in case they enhance their performance.

4.6.3. Reputation as an Agent Feedback

Even though we have not considered this as part of this dissertation, agents' reputations can act as feedback on their prior decisions. Lower reputations indicate that the agent is making wrong decisions. Thus, agents can update model algorithms to perform better. Since higher reputation indicates a higher effect on the system's decisions, enhancing agent prediction would be desirable for agent popularity in the system.

4.6.4. RPMC-EWA Use for Mining

Reputation-based consensus algorithms have been shown to exhibit good performance in the earlier works. Thus, the proposed reputation framework can be used as a standalone consensus algorithm or combined with other techniques to achieve a lower computational requirement or a higher security guarantee. However, the notion of good and bad decisions has to be redefined to

be acceptable or unacceptable for mining. Further, a node that has mined recently should have a lower mining chance next time. In other words, the reputation framework, along with its requirements, should be modified before it can be used for mining.

4.7. Performance Evaluation

To analyze the performance of RPMC-EWA, we compare it to a baseline reputation function using several agents' attack strategies. The evaluation in this chapter focuses on malicious agent detection in knowledge-based blockchains, whereas Chapter 6 will discuss reputations as part of the summary function. All the analyses will be using Equation 1 rather than Equation 2 for this chapter. Chapter 6 and Chapter 8 will use Equation 2.

In the following, we start by discussing the baseline reputation algorithm used for comparison and the attack strategies to be used for evaluation. Then, we present two evaluation settings and their results to show the superior performance of RPMC-EWA compared to the baseline.

The objective of these evaluations is to show how fast RPMC-EWA can detect a malicious agent and the effect of eliminating such a malicious agent from the decision-making process.

4.7.1. Beta Reputation Algorithm

The beta reputation algorithm is one of the traditional and most widely adopted reputation formulation used in reputation systems. It is derived from Posteriori probabilities of binary events that can be represented by a beta distribution. The expected value of the beta distribution is given $\alpha/(\alpha+\beta)$ where α and β are shape parameters that are greater than 1 [76].

Given that $\alpha=p+1$ (i.e., correct decisions) and $\beta=n+1$ (i.e., wrong decisions), the resulting expected value for an agent's reputation follows the expected value of the beta distribution. That is, an agent's reputation at time *t* can be calculated as follows:

$$R_t = \frac{p+1}{p+n+2}$$

4.7.2. RPMC-EWA Algorithm Setting

For the malicious node detection application, we need a slow increase and rapid decrease setting. To meet this requirement, RPMC-EWA will use $\alpha=0.005$ and $\beta=0.3$, for the rest of the chapter, unless otherwise stated. As can be seen, we do not consider the security configurable parameter in this chapter and assume all mistakes are equally treated. The security configuration requirement will be considered in Chapter 6 and the analysis done in Chapter 8.

4.7.3. Attack Strategies

We assume that a normal agent is performing perfectly and making continuous correct decisions. If a node makes wrong decisions, it contributes to the decrease of its reputation. Thus, the assumption made above considers a worst-case scenario, where the attacker's target is the perfectly behaving node. In such a scenario, the agent's reputation is maximized as it has not made any mistake yet.

We followed four strategies for an attacker to use as follows:

1. **Continuous-flipping strategy:** This is the most straightforward strategy where a malicious agent continuously flips its decision. This strategy is easy to detect since the number of wrong decisions will be growing steadily without changes to the number of correct decisions. Thus, it is rarely followed in real life, but we discuss it as a base case scenario.

2. **Pattern-flipping-1 strategy:** In this strategy, the malicious or compromised agent will flip its decisions based on a predefined pattern. Such an adversarial setting results in an eventual increase in both correct and wrong decisions. Although it is harder to detect than the first strategy, it can also be detected by analyzing the agent's history. For this case, a malicious agent is flipping the decision every four decisions. That is, the malicious agent makes three correct decisions, followed by a wrong decision.
3. **Pattern-flipping-2 strategy**: This is similar to the first pattern flipping strategy, except that the agent makes ten correct decisions followed by one wrong.
4. **Random-flipping strategy:** In this strategy, a malicious agent randomly flips its decisions resulting in an eventual increase in both correct and wrong decisions. It is harder to detect than others, thus, followed mostly in real attack scenarios.

4.7.4. Malicious Node Detection Results

We compare RPMC-EWA with the beta reputation algorithm as a baseline reputation algorithm. An agent is detected as malicious if its reputation falls below 0.5, i.e., its starting reputation. We vary the number of correct decisions made before the agent turned malicious and evaluate whether the algorithm can detect the agent as malicious. The evaluation metrics used are the detection accuracy and the decision at which the agent is detected as malicious. The latter metric measures how fast the algorithm can detect malicious agents.

The results of the above experiments are shown in Table I. "Decision #" is the decision at which the agent was compromised, i.e., turned malicious, while "Detection decision #" is the decision at which the agent is detected as malicious.

As can be seen, RPMC-EWA outperforms beta reputation in all cases. Beta reputation can detect a malicious agent only when in continuous flipping strategy and some cases of random flipping strategy. On the other hand, RPMC-EWA performs well in all strategies except for the pattern-flipping-2 strategy. Furthermore, RPMC-EWA can detect a malicious agent much earlier than Beta in all cases.

Table 4.1 Results of RPMC-EWA and Beta algorithms performance for malicious node detection

Setting		**Beta reputation**		**RPMC-EWA reputation**	
Strategy	**Decision #**	**Accuracy %**	**Detection decision #**	**Accuracy %**	**Detection decision #**
Continuous flipping	1	100	3	100	2
	50	100	99	100	54
	100	100	199	100	106
	500	100	999	100	511
	1000	100	1999	100	1011
Random flipping	1	100	20 *	100	2 *
	50	10	92 *	100	59 *
	100	0	**	100	113 *
	500	0	**	100	525 *
	1000	0	**	100	1030 *
Pattern flipping 1	1	0	**	100	2
	50	0	**	100	72
	100	0	**	100	167
	500	0	**	100	586
	1000	0	**	100	1165
Pattern flipping 2	***	0	**	0	**

* an average value; **Cannot be calculated; *** For all numbers

Beta's bad performance is due to a small decrease in reputation when an agent makes any mistake. That is, an agent's reputation will be going up and down by a small amount, thus it never reaches 0.5 except if the agent continuously flips or makes wrong decisions. In contrast, RPMC-EWA with

rapid or even moderate decrease and slow increase can achieve 0.5 reputations faster as the decrease is much higher than increase.

Figure 4.2 illustrates the previous discussion by showing an agent's reputation changes in the case of random flipping. As can be seen, RPMC-EWA achieves the cutoff fast, while Beta did not even reach 0.9 reputation.

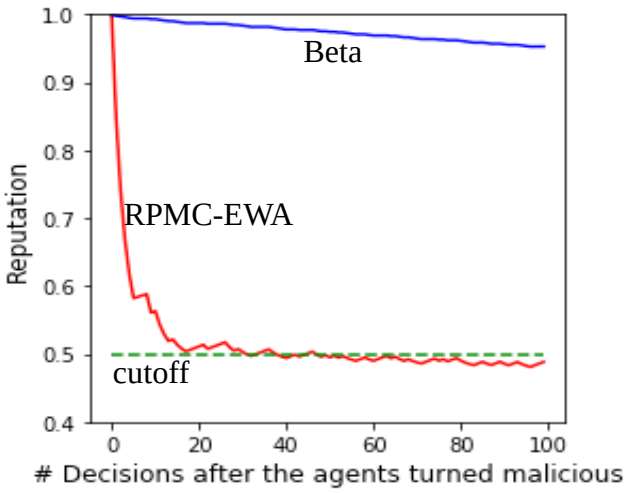


Figure 4.2 Reputation change in the case of random flipping strategy after 50 correct prior decisions.

4.7.5. Effect on Knowledge Summary

We also evaluate the effect of malicious node detection on the knowledge summary in knowledge-based blockchains. As the Beta algorithm did not perform well on malicious node detection, we excluded it in this experiment. We compare the knowledge summary value with and without malicious node detection.

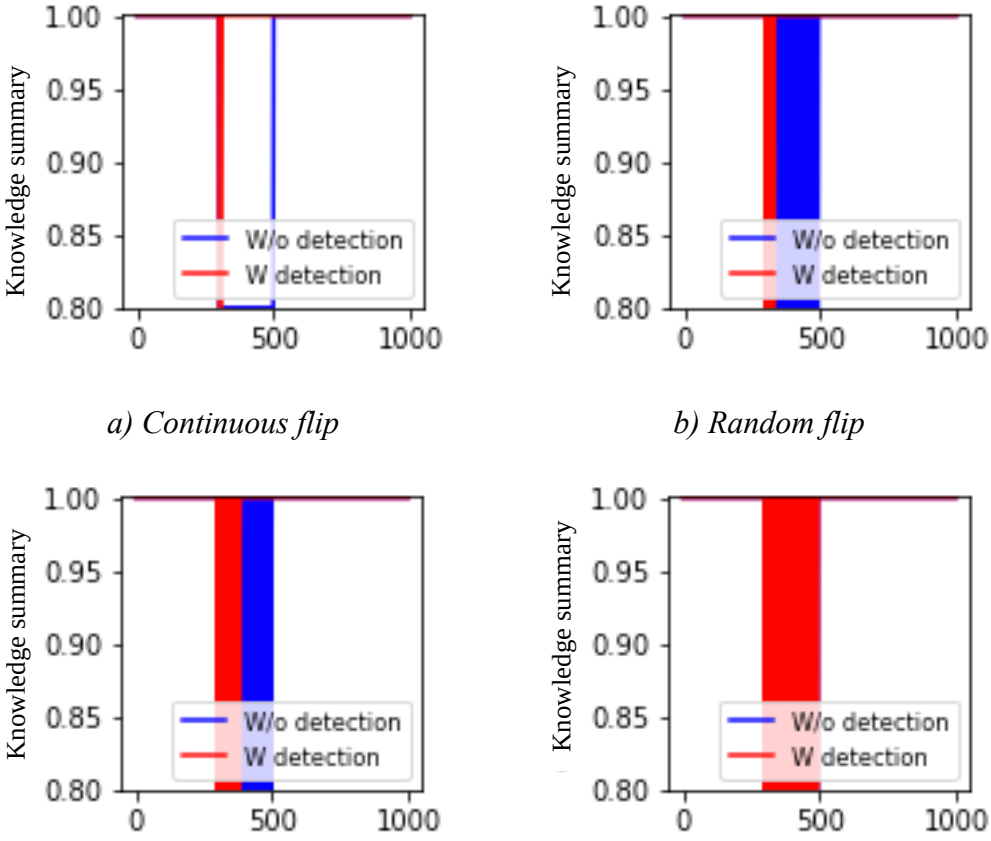
Chapter 3 has shown that knowledge-based blockchains are resilient against malicious nodes as soon as these nodes are less than half of the total number of agents. However, this resiliency applies to the summary interpretation rather than the probability of an event. For example, it was shown

that the summary decreased from 100% to 80% when 20% of the agents were malicious. Thus, even though the interpretation had not changed, there existed a loss of information with only a few malicious nodes. With malicious agents' detection, it is possible to recover such loss of information after the agent is detected as malicious and eliminated from the decision-making process.

Note that we are still using the arithmetic mean as a summary function. However, the miner will exclude the malicious agents from the decision process and knowledge summary calculation when using RPMC-EWA.

To show this discussion, we follow the same four strategies used in the first evaluation. Ten agents are participating with one malicious. Honest nodes give the same decisions with a probability of 1, while a malicious node flips the decision. Different probabilities may be used; however, we chose this case in the context of this evaluation. To harden the detection, an agent is turned malicious only for a limited time (Decision 300 to Decision 500). We evaluate the consensus with and without malicious node detection, as demonstrated in Figure 4.3. Besides, we show the reputation of both malicious and non-malicious agents in Figure 4.4.

As can be seen, the malicious agent was detected and excluded from the knowledge summary except for pattern-flipping-2. Considering a higher β facilitates the detection in the last case. In the case of continuous flip strategy, the node was detected right after it turned malicious. Whereas, in the case of the random flip pattern, the malicious node was detected after making around 25 decisions after turning malicious. For pattern-flipping-1, the malicious node is detected after approximately 50 decisions.

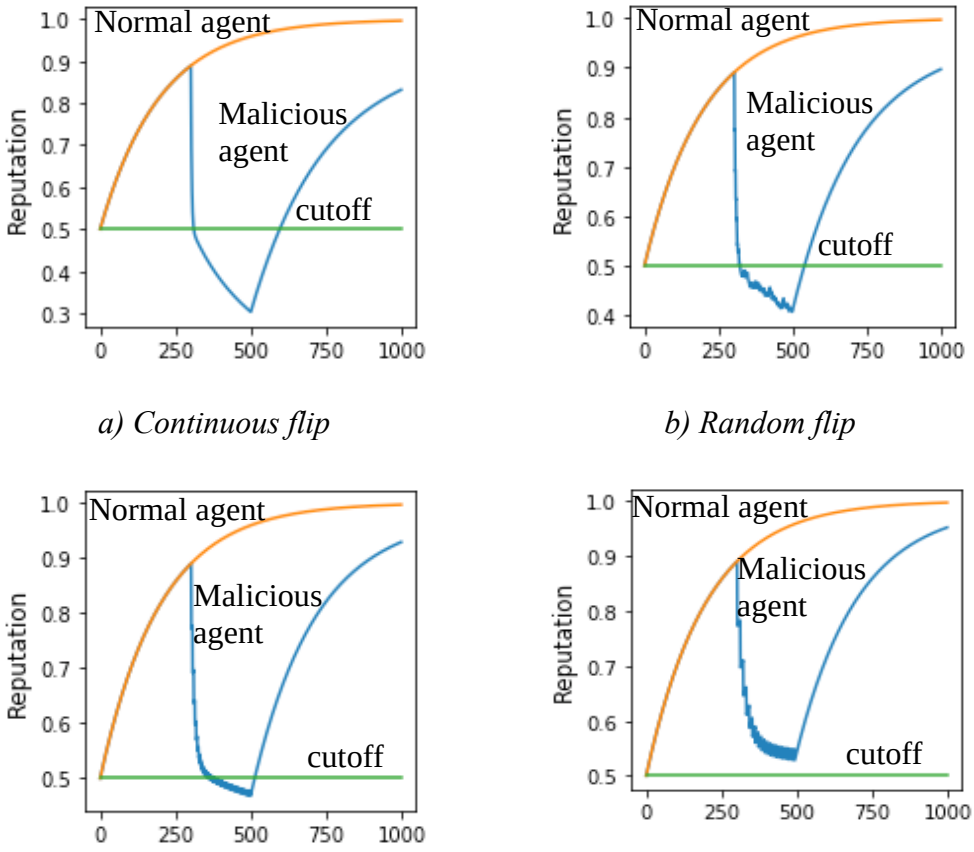


a) Continuous flip *b) Random flip*

c) Pattern flipping 1 *d) Pattern flipping 2*

Figure 4.3 knowledge summary with and without malicious node detection

To elaborate more on these results, we show the change in the agents' reputation for a normal and malicious agent in Figure 4.4. As can be seen, the agent's reputation dropped significantly in all cases as the agent turned malicious. The last case did not reach the cutoff, thus, the agent was not detected as malicious. The figure also shows that a malicious node can regain its reputation after again making correct decisions. After a maximum of 100 correct decisions, all cases show that the node has regained its reputation to above 0.5.



a) Continuous flip *b) Random flip*

c) Pattern flipping 1 *d) Pattern flipping 2*

Figure 4.4 The change in agents' reputation after it turns malicious

4.8. Summary

Trust and reputation systems play an important role in distributed decision-making systems where multiple agents are involved, and a knowledge summary should be achieved. To achieve a good summary in the context of this dissertation and encourage agents to participate, agents' past performance should define their contributions to the knowledge summary. Good agents have a higher contribution to the summary. However, defining a set of requirements that can be used to establish a good reputation scheme for distributed decision-making and risk assessment application is a challenge. Further, as deploying blockchains for these applications are relatively new, the use

of reputation in blockchain-supported decision-making applications has not been investigated. Thus, there is a need to establish a reputation scheme that provides reputations to agents based on their performance.

This chapter proposed a reputation scheme that bases reputations on agent's past performance and can be used for knowledge-based blockchains applications, especially the security risk assessment ones. The scheme was applied to malicious node detection where malicious agents are excluded from the blockchain decision-making process and knowledge summary. Results showed that the proposed approach outperforms the traditional beta reputation system in detecting malicious agents. Further, detecting and excluding malicious agents illustrated a better performance in achieving the knowledge summary in knowledge-based blockchains.

The analyses made in this chapter have several limitations to be handed in further works. First, all analyses are done off-chain as the reputation scheme is yet to be integrated with knowledge-based blockchains. The changes are only in the mining process, and the results are expected to hold for any blockchain. However, the reputation framework integration with a blockchain platform is needed to validate the results. The challenge is in tracking the reputations and making sure that a consensus is reached about their values. This is to be done later in Chapter 6. In addition, the framework computation, overhead, and security analysis are still to be done in Chapter 6.

Other future works outside the scope of this dissertation may include comparing the proposed reputation framework to existing blockchain-based reputation solutions that are also desirable for the proposed approach. Finally, applying the proposed approach to other applications and attack scenarios can add a value to the proposed approach for decision-making applications.

Chapter 5:Performance Analysis of Quorum

In the next chapter, we will present CrowdFAB as a crowd-forecasting framework that combines the contributions of the previous two chapters. The framework can be implemented on any blockchain platform. To validate our approach, we will be implementing the framework on Quorum as an easy-to-use permissioned blockchain platform. However, before doing so, an analysis of Quorum's performance is necessary to verify its suitability for our application domain. Thus, in this chapter, we present the work we have done to define a set of performance metrics for blockchain systems and factors that affect their performances. This will not only help us in the later analysis but will also establish a reference for industries that seek to evaluate their blockchain-based applications. Following that, we evaluate Quorum's performance with its two consensus algorithms and in terms of throughput, transaction delay, and transactions' success probability. This can act as a reference for our later implementation and for others that implement their applications using Quorum.

5.1. Introduction and Related Works

Permissioned blockchain platforms, including Quorum, Hyperledger Fabric, and Parity, are increasingly getting popular to be used for a wide range of applications. They generally offer higher throughput and lower delay guarantees at the expense of censorship. These features are generally desirable for applications outside cryptocurrencies. Further, businesses are increasingly opting for permissioned blockchains so that the censorship can be used to selectively deploy restrictions and access controls over the activities involved [77].

Given this wide range of possible permissioned blockchain platforms, it is challenging for organizations to decide on the right platform to use. Many of these platforms are available as open-

source for the industry and offer different services [78]. Resiliency, elasticity, decentralization, availability, and scalability (REDAS) features in such platforms differentiate them from one another. Further, before switching to a blockchain-based solution, developers need to verify the platform's suitability for their application and its requirements. Thus, a well-formulated evaluation framework that defines available benchmarks, evaluation metrics, and parameters that affect the blockchain performance can put forward an answer to these concerns and allow developers to evaluate their applications before switching.

Available blockchain evaluation benchmarks, such as Blockbench [79] and Caliper [80], can perform the above analysis with some limitations. These benchmarks are generally driven from distributed database evaluation standards. They mostly evaluate blockchains in terms of delay and throughput, which are the most critical metrics for assessing any distributed system. However, at the time of this writing, these benchmarks have not been standardized or even well-formulated. Each benchmark covers only a few blockchain platforms and requires a substantial implementation to integrate other platforms. Further, they are designed for financial applications, neglecting other blockchain applications that are currently becoming popular.

Some of the earlier works have already tried to provide recommendations on blockchains evaluations by surveying research works on blockchain evaluation benchmarks, performance metrics, and evaluation factors [81, 82]. A blockchain evaluation, or any system performance evaluation, can be done either by empirical analysis or by analytical modeling. Empirical analysis includes example benchmarks and evaluation metrics. In contrast, analytical modeling covers the theoretical and stochastic modeling of the process involved in blockchains. Both [81, 82] have

covered how recent research works and industries had used empirical analysis and analytical modeling to evaluate blockchain platforms.

From an analysis point of view, several works have analyzed permissioned blockchains' performance, including Ethereum [83, 84, 80, 85, 86, 87, 88], Hyperledger Fabric [83, 84, 80, 89, 90, 91, 92, 93, 94, 85], Quorum [30], and Parity [83, 84, 95]. These works evaluated the effect of changing different parameters (such as transaction rate) on the performance metrics (such as transaction delay). For example, Baliga et al. evaluated the effect in transaction rate, transaction size, block size, and consensus algorithm on the delay and throughput in Quorum [30]. Nasir et al. measured the effect of increasing the number of nodes on the transaction delay and throughput in Hyperledger Fabric [90]. Similar works were done in [89, 96, 97]. The works in [83, 84] have compared the performances of Hyperledger Fabric, permissioned Ethereum, and Parity when the number of nodes and the transaction rates increase.

5.2. Motivation

Earlier works in [81, 82] lack comprehensive lists of possible evaluation metrics that can distinguish one platform from another and the possible parameters that affect blockchain performance. Such lists are critical for blockchain users, analyzers, and platform developers to understand the performance and the chosen platform bottlenecks. Further, these lists help accelerate the evaluation standardization process, which is currently desirable for blockchains.

Therefore, compressive lists of evaluation metrics and parameters are necessary to perform proper analysis of blockchain performance and facilitate standardization efforts toward a proper and easy to use and extend evaluation benchmark.

On the other hand, as we will be using Quorum to implement our proposed approaches, it is desirable to analyze its performance from different perspectives. This analysis is necessary to verify its suitability for our implementation. The works in [30] analyzed Quorum with different transaction rates and consensus algorithm. However, these works lacked the analysis in terms of the number of blockchain nodes, which is critical to measure the platform scalability. Further, the analyses in these works were performed on an older version of Quorum and significant changes have been introduced to the platform afterward. Thus, the results and conclusion presented in these earlier works may not apply to the new versions of Quorum.

5.3. Contributions

To that extent, this part of the research will make the following contributions:

1. Comprehensive lists of blockchain evaluation metrics and the parameters that may change its performance are first provided. Such lists can act as a reference for industries that seek the evaluation of their blockchain-based applications. Further, they help accelerate the evaluation standardization process, which is currently desirable for blockchains.
2. Blockbench is then used as an evaluation platform that evaluates Quorum to be used for later implementation. The objective is to analyze the platform's scalability when increasing the transaction rate and the number of nodes. The analyses show that Quorum scales well when increasing the number of blockchain nodes (i.e., miners and validators). However, they also show that it has limited scalability when increasing the transaction rates. This scalability is enough for our security risk assessment applications, as will be concluded from the analyses in this chapter and the next one.

Most of the work presented in the chapter is under submission for publication in [98].

5.4. Blockchain Performance Metrics

Metrics are those criteria that distinguish one system from another, thus, they can be used to assess the system's performance. In the performance analysis literature, a metric is derived from the system's services and their corresponding outcomes. In the blockchain case, these services are the transaction's write service and the transaction's query service. Transaction's write service allows clients to submit transactions, enabling information exchange between any two nodes. Transaction's query service facilitates transaction reading requests and lets clients check if a transaction exists in the chain or not [54]. In the following, we discuss a comprehensive list of these metrics that can be used to evaluate blockchain performance.

5.4.1. Transaction Delay

Response time and delay are two metrics that measure the same criteria, so one of them is sufficient. Response time is the time taken by the system to respond to a transaction or a query request. In other words, it is the time between the transaction submission and the system reaction, which is the same as a transaction's delay. Response time is used by the system (e.g., the blockchain); the delay is used by the service (e.g., a transaction's write). For blockchain users, we usually care about the transaction delay rather than the response time.

The transaction's write delay for blockchains could have several definitions, as shown in Figure 5.1. All definitions start with the transaction submission time but may end at the block creation time, the validation time, the commit time, or the finalization time. The commit delay is the most important since the transaction cannot be removed or changed after that point. For

example, in a monetary exchange, commit delay is when the balances change to reflect the transaction.

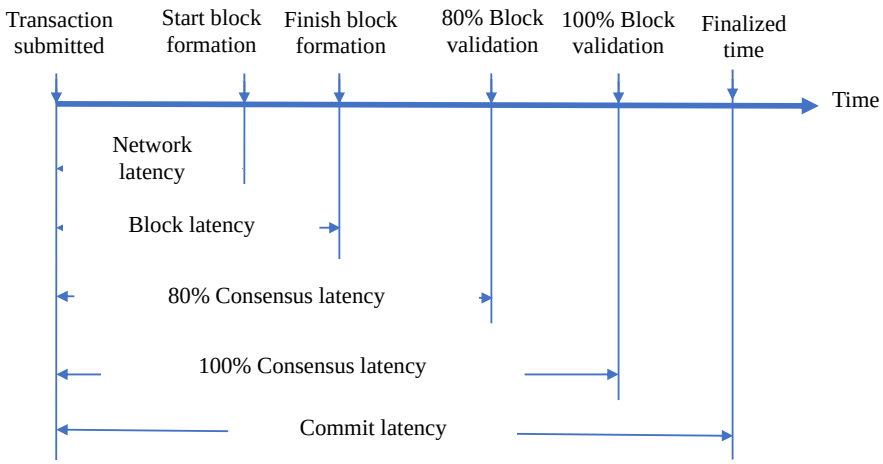


Figure 5.1 Different definitions of transaction's write delays

Lower delays are desirable for any blockchain system. It should be noted that this metric is measured per transaction and then averaged over all transactions. Other statistics, such as the minimum, maximum, standard deviation, or confidence interval of the delay, are also recommended to show the metric variability.

For the transaction's query delay, one can consider the overall delay, consisting of a time when a query is submitted until a response is received.

5.4.2. Throughput

For blockchain systems, throughput is the rate at which transactions are committed to (or read from) the chain. Like the delay, one can consider several forms of the throughput, including the consensus throughput and the commit throughput.

The commit throughput is the one on which a blockchain is mostly evaluated. Transactions that are invalid should be omitted from the commit throughput calculation. Furthermore, the commit

throughput should consider the consensus over all nodes rather than a single node. This means that the analysis should only consider the transactions committed at all nodes' databases at a particular time. The commit throughput for the transaction's write service can be expressed as:

$$\text{Commit Throughput} = \frac{\text{Number of committed transactions}}{\text{Total time in seconds}} @ \text{ all miners and validators}$$

The transaction's query throughput is applied to the read requests of the committed transactions. Generally, throughput is measured in transactions per second (TPS), where a high TPS is desirable for any blockchain system.

5.4.3. Transaction Success Probability

Transaction success probability is the percentage of transactions being successfully committed in transaction's writes and successfully returned in transaction's queries. Mathematically, the transaction's write success probability can be:

$$\text{Probability} = \frac{\text{\# of committed transactions}}{\text{\# of submitted transactions}} \times 100\%$$

Similarly, it can be calculated by the ratio of successfully returned transactions to the total number of queries for the transaction's query.

It is generally recommended that this metric is calculated at the end of the evaluation to consider all committed transactions. However, in some time-sensitive applications, this probability could be monitored as a moving average over different time windows.

This metric is calculated either by the success probability as the above or by the loss probability, which is the ratio of transactions not committed. In our later analysis, we will use the loss probability. However, the mapping between them is easy to understand.

5.4.4. Resource Consumptions

Consumed resources, such as network links, CPU, and memory, at different nodes, are other important metrics to consider when evaluating a blockchain platform. Some consensus algorithms, such as PoW, consume more CPU resources while others, like practical byzantine fault tolerance (PBFT), consume more network resources. Also, miners are expected to use more resources than other nodes. Thus, measuring resource consumption at different nodes is sometimes essential for the evaluation.

Lower resource consumption is desirable in any system. However, tradeoffs between different types of resources at various nodes should be considered.

5.4.5. Invalid Blocks Probability

An invalid block can be caused by an invalid transaction being included in a block. This can be considered as an attack by a malicious miner. The miner forms the invalid block, and if an improper validation process is used, this block may be committed to the chain. The probability of committing these invalid blocks should be measured for evaluation.

This metric can be measured by imposing a malicious miner and calculating the likelihood of successfully committing an invalid block.

As can be expected, lower invalid block probabilities are desirable for any secure blockchain system. As a matter of fact, this probability should tend to zero to prove that the system is secure against malicious miners and validators.

5.4.6. Query invalid/attack probability

An attack that targets a transaction's query can return an incorrect transaction that does not exist in the consensus chain. Similarly, a storage node can return a "no such transaction" response for a transaction that exists.

The success rate of such attacks is important when evaluating the security or resiliency of a blockchain platform. This metric should be extremely low for highly secure platforms.

5.4.7. The Ratio of Forked Blocks

A final essential metric to measure the platform's vulnerability (especially in permissionless blockchains) is the number of blocks involved in a blockchain fork, i.e., a divergence of the chain into two potential paths. These blocks can result in several well-known blockchain attacks, including selfish consensus and double-spending [99]. Thus, measuring their ratio, or the number of forked blocks to the number of total blocks, is a critical security measure.

As with the previous attack measurement metrics, this metric should be extremely low for highly secure platforms.

5.5. Blockchain Parameters and Factors

The performance of any system depends on many variables and specifications that affect its operations. As an example, the system hardware or software specifications can have a significant

effect on its performance. In performance analysis, these variables and specifications are commonly called parameters. They can be divided into system parameters and workload parameters. The parameters that have a substantial effect on the performance and are tuned during experiments are called ***factors.*** The values that a factor takes during evaluations are its corresponding ***levels***. Both factors and their levels should be chosen carefully based on their impacts on the system [54].

In the following, we discuss a list of system and workload parameters that affect blockchains.

5.5.1. Blockchain System Parameters

System parameters are the characteristics or system specifications that affect the performance regardless of its current load. These include the size and distribution of the nodes, the available resources, and blockchain configurations. For blockchains, the system parameters can be summarized as follows.

- **The number of different types of nodes:** The number of different types of nodes is a system parameter that affects the performance by shaping many of the metrics. For example, the number of miners may enhance the delay and throughput, but it increases resource consumption in most cases. The number of validators worsens the delay as more nodes will be included in the validation process. However, it hardens the attacks and thus it reduces their success probabilities. The number of users also affects the delay, throughput, and resource consumption due to the collision probability of the transactions and the increased overhead.
- **Network size and geo-distribution of nodes:** Another system parameter that significantly affects the performance is the network size. Network size is defined by the network's diameter,

which is the distance between its two farthest nodes. Thus, it depends on the geo-distribution of all nodes. This metric affects delay, throughput, and resource consumption. The delay increases while the throughput decreases as the distance between nodes increases. This will also impact the network resources, as it will increase as the size increases. Earlier results have shown that the network diameter does not affect Bitcoin networks much due to its excellent connectivity [100]. However, for network connectivity found in most permissioned blockchains, this parameter can significantly affect the performance.

- **Blockchain specific configurations:** Some blockchains implement several configurations for consensus algorithms, transaction configurations, block size, block time, database type, and so on. Changing these configurations affects the performance of the blockchains. Thus, they are critical factors to consider.
- **Hardware and software resources:** The amount of resources reserved for different nodes affects the system's performance, and therefore, it is a system parameter. More network resources can result in better performance by increasing throughput. Also, more CPU resources can decrease the consensus time in some platforms and reduce the overall delay, but it increases resource consumption in most cases.
- **Fault tolerance:** Another system parameter that might be considered is the system's resiliency to its faults. These faults may happen due to nodes' crashing after some time, network delays of messages, or interrupting the message exchange by malicious nodes. As most blockchains are Byzantine tolerant, the system should not fail with these faults. Byzantine tolerant means the system continues its normal operation even if some of the nodes fail or act maliciously. However, it is expected that the throughput and delay would worsen.

5.5.2. Blockchain Workload Parameters

The workload parameters are the characteristics of the inputs to the system that affect its performance. The number of transactions, their sizes, and external system loads are all workload parameters for blockchain systems. In the following, we briefly discuss these parameters.

- **Transaction rate:** One of the most critical workload parameters to consider is the transaction input frequency or transaction rate. Theoretically, as the rate increases, a block can include more transactions; therefore, the throughput increases. However, this is limited by the block size configuration, as smaller blocks cannot include all transactions simultaneously. The blockchain might get highly overloaded after that limit, thus, the performance would either decrease or stay the same.
- **Transaction size:** The size of transactions is another workload parameter that affects the performance by increasing the delay and reducing the throughput.
- **The number of transaction's reads and writes:** The number of reads and writes required in a single transaction can be considered a workload parameter. It affects the performance by increasing the execution time and thus increasing the delay.

5.6. Blockbench and Quorum

Blockbench is a blockchain evaluation benchmark, i.e., an evaluation platform, that we will be using in our analysis for this chapter and the next one. Thus, in the following, we briefly describe the architecture of this benchmark which is illustrated in Figure 5.2 [83, 79].

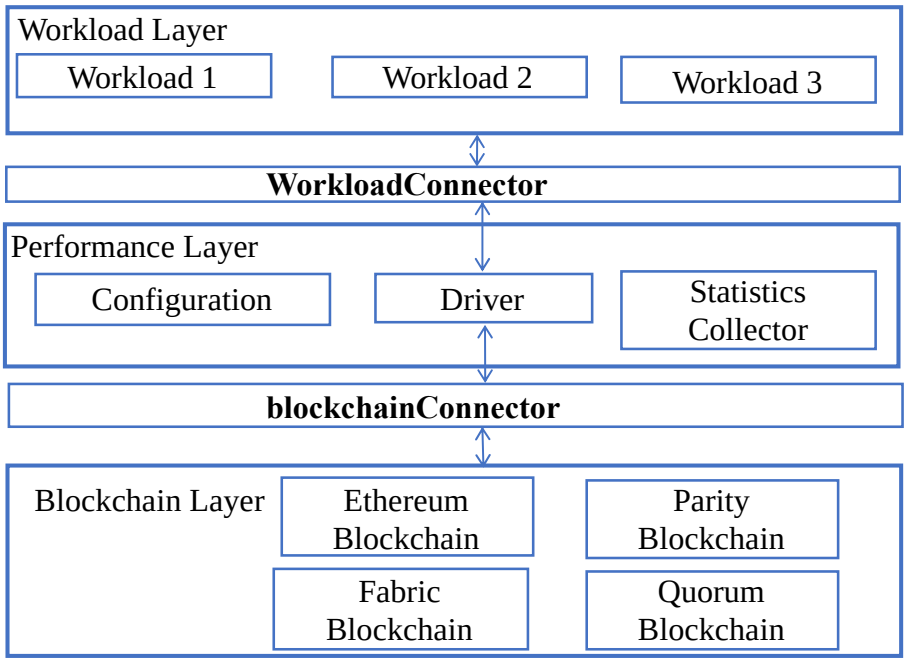


Figure 5.2 Blockbench evaluation benchmark

The workload layer is responsible for generating the required transactions to be submitted by users. These transactions can be standardized for specific applications (e.g., smallbank is a standard workload for banking transfer applications [101]), or it can be generalized for general applications. Workloads can be implemented using smart contracts and connected to the platform by the workload connector. Blockbench offers two already implemented workloads, namely smallbank for money exchange application and Yahoo! Cloud Serving Benchmark (YCSB) for general applications [102].

The performance layer calculates the performance metrics such as transactions' delays and resources' consumptions for each platform. In Blockbench, this layer consists of three modules, the configuration, the driver, and the statistics analyzer. The configuration module provides the setup of the blockchain platforms to all nodes in the system. This includes specifying the numbers of nodes, their types, and how they are connected to each other. The driver module is responsible

for connecting to the blockchain platform and passing transactions from the workload to the blockchain. Finally, the statistic collector is the module that collects all the performance metrics to be further analyzed by the benchmark users. Blockbench uses this layer to provide the results in terms of block analysis whenever a block is generated.

The blockchain layer is the underlaying blockchain platforms that can be evaluated using Blockbench. As can be seen from the figure, the current implementation of Blockbench can evaluate four platforms: Ethereum, Quorum, Fabric, and Parity. Other platforms can be added but have to be connected to the benchmark using the blockchainConnector and Driver modules.

As discussed, the platform of choice for our later implementation is Quorum as an easy-to-use platform that achieves high transaction rates with additional privacy guarantees and multiple consensus algorithms. The details of Quorum are out of the scope of this work and can be referred to in [103]. However, in the following, we briefly explain Quorum two consensus algorithms, namely Raft and Istanbul Byzantine Fault Tolerance (IBFT), to be used for the later analysis in this chapter.

Raft is a leader-based crash fault tolerance consensus algorithm adopted from the traditional PAXOS algorithm for fault tolerance systems [104]. In this algorithm, the first node that joins the systems becomes a miner (also named a leader or a minter). Other nodes, i.e., validators (also named followers or verifiers), verify the generated block and add it to the chain. After proving it is valid, the validators message back the miner to acknowledge the block's receipt. Raft is only fault-tolerant and does not assume that there can be adversary nodes on the network. As we will be implementing our approach with this algorithm, we have modified it to validate the generated

knowledge summary to handle this specific adversary setting. More details about that will be presented in the next chapter.

IBFT is another leader-based Byzantine fault tolerance consensus algorithm. A leader is elected periodically based on pre-specified time [105]. This leader proposes blocks and submits them to other nodes (validators) for validation. Once a validator receives the new block and proves that it is correct, it notifies the neighboring validators and waits for their subsequent notifications. When the validator receives enough notifications from the validators, it adds the block to its chain. Otherwise, the block is dropped, and a new leader is to be elected in the next round. Unlike Raft, this algorithm guarantees that the blockchain is tolerant to adversaries or faulty nodes. However, as the number of validators gets large, IBFT will introduce a network overhead and, consequently, result in transaction delays, as will be seen in the later analysis.

5.7. Performance Analysis

To build a Quorum blockchain network, we have used Cloudlab virtual machines provided by NSF [106]. The nodes are all located in Utah cloud servers and are connected in a peer-to-peer fashion with 1 Gbps links. Each blockchain node runs Ubuntu 18.04.1 LTS, as it is the latest supported version. All nodes are equipped with Blockbench and Quorum code and are able to run these codes independently. This setup is used for all of our analyses in this chapter and the next chapter. The Quorum version used for this analysis is v21.1.0.

In the following, we describe our chosen metrics, chosen factors and levels, workload, and finally, the results and comparisons.

5.7.1. Chosen Metrics

For the analyzes done in this dissertation, we care about the transactions' write service rather than transactions' query. This service is more critical for our objectives as we would later want to know how fast a knowledge summary (and a decision) can be generated and written to a block. On the other hand, the transaction query services can only return already made decisions or summaries, which is not of interest.

To analyze the performance of Quorum, we have chosen three metrics: *commit transaction delay, throughput, and transaction loss rate.* These metrics are the three most important ones in our case. The commit delay (subsequently named delay) provides an estimate on when a knowledge-summary can be made. The throughput can approximate how many response transactions can be made in a second. The transaction loss rate is necessary to analyze the platform's limits, and thus, will be used in this chapter only.

Note that the other metrics such as resource consumptions and invalid block rate are also important for proper analysis. However, we exclude them here as they provide minimal information specific to security risk assessment applications.

5.7.2. Smallbank Workload

Smallbank, as its name indicates, is a standard specification for applications that perform banking operations such as withdrawing money, depositing checks, and transferring between savings and checking accounts. The workload consists of actual transactions that reflect OnLine Transaction Processing (OLTP) operations. It consists of bank accounts and their types (savings or checking) and can perform six functions (amalgamate, get balance, deposit checking, send payment, transact

savings, and write a check). Further details about this workload can be found in [101]. This workload will be used for the analysis done in this chapter and the next one.

5.7.3. Chosen Factors, and Their Levels

We have chosen the transaction rate and the number of nodes as factors for our analysis. These two factors are the most crucial parameters to measure a blockchain platform's scalability, which is of interest to this research. The consensus algorithm is another factor that we consider for this chapter. The parameters, chosen factors, and their level for our experimental analysis are listed in Table 5.1.

Table 5.1 Parameters and levels for our experimental design

Parameter	**Level**		
Number of nodes (factor)	**Total**		8, 16, 32
	Validators		3, 7, 15
	Clients		3,7,15
	Miner (leader based)		1
Geo distribution	local (Utah servers)		
Blockchain specific configuration	**Consensus algorithm (factor)**		Raft, IBFT
	Block time (defaults)		IBFT= 1 s. Raft=50 ms
Hardware Resource	**CPU**	Eight-core Intel Xeon D-1548 at 2.0 GHz	
	RAM	64GB ECC Memory	
	Disk	256 GB NVM flash storage	
	Network	1 Gbps	
	NIC	Dual-port Mellanox ConnectX-3 10 GB NIC	
Transaction Rate (factor)	64, 128, 512, … 4094 (or 2^6, 2^7… 2^{12})		

Transaction size	smallbank workload sizes
Number of transactions	Transaction rate * 120

5.7.4. Effect of Transaction Rates on the Performance

In the first analysis, we vary the transaction rate with the levels indicated in Table 5.1 and measure the effect on throughput, delay, and transaction loss rate. We run the experiments as specified before with both RAFT and IBFT consensus algorithm. However, the number of nodes for this specific analysis is fixed to 16 total nodes, with one miner, seven validators, one requester, and seven clients. Each experiment is repeated 35 times to measure the variability of the results.

The effect on the throughput, delay, and success rates are shown in Figure 5.3, Figure 3.4, and Figure 3.5, respectively.

As can be seen from Figure 5.3, both consensus algorithms behave very similarly in terms of throughput at low transaction rates. The throughput increases steadily until it reaches a limit of about 1024 TPS, or 2^{10}. This limit is defined in performance analysis as *"the knee point,"* after which the throughput stabilizes and then decreases when the platform reaches "*its capacity"* [54].

The figure also shows that the throughput for Raft decreases after 2^{10} TPS. This drop indicates that the system has reached that capacity at which the throughput should start dropping after that. For IBFT, the capacity of the system is beyond 2^{12} (about 5000 TPS).

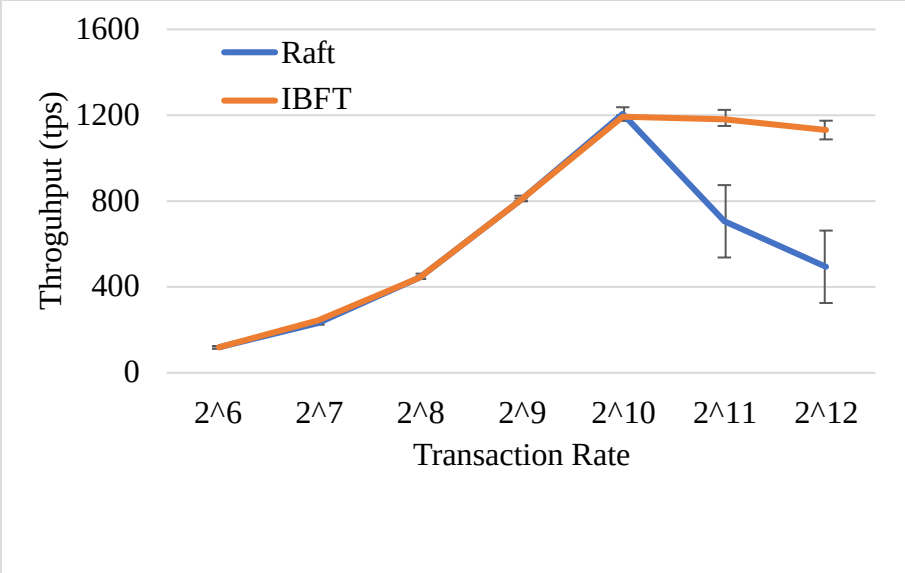


Figure 5.3 Effect of the number of nodes on the throughput

On the other hand, the delay, as shown in Figure 5.4, is minimal, stable, and comparable for the two algorithms before the knee point. This matches the performance analysis theory, which states that the delay is expected to be small and stable before the knee point and increase exponentially after the knee [54]. The figure shows the delay increases exponentially after the knee point and is expected to increase further or stabilize after 5000 TPS.

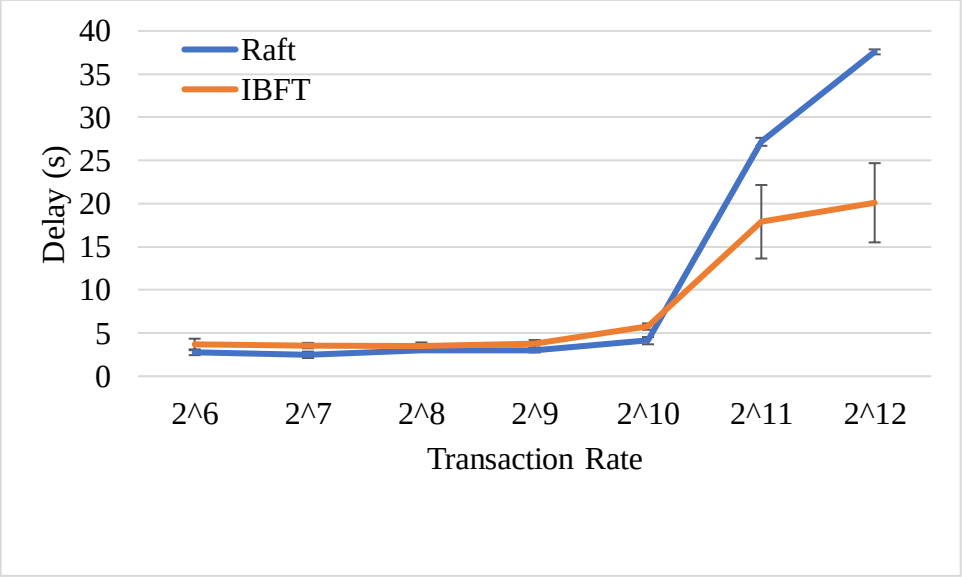


Figure 5.4 Effect of the number of nodes on the delay

In terms of the loss rate of transactions, as shown in Figure 5.5, Raft has a high increase in the loss rate, which is shown after the knee point, i.e., 2^{11} in the figure. At that point, the system has reached its maximum capacity and fails to get all transactions before their time out [54]. The figure also shows that IBFT has minimal transaction loss as the system has not reached its capacity yet, but the loss rate is expected to increase after 5000 TPS.

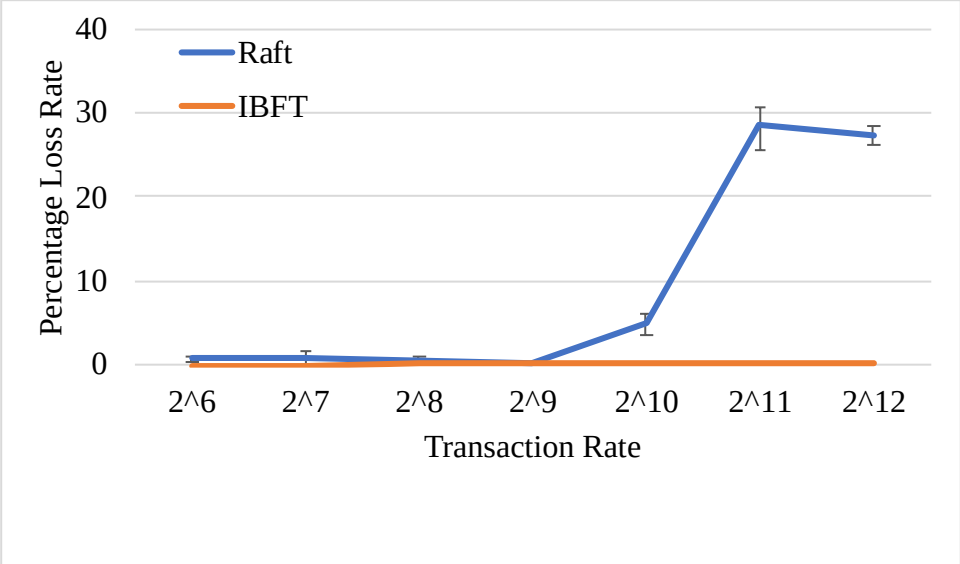


Figure 5.5 Effect of the number of nodes on the success rate

While these results seem to favor IBFT as it has higher capacity, the difference is that high and only about 4000 TPS. Both consensus algorithms would have high loss rates after 5000 TPS. That said, 5000 TPS, or even 1000 TPS for Raft, is reasonable to consider for security risk assessment and decision-making applications in general. Further, as will be seen in the next chapter, even with the system reaching its capacity, the delay in generating the knowledge-summary is less than two minutes which meets non-real-time requirements.

5.7.5. Effect of the Number of Nodes on the Performance

In this second analysis, we will vary the number of nodes with the level indicated in Table 5.1 and measure the effect on throughput, delay, and loss rate. Similar to the first analysis, both Raft and IBFT will be examined as consensus algorithms. The rate is fixed to 2^{12} TPS for this specific analysis. Note that, at that rate, Raft has reached its capacity while IBFT has not. Thus, it is expected that Raft will have a lower performance than IBFT for all metrics. However, the objective here is to see the change in the metric as the number of nodes increases.

The effect on the throughput, delay, and loss rates are shown in Figure 5.6, Figure 5.7, and Figure 5.8, respectively.

As shown in Figure 5.6, the throughput is stable for both algorithms as the number of nodes increases. Note that the throughput has slightly decreased for IBFT (due to network overhead), but at this stage, the difference is negligible given that the increase is still with the confidence interval range. It is expected that this decreased will be much higher after that. However, we could not show such a decrease after 32 nodes as we were limited by the number of virtual machines we could reserve in Cloudlab.

Raft's performance is stable (or even slightly better), which indicates that the system is more scalable in terms of the number of nodes.

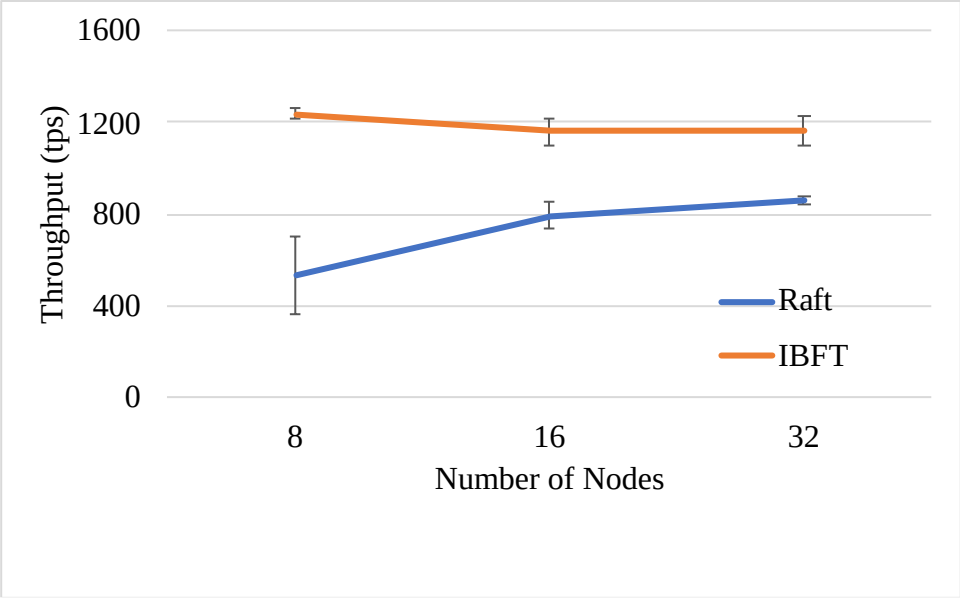


Figure 5.6 Effect of the number of nodes on the throughput

As shown in Figure 5.7, the delay for IBFT increases when the number of nodes increases. This metric clearly shows that IBFT is less scalable than Raft when considering the number of nodes. However, it should be remembered that the delay in Raft is not representing the performance at this transaction rate, as the loss rate is high. Nevertheless, the objective here is to show the change in performance rather than compare the two consensus algorithms. Thus, this experiment confirms that IBFT is less scalable than Raft.

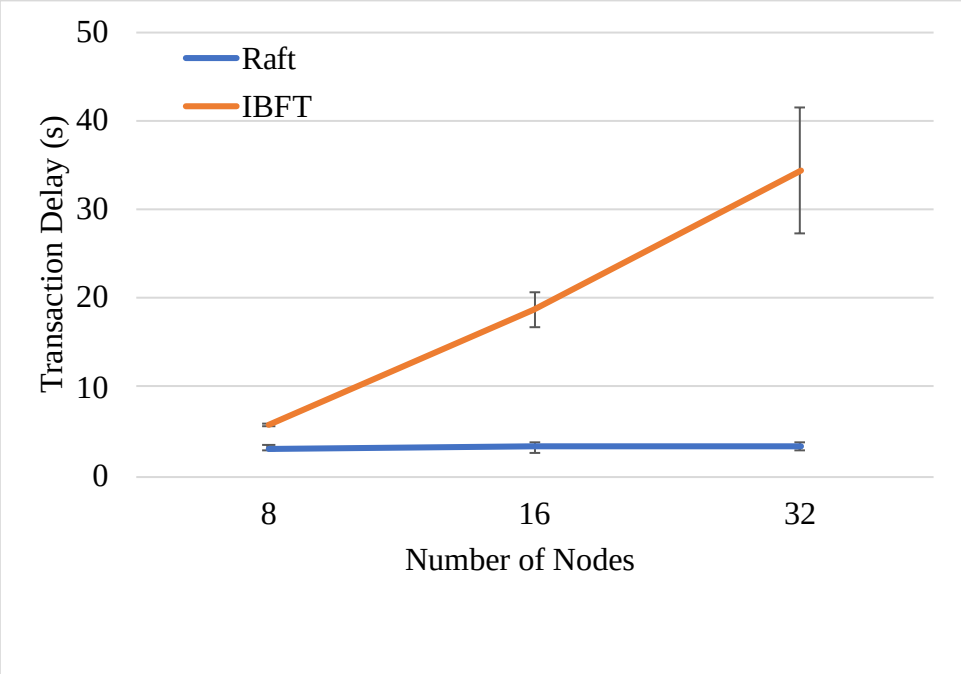


Figure 5.7 Effect of the number of nodes on the delay

For the transaction loss rate, as shown in Figure 5.8, both algorithms show a stable performance. Raft has a high loss rate in this case. However, such a steady loss is due to the chosen transaction rate, not to the increase in the number of nodes. On the other hand, IBFT showed a consistent performance for this specific case.

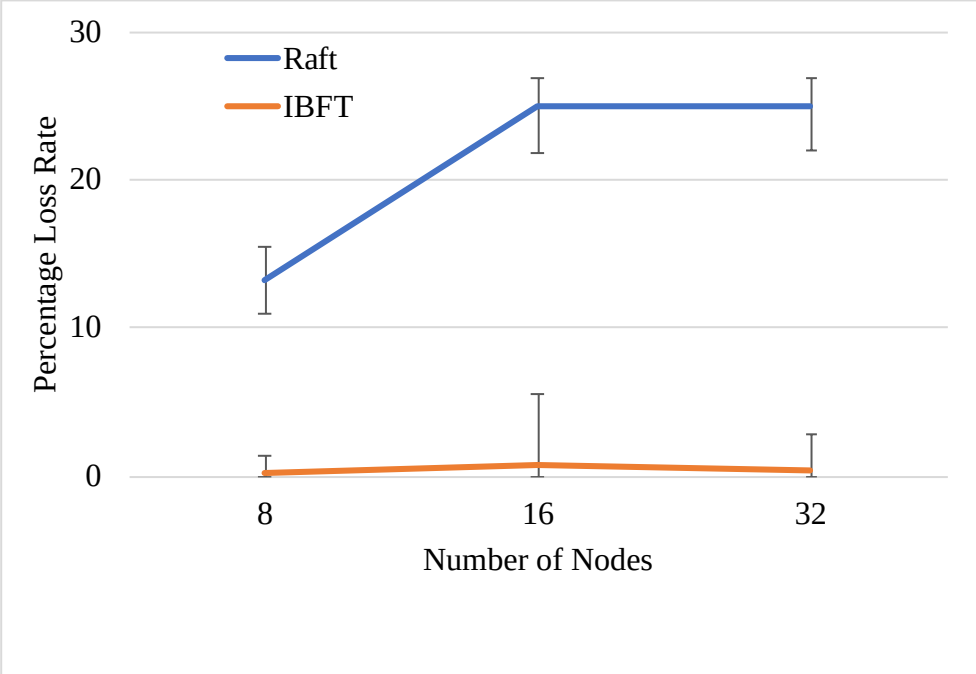


Figure 5.8 Effect of the number of nodes on the delay

The above results show that both IBFT and Raft have not reached their capacity performance with the chosen number of nodes. IBFT has demonstrated that the delay increases as the number of nodes increases, which is an indication that the system is not entirely scalable. This is a known problem for PBFT-like consensus algorithms as they fail to scale due to the network overhead resulting from message exchange between validators. On the other hand, Raft showed a stable performance for all metrics, which is an indication that it has better scalability.

The overall results from these analyses show that both algorithms scale better with the number of nodes than they do with the transaction overhead. Thus, Quorum is generally scalable with the number of nodes. This is a desirable feature for our decision-making and security risk assessment applications.

The two consensus algorithms are in general comparable in performance and both suit decision-making applications. Thus, in our later implementation, we will consider only Raft analysis due to its simplicity.

5.8. Summary

With the increasing number of permissioned blockchain platforms and their wide range of services, it is crucial to have an evaluation standard for companies to use when deciding on the blockchain platform for their applications. However, such a standard is still missing from the literature, indicating the urgent need to create one. To do so, one needs to know the metrics on which the blockchain systems can be evaluated and the parameters that affect their performances.

In this chapter, we presented comprehensive lists of blockchain evaluation metrics and parameters. We detailed how these metrics can be applied to the blockchains, and how the parameters are expected to affect the performance. The lists can act as a reference for industries that seek to evaluate their blockchain-based applications. Further, they help accelerate the evaluation standardization process, which is currently desirable for blockchains.

Following that, we used the discussed list to design and conduct an experimental analysis for Quorum as a blockchain platform of choice. Our results showed that Quorum scales better in terms of the number of nodes than it does with the transaction rate. They also showed that while IBFT generally performs better, its scalability is limited in terms of the number of nodes which is a critical issue for our target applications. Thus, moving forward, we will be using Raft for our implementation and analysis. As Raft showed to be exceptionally scalable in terms of the number of nodes, our later analysis will only include the transaction rate as a factor.

As will be highlighted in the next chapter, Raft is not the best choice for security in adversarial settings. We reduce that effect by the use of knowledge summary validation and reputation scheme. However, as future work, one could analyze the performance under adversarial settings and check how the metrics are affected by malicious miners (leaders in Quorum case) or validators.

Chapter 6:CrowdFAB: A Framework for Crowd-Forecasting Applications using Blockchains

Chapters 3 and 4 propose solutions that resolve two of the major challenges in blockchain-based security risk assessment solutions. Knowledge-based blockchains and a novel reputation scheme have been introduced to resolve the storage without knowledge and the need for reputation scheme challenges, respectively. In this chapter, we combine these two innovations to propose a new framework, named CrowdFAB, for crowd-forecasting applications using blockchains. Crowd-forecasting is defined as forecasting using a crowd of agents. A full system is defined, implemented, and analyzed in terms of security and performance overhead. As will be seen, CrowdFAB resolves the practical use-case challenge discussed earlier and achieves a system that is provably secure with a moderate overhead introduced. This finalizes stage one of the research that extends blockchains as an underlying infrastructure for any decision-making and risk assessment applications Even though the framework is generic, most of the discussion here will focus on security risk assessment applications.

6.1. Introduction and Related Works

Modern problem-solving applications, including traffic analysis, travel bookings, and job recruiting, are extremely complicated. They require users to collaborate with possibly thousands of others that could be at different levels of expertise. Crowdsourcing has emerged over the last decade as a distributed and efficient problem-solving model to support these applications [107]. Its traditional systems consist of three main parties: requesters, agents, and a centralized server. A requester submits a task or a set of tasks that are challenging for him but easy to solve for others.

Agents (also referred to as workers in literature) compete to solve this task and submit solutions through the system. The centralized server regulates the communication, selects a "one-to-one matching solution," and grants the corresponding agent(s) a reward. As an instance, Uber requires riders (requesters) to request a ride, and several drivers (agents) can offer the ride while Uber chooses one driver to be offered to the rider [108]. Other examples include Upwork for hiring tasks and Airbnb for hotel reservations [109, 110].

An emerging subset of crowdsourcing applications are those that involve forecasting and crowd intelligence or wisdom, where agents input their predictions as responses to specific questions, events, or metrics. We refer to these applications as *crowd-forecasting applications*. In these applications, requesters post questions and act depending upon several responses that might involve risk or uncertainty. A recent example is the Facebook forecasting application for predicting world-leading events, including disease outbreaks and elections [111].

Security risk assessments are considered as other examples of these types of forecasting applications. For instance, tracking and detecting malicious applications or network flow in mobile platforms and networks can be solved by crowd-forecasting. Thousands of distributed malware detectors (agents) individually evaluate whether an application is malicious or not and the system will reach a collaborative response about the application [13]. Similarly, various nodes can vote if an active node is misbehaving or if an account or an identity is fake, which would help detect malicious nodes in distributed networks. PolySwarm [112], Synack [113], and Virustotal [34] are examples of crowd-forecasting security risk assessment industries that have emerged in recent years. Similar research works are present in [114, 115, 116].

Unlike Uber and other traditional crowdsourcing applications, crowd-forecasting applications do not offer a one-to-one matching between requesters and agents. Instead, a collective summary or decision, i.e., knowledge summary in the context of this dissertation, is reached from multiple agents' answers [117]. Thus, crowd-forecasting models do not only communicate requests and responses but also process these responses to get a collaborative solution.

In other words, crowd-forecasting models need processing and knowledge-based systems rather than simple storage systems needed in traditional crowdsourcing.

Despite their popularity, traditional crowdsourcing and crowd-forecasting models have several weaknesses. The majority of these models *require a centralized party* that serves as a control and communication channel between requesters and agents [118]. This centralization leads to many challenges that were discussed earlier, including a single point of failure, trust issues, and potential attack surfaces. Further, most of these systems save user information such as name, address, and phone number along with requests or responses. This leads to *weak anonymity and many privacy disclosure concerns* [119]. Finally, *the knowledge summary in crowd-forecasting applications is vulnerable to manipulation*, given the centralized nature of such models [120].

With no surprise, blockchains have the potential to resolve some of the above challenges, as was emphasized throughout the earlier chapters. Many earlier works have emerged over the last three years to build proper blockchain-based crowdsourcing systems [121, 122, 123, 124, 125]. For example, CrowdBC was the first attempt to build a concrete blockchain-based framework for crowdsourcing applications. It utilized smart contracts to perform traditional crowdsourcing processes and was implemented on the Ethereum platform [121].

Some others have focused on the privacy problem of traditional crowdsourcing systems. For example, ZebraLancer advanced CrowdBC by providing privacy and strong anonymity using Zero-Knowledge proofs [126, 127]. In addition, the works in [128, 119, 129, 130, 122, 131, 132, 14] used other privacy-preserving mechanisms to provide privacy guarantees for crowdsourcing applications.

Some earlier works focused their research on building blockchain-based approaches for specific applications. For example, CrowdJudy was the first attempt to make a blockchain-based crowdfunding application [133]. Haung et al. proposed a blockchain-based approach for crowdsensing in industrial systems [134]. In addition, MCSChain, presented by Fang and Yan, discussed blockchain-based crowdsourcing for mobile applications [118].

6.2. Motivation

None of the earlier works have targeted crowd-forecasting applications and mostly focus on the traditional one-to-one solution for crowdsourcing applications. They use the blockchains only for gathering responses as a storage system. However, as discussed earlier, forecasting applications need a knowledge-based system to process the decisions and achieve the required "knowledge summary."

In other words, to support blockchain-based forecasting applications, there is a need to transfer blockchains from simple storage to processing and knowledge-based systems. This calls for the use of knowledge-based blockchains proposed and evaluated in Chapter 3.

In addition, earlier works rarely discuss the reputations of agents as an active part of the decision-making process and mostly base the reputations of evaluation of requesters who could be

malicious. However, for forecasting and decision-making applications, it is important to consider reputations based on actual performance rather than on the subjective requesters' evaluation.

The proposed reputation scheme in Chapter 4 has some potential to be used for the applications here.

To that extent, this chapter combines the works in Chapter 3 and Chapter 4 to propose CrowdFAB, a framework for Crowd-Forecasting Applications using Blockchains. CrowdFAB fills the gaps discussed above and proposes a blockchain-based solution designed explicitly for crowd-forecasting applications that securely and efficiently reach the knowledge summary. We will be using this framework for a security risk assessment application (specifically target a distributed malware detection use case) in Chapter 8.

6.3. Contributions

The work presented in this part of the research will make the following contributions:

1. CrowdFAB is first conceptualized as a blockchain-based paradigm for crowd-forecasting applications. CrowdFAB inherently resolves the centralization challenges discussed earlier. It also relies on blockchains' pseudo-anonymity to provide a level of user privacy. More importantly, CrowdFAB uses our extended knowledge-based blockchain paradigm, which resolves the issue of processing the decisions and achieving the required "knowledge summary."
2. The framework is then formulated and implemented on top of the Quorum blockchain platform to evaluate the introduced overhead. We emphasize that the framework can be deployed on

any other blockchain platform, given that it satisfies the system assumptions and requirements of the target applications.

3. The security of the proposed scheme is analyzed, proving that it is fraud resilient. This is achieved by two features: resiliency against malicious agents who try to control the knowledge summary and resiliency against malicious nodes which try to downgrade good agents' performance. In addition, the system has many appealing security characteristics that are inherited by the nature of blockchains.

The later contribution is expected to resolve the third and final blockchains challenge discussed in Chapter 2. That is, it provides a proper analysis that guarantees the security of the knowledge summary with provable minimal overhead.

The work presented in this chapter is under submission for publication [17].

6.4. CrowdFAB Framework Modeling

In the following, we discuss the framework model, architecture, and system and security assumptions. To simplify the discussion, we focus on malware detection in files as an example application and use it to describe the framework, which can be applied to numerous other applications. Table 6.1 presents some notations that we will be using throughout this chapter.

It should be recalled that *we refer to various blockchain nodes as requesters, agents, miners, validators. We also have two types of transactions: request and response transactions*. Also, to be consistent, *subscript i* will be used for requests, *subscript j* will be used for responses/agents, *subscript k* will be used for miners, and *subscript l* will be used for validators. *Subscript ij* will be used for the response of the j^{th} agent for the i^{th} request.

Table 6.1 Notations used in this chapter

Notation	Explanation
Q	A requester
$\boldsymbol{T}=\{T_1, T_2, \ldots, T_t\}$	Set of t sequential request transactions submitted by the requester. i^{th} request here consists of $file_i$ to be inspected. In a general application, a request would be a question posted by a requester. Each request ($file_i$) has an ID that is donated by $ReqID_i$
$\boldsymbol{F_i}=\{F_{1i}, F_{2i}, \ldots, F_{fi}\}$	Set of f features for $file_i$. Agents can use these features to determine their responses
$\boldsymbol{N_i}=\{N_{i1}, N_{i2}, \ldots, N_{in_i}\}$	Set of n_i agents that respond to the i^{th} request
$\boldsymbol{D_i}=\{D_{i1}, D_{i2}, \ldots, D_{in_i}\}$	Set of n_i Responses submitted by n_i agents responding to the i^{th} request
$\boldsymbol{P_i}=\{P_{i1}, P_{i2}, \ldots, P_{in_i}\}$	Each response D_{ij} includes a probability P_{ij} that the file is malware.
C_i	Knowledge summary of responses for $file_i$.
R_j	Reputation of $agent_j$
$\boldsymbol{V}=\{V_1, V_2, \ldots, V_v\}$	Set of v validator nodes
$\boldsymbol{M}=\{M_1, M_2, \ldots, M_m\}$	Set of m miners
G	The summary function specified by the requester to summarize *responses*

6.4.1. System Components

In a blockchain-based malware detection system, the requests consist of the software application, file, or update to be inspected. Requesters are organizations and individuals posting requests. Without loss of generality, we will be using only one requester Q. For our malware detection example, each request consists of several file features $\boldsymbol{F_i}=\{F_{1i}, F_{2i}, \ldots, F_{fi}\}$ that can be used by the detection agents to identify if the file is malicious or not. These features will be included in a request transaction T_i.

Agents, identified by $\boldsymbol{N_i}=\{N_{i1}, N_{i2}, \ldots, N_{in_i}\}$, are those who inspect T_i submitted by the requester Q. n_i here refers to the number of agents that respond to T_i. For malware detection, these agents are the security experts who can inspect a file for malware possibility based on predetermined

rules, ML models, or any other inspection strategy. Agent *j* submits his/her response transaction D_{ij} consisting of probability P_{ij} that this file has malware along with other conventional variables such as signature and timestamp. Thus, a single request results in a response transaction set $\boldsymbol{D_i}=\{D_{i1}, D_{i2}, ..., D_{in_i}\}$ along with a set of probabilities $\boldsymbol{P_i}=\{P_{i1}, P_{i2}, ..., P_{in_i}\}$. We allow all willing agents to participate in the request. Based on their performances, each agent gets a reputation R_j that determines the "goodness" of its predictions. An agent may choose to respond to all submitted requests, or only some requests, or not at all.

Miners, denoted by $\boldsymbol{M}=\{M_1, M_2, ..., M_m\}$, and validators, donated by $\boldsymbol{V}=\{V_1, V_2, ..., V_v\}$, are the blockchain nodes that generate and verify blocks. Compared to traditional blockchains, CrowdFAB miners/validators have an additional task of creating/verifying a knowledge summary of all responses received.

6.4.2. CrowdFAB System Architecture

CrowdFAB architecture consists of two layers: the blockchain layer and the crowd-forecasting layer, as illustrated in Figure 6.1. The crowd-forecasting layer consists of the crowd entities, i.e., requesters and agents. These entities connect to the underlying blockchain infrastructure by a web interface, hardware, or an API.

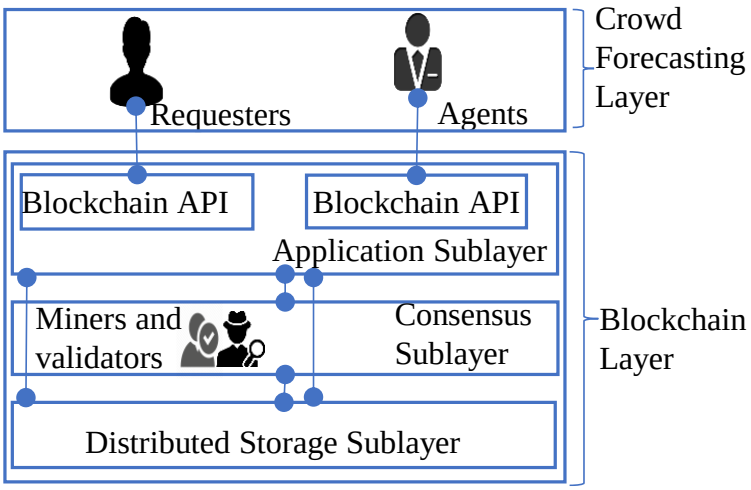


Figure 6.1 CrowdFAB architecture

The blockchain layer consists of the application sublayer, the consensus sublayer, and the distributed storage sublayer [135]. *The application sublayer* includes the blockchain interface with its specific functionalities, including transactions, smart contracts, underlying rules, and chain codes. *The consensus sublayer* consists of the miners and validators that agree on one consistent state of the blockchain database. These two sublayers, i.e., the consensus sublayer and the application sublayer, perform the logical operations of CrowdFAB, including collecting requests and responses and achieving the knowledge summary. *The distributed storage sublayer* is the shared layer that includes the database system stored on the nodes that maintain the blockchain database. All sublayers are connected to maintain the blockchain functionalities and provide a secure and efficient underlying infrastructure for the crowd-forecasting layer.

6.4.3. System and Security Assumptions

CrowdFAB has some system assumptions for the range of applications that can use the proposed concepts. These assumptions include passive processing and delay constraints, majority efficient

participants, no monetary rewards, and enough storage and processing powers. In the following, we briefly detail these assumptions.

- **Passive processing:** Due to the distributed nature of the blockchains, meeting real-time requirements and processing is still a significant challenge in blockchain-based solutions. In general, real-time crowdsourcing is a considerable research challenge to-date [136]. Thus, we *assume non-real-time and passive crowd-forecasting security applications* in this dissertation. As will be discussed in Chapter 8, malware detection could passively analyze files or applications to find encountered breaches or malicious activities. This also applies to all other applications discussed in Chapter 9.
- **Majority efficient participants:** CrowdFAB *assumes that the majority of agents, miners, and validators are honest. Dishonest members are a minority.* This assumption allows for boosting the reputations of honest agents and makes the decision-making more efficient. Dishonest or incompetent agents have their reputations reduced. Note that we do not require that the agents make correct responses all the time, but the majority of them make correct responses at a time.
- **Reputation rather than money reward:** We *assume no monetary rewards and limit the rewards in terms of reputations and popularity in the system*. We argue that reputations and popularity benefits are as critical as monetary rewards in decision-making systems. Further, most existing crowd-forecasting decision-making systems are free of charge, especially those targeting security applications and malware detection such as Virustotal [34].
- **Enough storage and processing power:** Finally, *we assume that blockchain nodes (miners and validators) have enough storage/processing power to perform blockchain operations* and hold the chain when needed. In addition, they should have enough storage for agents'

reputations and computational power to calculate the summary. This assumption applies to all nodes in the blockchain layer, while nodes in the crowd-forecasting layer do not necessarily need these capabilities.

For the security of CrowdFAB, we assume that the underlying blockchain uses elegant cryptographic techniques for transactions and block signatures. In other words, *we assume that the blockchain is secure against crypto-breaking attacks* that try to change the signature or change the transaction data while keeping valid signatures. Elliptical curve cryptography (ECC) is used in most blockchain platforms, and that is currently difficult to break [137]. We *also assume that the' private keys of nodes are securely stored and resilient against hardware attacks.*

For the privacy of agents and requesters participating in the system, *we only require pseudo-anonymity as provided by the underlying blockchain.* Quorum and Corda blockchain platforms have additional privacy guarantees, hence, they are more suited for CrowdFAB. However, providing fully private and encrypted data is not an objective of ours. That said, in Chapter 9, we will discuss how to choose a summary function that provides knowledge-summary for private and encrypted agents' responses or decisions.

6.4.4. Threat Assumptions

Attackers in the system can use different roles to launch threats or malicious activities that maximize their benefits in terms of reputation or popularity. In the following, we state our threat assumptions by elaborating on how different roles can impose various threats on the system.

- *We consider malicious agents who try to maximize their reputations by manipulating the knowledge summary to match their response*. This can be done either individually or

collaboratively with other agents. Individually, they can provide one response that manipulates the knowledge summary if an inefficient summary function is used. As discussed in Chapter 3, the summary function that summarizes response transactions should be elegant and not easy to manipulate. Further, agents can send many response transactions for the same request to dominate the summary with their responses. Multiple agents may also collaborate to match their responses that falsify the summary.

- Underlaying blockchain roles that include miners and validators are also considered in our threat assumptions. We *consider malicious miners who maximize their profits by colluding with other requesters or agents to break the system operations.* They can also individually try to force a non-desirable summary by proposing falsified blocks. Note that here we only care about the threats related to crowd-forecasting. Other malicious miners' threats are platform-dependent, and some of them have been discussed in [138]. *However, we do assume correct miner operations in the case of leader-based consensus algorithms.* These operations do not include the generation of a correct knowledge summary generation, as this is to be validated by validators. But they include bundling all submitted transactions and not discriminating agent responses by not bundling their transactions. In other words, *we do assume honest miners who are curious to manipulate the knowledgeable summary* by generating a falsified one.
- Similarly, *validators in the system can collude with others to enforce a specific knowledge summary.* Again, we only consider the threats related to crowd-forecasting, i.e., the ones associated with colluding with others to fail CrowdFAB operations either by manipulating the summary or imposing wrong agents' reputations.

It should be noted that we do exclude some threats from the system. First, *a malicious agent may maximize its reputation by delaying its response.* In this case, the agent can calculate the summary after many responses have been made and use it as its submitted response. These agents are generally called *curious agents*. Privacy-preserving protocols such as multi-party computations can help to resolve this issue. However, due to our earlier privacy assumptions and the non-harmful nature of curious agents, we will not consider this a threat.

Further, *we do not consider Sybil attacks in our threat assumptions.* In such attacks, attackers can join the network with a considerable number of accounts and submit false responses [28]. This can lead to falsified knowledge summary as these "fake" agents will dominate the submitted responses and manipulate the summary to their desire. Such an attack can be avoided using identity authentication or monetary bids for each response made. Identity authentication validates the agents' legibility before they can join the system. This leads to permissioned blockchains assumption as each agent will join only by permission that validates its legibility. Monetary bids can be used for permissionless blockchains, which can follow already available blockchain bidding solutions [139].

To generalize CrowdFAB regardless of the underlying blockchain, we do not address this attack and consider it a future direction. That said, we will be implementing CrowdFAB on top of Quorum which is a permissioned blockchains. Thus, our implemented solution is protected against Sybil attacks by identity authentication providing by permissioned systems.

6.4.5. Security Goals

Our CrowdFAB framework should be resilient against summary manipulations and reputation manipulation. We define these two security properties followed by an overall security goal as follows:

Definition 1. Knowledge summary manipulation resiliency.

The system is resilient against knowledge summary manipulations if ***it can prevent*** the occurrence of the following conditions:

- An agent controls the knowledge summary by sending one or multiple false responses
- Several agents collaboratively agree on one false response that dominates, and thus controls, the knowledge summary
- Miners manage to force a random falsified knowledge summary that favors an agent where it should not happen.
- Validators collaboratively validate an incorrect knowledge summary or invalidate a correct summary generated by a miner.

Definition 2. Reputation manipulation resiliency:

CrowdFAB system rewards agents by increasing their reputations, and thus popularity in the system. The system is reputation manipulation resilient if:

- Miners and validators cannot decrease agents' reputations, thus, they deny agents' reputations and individual responses.

Definition 3: Full fraud-resiliency:

A full fraud-resilient system is a system that satisfies both of the following conditions:

- It is summary manipulation resilient
- It is reputation manipulation resilient

For this dissertation, we target ***full fraud-resiliency as a security goal for CrowdFAB***. The discussion on how CrowdFAB meets this goal and other security properties will be in Section 6.6.3. of this chapter.

6.5. CrowdFAB Specification and Implementation

In the following, we define the specification of how CrowdFAB works based on its dataflow, summary functions for knowledge-based blockchains, and the reputation scheme used from RPMC-EWA. Then, we discuss the CrowdFAB algorithm along with its implementation on top of Quorum blockchains. This will help in CrowdFAB analysis that will be presented later in the chapter and Chapter 8.

6.5.1. CrowdFAB Data Flow

The proposed data flow for CrowdFAB can be formulated, as shown in Figure 6.2. Requesters, agents, miners, and validators do not necessarily trust and possibly compete with each other. Thus, as seen in the figure, a trust boundary that defines the trust zone between nodes exists. Members of each zone have to ensure that members of other zones are not manipulating the results.

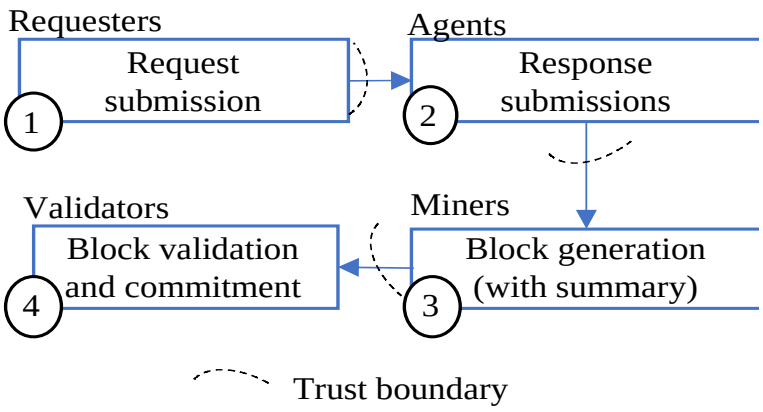


Figure 6.2 CrowdFAB data flow

The numbers in Figure 6.2 indicate the four steps in CrowdFAB. Its main functionalities lie in Steps 3 and 4 as these two steps build the underlying infrastructure for crowd-forecasting applications. In the following, we detail these four steps of the data flow process.

1. **Requests' submissions:** In the first step, a requester Q submits the request in a request transaction T_i. This transaction is broadcast to all nodes in the blockchain and will be visible to all agents in the system. A request may include a deadline which the knowledge summary should meet, and no further response will be validated.
2. **Responses' submissions:** Upon receiving T_i, an interested agent N_j, will evaluate the request and broadcast a response transaction D_{ij} that includes its response along with a probability P_{ij}. Different agents would assess the same request and return their response transactions, all together forming the set of agents, $\boldsymbol{N_i}$, and a response transactions set $\boldsymbol{D_i}$. These will also be visible to everyone else in the system.
3. **Block and summary generation:** A miner, M_k, forms blocks from the submitted response transactions in this step. The formed block will also include the knowledge summary C_i in its

headers. For example, the formed blocks can include the mean of the set of probabilities $\boldsymbol{P_i}$ for all request transactions $\boldsymbol{T_i}$s included in the block.

4. **Block validation and commitment:** The formed block is broadcast and validated by validators, $\boldsymbol{V}$, as discussed earlier. If the block and its knowledge summary are valid, it will be added to the chain, making the summary visible to all nodes.

6.5.2. Summary Function in Knowledge-based Blockchains

As discussed earlier, each agent's response is broadcast as a response transaction in CrowdFAB. Each request T_i will have a vector of response transactions $\boldsymbol{D_i}$ along with probabilities $\boldsymbol{P_i}$. A summary function, G, will be used by miners and validators to summarize $\boldsymbol{D_i}$ for a particular request. In Table 6.1, we referred to this knowledge summary earlier as C_i, were

$$C_i = G(\boldsymbol{D}_i).$$

G should meet the earlier summary function requirement discussed in Chapter 3 (specifically Section 3.3.2) [14]. The function that will be for the implementation in this chapter is the mean, which summarizes the probability vector $\boldsymbol{P_i}$ constructed from all responses to the i^{th} request by taking their mean. This function will be used along with their weighted version, where the weights are agents' reputations calculated by the RPMC-EWA reputation scheme discussed in Chapter 4 and highlighted next.

6.5.3. RPMC-EWA Reputation Scheme

CrowdFAB builds its reputation scheme based on the RPMC-EWA reputation scheme discussed in Chapter 4. Specifically, we donate the j^{th} agent's reputation at time t by R_{jt}. However, for simplicity, we omit the subscript j and write it as R_t here. The reputation based on Equation 2 from Chapter 4 can be updated as follows:

$$R_t = \frac{\begin{cases} 0 & p{=}0,\ n{=}0,\ t{=}0 \\ \alpha\dfrac{p}{p+n} + (1-\alpha)(2R_{t-1}-1) & \text{correct response} \\ \beta\dfrac{-n}{p+n} + (1-\beta)(2R_{t-1}-1) & \text{False Positive} \\ \gamma\dfrac{-n}{p+n} + (1-\gamma)(2R_{t-1}-1) & \text{False Negative} \end{cases}}{2} + 0.5$$

where α, β, and γ are configurable parameters for the increase in reputation when a correct response is made, decrease where a false positive (FP) error is made, and decrease when a false negative (FN) error is made, respectively. *p* and *n* are the numbers of correct and incorrect predictions, where the generated summary is compared to the agent's decision. An agent response is set to 1 (e.g., malware class in malware detection example) if his/her response's probability is greater than 0.5; otherwise, its response is 0.

Note that we are dealing with 1's and 0's rather than the probability or closeness of the prediction. An alternative approach can compare the probability in knowledge summary to the individual agent's probability and use fractional probabilistic *p* and *n*. However, as reputations can be generated by comparing individual agent's responses to ground truths with no probabilities, we consider the mapping of probabilities rather than exact probabilities. It should be emphasized that this step is only for the reputations. The individual responses, along with the knowledge summary, are still probabilistic.

Unlike traditional crowdsourcing schemes that rely on third parties, CrowdFAB, and other blockchain-based crowdsourcing applications, should implement the reputation within the blockchain system. In CrowdFAB, $\boldsymbol{R}$ (reputations R_j for all agents) is used by miners when summarizing responses and validators when validating the summary. It can also be visible to

requesters if they want to verify the summary and agents who can use it to enhance their future performance. Therefore, ***R*** should either be recorded in the blockchain or more likely kept on an off-chain decentralized database with a pointer to it and a hash-like change protection mechanism stored in the blockchain. In either case, the database contains *p, n,* and R_j for each agent.

For this work, reputations are stored off-chain in a decentralized database (maintained by miners and validators) and a hash-like change protection mechanism stored in the blockchain. They can be updated by miners and validators each time a knowledge summary is committed.

6.5.4. CrowdFAB Algorithm

CrowdFAB framework implements the data flow presented in Figure 6.2. In the CrowdFAB framework, Step 1 and Step 2 are request and response transactions collection, and they can be done using the knowledge-based blockchains discussed in Chapter 3. So, our focus here for CrowdFAB is on Step 3 and Step 4. The framework consists of five functions that are discussed below with a pseudo-code provided in Algorithm 6.1.

- **CalculateSummary function:** A function that takes all response transactions $\boldsymbol{D}_i$s (i.e., $\boldsymbol{D}_i$ for each T_i) in a block and returns their knowledge summary. It first gets the reputations for agents in these response transactions using the GetReputation function that is discussed next. Following that, it extracts all requests *(i's*) included in $\boldsymbol{D}_i$s by calling the CollectReqID function and gets the prior summary of these requests if present. To do so, the CalculateSummary function calls the GetPreviousSummary function to check if the request has been summarized in the last 50 blocks, where 50 is an implementation choice that can be adjusted. Finally, the CalculateSummary function generates a new summary using the prespecified summary function, the latest transactions, and the prior summary if it exists.

- **UpdateReputation function:** This function is responsible for updating the reputations of participating agents after the summary has been made. It takes the summary and $\boldsymbol{D}_i$s and does not return any variable. For each agent, it compares the generated summary to the agent's response and calculates the reputation. The reputations are then updated in the off-chain database along with *p* and *n*, if applicable.
- **GetReputation function:** A simple function that takes the list of response transactions and returns updated reputations. The CalculateSummary function uses this function to calculate the next summary that involves agents' updated reputation as weights, as discussed later in the case study.

Algorithm 6.1 CrowdFAB Framework Pseudo-code

Input: $\boldsymbol{D}_i s$*(all* response transactions, i.e., $\boldsymbol{D}i$ for each T_i), *S*(Request transactions)
Output: New valid block added to the chain

Function CollectReqID ($\boldsymbol{D}_i s$)
#Get the requests that have some responses
For *i* in $\boldsymbol{D}_i s$
ID.append(T_i.*ReqID*)
Return ***ID***

Function GetPreviousSummary (ReqID)
Blocks ←Get last 50 blocks in the blockchain
For *B* in *Blocks*
Cast.ReqID is stored in the block header
Cast is C_i for all i
ID.append*(B.Header*.Cast.*ReqID)*
For *id* in *ReqID*
If *id* in ***ID***
Summaries.appned(*id*, Cast.*Summary)*
Return *Summaries*

Function GetReputation ($\boldsymbol{D}_i s$)
For *d* in $\boldsymbol{D}_i s$
R_{agent} ←Get R_j for *d.ID* from the off-chain database
R.append(R_{agent})

```
    Return R

Function CalculateSummary (Dis)
    R←GetReputation(Dis)
    ReqID←CollectReqID (Dis)
    previousSummary←GetPreviousSummary(ReqID)
    For i in Dis #i is all the requests in Dis
        If d.ReqID in previousSummary
            Summary←previousSummary(d.ReqID)+G(R, Di)
        Else
            Summary←G(R, Di)
    #Summary has ReqID and the summary based on G
    #Summary Example is [ReqID, mean]
    Return Summary

Function UpdateReputation (newSummary,Dis)
    For d in Di
      If d.vote==newSummary(d.ReqID)
        # increase the agent's positives in off-chain database
        p← p+1
      Else
        # increase the agent's negatives in off-chain database
        n← n+1
    Rt←Calculate Rt from Section 5.2

Function ConstructBlock(Dis, S)
    NewSummary ←CalculateSummary (Dis)
    UpdateReputation (newSummary,Dis)
    newBlock.Header.Cast←NewSummary
    #Conventional Block construction process

Function ValidateBlock(newBlock)
    ValidSummary←CalculateSummary(Dis)
    UpdateReputation (ValidSummary,Dis)
    If ValidSummary!= newBlock.Header.Cast
        return false
    Else
        #Conventional validation process
```

- **ConstructBlock function:** A function that is run by miners (Step 3 in Figure 6.2) to form the block from the available transactions. It takes all available request and response transactions and does not return any variable. It first summarizes the response transactions using the

CalculateSummary function. Following that, it updates agents' reputations using the UpdateReputation function. The new summary, referred to as *Cast* in algorithm 6.1, is added to the new block header. Then, it follows the conventional blockchain process, bundling all transactions together to generate a new block.

- **ValidateBlock function:** a function that is used by validators to validate the newly generated block, Step 4 in Figure 6.2. It takes the new block and returns if the block is valid or not. It calls the CalculateSummary function to calculate the summary of transactions included in the new block. Then, it compares the generated summary to the summary in the block header. If the summary is incorrect, it returns false. Otherwise, it follows the normal blockchain process for validation.

6.5.5. CrowdFAB Implementation

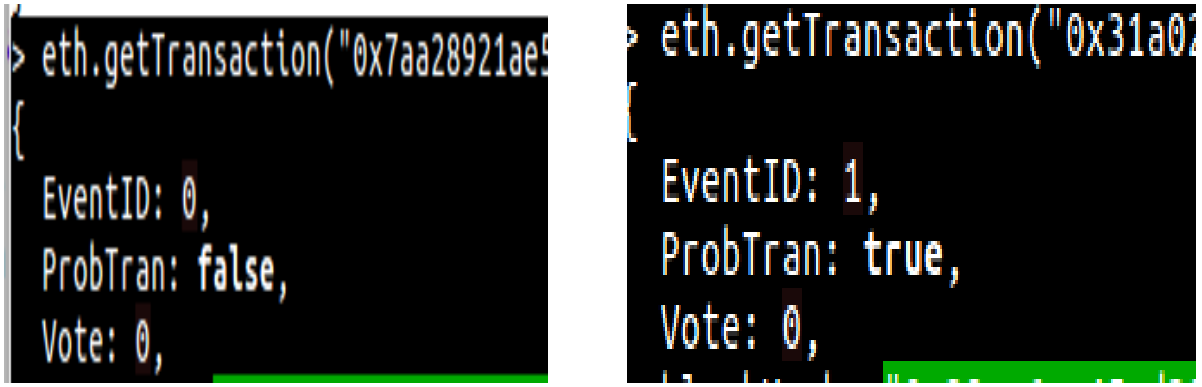
We extended the traditional Quorum blockchain to a Quorum-based CrowdFAB. A brief disruption of Quorum has been presented in Chapter 5, while the detailed discussion of Quorum can be referred to in [103]. The implementation in this chapter aims to determine the feasibility of CrowdFAB and the overhead it introduces over traditional Quorum.

We first extend Quorum to knowledge-based blockchain Quorum, where transactions and block architectures have been modified as follows.

At the transaction level, a Boolean, an integer, and a float variable are added to the transactions' data. The Boolean variable defines if the transaction is a request or a response transaction. The integer, *ReqID* (or *EventID* in the following figures), is a reference number for the request and will be used by agents in their responses. The vote is the response made by agents, which may be probabilistic.

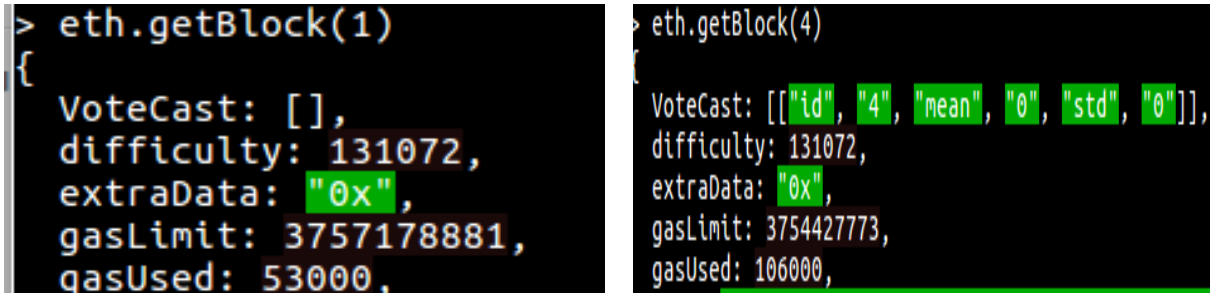
A list of summaries is added to block headers. The list (VoteCast in the following figures) in block headers would be the knowledge summary of response transactions, if there were any. This summary is prepared by miners and validated by validators.

Screenshots of a request transaction, a response transaction, a block with only request transactions, and a block that has some response transactions are shown in Figure 6.3 and Figure 6.4.



a. A request transaction *b. A response transaction*

Figure 6.3 Screenshot of request and response transactions



a. A Block without response transactions *b. A Block with some request transactions*

Figure 6.4 Screenshots of block with and without response transactions

We implement Algorithm 1 with a summary function of *ReqID*, mean, and standard deviation tuple. The ConsturctBlock and ValidateBlock functions extend the conventional block generation

and validation processes, respectively. Meanwhile, CalculateSummary and its associated functions have been added as additional functions used by miners and validators.

Even though they are currently not used by the summary function, the UpdateReputation and GetReputation functions are implemented to check agent reputations, as discussed in the next section.

6.6. Performance and Security Analyses

Now that CrowdFAB has been implemented, we can analyze its performance. The first analysis will cover CrowdFAB overhead, which compares Quorum-based CrowdFAB to conventional Quorum in terms of transaction delay and throughput. We also present an analysis of the reputation scheme for CrowdFAB, showing how an agent's reputation changes in different scenarios. Finally, we discuss CrowdFAB security features proving that the framework is fully fraud-resilient.

6.6.1. Analysis of CrowdFAB Overhead

The first analysis measures the overhead of Quorum-based CrowdFAB compared to the conventional Quorum. This is done in terms of throughput and transaction delay (standard transaction delay in Quorum and response transaction delay in CrowdFAB). In other words, we measure how many transactions can be made per second and how fast a knowledge summary can be seen in blocks. The details of throughput and delay definitions have been presented in Chapter 5.

Similar to the work presented in Chapter 5, we have used Blockbench as a benchmark to measure the performance with multiple settings [84]. The chosen consensus algorithm for these analyses is Raft, [104] as discussed before. The block generation time, or the time between two blocks, is set to the default 50 milliseconds.

It should be noted that Raft assumes an honest leader or miner who performs correct operations to generate the block. Thus, there is a general assumption to trust the leader. As discussed earlier, in the specific case of leader-based consensus algorithms, we assume that the miner is honest in including all transactions but curious in manipulating the summary. However, this is not a general assumption for CrowdFAB.

Our setup includes 16 blockchain nodes divided as follows: one miner, seven validators, one requester, and seven agents. We have used 16 Cloudlab virtual machines as blockchain nodes connected in a peer-to-peer fashion [106]. The other details of the parameters and factors are presented in Table 6.2.

Table 6.2 Parameters and Factors for the Experimental Design

Parameter	**Level**		
Number of nodes (factor)	**Total**		16
	Validators		7
	Clients		7
	Miner (leader based)		1
Geo distribution	local (Utah servers)		
Blockchain specific configuration	**Consensus algorithm (factor)**		Raft
	Block time (defaults)		50 ms
Hardware Resource	**CPU**	Eight-core Intel Xeon D-1548 at 2.0 GHz	
	RAM	64GB ECC Memory	
	Disk	256 GB NVM flash storage	
	Network	1 Gbps	
	NIC	Dual-port Mellanox ConnectX-3 10 GB NIC	

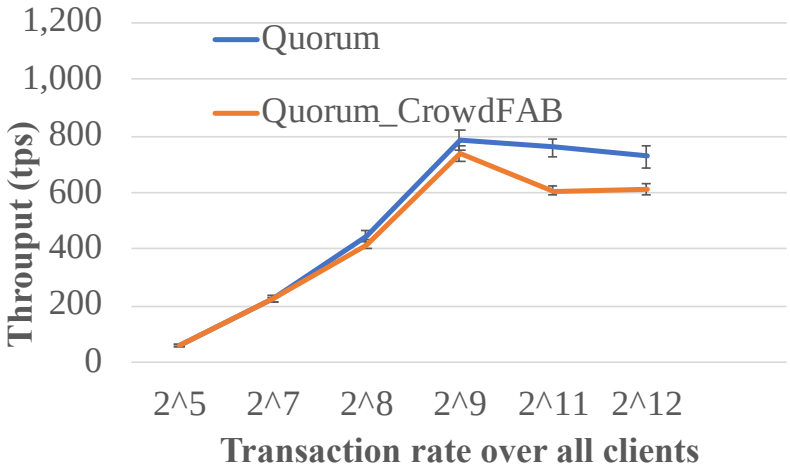
Transaction Rate (factor)	64, 128, 512, … 4094 (or 2^6, 2^7… 2^{12})
Transaction size	smallbank workload sizes
Number of transactions	Transaction rate * 120

We vary the transaction rate and measure the delay and throughput as discussed next. Each experiment has been repeated 35 times to estimate the variability and consistency of the results.

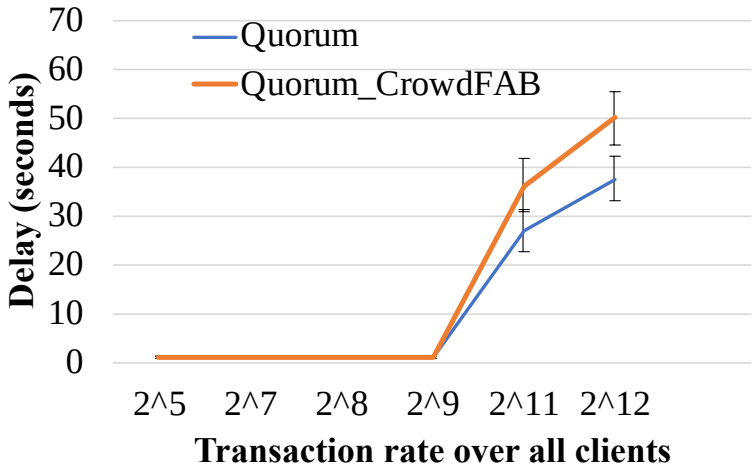
All the measurements are done at the requester node. The throughput is measured by summing the number of transactions available in blocks received in one second. The delay is measured by assuming that the difference between when the request appears in a block and when the first knowledge-summary appears, assuming that the first response transaction is made in one block after the request has been made. That is, we measure the delay between the request and the first response received at the requester node.

The results from the above experimental designs are presented in Figure 6.5. Comparing Quorum-based CrowdFAB to the conventional Quorum performance, as can be seen from the figure, the throughputs are very close at lower transaction rates with an overhead of than 1%. The throughputs in both cases increase steadily until they reach a limit of 512, or 2^9, transactions per second (TPS). This limit is the knee point, as explained before in Chapter 5.

After the knee point, CrowdFAB throughput is about 500-700 TPS while it is higher in the conventional Quorum. Thus, CrowdFAB overhead shows up after that knee performance, resulting in about a 20% decrease in throughput. This decrease is reasonable given the amount of processing in generating the knowledgeable summary when the number of response transactions is high. Note that the results here were done using an older version of Quorum, and so they will not match the results discussed earlier in Chapter 5.



a. Throughput analysis



b. Delay analysis

Figure 6.5 CrowdFAB overhead analysis on Quorum platform

The delays, as shown in Figure 6.5, behave similarly before the knee point, where they average about 1 second with an overhead of about 1%. After the knee, transactions delay increases rapidly for both Quorum and CrowdFAB. This is to be expected as the delay should be constant and minimal before the knee and increase exponentially after the knee.

CrowdFAB shows about a 30% increase in delay, jumping from about 30 seconds to 50 seconds at 2^{12} transaction rate. Again, such overhead is reasonable due to the amount of processing required in summarizing a larger number of transactions.

Overall, these results show that CrowdFAB has a throughput overhead of less than 20% and a delay overhead of less than 30%. This overhead is reasonable given the processing introduced by CrowdFAB. Nevertheless, it can be seen that the delay in generating the summary does not exceed one minute, or 60 seconds, which is typical for crowdsourcing applications [136].

Hence, the overhead that CrowdFAB introduces is minimal compared to its gains in providing the security guarantees needed for crowd-forecasting applications.

6.6.2. Agents' reputation Analysis

Next, we analyze RPMC-EWA in CrowdFAB to show different competent and incompetent agents' behavior/reputations. By trying different values for the configurable parameters detailed in general in Section 4.3.7 and highlighted for CrowdFAB in Section 6.4.3, we determined that the requirements can be best achieved by setting $\alpha = 0.05$, $\beta = 0.3$, $\gamma = 0.7$. As discussed before, α, β, and γ are respectively configurable parameters for the increase in reputation when a correct response is made, decrease where an FP error is made, and decrease when an FN error is made.

With $\alpha = 0.05$, the reputation increases slowly for a good agent, as shown in Figure 6.6. This slow increase helps avoid malicious behaviors where an agent may provide wrong responses randomly or in a specific pattern [16].

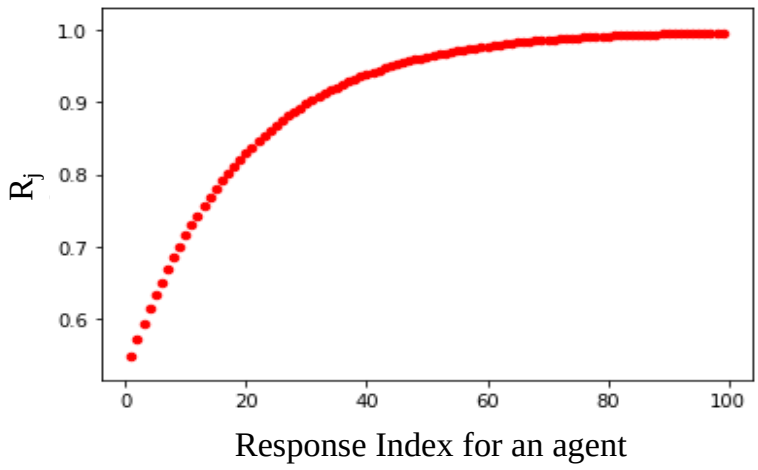


Figure 6.6 With α =0.05, the reputation of a competent agent increases slowly

By setting $\gamma = 0.7$, one can rapidly decrease the reputation when an FN mistake is made, which is desirable for security applications. As shown in Figure 6.7 at response index 3, the reputation has jumped from 0.6 to less than 0.1 with only two FN mistakes.

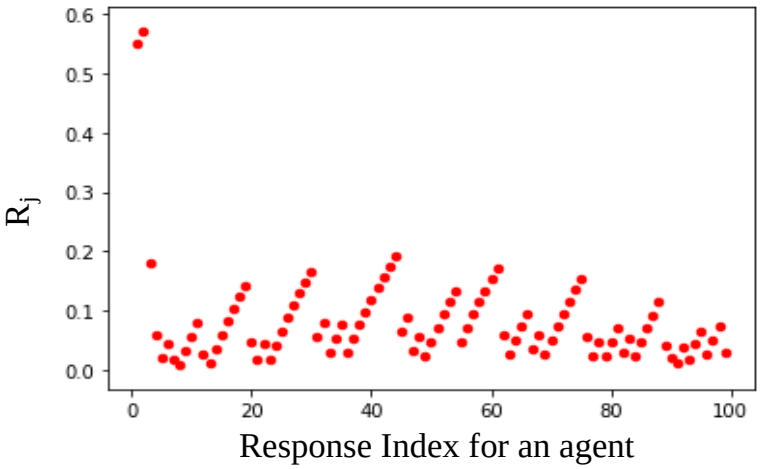


Figure 6.7 With γ=0.7, the reputation decreases rapidly on an FN mistake (at response index=3)

Meanwhile, with $\beta = 0.3$, the agent reputation can decrease moderately when an FP is made. As shown in Figure 6.8 at response index 7, the reputation has decreased from 0.6 to more than 0.2 and then increased steadily.

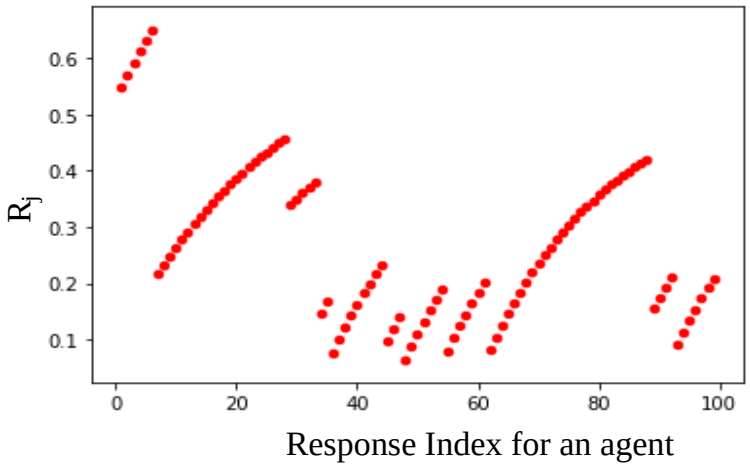


Figure 6.8 With β=0.3, the reputation decreases moderately on an FP mistake (at response index=7)

6.6.3. Security Analysis of CrowdFAB

CrowdFAB design fulfills our earlier security goals in addition to providing other security properties. We discuss these properties in the following.

1) Knowledge summary manipulation resiliency:

According to Definition 1, the knowledge summary can be manipulated if agents can individually or collaboratively send falsified transactions or collaborate with miners or validators to force a random falsified summary. Theorem 6.1 will analyze how CrowdFAB is resilient against summary manipulation, assuming an honest majority of agents.

Theorem 6.1. CrowdFAB is resilient against knowledge summary manipulation attacks

Proof. To prove this resiliency, we show that each of the four conditions discussed in Definition 1 (Section 6.3.5) cannot be met or may occur with a minimal probability.

Chapter 3 discussed some requirements on the summary function that includes it should not be easy to manipulate. With the use of a proper summary function that satisfies this requirement, an agent cannot control the summary by sending one or a few falsified transactions.

We have also assumed no Sybil attacks, so an agent cannot generate these falsified responses from multiple accounts, i.e., fake agents.

Let's assume that there are n good agents that responded correctly to a request. These responses were donated earlier by $\boldsymbol{N_i}$. To flip the summary, a malicious agent needs to submit at least n falsified responses, assuming that the mode is used as a most straightforward summary function. As n gets large, the probability of the malicious agent succeeding in launching the attack in a limited time goes to zero. Further, with the use of reputation schemes, the reputations will make it almost impossible to launch the attack for an extended time. Thus, one malicious agent cannot flip the knowledge summary.

The same reasoning applies to collaborative attack launches, where multiple malicious agents would send falsified responses. Again, as n gets large, the number of malicious agents required for the attack to succeed becomes large. Also, with proper reputation management, such attacks can be avoided.

Another way to manipulate the summary is by targeting the underlying blockchain security, i.e., by controlling miners or validators. Miners can generate wrong summaries; validators can invalidate a correct summary or validate an incorrect summary. These can be done by the CalculateSummary function in Algorithm 6.1, which is called by both miners (ConstructBlock function) and validators (ValidateBlock function). Since the summary is part of the validation

process, a wrong summary would result in an invalid block by the process of ValidateBlock function in Algorithm 6.1. Thus, the generated block is dropped, and the summary is not committed to the chain.

With the immutability feature of blockchains, changing the knowledge summary after it has been committed is also impossible or can occur with minimal probability, as many earlier studies have proven for different blockchains.

Thus, CrowdFAB is resilient against summary manipulation attacks. ■

2) Reputation manipulation resiliency:

According to our Definition 2 (Section 6.3.5), reputations can be manipulated if miners and validators can unreasonably decrease agents' reputations. In the following, we discuss how CrowdFAB can be resilient against such manipulation.

Theorem 6.2. CrowdFAB is resilient against reputation manipulation attacks.

Proof. Reputation manipulation can happen if miners and validators collaborate to decrease or increase the reputation. We have discussed that reputations are stored off-chain in a decentralized database (stored at miners and validators) and a hash-like change protection mechanism stored in the blockchain. They can be updated by miners and validators each time a knowledgeable summary is committed. Thus, reputation manipulation can happen either by changing the already stored reputations or when updating the reputations.

The change of already stored reputations is protected by the hash and blockchains' nature since blocks are signed, resulting in non-repudiation guarantees. In this case, reputation manipulation

cannot happen unless a node manages to manipulate the hash stored in the blockchain, i.e., changing the blockchain data. With public blockchains, this can happen by a "chain fork" that occurs if a valid but modified chain is longer than the healthy chain [140]. With the honest majority assumption, the success probability of malicious fork is very low [121]. For Quorum CrowdFAB implementation, one may change the hash if the miner (leader) is corrupted. As this invalidates our assumptions, CrowdFAB is resilient against changing the already stored reputations.

Falsified reputation updates lead to invalid blocks by the process of the CrowdFAB framework. As shown in Algorithm 6.1, the UpdateReputation function is called by miners and validators when calculating the summary (CalculateSummary function). False reputations by miners result in an incorrect knowledge summary, and hence an invalid block, as discussed in Theorem 6.1 and shown by ValidateBlock function in Algorithm 6.1. Therefore, CrowdFAB is protected against falsified reputation updates.

By proving that reputations cannot be changed while stored or updated, we can say that CrowdFAB is resilient against reputation manipulation attacks. ■

3) Full fraud-resiliency

By proving that CrowdFAB is summary and reputation manipulation resilient, we can say that the system is full fraud-resilient, according to Definition 3.

4) Security against free-riding attacks

A free-riding attack occurs when an agent attempts to increase its reputation without making proper responses, i.e., making random predictions. With the use of our reputation function, incorrect responses result in a substantial decrease in reputation. Further, the slow increase will

prevent the agent from even affecting the knowledge summary unless he/she makes many good responses in the future. Thus, CrowdFAB is resilient against free-riding attacks.

Note that an agent can predict this summary from the responses made so far, therefore, increasing its reputation without having a proper model. However, this requires the agent to collect responses and predict the summary. Since this action has a minimal effect on the knowledge summary, we do not consider it an attack.

5) No single point of failure and no trusted third party

With the decentralized nature of blockchain-based systems, CrowdFAB guarantees no single point of failure. Many nodes hold the blockchain database, which makes requests and responses available to query anytime. On the assumption that there are m (m >2) blockchain nodes (miners and validators) in CrowdFAB and more than $m/2$ are honest, there exists at least one node that can be accessed for CrowdFAB services. Note that reputations are stored in a distributed database maintained by different nodes. Thus, accessing and changing reputations is not a single point of failure.

The system has not trusted a third party, which again comes from the inherited design of blockchain systems. They work without requiring a trusted third-party.

6.7. Summary

Despite extensive adaptations of crowd-forecasting models, they suffer several challenges, including centralization, potential attack surfaces, and inefficient summarization. Existing blockchain-based solutions mainly focus on traditional crowdsourcing applications. Unlike traditional crowdsourcing applications that only communicate information, crowd-forecasting

applications need to communicate and intelligently process information to achieve the collaborative assessment. Hence, they require knowledge-based systems instead of simple storage-based systems needed for crowdsourcing applications.

By combining our proposed knowledge-based blockchains discussed in Chapter 3 and the reputation scheme discussed in Chapter 4, we presented the design of CrowdFAB, a novel framework for crowd-forecasting applications using blockchains. CrowdFAB uses knowledge-based blockchains as a novel blockchain paradigm that transforms blockchains from the storage system to knowledge and processing systems. Further, it builds a reputation scheme that assigns reputations to agents based on their past performance. This allows requests to be submitted to a crowd of agents who provide their responses without the need for a trusted third party. An efficient and secure summarization can be done in a fully decentralized manner.

The framework is applicable to any risk management or forecasting application, and we specifically targeted security assessment in this chapter. We implemented the CrowdFAB framework on top of the Quorum blockchain platform to show the feasibility and efficiency in terms of reasonable overhead. We further analyzed the security of CrowdFAB to prove that it is fraud resilient, i.e., satisfying our security goal, along with several other properties that are inherited by the use of blockchains or knowledge-based blockchains.

We expect CrowdFAB, and generally blockchain-based crowd-forecasting approaches, to grow as an alternative for traditional crowd-forecasting applications. In Chapter 8, we will discuss the use of CrowdFAB for a distributed malware detection use-case as a security risk assessment application. However, outside the scope of this dissertation, CrowdFAB can be used for other non-

security applications, including financial predictions and recommendation systems. Some examples of those will be provided in Chapter 9.

As a future work for this part of the dissertation, one could evaluate how CrowdFAB can be used for financial predictions and recommendation systems. In addition, analysis of CrowdFAB for public blockchains can be done and compared to Quorum-based CrowdFAB or other traditional blockchain systems. Finally, the system can be compared to other collaborative risk assessment approaches presented earlier in the related works.

Chapter 7:Safety Score as an Evaluation Metric for Security Risk Assessment Applications

The previous chapters have built the undelaying infrastructure that extends blockchains for decision-making and risk assessment applications. Thus, from this chapter and the forward, the research will focus on the second stage of the dissertation that targets security risk assessment applications.

To build intelligent security solutions, participating agents who underlie the blockchain-based security risk assessment should be intelligent. They are expected to develop their assessment models using state-of-the-art rule-based or AI-based algorithms. However, as discussed in Chapter 2, a major challenge is to evaluate assessment models built for security applications using a proper metric that reflects the actual performance in terms of cost. In this chapter, we present the work we've done to address this issue and propose Safety score as an evaluation metric suitable for security risk assessment applications. We show that conventional metrics, including accuracy and F_1 score, do not necessarily represent risks in security applications and may result in models that are not optimal. The proposed 'Safety score' is an evaluation metric that incorporates the cost associated with model predictions, making it easy to explain to system administrators. The discussion in this chapter focuses on evaluating a single agent model that uses ML to make its decisions. The next chapter will present the work in the targeted distributed setting.

7.1. Introduction and Related Works

Security risk assessment solutions are among computer science fields that have widely emerged to secure modern applications against threats and cyber-attacks. Traditional approaches in these fields heavily rely on rule-based and signature-based approaches that detect threats or attacks based

on predefined rules or well-known signatures. With the current continuously evolving and rapidly changing threats, these approaches fail to provide elegant and proper defense mechanisms. The high dimensionality and perpetual dynamicity of modern threats are among the leading causes why these traditional approaches strive to provide sufficient defense.

AI and ML have proven their undeniable benefits to resolve these challenges by defining insights from high-dimensional and perpetual dynamic data. Thus, even though they were not invented by these applications, security risk assessment applications are considered among the networking fields that have been revolutionized with these breakthrough algorithms. As we have discussed in Chapter 2, most state-of-the-art techniques in security risk assessment applications utilize AI and ML to provide elegant solutions.

Several evaluation metrics have been formulated to evaluate the performance of ML and AI models. The performance of a model is generally a function of how many mistakes it makes in its predictions. Some classical metrics, including true positive rate (TPR), true negative rate (TNR), focus on the mistakes made in one class, i.e., only the positive or the negative class. Other classical metrics, such as accuracy, base the evaluation on mistakes made in all classes.

Most current research in ML-based security applications heavily relies on these conventional metrics to evaluate models built for security applications. Accuracy, TPR, TNR, and area under the curve (AUC) are the most commonly used metrics [141, 142, 143, 144, 12]. However, as these metrics do not consider the difference in cost when making predictions, they fail to accurately evaluate the model performance in terms of cost, which will be detailed in this chapter.

Kim et al. have gone out of the traditional metrics and considered model efficiency as an evaluation metric [145]. They define efficiency as a function of FNR and FPR, or detection rate and false

alarm rate. The used metric for that specific work is simply a weighted multiplication between the FNR and FPR [145]. However, this metric still miss the security requirements that will be discussed later since it gives all classes the same weights.

Other traditional intrusion detection systems, i.e., without the use of ML and AI, have considered cost as an important metric for evaluation. For example, the minimization of the cost associated with attacks and begin traffics has been used as a metric in [146]. A function of operational, damage, and challenge costs has been used to evaluate [147]. The challenge with these approaches is that they do not have a relative measure on how good a model is, i.e., without giving a minimum or maximum cost to compare to.

7.2. Motivation

To evaluate an ML model built for a security application, two requirements should be considered. First, given that breach (or a security risk) and benign activities have different effects on the systems, the mistakes made in them should not be treated equally. Second, correctly classifying a sample in a class should not be the same as a mistake made by misclassifying in the same class.

To explain the rationality behind the previous two requirements, consider Capital One's data breach in March 2019. The breach, caused by a misconfigured Firewall that an attacker exploited [148], exposed about 80,000 linked account numbers costing the company more than 100 million dollars [149]. The company could have saved millions of dollars if the exploit was detected and mitigated. Similarly, incorrectly labeling a benign activity as an exploit would have cost drastically less than the 100 million dollars.

Note that the cost here represents the loss or damage due to misclassification. Hence, when evaluating a model built for a security application, the mistakes made in detecting a breach (or a risk) activity should penalize the performance much more than others.

None of the earlier discussed metrics meets the two requirements for security applications, namely missing breaches should penalize the model's performance more than benign misses and correct breach detections. This motivates us to propose a meaningful assessment metric of an ML model built for security risk assessment applications.

7.3. Contributions

The contributions of this part of the research are as follows:

1. First, by using illustrative examples, it is shown that conventional evaluation metrics fail to efficiently evaluate the performance of a model designed for security risk assessment applications. These metrics generally underrepresent the loss or damage that a model would cause when making security decisions.
2. Safety score is then proposed as an evaluation that is tailored for cost-sensitive security risk assessment applications. The goal of the new metric is to reflect the risk associated with models' correct or incorrect predictions in the performance indicators. The metric can be adjusted for different security applications ranging from national defense to patient care.
3. ML models designed for two security risk assessment applications, namely general intrusion detection and injection attack detection in healthcare networks, are then evaluated. Results, shown by examples and IDS dataset evaluations, prove the efficiency of Safety score in representing the actual models' performance in terms of costs or system effects.

The work presented in this chapter has been published in [18].

7.4. Conventional Evaluation Metrics

In the following, we provide a brief background on the ML model prediction outcomes and some of the commonly used evaluation metrics. This background will facilitate the discussions in this chapter, where we illustrate the challenges and propose Safety score as a new metric.

7.4.1. ML Prediction Outcomes

An ML model can be trained for network intrusion detection. This model makes its prediction evaluating whether a network flow is an attack or not. We refer to this prediction as the model outcome for that flow. A model outcome can fall into one of four categories: true positive (TP), true negative (TN), false positive (FP), or false negative (FN).

For security risk assessment applications: TP is a breach sample that has been correctly detected; TN is a correctly classified benign sample; FP is a benign sample that has been categorized as a breach; FN is a breach sample that is classified as benign.

7.4.2. Accuracy

As discussed before, accuracy is the most commonly used metric that measures the rate of correctly classified samples in both classes. That is:

$$Accuracy\ \% = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

The challenge in using accuracy is the imbalance of data when testing security risk assessment applications in realistic settings. The number of benign samples dominates the data with sometimes more than 95%. Thus, even if the model missed all breaches or security risks, it is still 95% accurate.

7.4.3. Weighted Accuracy

Weighted accuracy is an extended accuracy metric that considers weights per class. For example, weighted accuracy for security applications considers higher weights for breach or attack samples. A weight w_i is assigned to each class such that the sum of all weights is 1.

In security risk assessment applications, there are two classes: breach and benign. We assign breach samples a weight, which is 100 times that of the benign samples. By normalization, breach samples will weight 100/101while benign samples will weight 1/101. Both weights sum up to 1.

Even though this metric can distinguish FP from FN, it treats FN the same as TP. Thus, it does not meet the second requirement discussed earlier for security applications. As will be seen throughout the chapter, this metric mostly underrepresents the risk of the model for security applications.

7.4.4. AUC

AUC computes the area under the receiver operating characteristic (ROC) curve, which visualizes TPR versus false alarm rate. The area under that curve can summarize the curve characteristics in a single value. A 0.5 AUC shows lousy performance, while 0.7 is good, and 0.85 is excellent [150].

Again, this metric can resolve the unbalancing issue with the dataset and provide a mid-range value if the model performs poorly with the minority class. However, it does not differentiate different model outcomes, thus, not suitable to represent the performance for security risk assessment applications

7.4.5. Precision, Recall, and F1 Score

Precision in security applications is the fraction of samples that a model has classified as security breaches that were actually breaches. The recall is the ability of a model to detect all breach samples. F_1 score is the harmonic mean between the recall and the precision. That is:

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

$$Recall = \frac{TP}{TP + FN}$$

$$F_1\ score = \ 2 * \frac{Precision\ \times Recall}{Precision + Recall}$$

For the rest of this chapter, we will be using only the F_1 score as a representative of these three metrics. It should be noted that the discussion on the F_1 score applies to precision and recall as well. Recall gives lower metric values in most of the presented cases; however, the challenge in meeting the security requirements still holds for all these metrics.

7.5. The Challenge and The Proposed Metric

Now that we have discussed the frequently used metrics, we can illustrate the common challenges in using these metrics for security risk assessment applications and propose Safety score as a metric that overcomes these challenges.

7.5.1. An Illustrative Example

Consider a test dataset for a security application with 100 samples, consisting of 20 breach samples and 80 benign samples. Two ML models are used to label these samples as a breach or benign. Model 1 misclassifies one benign sample as a breach, while Model 2 misclassifies one breach

sample as benign. That is, Model 1 has a 1.25% error rate in the benign class, and Model 2 has a 5% error rate in the breach class. Table 7.1 shows how the conventional metrics would compare the performance of these two models.

Table 7.1 Illustrative example evaluation with traditional metrics

	Model 1	**Model 2**
Classification Outcome		
TP	20	19
TN	79	80
FP	1	0
FN	0	1
Performance Statistics		
Breach error rate	0.00%	5.00%
Benign error rate	1.25%	0.00%
Cost ratio	1.075	1.816
Evaluation Metrics		
Accuracy	0.990	0.990
Weighted accuracy	0.999	0.952
AUC	0.994	0.975
F_1 score	0.976	0.974

As can be seen, accuracy is a bad metric of choice since it treated both 5% and 1.25% mistakes equally. The weighted accuracy, F_1 score, and AUC have panelized Model 2 more than Model 1, with weighted accuracy having a higher penalty. In this case, weighted accuracy is the best in differentiating between the two models.

Even so, none of the metrics has more than a 0.04 decrease (a tiny 4% reduction) with Model 2 misclassifying 5% (a large fraction) of the breach samples. As missing a breach can result in excessive loss, a model with a 5% breach error rate is highly undesirable for security applications.

7.5.2. The Performance in Relation to Cost

Based on the earlier Capital One data breach example, the cost of (or loss caused by) misclassifying a breach is 100 million dollars. Let us assume that correctly classifying a breach (detecting a breach and taking countermeasures) would cost two million dollars to counter that breach. To penalize misclassifying a benign sample more than correctly identifying a breach, we assume that its cost is ten million. Correctly classifying a benign sample costs one million for keeping the system running. In terms of model outcomes, TN costs $1M, TP costs $2M, FP costs $10M, and FN costs $100M.

Applying these costs to our illustrative example, a perfect model costs $120M, Model 1 costs $129M, and Model 2 costs $218M. Thus, the cost ratio of Model 1 would be 129/120, which is 1.075, while it is 1.816 (218/120) for Model 2.

None of the metrics in Table 7.1 represents such an increase in cost since they showed a maximum of 4% difference when the cost ratio increased from 1.075 to 1.816. However, the cost is a crucial factor in assessing any application deployment.

7.5.3. Proposed Safety Score Metric

In our following discussion, we assume a linear relationship between the loss (or the cost) and the model outcome. If one FN costs $100M, two FN will cost $200M. This assumption is based on the belief that breaches are independent, and two breaches will result in twice as much loss as one

breach. Other relations, such as exponential or squared, may be assumed for other applications. Our proposed solution approach will still apply.

We define our proposed Safety score as a function of the cost associated with each model outcome. Safety score extends accuracy by giving weights to each outcome based on its cost. If w_x is the weight associated with outcome *x*, Safety score is:

$$Safety\ score = \frac{w_{TP}TP + w_{TN}TN}{w_{FP}FP + w_{FN}FN + w_{TP}TP + w_{TN}TN}$$

As can be seen, the weights in Safety score are adjustable, allowing different applications to utilize the metric with their own costs. Note that higher Safety scores indicate better models.

7.5.4. Weight Assignment Procedure

To assign weights for Safety score, a network administrator needs to estimate the costs of each of the four possible outcomes. These costs can then be mapped to weights that are assigned to each FN, FP, TN, and TP outcome in evaluation. To do so, we divide the cost of an outcome by the sum of all costs.

For our example, the four outcomes costs are as follows: FN costs \$100M, FP costs \$10M, TP costs \$2M, and TN costs \$1M, respectively. The sum of all costs is \$113M. Therefore,

$$w_{FN} = \frac{100}{113}, w_{FP} = \frac{10}{113}, w_{TP} = \frac{2}{113}, \text{and } W_{TN} = \frac{1}{113}$$

Going back to our example, the Safety score for Model 1 is 0.922, while it is 0.541 for Model 2. Hence, Model 2 performs much worse than Model 1. The cost ratio of Model 2 to Model 1 is 1.816:1.075, which is similar to their Safety scores ratios 1/0.541:1/0.922.

Thus, the Safety score is easy to explain to system administrators who care for the cost of security.

7.5.5. Extending the Illustrative Example

To extend the illustrative example with a higher number of mistakes, we add two more models. Model 3 makes ten benign mistakes, and Model 4 misclassifies ten breaches. The statistics and metrics for the four models are presented in Table 7.2.

Table 7.2 Extended illustrative example evaluation

	Model 1	**Model 2**	**Model 3**	**Model 4**
Classification Results				
TP	20	19	20	10
TN	79	80	70	80
FP	1	0	10	0
FN	0	1	0	10
Performance Statistics				
Breach error rate	0.00%	5.00%	0.00%	50.00%
Benign error rate	1.25%	0.00%	12.50%	0.00%
Cost ratio	1.075	1.816	1.750	9.167
Evaluation Metrics				
Accuracy	0.990	0.990	0.900	0.900
Weighted Accuracy	0.999	0.952	0.954	0.519
AUC	0.994	0.975	0.930	0.750
F_1 score	0.976	0.974	0.800	0.667
Safety score	0.922	0.541	0.524	0.090

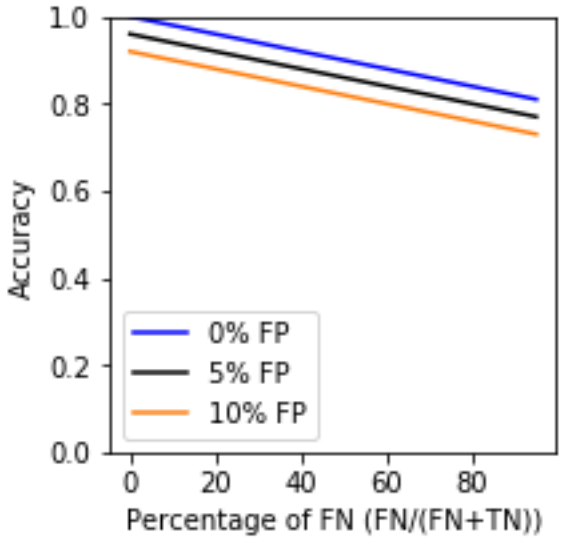
As seen, even with a 9.1667 cost ratio, i.e., more than 800% increase in cost, the decrease in the performance indicators is less than 40% for most conventional metrics. The weighted accuracy shows a 0.5 decrease, which is reasonable for this specific case. However, when the breach error rate is significantly lower, which is typical in ML models, weighted accuracy fails in giving reasonable measures in terms of the cost increase. This failure will be shown later in this chapter.

Safety score has gone down to less than 0.1, which represents the high amount of damage caused by Model 4. Model 4 has a 50% error rate in the breach class, which defeats the objective of any

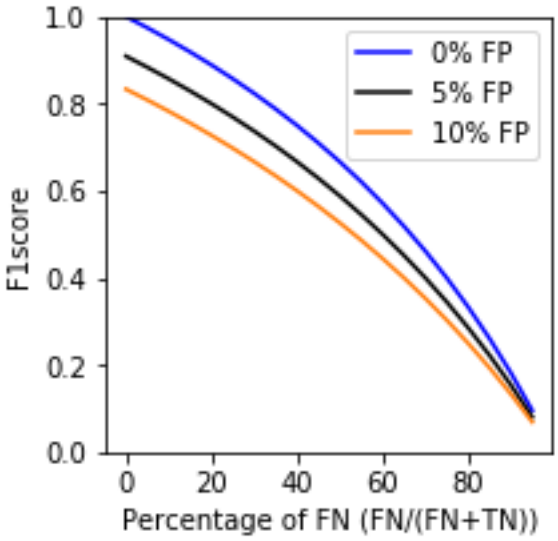
model designed to detect security risk assessments. Thus, the model should have very low-performance indicators.

7.5.6. Metrics Virtualization with Different FN Rates

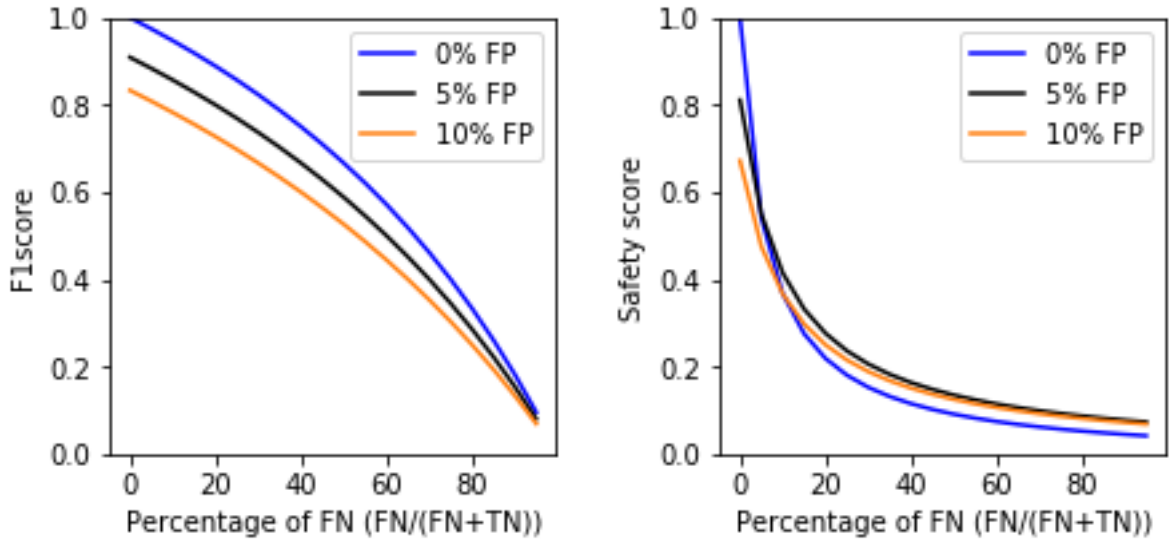
Figure 7.1 illustrates the effect of FN and FP rates on the metrics. When the model makes a 50% FN rate, conventional metrics are all above 0.5. On the other hand, Safety score rapidly decreases as the FN rate increases, resulting in a lower than 20% score when the FN rate is higher than 40%. That is, the Safety score is representing the cost accurately since a 40% FN rate implies a model that is not suitable for security risk assessment applications. Note that the weights here are the same ones discussed earlier.



a. Accuracy



b. AUC



c. F_1 score *d. Safety score (Proposed)*

Figure 7.1 The change in metrics as the percentage of FN increases

7.6. Case Study 1: General Breach Intrusion Detection

Our first evaluation case study targets a general IDS)as a security risk assessment application. We first discuss the used dataset, the ML models, and the weights setup. Then, we evaluate the used ML models using the above metrics to show the Safety score's outstanding performance.

7.6.1. Datasets Used

We use the UNSW-NB15 dataset that was discussed earlier in Section 3.5.3 for this case study. The dataset has 45 features extracted from network traces and nine types of breach traffic in addition to the benign traffic [151]. For our evaluation, we have used only the DoS breach along with the benign samples. This led to a smaller dataset that closely represents a real-world DoS IDS. The statistics of the dataset used are presented in Table 7.3.

Table 7.3 Dataset statistics of UNSW-NB15 DoS attack

Type of traffic	Total number of samples	Percentage
Benign	90191	86.58%
DoS breach	13982	13.42%

7.6.2. ML Models and Weights Setup

ML algorithms either return the classification output, i.e., if a sample is a breach or not, or returns the probability of a sample falling in a class. Following the general evaluation strategy, we have mapped these probabilities to a class by considering a specific threshold. Thus, when calculating the prediction outcomes, we compare the prediction outputs to the ground truths, i.e., the correct class. This is what has been done for all metrics.

However, it is possible that a 90% probability for a sample being a breach results in a different cost than 80% probability. To handle such cases, the Safety score metric can be extended easily by considering the probability of a sample falling in a certain outcome and using that to calculate that sample's cost.

As an example, consider a prediction of 20% attack and 80% benign where the ground truth is benign. In this case, the sample has outcome probabilities of TN=0.8, FP=0.2, FN=0, TP=0. These probabilities are then multiplied by outcome weights to get the cost of a sample. The Safety score is the sum of costs for all samples. This example assumes a linear relationship between the probability of an outcome and its cost. It is easily extended to non-linear relationships if those relationships are specified.

We used logistic regression (LogR) [10], random forest (RF) [59], and K-nearest neighbors (KNN) [152] to build three learning models that detect DoS breaches from the dataset. Models are trained

on 75% of the dataset and the rest is used for testing. The objective here is not to build perfect performing models but to illustrate the metrics change with different algorithms performance.

The discussion in the illustrative example costs applies to this case study as well. Thus, the weights used for Safety score are the same ones discussed earlier.

7.6.3. Results and Analysis

The performance of the models discussed above is illustrated in Table 7.4. As can be seen, even with more than 200 cost ratio, i.e., more than a 100% cost increase, the conventional metrics give an indicator of 0.7 or more. On the other hand, Safety score has gone below 0.5, which shows a better representation of the cost.

Table 7.4 Statistics and results for UNSW-NB15 DoS dataset

	RF	**KNN**	**LogR**
Classification Results			
TP	3393	3318	3297
TN	22407	22011	20948
FP	167	563	1626
FN	77	152	173
Performance Statistics			
Breach error rate	2.22%	4.38%	4.98%
Benign error rate	0.74%	2.49%	7.20%
Cost ratio	1.307	1.676	207.030
Evaluation Metrics			
Accuracy	0.990	0.972	0.930
Weighted Accuracy	0.979	0.957	0.949
AUC	0.985	0.966	0.903
F_1 score	0.965	0.903	0.785
Safety score	0.757	0.579	0.450

Note that 0.45 is still high compared to LogR's bad performance and its high cost. Similarly, the decrease in performance between KNN and LogR is less than 0.15, which is negligible compared

to the jump in cost ratio. However, these increases are due to the high FP rate, which is not our target for security risk assessment applications. As will be seen, such an increase in cost significantly affects the performance when FP is essential.

7.7. Case Study 2: Injection Breaches in HIoT

Our second use case targets injection attacks in healthcare Internet of Things (HIoT) networks as another example of security risk assessment applications. Attacks in HIoT have different effects, resulting in different requirements where both FP and FN are crucial. Thus, Safety score weights should be adjusted. We first discuss the healthcare system requirements, the used dataset, the ML models, and the weights setup. Then, we evaluate the used ML models using the above metrics to show the outstanding performance of Safety score.

7.7.1. Healthcare Networks and Their Requirements

To provide a timely diagnosis for patients with affordable costs, modern healthcare systems rely increasingly on technology. Next-generation healthcare monitoring systems are expected to incorporate sensing devices that connect patients to their clinics without the need for the physical presence of patients. This creates HIoT systems where patient's bio-markers (e.g., heart rate) are shared over the network with the clinic. Whenever necessary, the doctor will suggest treatments to patients with urgent needs.

An attacker can launch an injection breach that changes the patient biometrics, resulting in undesirable treatments that may threaten the patient's life. If an IDS is involved and has missed such an attack, this is considered an FN outcome. On the other hand, if the biometrics have changed and the IDS detects that as a breach, i.e., an FN outcome, the changes will be ignored, and the

necessary treatments will not be given. In this case, both FN and FP are risky and should be minimized.

Even if FP and FN are equally dangerous, TN and TP are not as crucial as FP and FN. Thus, FP and FN would have the same costs while TN and TP will have a much lower cost. Assuming that the cost of FP and FN is $100M while the cost for TP and TN is $1M, the weights for Safety score here can be adjusted as:

$$w_{FN} = \frac{100}{202}, w_{FP} = \frac{100}{202}, w_{TP} = \frac{1}{202}, and\ w_{TN} = \frac{1}{202}$$

7.7.2. Healthcare Testbed and Dataset

We have used a dataset collected from a healthcare testbed built at Washington University in St. Louis. The testbed includes monitoring sensors that collect the patient's biometrics and connect to a server in a lab-scale HIoT system [153, 10]. The dataset consists of 44 features and includes man-in-the-middle injection breaches and benign HIoT traffic. The statistics of the dataset used for testing are presented in Table 7.5.

Table 7.5 Statistics of HIoT dataset

Type of traffic	**Total number of samples**	**Percentage**
Benign	14272	87.50%
Attack	2046	12.50%

7.7.3. Results and Analysis

To build the models, we have used the same ML algorithms and settings used earlier. The results are presented in Table 7.6. As can be seen, all algorithms perform poorly, giving an error of about 10% in the benign class and 30% in the breach class. This error results in an extreme loss with more than a 600 cost ratio, or a 500% increase in cost. These high error rates can be due to a noisy

dataset or improper choice of features. Since our objective is not to build perfectly performing models, we show the results without any model tuning to enhance the performance.

Table 7.6 Statistics and results for HIoT dataset

	RF	**KNN**	**LogR**
Classification Results			
TP	358	425	268
TN	3439	3156	3295
FP	134	417	278
FN	149	82	239
Performance Statistics			
Breach error rate	29.39%	16.17%	47.14%
Benign error rate	3.75%	11.67%	7.78%
Cost ratio	699.740	2065.92	2104.77
Evaluation Metrics			
Accuracy	0.930	0.885	0.873
Weighted Accuracy	0.730	0.841	0.555
AUC	0.834	0.862	0.725
F_1 score	0.717	0.655	0.509
Safety score	0.128	0.071	0.065

Even with such high loss, accuracy has not gone lower than 0.87 when the cost ratio is more than 2000. AUC indicates that all models have comparable performance. F_1 scores and weighted accuracies reflect the bad performance for all models; however, they are still above 0.5. Safety scores show the most realistic result, with a score less than 0.1 for all models.

It should be emphasized that the proposed metric is useful only when evaluating the loss or damage that a model takes in security applications. Further, the Safety score is the only presented metric that meets the requirements discussed earlier. However, if an application considers the overall error as the most important criteria and the type of error does not matter, accuracy would be a

useful metric. If the errors in classifying breaches are the only concern, weighted accuracy or F_1 score would accurately represent the performance.

The Safety score is more general than these other metrics. It can behave the same as these metrics in some applications. However, in security applications, the costs of the four outcomes are different, and so Safety score the only metric that can represent the true cost.

7.8. Summary

AI and ML algorithms have recently gained attention in building elegant security risk assessment solutions that are intelligent and able to define insights in high dimensional and perpetual dynamic data frequently found in security risk assessment applications. Such algorithms and their resulting models will be used by agents in our system to achieve our objective in building an intelligent, collaborative, distributed, and hostile node resilient solution that can be to build efficient security assessment solutions. However, our research in ML-based security applications has shown that conventional metrics used to compare ML models are not always suitable to evaluate security applications.

In this chapter, we first showed that conventional metrics underrepresent the loss or damage that a model can cause when making security decisions. We then proposed Safety score as a metric that bases the performance on the costs associated with model outcomes. The metric can be adjusted for different security applications ranging from national defense to patient care. Results, shown by examples and IDS dataset evaluations, proved the efficiency of Safety score in representing the actual models' performance in terms of costs or system effects. This makes the metric easy to explain to network administrators who care about security cost the most.

It should be noted that a limitation of Safety score and other evaluation metrics is that they cannot measure the model performance outside the used dataset. That is, a model performance with zero-day or unknown breaches is not predictable and not reflected by the available metrics unless these breaches are part of the test. This is due to the unavailability of the ground truth in such cases, thus, the inability to know the outcome used in evaluation metrics.

Nevertheless, Safety score has the promise to be used as a cost-representative metric to evaluate any security risk assessment applications. Thus, we expect it to be used as a new metric when evaluating future AI-based or non-AI-based security applications. The metric will be further used in the next chapter when we present our distributed and blockchain-based malware-detection use case as a security risk assessment application targeted in this dissertation.

Chapter 8: Malware Detection as a Security Risk Assessment Case Study

In previous chapters, we have discussed our approaches to resolve the challenges of state-of-the-art blockchain-based security risk assessment solutions. CrowdFAB is our proposed framework that addresses the blockchain-related challenges while Safety score addresses the proper evaluation metric challenge. This chapter presents our first security risk assessment use-case that utilizes CrowdFAB as an underlying platform. Namely, we discuss blockchain-based passive malware detection in mobile applications. The objective here is to verify the proposed approach's efficiency as an underlying secure infrastructure for security risk assessment applications. Using ML models learned from previously published datasets, we simulate multiple malware detection agents that can reflect real-world scenarios. We then empirically evaluate the system in terms of several performance metrics, including Safety score that we discussed earlier. Our results show that using the proposed approach enhances the prediction performance over earlier works and has a close to optimal performance.

8.1. Introduction and Related Work

Malware detection in mobile applications has been a challenging research issue, especially with the open-source nature of mobile platforms such as Android. Such nature marks the platform as a notable target of malicious application developers [154]. According to the Mobile Malware Evolution report in 2020, Kaspersky (a famous malware application detection platform) has detected more than 5.6 million malicious installation packages, including around 156,000 mobile banking Trojans and about 20,000 mobile ransomware Trojans, which is an increase of 50%

compared to 2019 [155]. Although Google has developed Bouncer as a prominent malicious application detector, attackers can still bypass the detector to send malicious app [156].

Therefore, the detection and classification of malicious applications is still an important research area of interest to academia and industries. In general, an adequate solution should be able to detect a wide range of known malware families or programs as well as new zero-day malicious families that evolve on a daily basis.

As discussed in Chapter 2, solutions for the problem can range from single fully centralized approaches using powerful machine learning algorithms to fully distributed assessment. In general, AI-based sophisticated solutions have the potential to dynamically detect malicious applications [157, 158, 115, 159]. However, the lack of data, especially for zero-day attacks, is a major challenge. Distributed and collaborative approaches, such as VirusTotal, [34], and hybrid analysis, [160], have the promise to extend the learning data, thus, reduce the possibility of zero-day attacks. However, as discussed earlier, the challenges in these prior approaches include centralization, trust, and a single point of failure.

As one may expect, blockchains can resolve these challenges and pose a new distributed solution. Thus, some existing works have used blockchains for a distributed and efficient malware detection in mobile applications. For example, Gu et al. proposed consortium blockchain for malware detection and evidence extraction as a framework where detected malware applications' codes are shared by the prespecified consortium blockchain nodes and can be accessed by any other public blockchain node [47]. The idea is to store detected malware metadata, including name, type, and other features, in blockchain transactions that any node in the system can access. Similarly, the

works in [161], [162], [163], and [164] presented blockchain-based malware detection approaches where the malware information is stored on the blockchain after being detected by an ML model.

On a similar line of work, Raje et al. discussed a distributed firewall for malware detection, where a requester submits a file to be inspected and receives many responses from malware detection agents [46]. Unlike the previously discussed blockchain works, this work is unique in sharing the predictions for malware detection rather than the malware metadata, which provides higher privacy guarantees for participating agents.

8.2. Motivation and Contributions

Earlier works on blockchain-based malware detection systems share the same challenges with state-of-the-art blockchain-based security assessment solutions. Namely, the blockchain is used to store malware metadata, resulting in storage without knowledge challenge. Assessing the quality of the added malware metadata and detecting whether it is valid or not is still missing, resulting in a proper reputation scheme challenge. Finally, as these works are still in their fancy and there is still insufficient proof-of-concept implementation, the practical use-case challenge still exists for malware detection applications.

It should be noted that the work by Raje et al. is the closest to the work presented in this dissertation. Knowledge-based blockchains also allow sharing agent's predictions for malware rather than malware metadata. However, the work still missed the processing and achieving the collaborative decisions. Thus, we are motivated to propose an alternative blockchain malware detection solution, where several agents share their predictions and a knowledge summary should be achieved within the blockchain process, making it easier for everyone to access.

By utilizing the CrowdFAB framework presented in Chapter 6, the contribution here is to present a fully distributed malware detection system where requesters can submit requests to analyze a mobile application or software. Multiple malware detection agents can analyze the request to provide elegant analyses using proper ML techniques. Finally, the system achieves a collaborative knowledge summary about the request within the blockchain process and adequate security and low overhead guarantees.

The work presented in this chapter is under submission for publication [17].

8.3. Malware Detection Application Setup

In the following, we discuss the application setup to be used for our empirical evaluation. This includes discussing the target application scenario, the malware dataset used, the agents' setup, and the CrowdFAB setup.

8.3.1. Application Scenario

A workflow of a blockchain-based malware detection application is shown in Figure 8.1. The numbers in the figure represent the workflow of the systems. As can be seen, an application name and other identification data to be inspected by the system are first received from a requester. Such information is submitted in request transactions. The participating agents will analyze the submitted application and make their assessments in response transactions used in knowledge-based blockchains and CrowdFAB. When a miner generates a block, these transactions are summarized in a knowledge summary.

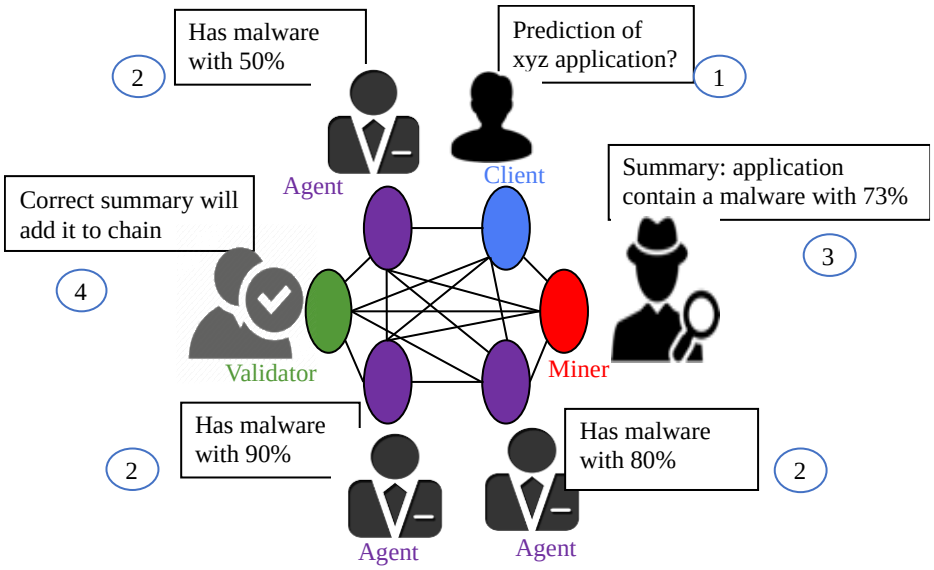


Figure 8.1 A workflow for blockchain-based malware detection for mobile applications

We emphasize that we consider passive malware analysis. This should be distinguished from a real-time malware detection application where the scan is performed online by a central agent or computer.

8.3.2. Agents' Setup

To empirically evaluate this case study, we have used a previously published malware dataset named Drebin [165, 166]. We used a portion of the dataset consisting of 215 features with 5560 malware apps and 9475 benign apps [167]. The dataset is divided into training and test sets where the assumption is that the test set will be the requests while the training set is preliminary data held at different agents, which they use to develop their expertise.

To simulate a large number of agents with many experience levels, we have used the training data with two different strategies, i.e., *horizontal and vertical partitions*.

In a horizontal partition, agents are given access to only a subset of the training dataset to train their models. This reflects on *their experience level* in the sense that an "old" agent would have

seen more malicious and benign data than a "new" agent. We believe that using this reflects on the real-world scenarios since no agent can have full or "perfect" data to train on, and just starting agents are assumed to have minimal training data. For this strategy, agents are randomly given 33%, 66%, and 90% of the training dataset.

The vertical partition allows agents to access only a subset of the features to train individual models. This partition simulates *different complexities of agents' models* as more features will result in a higher complexity, i.e., expensive or higher-cost models. Thus, some low resources agents train on only a few features that might not be sufficient. Others, with many resources, would choose to build complex models with a significant number of features. However, it should be noted that we do not mean here that having all features would result in better agents. Instead, this is just a reflection of the agents' model complexity. To simulate this, the number of features that agents use is randomized between 1 and 200 features, where the original dataset has 215 features. An agent with above 100 features performs relatively well for the used dataset, but this comes with a complexity cost.

Using the above two strategies, we simulate 1000 individual agents that train their models based on the provided training set and features. We have unified the learning algorithm to the random forest at this stage, given that the agents' performance can be randomized with the horizontal and vertical partitions. Random forest performs relatively well in malware detection applications and has a lower computation cost than more sophisticated learning algorithms [158, 168].

After the training has finished, the test set is used as a set of requests for agents to analyze if a particular application is malicious or not. We assume that all agents participate in all requests.

8.3.3. CrowdFAB Setup

After the agents make their predictions in response transactions, the system, using the CrowdFAB framework, summarizes these responses using six different summary functions. These functions are summarized as follows:

1. **Mean:** the mean summarizes the probability vector $\boldsymbol{P_i}$ constructed from all responses to the i^{th} request by taking their mean. Formally, the mean summarizes $Application_i$ with C_i as follows:

$$C_i = P\ (Application_i\ is\ malicious) = \frac{1}{n_i}\sum P_j\ (Application_i\ is\ malicious)$$

Where $Application_i$ is the inspected applications, $P_j\ (Application_i\ is\ malicious)$ is the individual agent decision, and n_i is the number of agents who have evaluated the same application.

2. **Median:** The median summarizes these probabilities $\boldsymbol{P_i}$ by their median value.
3. **Mode:** The mode is the value at which the probability mass function of $\boldsymbol{P_i}$ takes its maximum value.
4. **RMean:** RMean is the weighted version of the mean, where the weights here are agents' reputations calculated by the reputation scheme discussed in Section 5.5.3. and then normalized. Thus, at time t:

$$R_{j,t} = \frac{\begin{cases} 0 & p=0,\ n=0,\ t=0 \\ \alpha\frac{p}{p+n} + (1-\alpha)\left(2R_{j,t-1}-1\right) & \text{correct response} \\ \beta\frac{-n}{p+n} + (1-\beta)\left(2R_{j,t-1}-1\right) & \text{False Positive} \\ \gamma\frac{-n}{p+n} + (1-\gamma)\left(2R_{j,t-1}-1\right) & \text{False Negative} \end{cases}}{2} + 0.5$$

$$w_j = \frac{R_{j,t}}{\sum_{s=1}^{n} R_{s,t}}$$

5. **RMedain:** Similarly, RMedian is the weighted version of the median where the weights here are normalized agents' reputations.
6. **RMode:** RMode is the weighted version of the mode, or a weighted majority vote [169], where the weights here are normalized agents' reputations.

8.4. Performance Evaluation

To analyze the performance of our setup, we have compared it to several earlier single algorithm works and using several metrics. In the following, we will discuss these in addition to the results from this case study.

8.4.1. Baseline Algorithms

To show the superior performance of CrowdFAB-based malware detection, we will use three baselines:

1. Three earlier works that have used the same dataset, namely [170, 158, 47].
2. The average performance of the agents involved in our system.
3. An optimal model that trains on the whole dataset. Note that the optimal model, which is expected to perform the best, is rarely achievable in a real-world setting, given that none of the agents (or a trainer) has access to all training data. It is used here to show CrowdFAB superior performance

8.4.2. Metrics for Evaluation

To evaluate the performance, we have used three metrics: accuracy, F_1 score, and Safety score presented in Chapter 7. Note that Safety score is the only cost-representative metric that is usable for these applications. However, we show the conventional metrics to show that Safety score is

not tailored for our objectives and the analysis applies to all metrics. To recall, these metrics can formally be represented as follows:

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

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

$$Recall = \frac{TP}{TP + FN}$$

$$F_1\ score = 2 * \frac{Precision \times Recall}{Precision + Recall}$$

$$Safety\ score = \frac{w_{TP}TP + w_{TN}TN}{w_{FP}FP + w_{FN}FN + w_{TP}TP + w_{TN}TN}$$

For a malware application, TP is a malware sample that has been correctly detected. TN is a correctly classified benign sample. FP is a benign sample that has been categorized as a malware sample. FN is a malware sample that is classified as a benign sample.

For our case study, FNs have high costs as malware samples are missed. Thus, they should have high weights. FPs are less critical, but they cost more than TP and TN, and their weights should be lower than FN. Similarly, TP should have weights higher than TN. Given these considerations, we set Safety score weights for our application as:

$$w_{FN} = 5, w_{FP} = 1, w_{TP} = 0.2, and\ w_{TN} = 0.1$$

8.4.3. Results and Analysis

The results for the experimental setup are presented in Table 8.1. To analyze the results, we first look at a simple summary function without reputations (i.e., mean, median, and mode). As can be seen, using CrowdFAB as an underlying system can enhance the average prediction performance

of individual agents. The mean, which performs the worst as a summary function for this use case, improves the performance over the average performing individual agent by 20% in terms of Safety score and performs slightly better in accuracy and F_1 score. The mode, which performs the best (among the ones we tried) for this specific use case, enhances the performance by 7% in accuracy, 8% in F_1score, and more than 20% in Safety score.

Table 8.1 Performance of CrowdFAB malware detection with different summary functions

Algorithm	**Accuracy**	**F_1score**	**Safety score**
Earlier Works Performance			
[170]	97.7	97.12	-
[158]	97.24	97	-
[47]	94.6	-	-
Optimal Performance			
Optimal model (train on all data)	98.60	98.10	70.07
CrowdFAB Performance			
Average of involved individual agents	91.89	92.20	54.56
Mean	95.88	94.85	73.32
Median	97.92	96.47	76.97
Mode	98.67	98.25	74.60
RMean	97.29	96.51	78.03
RMedian	97.92	97.27	79.69
RMode	98.67	98.25	72.47

Compared to earlier works, the mode and the median have similar performance while the mean has a lower performance. The accuracy has been slightly enhanced using the median and the mode, while the F_1 score has slightly lower performance when using the median. The mode overperforms the earlier works in all cases. Note that we are not showing Safety score analysis for earlier works since this is a recently proposed metric that has not been used in both earlier works.

Compared to the optimal model, the mode and the median functions have comparable performance in terms of accuracy and F_1 score and better performance in terms of Safety score.

More importantly, it can be seen that summary functions with the proposed reputation management outperform, or at least perform similarly to, the same summary functions without reputations. The RMean and the RMedian have better performance, where the enhancement is about 3% for all metrics. RMode has a similar performance to the classic mode.

Compared to earlier works and the optimal model, the summary functions with reputations all have good performance in terms of accuracy and F_1 score and even outperform the optimal model in terms of Safety score.

8.4.4. Discussion

The results presented in this chapter show that the mode, or the majority vote, performed the best with this dataset. However, it should be emphasized again that the summary function is application-dependent. The best function would change from one application to another. We will be putting forward a summary function recommendation set in Chapter 9.

That said, our objective from these analyses is not to show the best summary function. Instead, we aimed to show that *any summary function overperforms the system's average agent performance and can perform better than earlier individual decision works*. The analyses in this chapter indicate that using a blockchain-based collaborative malware detection system can enhance the prediction performance over a singular almost perfect model. Further, the strengths of our approach are not in prediction performance but its security and decentralization approach. We expect any prediction

enhancement that system makes is minimal and restricted by the best available predictor or agent in the system.

While the analysis in this chapter is done empirically and with agents' simulations, it is easy to extend the results to practical applications using the response transactions and knowledge summary of knowledge-based blockchain and CrowdFAB. As was shown in Chapter 6, sending response transactions and achieving the knowledge summary is feasible with reasonable overhead.

Based on the earlier overhead and performance analysis, we expect the delay in detecting a malicious app to be within minutes in a reasonable size network, which is considered reasonable and outperforms other blockchain-based malware detection approaches.

8.5. Summary

Malicious mobile applications are increasing in volume and variation and resulting in an ever continuous personal data and financial losses threats. Traditional solutions, such as anti-virus scanners, fail to cope with the volume and diversity of malware in such systems. AI, ML, and traditional blockchain solutions have the potential to dynamically detect malware applications and resolve the problem. However, they come with the cost of centralization, trust, and inefficient decision making.

In this chapter, we have used our CrowdFAB framework to build an efficient blockchain-based malware detection system for mobile applications. We consider a security risk assessment use case where several malware detection agents analyze if a particular mobile app is malicious or not. These analyses are broadcasted using response transactions in CrowdFAB, and a knowledge summary is achieved by the system on whether the request is malicious or not. Results shown by

empirical analyses have proven the sustainable performance of the proposed method compared to single-agent detections. This, combined with the added security and decentralization features, shows our approach's superiority compared to existing approaches.

Future work for this part of the research can analyze the performance with various datasets and other general malware detection applications. Further, a practical implementation of the system, where the agents are malware detectors using different detection approaches and sending responses to CrowdFAB implementation, is desirable to extend the feasibility. Finally, the proposed approach can be applied directly to Android-based mobile applications to achieve a better secure, private, and efficient detection approach.

Chapter 9:Application Domains in and Beyond This Dissertation

In the previous chapter, we have discussed and evaluated malware detection in mobile applications as one security risk assessment study for CrowdFAB and the work proposed in this dissertation. However, CrowdFAB applications can go way beyond this application and even beyond security risk assessment applications. This chapter presents some security risk assessment applications that can use CrowdFAB in different networking domains. We further discuss other non-security risk assessment applications beyond this dissertation. These applications open the road to fully integrate blockchains and AI technologies to build better distributed, intelligent, secure, and agile platforms that underlie any decision-making or risk assessment application.

9.1. Security Risk Assessment Applications

In the following, we discuss some potential security assessment applications that can benefit from CrowdFAB and other proposed approaches. These applications include cyber threat information exchange, new node admission and access control, money laundering, trustworthiness sharing, spam filtering, and fraud detection. For each application, we discuss how it is done currently, its challenges, and how the proposed solution can resolve these challenges. It should be noted that the discussion here is conceptual and without validation; however, we expect the results from the previous chapter to apply to most applications presented here.

9.1.1. Cyber Threats Information and Knowledge Exchange

As defined by NIST, a cyber threat is "any event with the potential to adversely impact organizational operations, assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of

information, and/or denial of service" [171]. Cyber threat information is any kind of related information that helps the organization detect and protect itself from such a threat. The information includes attack indicators such as source IPs or Domain Name System (DNS), tactics or attacker behaviors, the security alerts and the threat intelligence reports received by the organization, and the later configuration to mitigates such an attack in the future [172]. Many organizations share and process such information internally among their security team, network administrators, and other relevant employers.

Sharing threat information with other organizations and getting their insights is sometimes crucial to avoid future attacks. The sharing can result in a global situational awareness, improved security posture, increased value of information, and greater defensive agility. Global situational awareness is achieved as threat information sharing enables organizations to leverage and process collective knowledge from the indicators and experiences of other organizations, thereby raising awareness of new attacks and countermeasures. In addition, while sharing the threat information, organizations can gain better insights into the threat information and develop an improved security posture to counter future attacks. The security posture results in an increased value of information and better defense strategies by knowing best practices and the actors and indicators that result in the attack for other organizations [172].

However, threat information sharing is currently restricted due to trust issues, privacy concerns, the sensitivity of the information, and many other reasons. As organizations do not trust and possibly do compete with each other, such information sharing is currently done by a third party which results in the trust issue. Further, privacy concerns arise as such information will be public to other organizations and individuals who may use it to launch more sophisticated attacks. The

information is by nature sensitive and may result in financial loss, violation of sharing agreements, and loss of reputation. Thus, threat information sharing is challenging with the currently available solutions.

Using blockchains, the challenges above can be resolved due to the distributed nature and pseudo anonymity features. Furthermore, by using CrowdFAB and knowledge-based blockchains, other organizations can share their insights about the same threat. Therefore, a collaborative understanding of the threat and its necessary countermeasures can be achieved in a secure, anomalous, and efficient way. *The summary function, in this case, can be based on machine learning algorithms that learn from the shared data*. By the end, a knowledge summary about the threat will be stored in the blockchain header. Hence, the summary can be accessed by other organizations who can use it to enhance their security measures to defend against a threat considered a "zero-day attack".

9.1.2. Node Admission and Access Control

Another potential application that can benefit from knowledge-based blockchains and CrowdFAB is authorization and admission of nodes in permissioned-like distributed systems. For example, healthcare systems are permissioned systems, where patients, doctors, and hospital staff members join the network only by permission, and even their functionalities should be predefined. A permissioned blockchain system is another permissioned-like distributed system example that allows only authorized nodes to access the chains and performs certain functionalities. In any case, adding a node and providing the correct permissions or allowed functionalities is a significant challenge. We focus the following on authorization and access control in permissioned blockchains but the discussion applies with minimal changes to other permissioned-like distributed systems.

Different permissioned blockchain platforms handle node admission differently. In most cases, it is up to a consortium running the blockchain to build the network and give permissions to nodes. For example, as a permissioned blockchain cryptocurrency system, Ripple defines its own validators and miners by the company. Other nodes can join the networks by connecting to enough validators. Quorum, as another permissioned blockchain platform, allows a designated authority or application developers to define participants along with their access permissions.

However, the above approaches suffer a centralization issue, indicating that decisions in permissioned blockchains are no longer fully decentralized. In addition, admissions may involve human interactions to approve the nodes, which introduces delay and lead to censorship.

A better approach may allow the current blockchain nodes to make a decision based on their local information. Thus, a collaborative decision on the new node's authorization and correct access control can be agreed upon among nodes. Knowledge-based blockchains and CrowdFAB can achieve this by allowing the blockchain nodes to make individual decisions on whether to add the new node or what rule to give in the system. Moreover, the existence of trust mechanisms can give higher contributions to consortium nodes, allowing the proposed solution to be compatible with applications that require only a few nodes to make the decision. *In general, the best summary function for such group decision making would be the weighted mode of the individual decisions made at a time.*

9.1.3. Fraud Identification and Spam Filtering

Other security risk assessment applications that are similar to threat information exchange and can widely benefit from crowd-forecasting models are fraud detections and spam filtering. Using crowdsourcing models, companies and individuals who have been targeted for fraud or spam can

share the information with others in the crowd. Similarly, suspected emails or fraud activities can be shared as requests over a crowd-forecasting model, and different agents can inspect if the submitted request is malicious. In this way, the detection task is mitigated to the crowd for better analysis and a variety of opinions.

However, a major challenge with crowd-forecasting models for these applications is the privacy of both requesters and agents. In general, a company or an individual would not share fraud information with others to keep their popularity and reduce the loss from such attacks. However, the information can be critical in building better detection models for these attacks and prevent them in the future. Thus, sharing this information while maintaining the privacy of the sharer is crucial for such systems.

Blockchains, and consequently knowledge-based blockchains and CrowdFAB, can resolve the above privacy challenge by providing pseudo-anonymity. Further, with CrowdFAB, one can summarize different opinions about the fraud and provide better insights while keeping privacy guarantees for both requesters and agents. *The summary function for this application can be an ML learned model if the shared information is spam or fraud information or a simple weighted mean if the shared information is predictions.*

It should be noted that the above solution comes with the cost of delay, thus, it is not suitable for real-time detections, and one should expect some delay before getting a proper collaborative answer.

9.1.4. Content Sharing and Trust in Vehicular Networks

Another application that we briefly mentioned before is a content caching and sharing in vehicular networks and intelligent transportation systems. Such systems are among the key component of smart cities and are thus expected to affect our daily life in the future. Content sharing between vehicles, or vehicles and base stations, currently enables a wide range of applications, from entertainment and multimedia to cooperative driving assessment and other sophisticated applications.

However, security and privacy issues are significant challenges in content sharing environments. Whether the vehicle or base station is trusted or not is typically delegated to a trusted authority that decides on whether the content can be shared or not. However, this solution suffers from the centralized party, the lack of privacy, the scalability, and many other challenges.

As an alternative approach, whether to trust the other vehicle can be delegated to knowledge-based blockchains and CrowdFAB in a vehicle network setting. Such a setting is expected to utilize a permissioned blockchain system that allows only authorized nodes (vehicles and base stations) to join the network. With the pseudo-anonymity feature of blockchains, the system is ensured to be anomalous when sharing the data, providing higher privacy guarantees over traditional centralized systems.

After requesting the trust of a vehicle or a base station, several other nodes or agents can evaluate if the nodes are trusted or not. These evaluations could be broadcast in response transactions. The evaluations can be different and are based on nodes' opinions from their previous interactions with the evaluated car or base station. In any case, CrowdFAB gathers these responses securely, achieves a collaborative trustworthiness value to the requester, and recommends whether to share

the content with the evaluated base station or car. *The summary function used here can be the weighted mean or any function that summarizes probabilistic trustworthiness values.*

9.1.5. Anti-Money Laundering

Money laundering is the process of illegally concealing the source of a large amount of money generated by criminal activities (such as drug setting or terrorist funding) by passing it through a sequence of smaller money transfers to multiple banks, thus, making it appear as coming from a legitimate sources [173]. To detect such activities by analysis or machine learning, a bank needs to get and share information about common users with multiple banks. Such collaboration is vital to enhance detection accuracy by order of magnitude. However, as these banks do not trust and possibly compete with each other, data sharing becomes a challenge.

As one would expect, CrowdFAB and knowledge-based blockchains can build efficient anti-money laundering systems, where different banks use their local models to predict such activities and collaborate with other banks to achieve a knowledge summary. As banks, in this case, share only predictions, the user data along with individual bank models are still private. This may enhance the prediction performance, as seen in the last chapter. However, since data is not shared, the individual models are still not improved.

As an alternative that is specifically helpful in this application, banks can share their model parameters and achieve a global detection model through the concept of federated (or distributed) learning. They can also privately share the actual training data instead of the model parameter. Using advanced privacy techniques such as deferential privacy on the user data along with encryption for model parameters, the banks can share the information without losing model or data privacy. CrowdFAB or knowledge-based blockchains, in this case, could achieve a global

detection model from these individual models or shared data. *In this case, the summary function should be modified to summarize encrypted information and build an efficient global model. Functions that involve federated learning with multi-party computations capabilities would be suitable in this case.*

9.2. Other Risk Assessment and Decision-Making Applications

As discussed, knowledge-based blockchains have the potential to build efficient, distributed, and secure decision-making processes for all risk assessment applications and decision-making systems. In the following, we discuss some distributed systems applications that can use the CrowdFAB and other approaches presented in this dissertation. It should be noted that these applications are not a target of this work; they are mentioned to extend CrowdFAB and knowledge-based blockchains applications beyond this dissertation and to introduce new blockchain application domains for distributed decision-making systems.

9.2.1. Aggregation in Wireless Sensor Networks

Wireless sensor networks consist of hundreds or thousands of inexpensive, low-powered sensors with limited memory, computations, and communication capabilities. These networks are getting popular to provide low-cost solutions for military and civilian applications, including habitat monitoring, disaster management, wildlife detection, traffic regulation, target tracking, etc. Due to the low power requirements, these sensors are generally deployed in a redundant fashion. Multiple sensors detect and sense the same information, and the data is aggregated to a central aggregator to be processed and then forwarded to a base station. However, this aggregator is generally centralized, leading to a centralized point of failure and trust challenges.

Using CrowdFAB and knowledge-based blockchains can resolve this challenge as the data can be aggregated within the blockchain process without the need for a central aggregator. With CrowdFAB elegant architecture, sensors do not have to be part of the blockchain and can communicate with the systems using simplified API. Thus, the additional overhead on the sensor is minimal and can include only sending the sensing data to one or multiple blockchain nodes. These blockchain nodes will be forming the blockchain transactions and broadcasting them to other nodes in the system. *The summary function for such application is generally the weighted mean, where weights are agents' reputations best calculated by iterative filtering, as was shown in* [68].

9.2.2. Stock Market Predictions

In a collaborative stock market prediction application, multiple agents predict whether the stock would rise or fall and the value by which the market will change. This prediction can be based on historical data, recent political news, or the geographic location of the predictor. Predictions can be shared over a crowdsourcing system, and this can help clients make conscious decisions about buying, selling, or holding their shares in that specific stock.

To achieve a collaborative market prediction, a centralized crowd-forecasting system is generally used to collect and aggregate several individual predictions. However, as with any centralized architecture, this comes with trust and privacy challenges.

CrowdFAB can replace these centralized crowd-forecasting systems and achieve distributed decisions and aggregations. *In this case, the summary function can be the average (arithmetic mean) or the weighted average, where agents are weighted based on their past performance.* It should be noted that the stock market requires fast decisions as the investors may lose millions in

a few seconds. Therefore, the on-the-go knowledge summary proposed by knowledge-based blockchains and analyzed in the previous chapters is expected to suit the requirements for such applications.

9.2.3. Disease Outbreak Prediction

Disease outbreaks can cause disastrous effects the life of citizens and the national economy. The use of machine learning in predicting such outbreaks and recommending vaccination sites is undeniable. The technology can help to indicate the locations of the outbreaks, fatality rates, and many other valuable metrics that help authorities know where to distribute the vaccine.

For a medical organization to compute a decent flu outbreak prediction model, it will need to use patient data (including age, sex, geographical locations, etc.) from multiple hospitals in the area and across the states. However, this brings up privacy concerns with sharing data and violating regulation policies. Thus, a trusted governmental third party is generally involved in gathering such data and building prediction models. This comes with trust, cost, and liability issues, making these models mainly untrusted.

Knowledge-based blockchains can enable generating such models securely and efficiently, thus, resolving the above challenge. As with money laundering cases, hospitals can build their own local models and aggregate only model parameters to the blockchain. Several other hospitals in the area do the same. Then, CrowdFAB can summarize the model parameter from multiple hospitals and achieve global prediction models that can be used by authorities for vaccine distribution or by citizens to avoid the locations of such outbreaks. *In this case, the summary function can be adopted from federated learning techniques, such as model average, gradient average, or even secure*

multi-party computation if modes are private and their corresponding parameters are encrypted [174].

9.2.4. Recommendation Systems

Distributed recommendation systems have been a research interest for a while. Many approaches have been proposed, and many websites are following these approaches. Examples include Expedia, Trivago, Booking.com, hotel.com, and so on. The problem with these approaches is that they involve a trusted third-party or a broker (e.g., Expedia) that collects the data and makes the decisions. This imposes a trust issue as recommenders are competing and do not share their datasets with others.

With the use of knowledge-based blockchains and CrowdFAB, agents or recommenders share their opinions or recommendations to the system without violating their privacy. To illustrate the suitability, consider a hotel rating application. Agents in this application can be users from Expedia, Trivago, Booking.com, hotel.com, etc. As can be noted, these agents do not trust and possibly compete with each other. Nevertheless, they can use CrowdFAB to exchange recommendation information and achieve a global summary that is visible to companies as well as users who want to use the hotel. *The summary function, in this case, can follow any aggregation function that is used by recommendation systems, and collaborative filtering is one of the most commonly used functions.*

9.2.5. Reinforcement Learning Applications

Reinforcement learning is a type of machine learning that builds models by taking action and receiving feedback from the system. Each action moves the system to a particular state, and the feedback comes from system experts regarding reward/punishment for the specific action. One of

the significant challenges in these systems is trust, i.e., trusting the system's feedback and trusting the agent's decision is authenticated and non-reproducible [175].

In addition to aiding the trustless decision-making process, CrowdFAB can act as systematic, non-reproducible, and authenticate feedback for reinforcement learning applications. The model learns adaptively according to the knowledge-summary as feedback and the reputation as rewards from the system. *The summary function in this specific case is application dependent, i.e., it depends on the application for which reinforcement learning will be used.* If the decision made by the agent is extremely different from the knowledge summary and the reputation has been reduced, the model is updated. Otherwise, the model is updated to reflect this new knowledge. In both cases, the model is updated so that the models' decisions will match more often in the future.

9.3. Summary Functions Recommendations

As discussed previously and shown in this chapter, the summary function used in knowledge-based blockchains and CrowdFAB is application-dependent. In the following, we summarize some recommendations on the choice of a good summary function based on the application type.

1. **When data privacy is involved:** In some applications, such as disease outbreak prediction, it is vital to keep data privacy guarantees. Thus, the responses that are shared via the blockchains can be encrypted, and the summary function should adapt to encounter that. In such a case, multi-party computation and homographic encryption techniques are recommended to be used as summary functions.
2. **When global models are to be built:** These applications involve the agents sharing their local models and trying to achieve a global (federated) learning model that is shared between

them. In such a case, federated learning techniques, such as model average, gradient average, or even secure multi-party computation if models' parameters are encrypted, can be used as summary functions.

3. **When the incremental property can be relaxed:** We have discussed before that the summary function must be incremental in the sense that a new summary can be calculated from the previous summary and the newly added responses. However, some applications such as recommendation systems can tolerate a delayed knowledge-summary and computationally expensive miners as a tradeoff for a better decision process that provides better aggregations and decision trustworthiness. In addition, some other applications, such as wireless sensor networks, are generally expected to get responses simultaneously. Thus, the knowledge summary does not have to be incremental, and the miner can go back to only a few blocks. In these cases, it is recommended to use iterative or collaborative filtering as sophisticated summary functions.
4. **When reputation matters:** Most decision-making applications will fall under this category where agents' performances are highly distinguished by their experience and adversarial level. At the same time, they need a fast and incremental summary function that meets the requirements we have discussed. In this case, weighted mean, weighted median, or weighted mode is recommended to be used.
5. **Agents are equally treated:** These kinds of simple functions can be used when the general applications are simply sharing predictions, where agents can be equally treated, and the system is trying to achieve a fast, incremental, and moderately efficient knowledge-summary. In such cases, mean, mode, or median can be very simple and fast summary function options.

Table 9.1 lists the summary functions that have been used or suggested through this dissertation based on the above recommendations.

Table 9. 1 Summary functions used in this work

Application	Function	Justification
Malware detection in mobile applications	Reputation-based mean, mode, or median	Agents' reputation matter and we need slightly fast decisions to be made
Cyber threat information sharing (predictions are shared)	Reputation-based mean, mode, or median	Agents' reputation matter and we need slightly fast decisions to be made
Cyber threat information sharing (threat are shared)	Machine Learning algorithm	A machine learning algorithm should learn from the shared data
Node admission in permissioned blockchains	Weighted mode	Weighted modes can give higher weights to administers for this specific application.
Fraud identification (predictions are shared)	Reputation-based mean, mode, or median	Agents' reputation matter and we need reasonably fast decisions to be made
Fraud identification (threat are shared)	Machine Learning algorithm	A machine learning algorithm should learn from the shared data
Trustworthiness in Content Sharing	Weighted mean	Agents' reputation matters
Money laundering	Federated Learning with multi-party computation	Local machine learning parameters are shared (possibly) with privacy concerns
Aggregation in wireless sensor networks	Iterative filtering	Sensors are expected to send their data around the same time
Stock market prediction	Weighted mean	Agents' reputation matters and the value is an integer
Disease outbreak prediction	Federated Learning with or without multi-party computation	Local machine learning parameters are shared, and this could be with or without privacy concerns
Recommendation system	Collaborative filtering	The application can tolerate delay as a tradeoff of better decisions

9.4. Summary

Knowledge-based blockchains and CrowdFAB have many potentials in building efficient decision-making and risk assessment applications. They provide better distributed, intelligent, secure, and agile platforms that underlie any decision-making or risk assessment application. This resolves many of the challenges in traditional intelligent systems where a centralized trust third party is involved for aggregations. Further, they open the road to fully integrate blockchains and AI to build blockchain-based distributed AI at learning and prediction phases.

This chapter discussed some of these applications in both security and non-security fields. This includes cyber-threat information sharing, node addition in permissioned system, anti-money laundering, fraud and spam detection, disease outbreak prediction, and many others. In each application, we discussed the current approaches along with their challenges and how knowledge-based blockchains and CrowdFAB can be used to resolve these challenges. Following that, a set of recommendations for the summary functions use for different application domains is provided. It should be noted that this set is based on the applications that we have dealt with so far and thus it can be adapted and future recommendations may be added for other application domains.

Chapter 10: Conclusion and Future Work

Security risk assessment solutions, including malware detection, intrusion detection, threat intelligence, and access control, are mostly centralized even with the current fully distributed networking applications. This centralization leads to lack of data, a single point of failure, trust issues, and many other challenges. On the other hand, blockchain experience, shown by the success of Bitcoin along with recent wide-adaptation by non-financial applications, has proven its potential to resolve the centralization challenge and provide a fully distributed, secure, and agile infrastructure that underlies networking applications.

In an effort to use blockchains for security risk assessment applications, we have identified several challenges and possible extensions to the technology. First, blockchains are currently used to store a vast amount of information in transactions and smart contracts without processing. However, it is important for AI-based security risk assessment solutions to process stored information (predictions or threat-related information) and achieve meaningful insights. This leads to the first challenge, namely the storage without knowledge challenge. Second, to build an elegant risk assessment or decision-making infrastructure, it is important to distinguish between several participants based on their performances. Thus, a reputation assignment scheme is needed to assign reputations to participants based on their past performances. Third, a proper analysis of the system performance and security requirements and guarantees with provable minimal overhead is required for the success of these blockchain-based solutions. Finally, a cost-representative metric is needed for a practical and transformative evaluation of any security solution built by AI and blockchains.

10.1. What Have We Achieved in This Dissertation?

This dissertation has aimed to resolve the above challenges and to build an intelligent, collaborative, distributed, and hostile node resilient solution that can be used for efficient security assessment suitable for next-generation distributed applications. In particular, the research focus can be divided into two stages: (1) extending blockchains for decision-making and risk assessments applications; (2) applying the proposed extensions to malware detection in mobile applications as an example of security risks assessment applications.

To resolve the storage without knowledge challenge, it is crucial to extend blockchains from simple storage-based systems to knowledge-based systems. The knowledge-based blockchains concept is introduced to make such an extension. It establishes a new transaction type that extends traditional transactions to include agents' decisions (risk assessments) along with their confidence. Moreover, transactions are processed by the miners/validators when generating/validating a new block. A knowledge summary is formed and stored in the new block header, making it available to other nodes. Thus, knowledge-based blockchains summarize the decisions included in the block, and knowledge is achieved within the blockchains process. The proposed concept is simple, yet it can accomplish a crucial task and introduces a new blockchain generation designed for collaborative decision-making and risk assessment applications. Details of knowledge-based blockchains are presented in Chapter 3.

Since knowledge-based blockchains are designed for decision-making systems, they need to be topped with a reputation scheme that assigns reputations to agents based on their performance. The scheme should favor outstanding and experienced agents who provide correct decisions over both new good agents and old, but maliciously, behaving agents. RPMC-EWA is proposed in this

dissertation to build such a reputation scheme. The proposed scheme is used to detect malicious agents, define the contributions of agents to the knowledge summary, and act as feedback for agents who want to enhance their predictions. This work has been explained in Chapter 4.

Before combining and implementing knowledge-based blockchains and RPMC-EWA, one needs to evaluate a blockchain platform to be used for such an implementation. However, the challenge here is how to evaluate such systems given their distributed and decentralized nature. A benchmark should define a list of metrics on which the system can be evaluated and a list of parameters that affect any blockchain performance. In that sense, the research in this dissertation introduces such lists and evaluates Quorum as a potential platform for implementing our proposed approaches. Our results show that Quorum is scalable in terms of the number of nodes and performs well when increasing the transaction rate up to a reasonable limit. The details of this are discussed in Chapter 5.

Next, knowledge-based blockchains and RPMC-EWA are combined to propose CrowdFAB as a crowd-forecasting application using blockchains framework. The framework is implemented and analyzed in terms of security and overhead. Results show the feasibility of a secure framework with a reasonable overhead, thus, it resolves the practical implementation and proper analysis challenge. Details of this are elaborated in Chapter 6.

Then, in the second stage of the research described in this dissertation, Safety score is introduced to resolve the fourth and final challenge, i.e., the cost-representative metric for security risk assessment application. The newly proposed metric is cost-sensitive and accurately represents the cost of a model designed for such applications. The metric is expected to be used when evaluating future AI-based or non-AI-based security applications. Our work on this is explained in Chapter 7.

Finally, the dissertation discussed and applied the proposed approaches to practical security risk assessment use cases. First, CrowdFAB is empirically evaluated for practical malware detection in mobile applications as a security risk assessment case study. Results show a sustainable close-to-optimal performance that outperforms earlier blockchains and non-blockchains works. This is presented in Chapter 8. A list of potential assessment applications is then introduced to extend the applicability of the proposed approaches in this dissertation. These applications include those designed for security risk assessment and are targeted in this work and those non-security-oriented applications that go beyond the discussion of this work. A detailed discussion of this is presented in Chapter 9.

10.2.Future Work

The work done in this dissertation is only a step towards making it possible to fully integrate blockchains and AI technologies and to introduce a new blockchain generation designed for intelligent systems. However, any single research work cannot aim to completely solve the problem, especially when extending a decade-old technology that is as complex as blockchains. The road to realizing fully integrated blockchain-based AI techniques and applications is yet to come. In the following, we will try to identify two possible directions that the work can be taken beyond this dissertation.

First, practical integration of the proposed knowledge-based blockchains and CrowdFAB with permissionless systems is necessary to check on the feasibility when the network is geographically distributed and adversarial settings are sophisticated. The implementation of CrowdFAB in this work was on Quorum, a permissioned blockchain with a leader-based consensus algorithm. Thus, we expect CrowdFAB to work with any permissioned-like system with a leader-based consensus

algorithm. However, for more complex networks such as Ethereum, there might be some challenges. As response transactions arrive at very different times due to the global distribution of the network, achieving a knowledge summary that can be validated by others will be a challenge. We have proposed some approaches to resolve these challenges and validated them in the context of this dissertation. However, this is still to be validated for permissionless blockchains. These systems typically come with much more sophisticated adversarial settings and attack strategies for miners, validators, and agents in the system. Thus, accounting for such settings is still to be investigated in the future.

From another point of view, the work proposed in this dissertation mainly targeted AI-enabled security risk assessment at the prediction/testing phase. Namely, our work assumed that agents are already trained to make assessments, and the system achieves a knowledge summary from these assessments. A future direction can take it back to the training phase, building what is referred to as blockchain-based federated or distributed learning environments. In these cases, agents share either data or model parameters instead of predictions. Current approaches for federated and distributed learning are centralized as they involve an aggregator [174]. However, the work proposed in this dissertation has the potential to resolve the centralization challenge and achieve a global model that is fully distributed. We have briefly discussed this direction in Chapter 9 for money laundering and disease outbreak applications. However, the evaluation and practicality of the discussed solutions are yet to come and can be seen as a future direction for ourselves and other researchers to take. Blockchain-based federated learning has been discussed before in [176, 177, 178]. However, the work is still in its fancy and does not achieve a global model by making

knowledge of the shared information. Much more research is required to make these systems come to reality with proper applications.

Bibliography

[1] M. Pilkington, Blockchain technology: principles and applications, Research handbook on digital transformations, 2016.

[2] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Munich, 2016.

[3] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292-2303, 2016.

[4] B. Betts, "Blockchain and the promise of cooperative cloud storage," Computerweekly, Augest 2016. [Online]. Available: http://www.computerweekly.com/feature/Blockchain-and-the-promise-of-cooperative-cloud-storage. [Accessed Febreuary 2019].

[5] J. Brantley, "Blockchain can help transform supply chain networks in the chemicals and petroleum industry," IBM Cross Business Unit White paper, 2017. [Online]. Available: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=CHJ12351USEN. [Accessed 22 Febreuary 2019].

[6] J. Frankenfield, "Blockchain-as-a-Service (BaaS)," investopedia tech, 17 May 2018. [Online]. Available: https://www.investopedia.com/terms/b/blockchainasaservice-baas.asp. [Accessed 22 Febreuary 2019].

[7] R. Sobers, "134 Cybersecurity Statistics and Trends for 2021," varonis, 22 Febreuary 2021. [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/. [Accessed 12 March 2021].

[8] K. Haug, "The Risks and Benefits of Decentralized Information Security," 16 March 2017. [Online]. Available: https://www.klogixsecurity.com/blog/the-risks-and-benefits-of-decentralized-informati. [Accessed 12 March 2021].

[9] N. Kaloudi and J. Li, "The AI-Based Cyber Threat Landscape: A Survey," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1-34, 2020.

[10] A. A. Hady, A. Ghubaish, T. Salman, D. Unal and R. Jain, "Intrusion Detection System for Healthcare Systems Using Medical and Network Data: A Comparison Study," *IEEE Access*, vol. 8, pp. 06576-106584, 2020.

[11] M. A. S. T. Teixeira, M. Zolanvari, R. Jain, N. Meskin and M. and Samaka, "SCADA system testbed for cybersecurity research using machine learning approach," *Future Internet*, vol. 10, no. 8, pp. 1-15, 2018.

[12] T. Salman, D. Bhamare, A. Erbad, R. Jain and M. Samaka, "Machine Learning for Anomaly Detection and Categorization in Multi-Cloud Environments," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, New York, NY, 2017, pp. 97-103.

[13] A. Raff, D. Peri and A. Lotem, "System and methods for malware detection using log based crowdsourcing analysis". US Patent No. 10,397, 27 Aug 2019.

[14] T. Salman, R. Jain and L. Gupta, "Probabilistic Blockchains: A Blockchain Paradigm for Collaborative Decision-Making," in *2018 9th IEEE Annual Ubiquitous Computing,*

Electronics & Mobile Communication Conference (UEMCON), New York City, NY, USA, 2018, pp. 457-465.

[15] R. Jain and T. T. Salman, "Systems and methods for probabilistic blockchains". US Patent 16671780, 2019.

[16] T. Salman, R. Jain and L. Gupta, "A Reputation Management Framework for Knowledge-Based and Probabilistic Blockchains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, Atlanta, GA, USA, 2019, pp. 520-527.

[17] T. Salman, G. A., R. Jain and M. Baza, "CrowdFAB: A Framework for Crowdsourced Forecasting Applications using Blockchains and its use for Security," *under preperation.*

[18] T. Salman, A. Ghubaish, D. Unal and R. Jain, "Safety Score as an Evaluation Metric for Machine Learning Models of Security Applications," *IEEE Networking Letters*, vol. 2, no. 4, pp. 207-211, 2020.

[19] T. Salman, M. Zolanvari, A. Erbad, R. Jain and M. Samaka, "Security Services Using Blockchains: A State of the Art Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 858-880, 2019.

[20] BitcoinWiki, "Proof of Work," [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_work. [Accessed 28 October 2018].

[21] Wekipedia, "Proof-of-stake," [Online]. Available: https://en.wikipedia.org/wiki/Proof-of-stake. [Accessed 22 May 2019].

[22] Wekipedia, "Proof-of-space," [Online]. Available: https://en.wikipedia.org/wiki/Proof-of-space. [Accessed 22 May 2019].

[23] Wikipedia, "Proof-of-importance (PoI)," [Online]. Available: https://golden.com/wiki/Proof-of-importance_(PoI)-639YX6M. [Accessed 13 March 2021].

[24] G. Zyskind, O. Nathan and A. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," in *2015 IEEE Security and Privacy Workshop*, San Jose, CA, 2015, pp. 180-184.

[25] G. Paul, P. Sarkar and S. Mukherjee, "Towards a more democratic mining in bitcoins," in *0th International Conference on Information Systems Security (ICISS'14)*, Dec. 2014, pp. 185-203.

[26] Wikipedia, "Byzantine fault," Weki, [Online]. Available: https://en.wikipedia.org/wiki/Byzantine_fault. [Accessed 22 May 2019].

[27] L. M. Bach, B. Mihaljevic and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),*, Opatija, 2018, pp. 1545-1550.

[28] Wikipedia, "Sybil attack," 15 January 2021. [Online]. Available: https://en.wikipedia.org/wiki/Sybil_attack. [Accessed 25 January 2021].

[29] J. Kreps, N. Narkhede and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, 2011, pp .1-7.

[30] A. Baliga, I. Subhod, P. Kamat and S. Chatterjee, "Performance Evaluation of the Quorum Blockchain Platform," *arXiv preprint arXiv:1809.03421,* 2018.

[31] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and A. A.-N. a. S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access,* vol. 7, no. 1, pp. 41525-41550, 2019.

[32] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos and P. Burnap, "A Supervised Intrusion Detection System for Smart Home IoT Devices," *IEEE Internet of Things Journal,* vol. 6, no. 5, pp. 9042-9053, 2019.

[33] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computers & Security,* vol. 92, no. 1, pp. 1-15, 2020.

[34] Wikipedia, "VirusTotal," 9 June 2020. [Online]. Available: https://en.wikipedia.org/wiki/VirusTotal. [Accessed 1 July 2020].

[35] G. Yang, S. He and Z. Shi, "Leveraging Crowdsourcing for Efficient Malicious Users Detection in Large-Scale Social Networks," *IEEE Internet of Things Journal,* vol. 4, no. 2, pp. 330-339, 2017.

[36] I. Burguera, U. Zurutuza and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices,* Chicago, 2011, pp. 15-26.

[37] Mamu, M. S. Islam and A. S. Kabir, "Hierarchical design based intrusion detection system for wireless ad hoc sensor network," *International Journal of Network Security & Its Applications (IJNSA),* vol. 2, no. 3, pp. 102-117, 2010.

[38] A. M. L. Ahmim, M. A. Ferrag, M. Derdour and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *In 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS) ,* 2019, pp. 228-233.

[39] Q. Schueller, K. Basu, M. Younas, M. Patel and F. Ball, "A hierarchical intrusion detection system using support vector machine for SDN network in cloud data center," in *018 28th International Telecommunication Networks and Applications Conference (ITNAC),* 2018, pp. 1-6.

[40] Q. Ling, Z. Tian and Y. Li, "Distributed decision-making in wireless sensor networks for online structural health monitoring.," *Journal of Communications and Networks,* vol. 11, no. 4, pp. 350-358, 2009.

[41] M. Z. A. Bhuiyan, J. W. G. Wu and J. Cao, "Sensing and decision making in cyber-physical systems: the case of structural event monitoring," *IEEE Transactions on Industrial Informatics,* vol. 12, no. 6, pp. 2103-2114, 2016.

[42] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó and M. Mühlhäuser, "Towards Blockchain-Based Collaborative Intrusion Detection Systems," in *International Conference on Critical Information Infrastructures Security,* Lucca, Italy, 2017, pp. 107-118.

[43] M. Banerjee, J. Lee and K.-K. R. Choo, "A blockchain future for internet of things security: a position paper," *Digital Communications and Networks,* vol. 4, no. 3, pp. 149-160, 2018.

[44] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," *IEEE Access*, vol. 6, no. 1, pp. 10179-10188, 2018.

[45] A. Malvankar, A. Kundu, S. Chari, M. Mohania, J. Payne and K. K. Budhraja, "Malware Containment in Cloud Using Analyticsand Blockchain," in *IEEE International Conference on Collaboration and Internet Computing*, Los Angeles, California, 2019, pp.1-4.

[46] S. Raje, S. Vaderia, N. Wilson and R. Panigrahi, "Decentralised firewall for malware detection," in *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, Mumbai, 2017, pp. 1-5.

[47] J. Gu, B. Sun, X. Du, J. Wang, Y. Zhuang and Z. Wang, "Consortium Blockchain-Based Malware Detection in Mobile Devices," *IEEE Access*, vol. 6, no. 1, pp. 12118-12128, 2018.

[48] F. Samreen, G. S. Blair and M. Rowe, "Adaptive decision making in multi-cloud management," in *the 2nd International Workshop on Cross-Cloud Systems*, Bordeaux, France, 2014, pp. 1-6.

[49] P. Han, B. Xie, F. Yang and R. Shen, "A Scalable P2P Recommender System Based on Distributed Collaborative Filtering," *Expert Systems with Applications*, vol. 27, no. 2, pp. 203-210, 2004.

[50] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660-4670, 2018.

[51] Y. Lu, X. Huang, K. Zhang, S. Maharjan and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298-4311, 2020.

[52] P. McCorry, C. F. Shahandashti and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *International Conference on Financial Cryptography and Data Security*, Malta, 2017, pp. 357-375.

[53] Wikipedia, "Social Choice Theory," [Online]. Available: https://en.wikipedia.org/wiki/Social_choice_theory. [Accessed 21 Augest 2018].

[54] R. Jain, The Art of Computer Systems Performance Analysis, Wiley-Interscience, 1991.

[55] S. Mansfield-Devine, "The growth and evolution of DDoS," *Network Security*, vol. 2015, no. 10, pp. 13-20, 2015.

[56] M. Nour and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 3, pp. 18-31, 2016.

[57] M. Nour and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network dataset)," in *IEEE Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015, pp. 1-6..

[58] A. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naïve Bayes," *Advances in neural information processing systems*, vol. 14, pp. 841-849, 2002.

[59] J. Zhang, M. Zulkernine and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 4, pp. 649-659, 2008.

[60] N. B. Amor, S. Benferhat and Z. Elouedi, "Naive Bayes vs Decision Trees in Intrusion Detection Systems," in *ACM Symposium on Applied Computing*, Nicosia, Cyprus, 2004, pp.420-424.

[61] G. D. Putra, V. Dedeoglu, S. S. Kanhere and R. Jurdak, "Trust Management in Decentralized IoT Access Control System," in *IEEE International Conference on Blockchain and Cryptocurrency*, Virual Conference, 2020.

[62] J. Yu, D. Kozhaya, J. Decouchant and P. E. Verissimo, "RepuCoin: Your Reputation is Your Power," *EEE Transactions on Computers*, vol. 68, no. 8, pp. 1225-1237, 2019.

[63] C. De Kerchove and P. Van Dooren, "Iterative filtering in reputation systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 4, pp. 1812-1834, 2010.

[64] Y. B. Zhou, T. Lei and T. Zhou, "A robust ranking algorithm to spamming," *EPL (Europhysics Letters)*, vol. 94, no. 4, p. 48002, 2011.

[65] P. Laureti, L. Z. Moret, Y. C. and Y. K. Yu, "Information filtering via iterative refinement," *EPL (Europhysics Letters)*, vol. 75, no. 6, pp. 1006-1012, 2006.

[66] Y. K. Yu, Y. C. Zhang, P. Laureti and L. Moret, "Decoding information from noisy, redundant, and intentionally distorted sources," *Physica A: Statistical Mechanics and its Applications*, vol. 371, no. 2, pp. 732-744, 2006.

[67] R. H. Li, J. Xu Yu, X. Huang and H. Cheng, "Robust reputation-based ranking on bipartite rating networks," in *Proceedings of the 2012 SIAM international conference on data mining. Society for Industrial and Applied Mathematics*, 2012, pp. 612-623.

[68] M. Rezvani, A. Ignjatovic, E. Bertino and S. Jha, "Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks," *IEEE transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 98-110, 2014.

[69] F. Gai, B. Wang, W. Deng and W. Peng, "Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network," in *Database Systems for Advanced Applications*, Gold Coast, Australia, 2018.

[70] Github, "The official GoChain client. https://gochain.io," gochain, [Online]. Available: https://github.com/gochain-io/gochain. [Accessed 3 June 2019].

[71] D. Qin, C. Wang and Y. Jiang, "RPchain: A Blockchain-Based Academic Social Networking Service for Credible Reputation Building," in *International Conference on Blockchain*, Seattle, WA, USA, 2018.

[72] C. Huang, Z. Wang, H. Chen, Q. Hu, Q. Zhang, W. Wang and X. Guan, "RepChain: A Reputation based Secure, Fast and High Incentive Blockchain System via Sharding," *arXiv preprint arXiv:1901.05741*.

[73] 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. (to appear), 2019.

[74] 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, March 2019.

[75] Weki, "Exponential_moving_average," Wekipedia, [Online]. Available: https://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average. [Accessed 22 Febreuary 2019].

[76] A. Josang and R. Ismail, "The Beta Reputation System," in *15th Bled Electronic Commerce Conference,* Bled, Slovenia, 2002.

[77] S. Seth, "Public, Private, Permissioned Blockchains Compared," Investopedia, 11 January 2021. [Online]. Available: https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/. [Accessed 18 March 2021].

[78] T.-T. Kuo, H. Zavaleta Rojas and L. Ohno-Machado, "Comparison of blockchain platforms: a systematic review and healthcare examples," *Journal of the American Medical Informatics Association,* vol. 26, no. 5, pp. 462-478, 12 June 2019.

[79] Github, "BLOCKBENCH: A Framework for Analyzing Private Blockchains. Blockbench contains workloads for measuring the data processing performance, and workloads for understanding the performance of different layers of Blockchain systems.," [Online]. Available: https://github.com/ooibc88/blockbench. [Accessed 22 May 2019].

[80] Hyperledger, "Caliper Documentation," [Online]. Available: https://hyperledger.github.io/caliper/docs/1_Getting_Started.html. [Accessed 22 May 2019].

[81] C. Fan, S. Ghaemi, H. Khazaei and P. Musilek, "Performance Evaluation of Blockchain Systems: A Systematic Survey," *IEEE Access,* vol. 8, pp. 126927-126950, 2020.

[82] S. Smetanin, A. Ometov, M. Komarov, P. Masek and Y. Koucheryavy, "Blockchain Evaluation Approaches: State-of-the-Art and Future Perspective," *Sensors (Basel),* vol. 20, no. 12, pp. 3358-3378, 2020.

[83] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi and K. L. Tan, "BLOCKBENCH: A Framework for Analyzing Private Blockchains," in *2017 ACM International Conference on Management of Data (SIGMOD)*, Chicago, Illinois, USA, 2017.

[84] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi and K. L. Tan, "Untangling Blockchain: A Data Processing View of Blockchain Systems," *IEEE Transactions on Knowledge and Data Engineering,* vol. 30, no. 7, pp. 1366-1385, 2018.

[85] Y. Hao, Y. Li, X. Dong, L. Fang and P. Chen, "Performance Analysis of Consensus Algorithm in Private Blockchain," in *IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018.

[86] W. M. Shbair, M. Steichen, J. François and R. State, "Blockchain orchestration and experimentation framework: A case study of KYC," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, 2018.

[87] A. D. Gupta and A. Dickson, "Analyzing Performance in Blockchain-Based Systems," 21 November 2017. [Online]. Available: https://stratumn.com/thinking/analyzing-performance-in-blockchain-based-systems/. [Accessed 22 May 2019].

[88] Gitbub, "Blockchain benchmarking framework," [Online]. Available: https://github.com/dsx-tech/blockchain-benchmarking. [Accessed 22 May 2019].

[89] H. Sukhwani, "Performance Modeling & Analysis of HyperledgerFabric (Permissioned Blockchain Network)," Doctor of Philosophy Dissertation Thesis, Department of Electrical & Computer Engineering, Duke University, 2018.

[90] Q. Nasir, I. A. Qasse, M. Abu Talib and A. Nassif, "Performance Analysis of Hyperledger Fabric Platforms," *Security and Communication Networks,* vol. 2018, pp. 1-14, 2018.

[91] H. F. Leppelsack, "Experimental Performance Evaluation of Private DistributedLedger Implementations," MASTER'STHESIS ININFORMATICS, TECHNICALUNIVERSITY OFMUNICH, 2018.

[92] P. Thakkar, S. Nathan and B. Viswanathan, "Performance Benchmarking and OptimizingHyperledger Fabric Blockchain Platform," in *IEEE MASCOTS*, Wisconsin, USA, 2018.

[93] S. Pongnumkul, C. Siripanpornchana and S. Thajchayapong, "Performance Analysis of Private Blockchain Platforms in Varying Workloads," in *017 26th International Conference on Computer Communication and Networks (ICCCN)*, Vancouver, BC, 2017.

[94] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat and S. Chatterjee, "Performance Characterization of Hyperledger Fabric," in *018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, Zug, 2018.

[95] M. T. Oliveira, G. R. Carrara, N. Fernandes, C. V. N. Albuquerque, R. C. Carrano, D. S. V. Medeiros and D. M. F. Mattos, "Towards a Performance Evaluation of Private Blockchain Frameworks using a Realistic Workload," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Paris, France, 2019.

[96] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi and A. Rindos, "Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric),," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, Hong Kong, 2017.

[97] H. Sukhwani, N. Wang, K. S. Trivedi and A. Rindos, "Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, 2018.

[98] T. Salman, M. Zolanvari, A. Ghubaish, M. Hall and R. Jain, "An Overview of Current and Future Blockchain Evaluation Approaches," *under submission* .

[99] I. C. Lin and T. C. Liao, "A Survey of Blockchain Security Issues andChallenges," *International Journal of Network Security,* vol. 19, no. 5, pp. 653-659, 2017.

[100] C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," in *IEEE P2P 2013 Proceedings*, Trento, 2013.

[101] D. Difallah, A. Pavlo, C. Curino and P. C. Mauroux, "OLTP-bench: An extensible testbed for benchmarking relational databases," *Proceedings of the VLDB Endowment,* vol. 7, no. 4, pp. 277-288, 2013.

[102] B. F. S. A. Cooper, E. Tam, R. Ramakrishnan and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM symposium on Cloud computing*, Indianapolis, Indiana, USA, 2010, pp. 143-154.

[103] N. Singh, "Quorum Blockchain Ultimate Guide," 101 Blockchains, 15 March 2019. [Online]. Available: https://101blockchains.com/quorum-blockchain-tutorial/. [Accessed 25 January 2021].

[104] D. Ongaro and J. Ousterhout, "n search of an understandable con-sensus algorithm," in *USENIX ATC '14*, Philadelphia, PA, 2014, pp. 305–320.

[105] G. Shapiro, C. Natoli and V. Gramoli, "The Performance of Byzantine Fault Tolerant Blockchains," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, 2020, pp. 1-8.

[106] CloudLab. [Online]. Available: https://cloudlab.us.

[107] J. Howe, "The Rise of Crowdsourcing," *Wired Magazine*, vol. 53, no. 10, pp. 1-4, 2016.

[108] Wikipedia, "Uber," 28 June 2020. [Online]. Available: https://en.wikipedia.org/wiki/Uber. [Accessed 1 July 2020].

[109] Wikipedia, "Upwork," 22 June 2020. [Online]. Available: https://en.wikipedia.org/wiki/Upwork. [Accessed 1 July 2020].

[110] Wikipedia, "Airbnb," 29 June 2020. [Online]. Available: https://en.wikipedia.org/wiki/Airbnb. [Accessed 1 July 2020].

[111] S. Perez, "Facebook tests Forecast, an app for making predictions about world events, like COVID-19," 23 June 2020. [Online]. Available: https://techcrunch.com/2020/06/23/facebook-tests-forecast-an-app-for-making-predictions-about-world-events-like-covid-19/. [Accessed 1 July 2020].

[112] Polysawm. [Online]. Available: https://polyswarm.io/. [Accessed 1 July 2020].

[113] Wikipedia, "Synack," https://en.wikipedia.org/wiki/Synack, 30 April 2020. [Online]. [Accessed 1 July 2020].

[114] D. R. Choffnes, F. E. Bustamante and Z. Ge, "Crowdsourcing service-level network event monitoring," in *Proceedings of the ACM SIGCOMM 2010 conference*, New delhoi, India, 2010, pp. 387–398.

[115] C. Christoforidis, V. Vlachos and I. Androulidakis, "A crowdsourcing approach to protect against novel malware threats," in *2014 22nd Telecommunications Forum Telfor (TELFOR*, Belgrade, 2014, pp. 1063-1066.

[116] F. Saab, I. Elhajj, A. Kayssi and A. Chehab, "A crowdsourcing game-theoretic intrusion detection and rating system," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, Pisa, Italy , 2016, pp. 622–625.

[117] M. Benali, A. R. Ghomari and L. Zemmouchi-Ghomari, "Crowdsourced Collaborative Decision Making in Crisis Management: Application to Desert Locust Survey and Control," in *Computational Intelligence and Its Applications*, Oran, Algeria, 2018, pp 533-545.

[118] W. Feng and Y. Zheng, "MCS-Chain: Decentralized and trustworthy mobile crowdsourcing based on blockchain," *Future Generation Computer Systems,* vol. 95, no. 1, pp. 649-666, 2019.

[119] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi and W. Dou, "A Blockchain-Powered Crowdsourcing Method With Privacy Preservation in Mobile Environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1407-1419, 2019.

[120] N. Quoc Viet Hung, N. T. Tam, L. N. Tran and K. Aberer, "An Evaluation of Aggregation Techniques in Crowdsourcing," in *Web Information Systems Engineering -- WISE 2013*, Berlin, Heidelberg, 2013, pp.1-15.

[121] 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.

[122] S. Zou, J. Xi, H. Wang and G. Xu, "CrowdBLPS: A Blockchain-Based Location-Privacy-Preserving Mobile Crowdsensing System," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4206-4218, 2020.

[123] V. Jacynycz, A. Calvo, S. Hassan and A. A. S.́anchez-Ruiz, "Betfunding:A distributed bounty-based crowdfunding platform over ethereum,," in *Distributed Computing and Artificial Intelligence*, Sevilla, Spain,, 2016, pp. 403–411.

[124] H. Zhu and Z. Z. Zhou, "Analysis and outlook of applications of blockchain technology to equity crowdfunding in china," *FinancialInnovation*, vol. 2, no. 1, p. 29, 2019.

[125] F. Buccafurri, G. Lax, S. Nicolazzo and A. Nocera, "Tweetchain: Analternative to blockchain for crowd-based application," in *nternationalConference on Web Engineering*, 2017, pp. 386–393.

[126] Y. Lu, Q. Tang and G. Wang, "ZebraLancer: Private and Anonymous Crowdsourcing System atop Open Blockchain," in *018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS),*, Vienna, 2018, pp. 853-865.

[127] Wikipedia, "Zero knowledge Proof," 25 June 2020. [Online]. Available: https://en.wikipedia.org/wiki/Zero-knowledge_proof. [Accessed 1 July 2020].

[128] J. Wang, M. Li, Y. He, H. Li, K. Xiao and C. Wang, "A Blockchain Based Privacy-Preserving Incentive Mechanism in Crowdsensing Applications," *IEEE Access*, vol. 6, pp. 17545-17556, 2018.

[129] 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.

[130] S. Zhu, H. Hu, Y. Li and W. Li, "Hybrid Blockchain Design for Privacy Preserving Crowdsourcing Platform," in *2019 IEEE International Conference on Blockchain (Blockchain)*, Atlanta, GA, USA, 2019, pp. 26-33.

[131] C. Lin, D. He, S. Zeadally, N. Kumar and K.-K. R. Choo, "SecBCS: a secure and privacy-preserving blockchain-based crowdsourcing system," *Sci. China Inf. Sci.* , vol. 63, no. 3, pp. 1-14, 2020.

[132] D. G. Kogias, H. C. Leligou, M. Xevgenis, M. Polychronaki, E. Katsadouros, G. H. R. Loukas and C. Z. Patrikakis, "Toward a Blockchain-Enabled Crowdsourcing Platform," *IT Professional*, vol. 21, no. 5, pp. 18-25, 2019.

[133] F. Ast and A. Sewrjugin, "The Crowdjury, a Crowdsourced Judicial System for the Collaboration Era," 10 November 2015. [Online]. Available: https://medium.com/the-

crowdjury/the-crowdjury-a-crowdsourced-court-system-for-the-collaboration-era-66da002750d8. [Accessed 1 July 2020].

[134] J. Huang, L. Kong, H. Dai, W. Ding, L. Cheng, G. Chen, X. Jin and P. Zeng, "Blockchain Based Mobile Crowd Sensing in Industrial Systems," *IEEE Transactions on Industrial Informatics,* to appear.

[135] M. Ali, J. Nelson, R. Shea and M. J. Freedman, "Blockstack: A Global Naming and Storage System Secured by Blockchains," in *2016 USENIX Annual Technical Conference*, Denver, CO, USA, 2016, pp .181-194.

[136] N. Nandan, A. Pursche and X. Zhe, "Challenges in Crowdsourcing Real-Time Information for Public Transportation," in *014 IEEE 15th International Conference on Mobile Data Management*, Brisbane, 2014, pp. 67-72.

[137] Wikipedia, "Elliptic-curve cryptography," 28 June 2020. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic-curve_cryptography. [Accessed 1 July 2020].

[138] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang and A. Mohaisen, "Exploring the Attack Surface of Blockchain: A Systematic Overview," *IEEE Communications Surveys & Tutorials,* vol. (to appear), 2020.

[139] Y. Chen, S. Chen and I. Lin, "Blockchain based smart contract for bidding system," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, Chiba, 2018, pp. 208-211.

[140] Wikipedia, "Fork (blockchain)," 26 May 2020. [Online]. Available: https://en.wikipedia.org/wiki/Fork_(blockchain). [Accessed 1 July 2020].

[141] S. Otoum, B. Kantarci and H. T. Mouftah, "On the Feasibility of Deep Learning in Sensor Network Intrusion Detection," *IEEE Networking Letters,* vol. 1, no. 2, pp. 68-71, 2019.

[142] S. M. Srinivasan, T. Truong-Huu and M. Gurusamy, "Machine Learning-Based Link Fault Identification and Localization in Complex Networks," *IEEE Internet of Things Journal,* vol. 6, no. 4, pp. 6556-6566, 2019.

[143] D. Papamartzivanos, F. G. Mármol and G. Kambourakis, "Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems," *IEEE access,* vol. 7, pp. 13546-13560, 2019.

[144] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan and R. Jain, "Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things," *IEEE Internet of Things Journal,* vol. 6, no. 4, pp. 6822-6834, 2019.

[145] J. Kim, J. Kim, H. L. T. Thu and H. Kim, "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, Jeju, 2016, pp. 1-5.

[146] A. A. Cardenas, J. S. Baras and K. Seamon, "A framework for the evaluation of intrusion detection systems," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, Berkeley/Oakland, CA, 2006, pp. 15-77.

[147] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: results from the JAM project," in *Proceedings DARPA Information Survivability Conference and Exposition (DISCEX'00)*, Hilton Head, SC, USA, 2000, pp. 130-144.

[148] R. Siegel, "Capital One looked to the cloud for security. But its own firewall couldn’t stop a hacker.," 30 July 2019. [Online]. Available: https://www.washingtonpost.com/technology/2019/07/30/capital-one-looked-cloud-security-its-own-firewall-couldnt-stop-hacker/. [Accessed 2 Mar 2020].

[149] C. One, "Information on the Capital One Cyber Incident," 23 September 2019. [Online]. Available: https://www.capitalone.com/facts2019. [Accessed 18 March 2021].

[150] S. Safari, A. Baratloo, M. Elfil and A. Negida, "Evidence Based Emergency Medicine; Part 5 Receiver Operating Curve and Area under the Curve," *Emergency* , vol. 4, no. 3, p. 111–113, 2016.

[151] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, 2015, pp. 1-6.

[152] Wikipedia, "k-nearest neighbors algorithm," [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. [Accessed 2 March 2020].

[153] L. Gupta, "Management And Security Of Multi-Cloud Applications," Doctor of Philosophy (PhD) Dissertation at Washington University in St. Louis, St. Louis, MO, 2019.

[154] H. Bing, "Analysis and Research of System Security Based on Android," in *Proceedings of Fifth International Conference on Intelligent Computation Technology and Automation*, 2012, pp. 581-584.

[155] V. Chebyshev, "Mobile malware evolution 2020," Securelist, 1 March 2021. [Online]. Available: https://securelist.com/mobile-malware-evolution-2020/101029/. [Accessed 20 March 2021].

[156] N. J. Schulte and S. Percoco, "dventures in BouncerLand," in *Proc. Black Hat USA*, 2021.

[157] W. Peters, A. Dehghantanha, R. M. Parizi and G. Srivastava, "A Comparison of State-of-the-Art Machine Learning Models for OpCode-Based IoT Malware Detection," in *Handbook of Big Data Privacy*, Cham, Springer International Publishing, 2020, pp. 109-120.

[158] M. S. Rana, S. S. M. M. Rahman and A. H. Sun, "Evaluation of Tree Based Machine Learning Classifiers for Android Malware Detection," in *International Conference on Computational Collective Intelligence*, Bristol, United Kingdom, 2018, pp.377--385.

[159] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Software Quality Journal*, vol. 26, no. 1, p. 891–919, 2018.

[160] "Hybrid Analysis," [Online]. Available: https://www.hybrid-analysis.com/. [Accessed 20 March 2021].

[161] F. Gao, F. Jiang, Y. Zhang and R. Doss, "Quorum Chain-Based Malware Detection in Android Smart Devices," in *International Conference on Future Network Systems and Security*, Cham, 2019, pp. 212-224.

[162] S. Talukder, S. Roy and T. A. Mahmud, "An Approach For An Distributed Anti-Malware System Based on BlockChain Technology,," in *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, India, 2019, pp. 1-6.

[163] R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar and A. Sharif, "A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features," *IEEE Access*, vol. 7, no. 1, pp. 64411-64430, 2019.

[164] S. Homayoun, A. Dehghantanha, R. M. Parizi and K. R. Choo, "A Blockchain-based Framework for Detecting Malicious Mobile Applications in App Stores," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE),*, Edmonton, AB, Canada, 2019, pp. 1-4.

[165] D. Arp, M. Spreitzenbarth, M. Huebner, H. Gascon and K. Rieck, "Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket," in *21th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2014, pp. 35–40.

[166] M. Spreitzenbarth, F. Echtler, T. Schreck, F. C. Freling and J. Hoffmann, "MobileSandbox: Looking Deeper into Android Applications," in *28th International ACM Symposium on Applied Computing (SAC)*, Coimbra, Portugal, 2013, pp. 1808–1815.

[167] S. Yerima, "Android malware dataset for machine learning 2," Figshare. Dataset., 5 February 2018. [Online]. Available: https://figshare.com/articles/Android_malware_dataset_for_machine_learning_2/5854653. [Accessed 1 July 2020].

[168] M. S. Alam and S. T. Vuong, "Random Forest Classification for Detecting Android Malware," in *IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing,*, Beijing, 2013, pp. 663-669.

[169] Wikipedia, "Weighted majority algorithm (machine learning)," 4 July 2019. [Online]. Available: https://en.wikipedia.org/wiki/Weighted_majority_algorithm_(machine_learning). [Accessed 1 July 2020].

[170] S. Y. Yerima and S. Sezer, "DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection," *Transactions on Cybernetics*, vol. 49, no. 2, pp. 453-466, 2019.

[171] R. 1. NIST SP 800-30, "Guide for Conducting Risk Assessments," September 2012. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-30r1. [Accessed 20 March 2021].

[172] R. 1. NIST SP 800-150, "Guide to Cyber Threat Information Sharing," October 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-150.pdf. [Accessed 20 March 2021].

[173] Wikipedia, "Money laundering," 17 March 2021. [Online]. Available: https://en.wikipedia.org/wiki/Money_laundering. [Accessed 20 March 2021].

[174] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen and H. Yu, Federated Learning, Morgan & Claypool, 2019.

[175] T. T. Nguyen, N. D. Nguyen and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3826-3839, 2020.

[176] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng and Q. Yan, "A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus," *IEEE Network,* vol. 35, no. 1, pp. 234-241, 2021.

[177] U. Majeed and C. S. Hong, "FLchain: Federated Learning via MEC-enabled Blockchain Network," in *20th Asia-Pacific Network Operations and Management Symposium (APNOMS),* Matsue, Japan, 1-4.

[178] G. Hua, L. Zhu, J. Wu, C. Shen, L. Zhou and Q. Lin, "Blockchain-Based Federated Learning for Intelligent Control in Heavy Haul Railway," *IEEE Access,* vol. 8, no. 1, pp. 176830-176839, 2020.