Designing Game-theoretically Sound, Fair, and Private Multi-agent Systems

Thesis submitted in partial

fulfillment of the requirements of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Sankarshan Damle

2020801008

sankarshan.damle@research.iiit.ac.in

Advised by Dr. Sujit Prakash Gujar

International Institute of Information Technology

Hyderabad - 500 032, INDIA

June, 2024

International Institute of Information Technology

Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "*Designing Game-theoretically Sound, Fair, and Private Multi-agent Systems*" by Sankarshan Damle, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Advisor: Dr. Sujit P Gujar

To, my parents,

whose efforts and sacrifices have made this possible.

Acknowledgments

I would like to express my sincere gratitude to the following individuals and organizations who have played a significant role in the completion of this thesis.

I am deeply thankful to my Ph.D. advisor, Dr. Sujit Gujar, for his unwavering support, guidance, and invaluable feedback throughout the entire research process. His expertise and commitment to excellence have been instrumental in shaping the direction of my research career. I am grateful to him for allowing me to be a part of the Machine Learning Lab (MLL) at IIIT, Hyderabad, since 2017 and for creating a thriving research environment at MLL for us to learn and grow.

I would also like to extend my appreciation to Prof. Boi Faltings, Prof. Dimitris Papadopoulos, and Prof. Dimitris Chatzopoulos, who provided me with opportunities to work on and contribute to exciting research projects at EPFL and HKUST. The insights and learning from these stints have greatly enriched the quality of this thesis and my research aptitude in general.

I extend my heartfelt thanks to my colleagues and fellow researchers who collaborated with me on various research projects during my PhD journey. These include Manisha Padala, Moin Hussain Moti, Samhita Kanaparthy, Anurag Jain, Sambhav Solanki, Sanidhay Arora, Varul Srivastava, Aleksei Triastcyn and Vlasis Koutsos. I thank them for their collective effort and diverse viewpoints that have enhanced my research perspective.

To my family, my parents, brothers, sisters, sister-in-law, aunts, and grandfather, your unwavering support, encouragement, and understanding have been my pillars of strength. I am profoundly grateful for your love and belief in my abilities.

A special mention goes to my friends Kumar Abhishek, Dhiraj Jagadale, Namit Sawhney, Yaghyavardhan Singh Khangarot, Amardeep Kedia, Swapnil Daga, Priyansh Agrawal, and Rohan Singh, who provided moral support and shared both the joys and challenges of my IIIT journey all the way back from 2015. Your camaraderie made this endeavor more enjoyable. I also appreciate my MLL colleagues, Shaily Mishra, Sanjay Chandlekar, Shantanu Das, Debojit Das, and Shoeb Siddiqui, for making my MLL journey more enjoyable.

I would also like to acknowledge the financial support provided by the Ripple-IIITH PhD Fellowship for the duration of my Ph.D. journey. I also thank the Microsoft Research Travel Grant, ACM India Travel Grant, the AAAI and IJCAI Scholarships, and IIIT, Hyderabad, for supporting various conference visits.

I express my gratitude to the Thesis panel which included Dr. Satya Lokam, Dr. Sunil Simon, and Dr. Ashok Kumar Das. The panel's thoughtful feedback and guidance helped improve the thesis. I also thank IIIT, Hyderabad, for providing the necessary resources, facilities, and a conducive research environment. I thank the staff and the administration for their help throughout my (long) journey at IIIT, Hyderabad.

In conclusion, this thesis stands as a testament to the collective effort and support of the individuals and organizations mentioned above. Their contributions have been indispensable in bringing this research to fruition.

Abstract

Multi-agent systems (MAS) are distributed systems composed of multiple autonomous agents interacting to achieve a common or conflicting goal. MAS tackles complex and dynamic problems that a single agent cannot solve, resulting in better problem-solving skills, enhanced reliability, and improved scalability. This thesis explores the challenges facing MAS, particularly related to their game-theoretic, fairness, security, and privacy guarantees.

A game-theoretically sound MAS is one where the agent interaction can be modeled as a game and analyzed using game-theoretic concepts. This leads to a more stable and efficient system, as agents are incentivized to make decisions that align with the system goals. This thesis focuses on civic crowdfunding, a method for raising funds through voluntary contributions for public projects (e.g., public parks). Our work enriches the existing literature by designing more inclusive mechanisms and providing fairer rewards and efficiency over the blockchain.

Fairness is also an essential aspect of MAS as it ensures that the actions and outcomes of agents are equitable and just, resulting in MAS's long-term stability and sustainability. This thesis looks at fair incentives in Transaction Fee Mechanisms (TFM). Blockchains employ TFMs to include transactions from the set of outstanding transactions in a block. We argue that existing TFMs' incentives are misaligned for a cryptocurrency's greater market adoption. We propose TFMs that provide fairer rewards to the transaction creators and minimize the surplus collected to the creators.

Last, security and privacy are crucial aspects of MAS, as the autonomy and decentralization of agents in MAS can lead to the exposure of sensitive information. In this thesis, we specifically focus on privacy guarantees for MAS like (i) auctions, (ii) voting, and (iii) distributed constraint optimization (DCOPs). We propose privacy-preserving applications that preserve agents' sensitive information while proving the computation's verifiability.

Research Papers Based on the Thesis Work

Journal Paper

1. **Sankarshan Damle**, Aleksei Triastcyn, Boi Faltings, and Sujit Gujar. "Differentially Private Multi-Agent Constraint Optimization". Journal of Autonomous Agents and Multi-Agent Systems *(JAAMAS)*.

Conference Papers

1. **Sankarshan Damle**, Manisha Padala, and Sujit Gujar. "Designing Redistribution Mechanisms for Reducing Transaction Fees in Blockchains." In Proceedings of the 23rd International Conference on Autonomous Agents and MultiAgent Systems *(AAMAS '24)* pp. 416-424.

2. **Sankarshan Damle** and Sujit Gujar. "Analyzing Crowdfunding of Public Projects Under Dynamic Beliefs." In Proceedings of the 23rd International Conference on Autonomous Agents and MultiAgent Systems *(AAMAS '24)* pp. 2225-2227.

3. **Sankarshan Damle**, Varul Srivastava and Sujit Gujar. "No Transaction Fees? No Problem! Achieving Fairness in Transaction Fee Mechanism Design." In Proceedings of the 23rd International Conference on Autonomous Agents and MultiAgent Systems *(AAMAS '24)* pp. 2228-2230.

4. **Sankarshan Damle**, Manisha Padala, and Sujit Gujar. "Combinatorial Civic Crowdfunding with Budgeted Agents: Welfare Optimality at Equilibrium and Optimal Deviation." In Proceedings of the 37th AAAI Conference on Artificial Intelligence *(AAAI '23)* pp. 5582-5590.

5. **Sankarshan Damle**, Sujit Gujar, and Moin Hussain Moti. "FASTEN: Fair and Secure Distributed Voting Using Smart Contracts". In Proceedings of the 3rd IEEE International Conference on Blockchain and Cryptocurrency *(ICBC '21)* pp. 1-3.

6. **Sankarshan Damle**, Boi Faltings, and Sujit Gujar. "Blockchain-based Practical Multi-agent Secure Comparison and its Application in Auctions". In Proceedings of The 20th IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology *(WI-IAT '21)* pp. 430-437.

7. **Sankarshan Damle**, Aleksei Triastcyn, Boi Faltings, and Sujit Gujar. "Differentially Private Multi-Agent Constraint Optimization". In Proceedings of The 20th IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology *(WI-IAT '21)* pp. 422-429.

8. **Sankarshan Damle**, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Designing Refund Bonus Schemes for Provision Point Mechanism in Civic Crowdfunding." In Proceedings of the 18th Pacific Rim International Conference on Artificial Intelligence *(PRICAI '21)* pp. 18-32.

9. **Sankarshan Damle**, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Civic Crowdfunding for Agents with Negative Valuations and Agents with Asymmetric Beliefs." In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence *(IJCAI '19)* pp. 208–214.

10. **Sankarshan Damle**, Boi Faltings, and Sujit Gujar. "A Truthful, Privacy-Preserving, Approximately Efficient Combinatorial Auction For Single-minded Bidders." In Pro-

ceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems *(AAMAS '19)* pp. 1916–1918.

11. **Sankarshan Damle**, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Aggregating Citizen Preferences for Public Projects Through Civic Crowdfunding." In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems *(AAMAS '19)* pp. 1919–1921.

(Non-archival) Workshop Papers

1. **Sankarshan Damle**, Manisha Padala, and Sujit Gujar. "Welfare Optimal Combinatorial Civic Crowdfunding with Budgeted Agents". Appears in the 4th Games, Agents, and Incentives Workshop (*GAIW@AAMAS '22*).

2. **Sankarshan Damle**, Aleksei Triastcyn, Boi Faltings, and Sujit Gujar. "Differentially Private Multi-Agent Constraint Optimization". Appears in the 2nd AAAI Workshop on Privacy-Preserving Artificial Intelligence (*PPAI@AAAI '21*).

3. **Sankarshan Damle**, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Designing Refund Bonus Schemes for Provision Point Mechanism in Civic Crowdfunding.". Appears in the 2nd Games, Agents, and Incentives Workshop (*GAIW@AAMAS '20*).

Other Peer-reviewed Papers

Journal Paper

1. Vlasis Koutsos, **Sankarshan Damle**, Dimitrios Papadopoulos, Dimitris Chatzopoulos, Sujit Gujar. "AVeCQ: Anonymous Verifiable Crowdsourcing with Worker Qualities." IEEE Transactions on Dependable and Secure Computing *(TDSC)*.

Conference Papers

1. Samhita Kanaparthy, Manisha Padala, **Sankarshan Damle**, Ravi Kiran Sarvadevabhatla, and Sujit Gujar. "F3: Fair and Federated Face Attribute Classification with Heterogeneous Data." In Proceedings of the 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2023 *(PAKDD '23)* pp. 483-494.

2. Sambhav Solanki, Samhita Kanaparthy, **Sankarshan Damle**, and Sujit Gujar. "Differentially Private Federated Combinatorial Bandits with Constraints". In Proceedings of the 32nd European Conference on Machine Learning and 25th Principles and Practice of Knowledge Discovery in Databases, 2022 *(ECML PKDD '22)* pp. 620-637.

3. Anurag Jain, Sanidhay Arora, **Sankarshan Damle**, and Sujit Gujar. "Tiramisu: Layering Consensus Protocols for Scalable and Secure Blockchains". In Proceedings of the 4th IEEE International Conference on Blockchain and Cryptocurrency *(IEEE ICBC '22)* pp. 1-3.

4. Samhita Kanaparthy, **Sankarshan Damle**, and Sujit Gujar. "REFORM: Reputation Based Fair and Temporal Reward Framework for Crowdsourcing". In Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems, 2022, *(AAMAS '22)* pp. 1648–1650.

5. Samhita Kanaparthy, Manisha Padala, **Sankarshan Damle**, and Sujit Gujar. "Fair Federated Learning for Heterogeneous Face Data." In Proceedings of the 9th ACM IKDD CODS and 27th COMAD *(CODS-COMAD '22)* pp. 298-299.

6. Manisha Padala, **Sankarshan Damle**, and Sujit Gujar. "Learning Equilibrium Contributions in Multi-project Civic Crowdfunding". In Proceedings of The 20th IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology *(WI-IAT '21)* pp. 362-375.

7. Manisha Padala, **Sankarshan Damle**, and Sujit Gujar. "Federated Learning Meets Fairness and Differential Privacy". In Proceedings of the 28th International Conference on Neural Information Processing (ICONIP) of the Asia-Pacific Neural Network Society 2021 *(ICONIP '21)* pp. 692-699.

   Also received the Best Paper Award at the Deployable AI Conference (DAI ' 21).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Multiagent systems seem to be a natural metaphor for understanding and building a wide range of what we might crudely call artificial social systems."

– Michael Wooldridge [298]

"Privacy is not an option, and it shouldn't be the price we accept for just getting on the Internet."

– Gary Kovacs (CEO of Accela, former CEO of Mozilla Corporation)

⋆ ⋆ ⋆ ⋆ ⋆

Artificial Intelligence (AI) has become an integral part of our lives, influencing various aspects and bringing about significant changes. AI impacts our daily lives in several ways, including the smart devices we wear and the personal assistants we interact with through recommendations, cybersecurity, e-learning, language translation, and many, many more. Research in AI generally focuses on developing theories, techniques, and systems of one or more cognitive entities [269]. Over the last several decades, AI systems have matured, tackling complex and more realistic problems well beyond the scope of an individual agent.

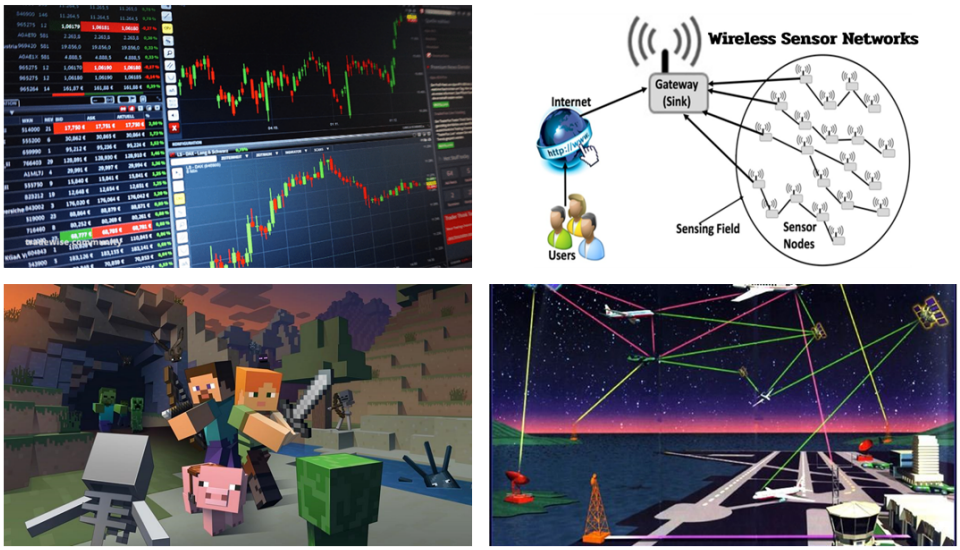


Figure 1.1: Representative real-world Applications of Multi-agent Systems. These include automated trading systems (image credit: [282]), distributed sensor networks (image credit: [277]), commercial games (image credit: [11]), and air-traffic management systems (image credit: [96]).

An *AI agent's* capacity is limited by its knowledge, computational resources, and perspective [269]. Simon [259] refers to this limitation as *bounded rationality*; this remains one of the key reasons for creating problem-solving organizations. Modularity and abstraction are powerful tools by which one can tackle complex problems.

1.1 Motivation

Such complex problem-solving organizations comprise dynamic environments where several agents interact, influencing each other actions and outcomes. These entities, whether individuals, organizations, or AI agents, contribute to a larger system where their decisions

impact themselves and the overall dynamics. Such systems are ubiquitous in modern times, including but not limited to traffic management, robotics, social networks, emergency responses, and financial markets.

1.1.1 Real-world Applications

Figure 1.1 provides an overview of some of these applications. We will describe some of them in detail next.

(1) **Automated Trading Systems:**

Automated trading systems, where each trader is an autonomous agent interacting with other agents and the market to make trading decisions [23, 171, 238]. The agents can include market-making, trend-following, or mean-reverting agents. These agents can buy and sell stocks, bonds, commodities, or currencies based on market conditions, risk preferences, and trading strategies.

(2) **Distributed Sensor Networks:**

Distributed sensor networks are networks comprising sensors as agents that can communicate with other sensors and perform tasks such as data collection, processing, and fusion [35, 170, 279]. The study of distributed sensor networks under the lens of agents interacting with each other in a dynamic environment is particularly relevant in applications like environmental monitoring, smart cities, surveillance, and industrial automation, where real-time data collection, analysis, and decision-making are critical.

(3) **Commercial Games:**

Video games are applications incorporating design patterns reminiscent of virtual dynamic environments. Entities like game objects or actors function as autonomous agents, engaging in interactions with one another to simulate intricate systems. These agents can interact with fellow characters and the environment, aiming to accomplish objectives, express emotions, and craft immersive experiences [122, 186].

(4) **Air-traffic Control Systems:**

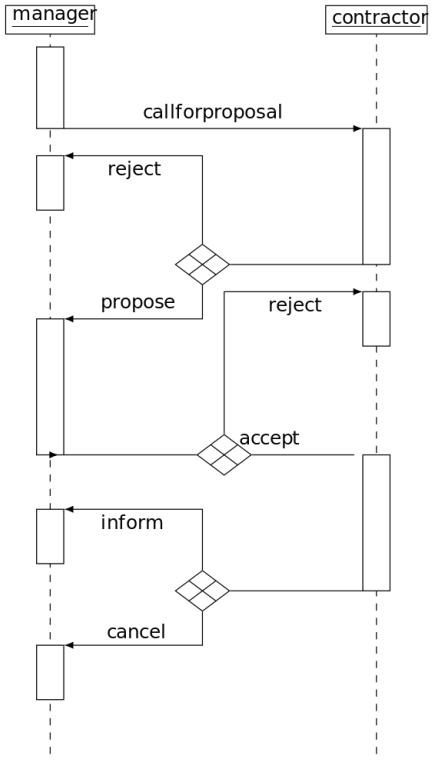


Figure 1.2: The Contract Net Protocol (image credit: [64])

Air traffic control systems are services that provide guidance to aircraft in controlled airspaces and information and support to pilots in uncontrolled airspaces. Such control systems aim to ensure safety, order, and efficiency for air traffic. We can imagine these as systems where each aircraft is an agent that can negotiate with other aircraft and ground stations to coordinate flight plans, avoid collisions, and optimize route and fuel consumption [37, 176, 210].

Next, we will examine classic case studies of such complex systems with multiple interacting agents.

1.1.2 Case Study: The Contract Net Protocol

The Contract Net Protocol, introduced by Smith [261] in 1980, is probably one of the earliest protocols to propose a distributed solution for task allocation among a group of autonomous agents. Task allocation is a common problem in various domains, such as distributed and/or cloud computing, manufacturing units, smart grids, and even search and rescue operations. The Contract Net Protocol provides a communication channel between the manager and the agents it wishes to allocate the tasks to (referred to as "contractors"). The manager proposes tasks to the participating contractors. These contractors send "proposals" to the manager based on which the manager chooses the allocations. Figure 1.2 depicts an illustration of the protocol.

The success of the Contract Net Protocol was instrumental in driving research in systems with multiple interacting agents. The protocol's ability to facilitate decentralized task allocation in multi-agent environments inspired further exploration into enhancing cooperation and coordination among autonomous agents. Furthermore, introducing *game theory* to these systems sparked interest in negotiation strategies without direct human intervention. This found practical applications in various domains, with e-commerce being a classic example [213].

1.1.3 Case Study: Llotja (a fish market) of Blanes

In the realm of online marketplaces, automated negotiation mechanisms, influenced by the principles of the Contract Net Protocol and game theory, have been employed to optimize resource allocation, pricing strategies, and overall system efficiency, showcasing the impact of these advancements in real-world scenarios. In one of the classic proposals in 1999, Noriega et al. [213] show how to take an instance of a traditional auction place, the Llotja (a fish market) of Blanes, and convert it into a virtual and dynamic electronic fish market. More concretely, the virtual market comprises autonomous agents – compromising customers and vendors – that perform the functions essential for an auction. These func-

tions include vendor/customer and goods registration, the bidding process, good delivery, and payment settlement.

Agents: Interacting, Coordinating, Competing. The common theme among these applications is that the agents interact with each other, either in collaboration or competition, with their actions and decisions having repercussions on both their own trajectories and the broader system. This "collective" behavior allows the agents to overcome computational limitations due to bounded rationality, lack of resources, and perspective. The dynamics arising from these interactions often lead to emergent behaviors and complex system-level outcomes. To further study the nature of agents' collective behavior, researchers introduced *multi-agent systems* (MAS) as an area that investigates the control and modeling of such complex systems made up of autonomous agents.

1.2 Multi-agent Systems (MAS)

The observation by Simon [259] that an individual agent is limited by its knowledge, resources, and perspective (i.e., bounded rationality) hinted that to cater to complex systems, it is essential to have modularity. Furthermore, as seen in Chapter 1.1, collaborative behavior is instrumental in various practical scenarios. This motivates the study of Multi-agent systems (MAS) that (i) offer modurality [269] and (ii) help formally study the behavior of collaborative agents in dynamic environments. In fact, from Sycara et al. [268], an effective way to solve a complex problem is by designing multiple functionally specific and (nearly) modular components (aka "agents") that specialize in solving a particular problem aspect. The general idea of a MAS is to create a decomposition in which agents use "the most appropriate paradigm for solving its particular problem" [269]. For interdependent problems, MASs require agents to coordinate with each other to ensure that the interdependencies get resolved.

1.2.1 Characteristics of MAS

The general characteristics of MAS include [269]: (i) agents have incomplete information or ability to solve the problem (i.e., their viewpoints are limited), (ii) there is no system of global control, (iii) data are decentralized, and (iv) computation is asynchronous. The first characteristic ensures the underlying problem is too complex for an individual agent to solve. The second characteristic allows room for interconnection and interpretation of existing legacy systems. The third characteristic results in scenarios naturally regarded as a society of autonomous and interacting agents. Last, the fourth characteristic promotes solutions that use information sources that may be spatially distributed. We further illustrate these characteristics through the following example of meeting-scheduling.

Example 1.1: Meeting-scheduling [113]

Consider the meeting-scheduling problem [113]. Each agent is a scheduler that manages the calendar of its user. We have independent agents for each user in the system that manage their calendars. Here, we have no global control and decentralized data. The agents can also be customized to reflect the constraints or preferences of their users. Thus, no single agent can derive an optimal schedule independently.

Meeting-scheduling is a quintessential example of a MAS [113, 269]. It highlights the importance of collaborative agents – no single agent can derive a useful schedule on its own. However, it also illustrates the need for the collaboration to be *private*. If the agents exchange information (i.e., their users' private calendars) in the plain, they risk leaking sensitive user information. So, while collaboration helps an MAS reach more useful outcomes, it is imperative that these MAS also satisfy certain desirable properties pertaining to agent behavior and information exchange (e.g., privacy of the user information). We will discuss some of these properties later in Chapter 1.4.

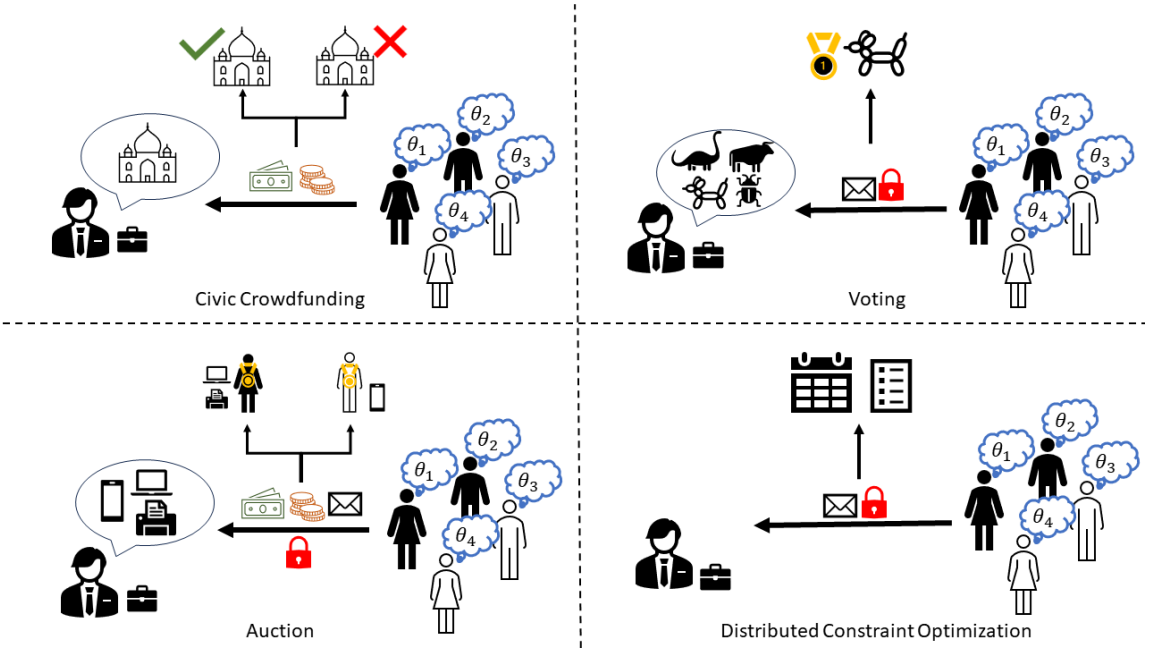


Figure 1.3: Overview of the Multi-agent System (MAS) applications studied in this thesis. In particular, we look at civic crowdfunding (top-left), voting (top-right), auctions (bottom-left), and distributed constraint optimization (bottom-right).

1.2.2 MAS: Other Real-world Applications

Post the formalization of multi-agent systems as an established research area, the *distributed vehicle motoring* (DVMT) [88, 89] emerged as one of its most well-known applications. In DMVT, a set of geographically distributed agents monitor the vehicles that pass through their areas. These agents attempt to interpret what vehicles pass through the global areas and track their movements. Other earlier examples include OASIS [177], an agent-based air-traffic control system used in Sydney, Australia, WARREN [268] which was a financial portfolio management system, and YAMS [227] whose aim is to efficiently manage the production process across multiple factories that also comprise independent components among themselves.

1.3 MAS: Applications In Focus

This thesis studies several practical multi-agent systems. Looking back at the meeting-scheduling problem from Example 1.1, one may observe that the independent agents must communicate among others to decide on an (i) "optimal" schedule that (ii) satisfies the individual user constraints. In the MAS literature, such problems are referred to as *distributed constraint optimization* (DCOP) problems. In addition, we also look at a couple of e-governance and e-commerce applications. First, *civic crowdfunding*, which is an effective method of using the "power of the crowd" to raise money specifically for public projects (e.g., public parks). Second, *voting*, particularly general elections. Last, we look at auctions, in particular, (i) *combinatorial auctions*, which involve agents bidding for more than one subset of items among those being auctioned, and (ii) *transaction fee mechanisms*, which involve adding transactions to the next block in a blockchain from the set of outstanding transactions. Figure 1.3 depicts these applications in terms of the agent interaction and the application outcome. Details follow.

1.3.1 Civic Crowdfunding

In civic crowdfunding (CC), a *social planner* seeks voluntary contributions from individuals specifically to crowdfund public projects. Typical examples include public parks, libraries, bridges, and other community welfare projects. It is important to note that such public projects are (i) *non-rivalrous*, i.e., an agent consuming the project (e.g., using a bridge for commute) does not decrease the amount for others, and (ii) *non-excludable*, i.e., an agent cannot be stopped from consuming the project.

Unfortunately, because of these properties, a *strategic* or a *rational* agent may prefer not to contribute to a project. Instead, the agent may believe that the other agents may contribute, and it can merely consume the project once it gets crowdfunded. The CC literature refers to this lack of incentive for an agent to contribute as "free-riding" [19].

Thus, there is a need to design incentive mechanisms for such strategic agents to contribute to the project.

To overcome free-riding, Zubrickas [312] proposes that the social planner provide contributing agents with *refunds* if the project does not meet its target by its deadline. The author's proposal induces a game among the agents such that they are incentivized to contribute. In fact, the game admits an equilibrium such that the project gets crowdfunded, and the social planner is not required to provide refunds! In this thesis, we look at several interesting new directions to the work by Zubrickas [312]. More concretely, we (i) relax the restricted assumptions made on agent behavior and the information they hold, (ii) look towards the efficiency of deploying the process over blockchain [203, 297], and (iii) extend the theory for the multi-project or combinatorial case. For further details, we refer the reader to Chapter 1.5.2.1, Chapter 1.5.2.2, Chapter 1.5.2.3 and Chapter 1.5.2.4.

1.3.2 Voting

Elections play a vital role in democratic governance, allowing societies to select representatives when direct democracy isn't feasible. Modern representative democracies have employed elections since the 17^{th} century, allowing eligible citizens to participate in decision-making by casting their votes. The key to designing a *fair* election is ensuring that voters can freely express their preferences.

A natural method to design such fair elections is to ensure (i) voter anonymity, (ii) vote concealment, (iii) vote immutability (i.e., a vote, once cast, cannot be altered), and (iv) double voting inhibition. In literature, works exist that ensure a subset of these properties [27, 136, 304] or do so at the cost of their scalability (i.e., the number of voters the system can practically process) [307]. Our goal in this space is to design a fair election protocol that is also scalable. For further details, we refer the reader to Chapter 1.5.3.1.

1.3.3 Auctions

As seen in Chapter 1.1.3, auctions are a classic MAS application. Typically, auctions involve agents acting as bidders who bid for their items of interest. The seller allocates the items to the (sub) set of bidders based on their bids and certain allocation rules. Auctions differ based on the allocation and payment rules (e.g., first-price auctions, Vickrey or second-price auction [284]) or the number of items being auctioned (e.g., single item & single unit, single item & multi-unit, combinatorial auctions). In this thesis, we focus on Transaction Fee Mechanisms (TFMs), a classic application of auctions for transaction inclusion in cryptocurrencies like Bitcoin [203] and Ethereum [297] as well as combinatorial auctions.

1.3.3.1 Transaction Fee Mechanism Design

In cryptocurrencies like Bitcoin [203] and Ethereum [297], the *miner*, i.e., the agent who proposes the next block, includes a set of transactions to its block from the set of outstanding transactions – referred to as the *mempool*. Naturally, as the miner is self-interested, it looks to maximize its revenue by adding the set of transactions that pay the maximum *transaction fees* to it. Zooming out, this means that the transaction creators 'bid' for a slot in the miner's block. The miner acts as the seller that allocates the transactions to its block and receives the transaction fee as payment. Roughgarden [245] introduces this miner-transaction creator interaction as a Transaction Fee Mechanism (TFM) design. E.g., Bitcoin's TFM can be considered a first-price auction where the miner optimally adds transactions that maximize its revenue, and the transaction creators pay the transaction fee they commit to.

While the transaction fee acts as an incentive that allows TFMs to satisfy several important agent-specific incentive properties [62, 103, 245], recent data suggest that transaction fees have grown considerably more than required [193]. This may limit the cryptocurrencies market penetration. For instance, the commission-less UPI payment network has

superseded the long-established commission-based debit/credit card payment network in India [1]. Motivated by this, we look at designing TFMs that look to reduce the transaction fees collected from the transaction creators as well as 'fair' TFMs that allow transactions with zero fees to have a non-zero probability of being included in the miner's block. For further details, we refer the reader to Chapter 1.5.2.5 and Chapter 1.5.2.6.

1.3.3.2 Combinatorial Auctions

We next focus on combinatorial auctions. These auctions allow bidders, agents in our language, to bid for multiple subsets of items simultaneously. Combinatorial auctions also generate more revenue than other type of auctions [292]. Popular examples include procurement auctions, spectrum auctions [190], and airport take-off slot auctions [237].

However, using auctions directly is a security and privacy risk for an agent's bid and its subset of items. Disclosure of an agent's public identity reveals its interest in acquiring auctioned items. The revelation of an agent's *bidding information* (bid value and the combination of preferred items) to an auctioneer or other participating agents may expose its profits, economic situations, and preferences for specific items to its contemporaries. An auctioneer may further exploit this information in future auctions. Consequently, we desire an auction protocol in which only the winning agents' combination of preferred items is made public while preserving the privacy of the identities and the bidding information of the other agents. Preserving the privacy of the participating agents and their private information is integral to such mechanisms. To summarize, we look to design a privacy-preserving combinatorial auction that preserves each agent's bidding information from the public and the auctioneer, even after the auction ends. For further details, we refer the reader to Chapter 1.5.3.2.

1.3.4 Distributed Constraint Optimization Problems (DCOPs)

At last, we take a re-look at DCOPs (see Example 1.1). DCOP is a problem where agents collectively compute their value assignments to maximize (or minimize) the sum of resulting *constraint* rewards. In DCOP, constraints quantify each agent's *preference* for each possible assignment. DCOPs help model various multi-agent coordination and resource allocation problems like meeting-scheduling and graph-coloring-related applications such as mobile radio frequency assignments. For instance, consider the problem of meeting-scheduling in which several Chief Executive Officers (CEOs) aim to decide a date and time to meet. Each CEO will have a constraint for each date and time slot assignment, quantifying its preference for the assignment. The preferences may depend on the CEOs' availability and favorable slots. In this scenario, the CEOs cannot directly employ a centralized coordinator to decide on an agreeable meeting slot. The coordinator will require information regarding the CEOs' availability – which is often sensitive. Alternatively, the CEOs can generate a suitable schedule by modeling the problem as a DCOP and using any DCOP-solving algorithms. However, researchers show that despite their distributed nature, DCOP algorithms may themselves leak sensitive information [100].

While several privacy-preserving algorithms exist [99, 100, 167, 272, 273], they rely on computationally expensive cryptographic primitives and secure multi-party computation protocols. This dependence results in the algorithms not being scalable, i.e., the algorithms are only practical to deploy for a small number of agents. In this thesis, we look at designing a private DCOP algorithm that preserves (constraint) privacy and is also scalable. For further details, we refer the reader to Chapter 1.5.3.3.

1.4 MAS: Desirable Properties

So far, we have introduced MAS and looked at several real-world deployments of them. Furthermore, we discussed MAS applications studied as part of this thesis. What remains

unclear is the requirements that an MAS should satisfy so that it is desirable for real-world deployment. In this thesis, we focus on the following desirable properties of a MAS.

Definition 1.1: What qualifies as a "good" Multi-agent System?

We say that a MAS is desirable if it satisfies all or any combination of the following properties:

1. Game-theoretic Soundness
2. Fairness
3. Security and Privacy

Given their increasing adoption, we believe that multi-agent systems must satisfy these properties. In fact, a large body of work pertaining to these properties can be found in contemporary AI research [3, 7, 86, 87, 138, 161, 195, 240, 289, 312]. We discuss these properties in detail next.

1.4.1 Game-theoretic Soundness

As alluded to briefly in Chapter 1.3, often in MAS, the participating agents may be strategic, i.e., the agents may opt for strategies (outside of the desired strategy) such that the "deviating" strategy increases their utility. Consequently, the rich game theory literature aims to tackle this strategic behavior through incentives or modeling agents' utility behavior. *Mechanism design* is a game-theoretic tool concerned with settings where a social planner faces the problem of aggregating the announced preferences of multiple agents into a collective decision when the agents exhibit strategic behavior. E.g., e-governance applications such as auctions and e-commerce applications such as civic crowdfunding. Through mechanism design, researchers aim to promote agents' equilibrium strategies into those desired.

Some popular game theory properties in mechanism design[1] include Dominant Strategy Equilibrium (DSE), Nash Equilibrium (Pure and Bayesian) (NE), Individual Rationality (IR), and Incentive Compatibility (IC) [205]. For instance, a strategy is DSE if the agent receives maximum utility with it irrespective of the strategy employed by the remaining agents. With a slight difference, a strategy is pure-NE if the agent receives maximum utility with it *only* if the remaining agents employ the same strategy. IR states that playing a given strategy guarantees non-negative utility to the agent. Lastly, IC states that an agent receives maximum utility when eliciting its true preference/valuation. From a mechanism design perspective, we examine civic crowdfunding (CC) and transaction fee mechanisms (TFMs), delving into the desirable properties they satisfy.

1.4.2 Fairness

Typically, fairness in MAS revolves around two categories: (i) *group fairness*, which involves fair classification in ML systems, and (ii) *algorithmic fairness*, which deals with fair resource allocation of goods (or chores).

1.4.2.1 Group Fairness

Machine Learning (ML) systems are frequently employed in decision-making. E.g., criminal risk assessment, credit approvals, and online advertisements. Researchers have observed that these ML systems inadvertently introduce societal bias [61][2]. For instance, the firm ProPublica conducted a study of a risk assessment tool that was widely used by the judiciary system in the US [234]. ProPublica observed that the risk values for recidivism estimated for African-American defendants were, on average, higher than for Caucasian defendants. Since then, research on fairness in ML gained traction, especially in quantifying and satisfying several notions of fairness. Popular group fairness notions require different sensitive groups (e.g., age or race) to receive beneficial outcomes in similar proportions.

[1] Chapter 2.1 provides a comprehensive summary of the game theory and mechanism design literature.
[2] nytimes.com/2019/12/19/technology/facial-recognition-bias.html

These include Demographic Parity, Disparate Impact, and Equalized odds. For details, we refer the reader to [191].

1.4.2.2 Algorithmic Fairness

This category revolves around allocating (say) m indivisible items among (say) n agents who report their valuations for the items. The objective is to ensure fair allocation for a desirable notion of fairness. Such a scenario often arises in the division of inheritance among family members, divorce settlements, and distribution of tasks among workers (e.g., spliddit.org).

Envy-freeness (EF) is the most common notion of fairness here. EF ensures that no agent has higher utility for other agent's allocation [109]. Other notions, such as Proportionality, ensure that every agent receives at least $1/n$ of its complete bundle value [265]. There is also a relaxation of PROP, the max-min share (MMS) allocation [18]. Imagine asking an agent to divide the items into n bundles and take the minimum-valued bundle. The agent would divide the bundles to maximize the minimum utility, which is the agent's MMS share. An MMS allocation guarantees every agent its MMS share. For further details, we refer the reader to [9, 17].

1.4.2.3 Other Categories of Fairness

Researchers have recently highlighted the need for fairness for algorithms in general [65]. Here, we look at two such applications based on *fair-incentive mechanisms*.

1. **Crowdsourcing.** Schmidt [253] introduce the idea of fair rewards in human-centered crowdsourcing . Fair crowdsourcing platforms are necessary to ensure the participation of the crowd. As a result, recent research has focused on mechanisms with the fair provision of rewards [118, 119, 137, 200].

2. **Cryptocurrencies.** In the world of decentralized cryptocurrencies like Bitcoin [203] and Ethereum [42], a lack of fairness can also correspond to strategic miners collecting

transactions with higher transaction fees and leaving out those with more minor (or no) transaction fees [257]. Naturally, works such as [257] increase the fairness in transaction order by proposing BitcoinF, which allocates a section of the block space that must include transactions in first-in-first-out order.

Chapter 2.4 formally describes the fairness notions presented here. Particularly, we look at fair incentives in TFMs – ensuring a reduction in the agent's transaction fee to increase a cryptocurrency's market penetration.

1.4.3 Privacy

As mentioned, MAS applications are widely used in various domains. For further acceptance, mostly when multiple agents interact with the system, we must aim to preserve the privacy of participants' information in such applications. Researchers resolve the privacy challenges in MAS either through (i) *cryptosystems* or (ii) *differential privacy* (DP).

Cryptosystems. This approach involves a fusion of various cryptographic primitives (e.g., encryptions, commitments, hashes, secure multi-party computation) to provide required privacy guarantees. Researchers have successfully used such an approach for several MAS applications including deep learning [60, 101, 198, 299, 308], reinforcement learning [145, 224, 266], and artificial intelligence [179, 178, 195, 225, 226, 274].

To assert the correctness of the computation over encrypted data/information, i.e., *security* of the privacy-preserving protocol, *zero-knowledge proofs* (ZKPs) are employed. ZKP is a method by which a party, called a *Prover* ($\mathcal{P}$), can convince another party, called a *Verifier* ($\mathcal{V}$), that it knows some information ω, without revealing ω (or any other information related to ω). A few standard ZKP techniques include commit-response methods [87, 195], Fiat-Shamir heuristic [104] and zk-SNARKs [178, 179].

Differential Privacy (DP). This technique involves randomizing computation or adding calibrated random noise to the statistics before releasing them [91]. DP has emerged as the premier alternative to traditional cryptosystems for privacy preservation in the last

decade. Similar to the earlier approach, DP also finds use in privacy-preserving MAS literature for deep learning [3, 206, 221, 256], reinforcement learning [58, 112, 286], and artificial intelligence [139, 288].

We refer the reader to Chapter 3 for a comprehensive summary of the required security and privacy preliminaries. We construct privacy-preserving applications for MAS, such as DCOPs, voting, and combinatorial auctions.

1.5 Our Goal and Contributions

We first define the broad goal of the thesis. Next, we zoom in a level and summarize the concrete contributions made.

1.5.1 Our Goal

We remark that MAS is a field of artificial intelligence (AI) that studies how multiple agents interact with each other and their environment to achieve specific objectives. MAS has applications in various domains such as finance, transportation, healthcare, etc. However, as these systems involve multiple agents with potentially conflicting goals and interests, we believe that ensuring their security and privacy, game-theoretic soundness, and fairness is of utmost importance.

This report focuses on these aspects, and Table 1.1 presents a taxonomy of our contributions in this space. Specifically, we work on popular MAS applications such as voting, where the goal is to determine the preference of a group of agents for a set of alternatives; combinatorial auctions, where agents bid on combinations of items; distributed constraint optimization, where agents work together to find solutions to a problem with constraints; and civic crowdfunding with refunds, where agents contribute money towards a collective goal and are refunded if the goal is not met.

Our contributions in these areas involve designing algorithms and protocols that ensure the security and privacy of agents' information, promote fair and efficient outcomes, and

MAS	Chapter	Security & Privacy	Game-theoretic Soundness	Fairness
Civic Crowdfunding [72, 74, 75, 76, 77]	Chapters 4, 5, 6, 7	✗	✓	✓
Transaction Fee Mechanism [78, 79]	Chapters 8, 9	✗	✓	✓
Voting [73]	Chapter 10	✓	✗	✗
Combinatorial Auction [70, 71]	Chapter 11	✓	✗	✗
Distributed Constraint Optimization [80]	Chapter 12	✓	✗	✗

Table 1.1: Taxonomy of this thesis' contributions and the subset of the MAS desirable properties they consider.

are robust to various manipulation and attacks. We analyze existing MAS applications, identify vulnerabilities or limitations, and propose solutions to these issues.

1.5.2 PART A: Game-theoretically Sound and Fair Mechanism Design

The first part of this thesis focuses on game-theoretically sound and fair incentive-based mechanism design. In particular, we focus on MAS, such as Civic Crowdfunding (CC) and Transaction Fee Mechanisms (TFMs). More concretely, PART A comprises the following contributions.

1.5.2.1 Civic Crowdfunding for Agents with Asymmetric Belief and Negative Preferences

Agent Preferences. We also focus on aggregating citizen preferences for public projects through civic crowdfunding. Existing civic crowdfunding mechanisms such as PPR and PPS consider only citizens with positive valuations towards the public project. Public projects aim to cater to the majority, so they should be provisioned only if the majority prefers it. For this, in Chapter 4 (based on [74, 75]), we propose a methodology to convert existing civic crowdfunding mechanisms for positive preferences to cater to markets having both types of agents. Specifically, we adapt existing PPR and PPS mechanisms to design

PPRN and PPSN that incentivize agents to contribute towards or against the project's provision based on their preference.

Agent Beliefs. Moreover, as mentioned earlier, the mechanisms assume that each agent has a symmetric belief about the project getting provisioned. To address asymmetric beliefs, in Chapter 4 (based on [74, 75]), we propose a novel reward scheme, Belief Based Reward (BBR), based on the Robust Bayesian Truth Serum (RBTS) mechanism. BBR rewards agents based on their belief in the project's provision. Using this reward scheme, we introduce a general framework for civic crowdfunding, allowing agents with asymmetric beliefs about the project to get provisioned and incentivizing them to contribute towards the project's provision. Based on this framework, we also design two mechanisms, PPRx and PPSx, adapting PPR and PPS, respectively, and prove that the project is provisioned at equilibrium in both mechanisms.

1.5.2.2 Efficient Civic Crowdfunding over Blockchain

In Chapter 5 (based on [76]), we identify essential properties a refund bonus scheme must satisfy to curb free-riding while avoiding the race condition. We prove Contribution Monotonicity and Time Monotonicity as sufficient conditions for this. We show these conditions are also necessary if a unique equilibrium is desirable. We propose three refund bonus schemes satisfying these conditions, leading to three novel mechanisms for CC - PPRG, PPRE, and PPRP. We show that PPRG is the most cost-effective when deployed as an SC. We prove that under certain modest assumptions, in PPRG, the project is funded at equilibrium.

1.5.2.3 Combinatorial Civic Crowdfunding with Budgeted Agents

The existing CC literature mentioned above (i.e., [52, 53, 74, 75, 76, 312]) only focuses on the crowdfunding of a single project. To this end, in Chapter 6 (based on [77]), we present several foundational results for CC for multiple projects. Given budget-constrained agents,

we argue that funding the welfare optimal subset may be desirable. We show that the optimal subset is funded at equilibrium under specific assumptions on the overall budget and agents' individual budgets. For any other scenario, an agent may be incentivized to deviate from funding the optimal set. However, computing the optimal deviation in the multi-project case is NP-Hard.

1.5.2.4 Analyzing Civic Crowdfunding under Dynamic Beliefs

As mentioned above, in Chapter 4, we relax the assumption on agents' beliefs by incorporating agents with asymmetric beliefs. However, the mechanisms proposed in Chapter 4 still assume that the agent's beliefs are static, i.e., they do not evolve with time. This assumption seems restrictive — in CC, agents can observe the remaining time to the deadline and the net contribution at any time. Thus, agent beliefs may evolve with time and be in line with the information at hand. In Chapter 7 (based on [72]), we argue that an agent's dynamic belief will evolve as a random walk. Based on this assumption, we characterize the equilibria of the CC game (i.e., the equilibrium point and agent strategies) if the dynamic belief evolves as a (i) martingale, (ii) sub-martingale, or (iii) super-martingale.

1.5.2.5 Achieving Fairness in Transaction Fee Mechanism Design

Prominent cryptocurrencies like Bitcoin and Ethereum handle over a million transactions daily, creating a scenario where strategic miners aim to maximize their utility by selecting transactions with higher fees. The transaction fee mechanism (TFM) design literature aims to understand miners' and transaction creators' optimal behavior. As such, it focuses on standard incentive properties, which may not be sufficient for a cryptocurrency's increased market penetration. In Chapter 8 (based on [79]), we argue that a TFM is deemed "fair" to transaction creators when it satisfies specific notions, including Zero-fee Transaction Inclusion and Monotonicity. We show that existing TFMs either fail to satisfy these notions or do so at a considerable cost to miners' utility. In response, we present

a new set of TFMs that satisfy the desirable incentive properties and our novel fairness notions.

1.5.2.6 Designing Redistribution Mechanisms for Reducing Transaction Fees in Blockchains

Recall that Transaction Fee Mechanisms (TFMs) manage agent transactions in blockchains, determining transaction fees. However, transaction fees have become high due to rising demand and limited block resources. To address this, in Chapter 9 (based on [78]), we introduce Transaction Fee Redistribution Mechanisms (TFRMs). TFRMs redistribute the VCG payments from TFMs as rebates to minimize fees using Redistribution mechanisms [129]. In particular, we propose two TFRMs that satisfy several desirable incentive properties while resulting in a non-zero reduction of transaction fees, even in the presence of a strategic miner.

1.5.3 PART B: Security and Privacy

PART B is the privacy-focused part of this thesis. We particularly focus on the privacy of interacting agents in MAS, such as voting, combinatorial auction, and distributed constrained optimization.

1.5.3.1 FASTEN: Fair and Private Voting

Electing democratic representatives via voting has been common since the 17^{th} century. However, these mechanisms raise concerns about fairness, privacy, vote concealment, fair calculations of tally, and proxies voting on behalf of the voters. Ballot voting, and in recent times, electronic voting via electronic voting machines (EVMs), improves fairness by relying on centralized trust. Homomorphic encryption-based voting protocols also assure fairness but cannot scale to large-scale elections such as presidential elections. To resolve these issues, Chapter 10 (based on [73]) leverages the blockchain technology of distributing trust

to propose a smart contract-based protocol, namely, FASTEN. There are many existing protocols for voting using smart contracts. We observe that these either are not scalable or leak the vote tally during the voting stage, i.e., do not provide vote concealment.

In contrast, we show that FASTEN preserves voters' privacy, ensures vote concealment and immutability, and avoids double voting. We prove that the probability of privacy breaches is negligibly small. Further, our cost analysis of executing FASTEN over Ethereum is comparable to most of the existing election costs.

1.5.3.2 STOUP: Secure and Trustworthy Combinatorial Auction

As mentioned, MAS applications are widely used in a variety of domains. For further acceptance, mostly when multiple agents interact with the system, we must aim to preserve the privacy of participants' information in such applications. Towards this, Yao's Millionaires' problem (YMP) [300], i.e., to determine the richer among two millionaires privately, finds relevance. To this end, Chapter 11 (based on [71]) presents a novel, practical, and verifiable solution to YMP, namely, Secure Comparison Protocol (SCP). We show that SCP achieves this comparison in a constant number of rounds without encryption and does not require continuous participant involvement. SCP uses semi-trusted third parties - which we refer to as privacy accountants - for comparison, who do not learn any information about the values. The probability of information leaks is negligible in the problem size. We leverage the Ethereum network in SCP for pseudo-anonymous communication, unlike computationally expensive secure channels such as Tor. We present a Secure, Truthful cOmbinatorial aUction Protocol (STOUP) for single-minded bidders to demonstrate SCP's significance. We show that STOUP, unlike previous works, preserves the privacies relevant to an auction, even from the auctioneer. We demonstrate the practicality of STOUP through simulations.

1.5.3.3 Differentially Private Multi-agent Constraint Optimization

Several optimization scenarios involve multiple agents that desire to protect the privacy of their preferences. There are distributed algorithms for constraint optimization that provide improved privacy protection through secure multiparty computation. However, it comes at the expense of high computational complexity. It does not constitute a rigorous privacy guarantee for optimization outcomes, as the result of the computation itself may compromise agents' preferences. In Chapter 12 (based on [80]), we show how to achieve privacy, specifically differential privacy, by randomizing the solving process. In particular, we present P-Gibbs, which adapts the SD-Gibbs algorithm to obtain differential privacy guarantees with much higher computational efficiency. Graph coloring and meeting scheduling experiments show the algorithm's privacy-performance trade-off for varying privacy budgets and the state-of-the-art SD-Gibbs algorithm.

1.6 Thesis Outline

We divide the thesis into two parts, with PART A on game-theoretic soundness and PART B on security, privacy, and blockchains.

- **Chapter 2.** With a particular focus on PART A, Chapter 2 presents the required preliminaries for game theory and mechanism design, combinatorial auctions, civic crowdfunding, and fair reward mechanisms.

- **Chapter 3.** Likewise, towards PART B, Chapter 3 provides the required cryptographic background, summarizing cryptographic primitives such as hash functions, encryptions, and commitments, as well as zero-knowledge proofs, differential privacy, blockchains and Transaction Fee Mechanisms (TFMs).

PART A: Game-theoretic Soundness

- **Chapter 4.** This chapter presents civic crowdfunding for agents with asymmetric beliefs and negative valuation.
- **Chapter 5.** Here, we present several efficient refund bonus schemes for blockchain-based civic crowdfunding.
- **Chapter 6.** With this chapter, we present our model of combinatorial civic crowdfunding with budgeted agents.
- **Chapter 7** Here, we characterize the underlying game induced in civic crowdfunding in the presence of agents with dynamic beliefs.
- **Chapter 8.** Pivoting to TFMs and fair incentive mechanisms, we present TFMs that satisfy novel fairness notions while retaining desirable game-theoretic properties.
- **Chapter 9.** For the last chapter for PART A, we look at reducing the explosion of transaction fees in TFMs. Particularly, we employ a Redistribution mechanism-based approach that offers rebates to agents to reduce the fee they pay for transaction inclusion.

PART B: Security and Privacy

- **Chapter 10.** Moving forward with PART B, this chapter presents our novel fair and private voting protocol, namely FASTEN.
- **Chapter 11.** This chapter presents STOUP, a blockchain-based private and verifiable combinatorial auction.
- **Chapter 12.** We end PART B by looking at differentially private multi-agent constraint optimization in this chapter.
- **Chapter 13.** We conclude this thesis with this chapter with a discussion on the contributions presented, their potential impact, and promising directions to further explore in the future.

Chapter 2

Preliminaries: Game Theory

This thesis chapter provides a comprehensive overview of fundamental concepts in game theory, mechanism design, civic crowdfunding, and fair reward mechanisms. Game theory is a theoretical framework to analyze strategic interactions among rational actors, providing insights into decision-making processes and outcomes. Mechanism design, an extension of game theory, is crucial for designing incentive-compatible mechanisms that encourage desirable behaviors. The chapter then delves into civic crowdfunding, a method to exhibit participatory decision-making in which communities collectively fund projects. By combining insights from game theory and mechanism design, the chapter presents a literature survey of known civic crowdfunding mechanisms that guarantee a project's funding at equilibrium. Additionally, we look at fair reward mechanisms for crowdsourcing, addressing challenges related to the equitable distribution of rewards among participants.

⋆ ⋆ ⋆ ⋆ ⋆

A_1 \ A_2	H	T
H	(-100, 100)	(100, -100)
T	(100, -100)	(-100, 100)

Table 2.1: Player Utilities for Matching Pennies (Example 2.1)

2.1 Game Theory

This thesis focuses on multi-agent systems. Naturally, we desire a tool that can model the interaction between these agents, depending on their type and environment. Game theory helps model the interaction between these agents. In general, game theory assumes that the agents part of the system are *rational*, i.e., they are interested in maximizing their *utility*. An agent's utility structure may be *cooperative* or *conflicting*.

Generally, a game is described by four properties [205]: (i) Agents: individual/group of individuals making decisions, (ii) Rules, (iii) Outcomes, and (iv) Preferences. For illustrative purposes, consider the following example.

Example 2.1: Matching Pennies

1. Agents: $\{A_1, A_2\}$
2. Rules: Each player simultaneously flips a coin
3. Outcomes: $\{(H, H), (T, H), (H, T), (T, T)\}$
4. Preferences: Table 2.1 gives the player's preferences. If the two coins get the same outcome, A_1 pays USD 100 to A_2; otherwise, A_2 pays USD 100 to A_1

Strategic Form Game. While Example 2.1 presents the constituents of a game, it is often useful to represent the game formally. That is,

Definition 2.1: Strategic Form Game [205]

A strategic form game Γ is the tuple $\langle A, \{S_i\}_{i \in A}, \{u_i\}_{i \in A} \rangle$ wherein,

1. $A = \{1, \ldots, n\}$ where $n \in \mathbb{Z}_{\geq 1}$ are the set of agents
2. $\{S_i\}_{i \in A}$ where each S_i is the strategy of agent i
3. $\{u_i\}_{i \in A}$ s.t. $u_i : S_1 \times S_2 \times \ldots \times S_n \to \mathbb{R}$ is the utility function of agent i

Every agent 'plays' their strategy simultaneously in the strategic form game and reports it to a central coordinator (often referred to as the *social planner*). The planner then computes the outcome and individual utilities. Let us look at the famous Prisoner's Dilemma[1] game to understand Definition 2.1 better.

Example 2.2: Prisoner's Dilemma

We have two prisoners, i.e., $A = \{1, 2\}$. Let the prosecutor act as the social planner without evidence to convict the prisoners. The prosecutor develops a clever plan to obtain a confession: The prisoners are made to sit separately so they cannot mutually communicate. Each prisoner i's strategy is either confessing or defecting. Their utilities are defined as follows.

- If both confess, they receive 5 years of jail time each, i.e., $u_1(C, C) = u_2(C, C) = -5$.
- If prisoner 1 confesses while prisoner 2 defects, prisoner 1 gets 1 year and prisoner 2 gets 10 years of jail time, i.e., $u_1(C, D) = -1$ and $u_2(C, D) = -10$.
- If prisoner 2 confesses while prisoner 1 defects, prisoner 2 gets 1 year and prisoner 1 gets 10 years of jail time, i.e., $u_1(D, C) = -10$ and $u_2(D, C) = -1$.
- If both defect, they receive 2 years of jail time each, i.e., $u_1(D, D) = u_2(D, D) = -1$.

Table 2.2 presents the prisoners' utility in the *matrix-form*. The row agent is Agent 1, and the first value corresponds to its utility, while Agent 2 is the column player, and the

[1] Veritasium's video is a great watch for those interested in the significance of Prisoner's Dilemma and game theory in general (youtube.com/watch?v=mScpHTIi-kM).

second value corresponds to its utility. C denotes the prisoner's choice to confess, and D denotes the prisoner's choice to defect.

Agent 1 \ Agent 2	D	C
D	(-2, -2)	(-10, -1)
C	(-1, -10)	(-5, -5)

Table 2.2: Player Utilities for Prisoner's Dilemma

Given such a setting, it is natural to wonder what strategies the prisoners should adopt to guarantee themselves a desirable (or *optimal*) outcome. Such a question leads us to the game theory concept of *equilibrium*.

2.1.1 Dominant Strategy Equilibrium

From Example 2.2, each agent's strategy comprises either confessing or defecting. Each agent wishes to choose a strategy that maximizes its utility. If there exists a strategy (e.g., confessing) that maximizes the utility of agent 1 irrespective of the strategy of agent 2, then we say that the strategy is **dominant**. Furthermore, we refer to the agents' dominant strategy profile as the dominant strategy equilibrium, defined as follows.

Definition 2.2: Weakly Dominant Strategy Equilibrium [205]

Given the game Γ, we refer to the strategy profile $s^\star = (s_1^\star, \ldots, s_n^\star)$ as dominant strategy equilibrium if $\forall i \in A$, the strategy $s_i^\star$ is a dominant strategy, i.e.,

$$u_i(s_i^\star, s_{-i}) \geq u_i(s_i, s_{-i}), \forall s_i \in S_i, s_i \neq s_i^\star \text{ and } s_{-i} \in S_{-i} \tag{2.1}$$

such that $\exists s_{-i} \in S_i$ for which $u_i(s_i^\star, s_{-1}) > u_i(s_i, s_{-i})$.

Here, s_{-i} denotes the strategy tuple of all agents except agent i. Adding to Definition 2.2, if the strict inequality exists $\forall s_i$ in Eq. 2.1, $\forall i$, we say that the profile is **strongly** dominant strategy equilibrium.

Dominant Strategy in Prisoner's Dilemma. We see from Table 2.2 that confess-confess is the dominant strategy profile in Prisoner's Dilemma. This is because, for Agent 1, $u_1(C, C) > u_1(D, C)$ and $u_1(C, D) > u_1(D, D)$. Likewise, for Agent 2, $u_2(C, C) > u_2(D, C)$ and $u_2(C, D) > u_2(D, D)$.

Pareto Optimality. The dominant strategy equilibrium utility for both the agents in the Prisoner's Dilemma is -5. However, Table 2.2 shows that defect-defect would have given the agents a higher utility of -2 each. We say that equilibrium is not **Pareto optimal** if (i) a change in the strategy profile (from $s^\star$ to $\tilde{s}$) improves the utility of at least one agent without affecting the utility of the remaining agents ("Pareto" improves) and (ii) $\tilde{s}$ is Pareto optimal if there are no possible Pareto improvements.

Dominant Strategy Equilibrium May Not Exist. While the dominant strategy equilibrium is useful for the agents to play (guaranteeing maximal payoff irrespective of the others' strategy), unfortunately, such an equilibrium may not always exist.

For instance, recall the matching pennies game from Example 2.1. Here, consider the agent A_1. If A_1 chooses H, its utility $u_1(H, H) = -100$ if A_2 also chooses H and $u_1(H, T) = 100$ if A_2 chooses T. That is, $u_1(H, T) > u_1(H, H)$. However, if A_1 chooses T, then $u_1(T, H) = 100$ and $u_1(T, T) = -100$. That is, $u_1(T, H) > u_1(T, T)$. So, there is no dominant strategy for A_1.

Given this non-existence, researchers look towards other equilibrium concepts that may provide the desirable properties of dominant strategy equilibrium and may possibly be guaranteed to exist for any instance of Γ. To this end, we next look at *Nash equilibrium*.

2.1.2 Nash Equilibrium

Nash equilibrium is a weaker equilibrium concept proposed by John Nash in 1950 [208], which states that a given strategy profile is Nash equilibrium if each agent following the strategy maximizes its utility, given that all the remaining agents are following their strategy from the same profile. More formally[2],

Definition 2.3: (Pure-strategy) Nash Equilibrium [205, 208]

Given the game Γ, we refer to the strategy profile $s^\star = (s_1^\star, \ldots, s_n^\star)$ as pure-strategy Nash equilibrium if $\forall i \in A$, the strategy $s_i^\star$ satisfies

$$u_i(s_i^\star, s_{-i}^\star) \geq u_i(s_i, s_{-i}^\star), \forall s_i \in S_i \tag{2.2}$$

In other words, it is Nash equilibrium for an agent to play the strategy specified by $s^\star$ if all the remaining agents follow $s^\star$. That is, it is the agent's *best response* as it cannot receive a higher utility by *unilaterally deviating* from $s^\star$.

(Mixed-strategy) Nash Equilibrium Always Exist. Our discussion so far has revolved around pure-strategy equilibrium wherein each agent plays a fixed strategy as defined in $s^\star$. Nash Jr [208] presents the general mixed-strategy Nash equilibrium where agents choose a probability distribution over possible pure strategies. As we have seen so far, agents may put 100% of the probability on one pure strategy. Thus, pure strategies are a subset of mixed strategies.

Notably, John Nash [208, 207] also showed that mixed-strategy Nash equilibria will always exist, unlike for the dominant strategy case.

Matching Pennies. Taking a re-look at Example 2.1, we see that the game has no pure-strategy Nash equilibrium as there is no pair of pure strategies such that neither agent would want to switch if told what the other would do. However, the game indeed has

[2]For completeness, the pure-strategy Nash equilibrium was first proposed by Cournot [66]. In his work, Nash Jr [208] generalized the concept for mixed strategies (agents randomly choose their strategy from a probability distribution).

a mixed-strategy Nash equilibrium wherein each agent chooses head or tail with equal probability, i.e., $\frac{1}{2}$. Such a probability distribution allows each agent to make the other agent indifferent about choosing heads or tails, implying that no agent has an incentive to try another strategy.

2.1.2.1 PPAD-Completeness

Nash [207] showed the existence of Nash equilibrium. However, the question of "how long does it take until economic agents converge to an equilibrium" [83] remained unclear. Daskalakis, Goldberg, and Papadimitriou [83] addressed this question by studying the complexity of computing the mixed Nash equilibrium in a game. Traditional computational problems fall into (i) polynomial-time solvable or (ii) NP-Hard. However, the authors argue that NP-hardness cannot be applied to this problem as we know that every game has a mixed-strategy solution. As such, Daskalakis, Goldberg, and Papadimitriou [83] propose a new class of problems called *PPAD*, a subclass of NP. They showed that computing the mixed NE of a game is PPAD-complete [83, Theorem 3.1].

PPAD-Completeness. While the proof of the above result is outside the scope of this thesis, we present an intuitive understanding of what it means for a problem to be PPAD-complete. PPAD (Polynomial Parity Argument, Directed version) is a complexity class dealing with decision problems related to finding fixed points of certain functions. PPAD completeness is a concept that is used to classify the difficulty of problems within this class. A problem is PPAD-complete if it is in PPAD and is as hard as any problem in PPAD. In other words, if you can efficiently solve a PPAD-complete problem, you can efficiently solve any problem in PPAD.

Example 2.3: PPAD-complete: Example

Consider the "End-of-the-line" (EOL) problem. In EOL, we are given a polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}^n$. The goal is to find an x such that $f(x) = x$ and the function f is such that there is at least one fixed point.

2.1.3 Sub-game Perfect Nash Equilibrium

The examples of Matching pennies and the Prisoner's Dilemma are *simultaneous-move* games. In these games, the agents simultaneously play their strategies. In a *sequential* game, where the agents may observe the actions of the other agents, we require a tweak to the classic Nash equilibrium concept. Instead, we require a strategy profile that is the best response of every agent during the game, i.e., the best response for every *sub-game* induced during it. Such a strategy profile is said to be a *Sub-game Perfect Equilibrium*.

Definition 2.4: Sub-game Perfect Nash Equilibrium [205]

Given the game Γ, we refer to the strategy profile $s^\star = (s_1^\star, \ldots, s_n^\star)$ as Sub-game Perfect Nash Equilibrium if $\forall i \in A$, the strategy $s_i^\star$ satisfies

$$u_i(s_i^\star, s^\star_{-i|H_{t_i}}) \geq u_i(s_i, s^\star_{-i|H_{t_i}}), \forall s_i \in S_i, \forall H_t. \tag{2.3}$$

Here, H_t is the history of the game Γ till time t. The history comprises the agents' information that has arrived till time t. More formally, H_t is the tuple that includes the time remaining before the deadline from t, the target, the refund bonus, and the total contribution made till t. Furthermore, $s^\star_{-i|H_{t_i}}$ implies that agents who arrive after t_i follow the strategy specified in $s^\star_{-i}$. A strategy profile is a Sub-game Perfect Nash Equilibrium if it is Nash equilibrium for each agent to follow the strategy given by the profile irrespective of the events till that time.

2.2 Mechanism Design

In Chapter 2.1, we introduce the concept of game theory and discuss several concepts with the idea of an agent maximizing its utility. However, several scenarios exist where decisions are made considering the interests of a group of people or community (e.g., public decision-making or resource allocation in organizations). In these scenarios, we can consider the people involved as agents. Each scenario comprises a set of possible outcomes. We say that each agent involved has preferences over the set of outcomes.

The social planner (e.g., the head of the organization) is tasked with designing a game among these (conflicting) agents. The planner must construct various rules/incentives to facilitate a desirable outcome in the presence of these rational agents. This "reverse engineering" of game theory is called *Mechanism design*.

2.2.1 Overview

Consider a classic auction setting. Here, we have multiple agents (acting as bidders), each bidding to own a single item being auctioned. Each agent has a private *valuation* for the item. The valuation represents the utility the agent will receive if it acquires the item. The auctioneer (acting as the social planner) wishes to find the optimal outcome, which in this case corresponds to the outcome where the agent with the highest valuation receives the item.

The auction setting described has two major challenges. Firstly, we have *preference elicitation* wherein the social planner must design the auction game to elicit each agent's true valuation. In general, the planner either uses monetary benefits ("mechanisms with money") or enforces certain constraints ("mechanisms without money") to enable *truthful* elicitation. That is, mechanisms that, at equilibrium, incentivize agents to report their true valuations. The second challenge is that of *preference aggregation*, which corresponds to computing the optimal outcome, given each agent's valuations.

With this backdrop, we next discuss the mechanism design environment required to model the auction setting discussed, or in general, any setting, as a game.

2.2.2 The Mechanism Design Environment

Consider the following definition, which provides a general setting for formulating, analyzing, and solving mechanism design problems.

Definition 2.5: The Mechanism Design Environment [205]

A typical mechanism design environment comprises:

1. The set of agents $\mathcal{A} = \{1, \ldots, n\}$. Each agent is assumed to be intelligent and rational.
2. The set of outcomes, or alternatives, $\mathcal{X}$. The agents collectively choose from $\mathcal{X}$.
3. Agents privately observe preferences over the alternatives in $\mathcal{X}$. Formally, each agent i derives a preference by observing a private signal or type θ_i.
4. The set Θ_i denotes the private values of agent i, $\forall i \in \mathcal{A}$. The set of all type profiles is $\Theta = \Theta_1 \times \ldots \times \Theta_n$. A typical type profile is denoted by $\theta = (\theta_1, \ldots, \theta_n)$.
5. The private values of the agents have a common prior distribution $\Phi \in \Delta(\Theta)$.
6. An agent's preference over the outcomes is represented by the utility function $u_i : \mathcal{X} \times \Theta_i \to \mathbb{R}$. That is, given $x \in \mathcal{X}$ and $\theta_i \in \Theta_i$, $u_i(x, \theta_i)$ denotes the utility that agent i having type $\theta_i \in \Theta_i$ receives from its choice $x \in \mathcal{X}$.
7. The set of outcomes $\mathcal{X}$, agents $\mathcal{A}$, types $\Theta_i, \forall i$, the distribution Φ and utility functions $u_i, \forall i$ are assumed to be *common knowledge* among the agents. But, the observed valuation θ_i by agent i is its private information.

To further understand the environment, consider the following example of an auction.

Example 2.4: Mechanism Design Environment for an Auction

Consider a simple auction setting with an auctioneer (agent 0) and two bidders (agents 1 and 2). We have $\mathcal{A} = \{0, 1, 2\}$. Also, $\Theta_0 = \{0\}$ as the auctioneer has no value for keeping the item with itself. Likewise, $\Theta_1 = [0, 1]$ and $\Theta_2 = [0, 1]$. The set of outcomes are $\mathcal{X} = \{(y_0, y_1, y_2, t_0, t_1, t_2) : y_i \in \{0, 1\} \mid \sum_i y_i \leq 1; t_i \in \mathbb{R}, \sum_i t_i \leq 0\}$.

In Example 2.4, $y_i = 1$ implies that agent i receives the item while the agent does not if $y_i = 0$[3]. Example 2.4 also introduces the *transfer variable* $t_i, \forall i$, which for an auction corresponds to the payment for the agent i. One can observe that if $\sum_i t_i > 0$, the mechanism will need money from an external source not part of the mechanism. In general, we have $\sum_i t_i \leq 0$. For the auction-specific case, the agents 1 and 2 pay for the item while agent 0 receives the payment as the auctioneer. In other words, $t_0 = -(t_1 + t_2)$, i.e., $\sum_i t_i = 0$.

Given this construction, we now take an in-depth look at the two challenges in mechanism design: (i) preference aggregation and (ii) preference elicitation.

2.2.3 Preference Aggregation

This is understood as the aggregation of agents' preference rankings of two or more social alternatives into a single, collective preference ranking (or choice) over these alternatives. Here, θ_i's generate preferences over outcomes for player i. To this end, we define a *social choice* function that takes the private valuations as inputs and outputs a collective decision.

[3]We also remark that depending on how the item is being allocated to the agents, the domain of y_i, or the value which y_i takes may also change. If the item is being allocated randomly, we have $y_i \in [0, 1]$. For the deterministic, non-divisible case, $y_i \in \{0, 1\}$.

Definition 2.6: Social Choice Function (SCF) [205]

Given a type profile $\Theta = (\Theta_1, \ldots, \Theta_n)$, an outcome $x \in \mathcal{X}$ is called a *social choice* or *collective choice*, where $f : \Theta \to X$ is called the SCF. That is f assigns to each possible type profile $(\theta_1, \ldots, \theta_n)$ a collective choice from the set of alternative $\mathcal{X}$.

We illustrate the social choice function using a *first-price* auction based on the setting described in Example 2.4.

Example 2.5: SCF for First Price Auction

For the mechanism defined in Example 2.4, let $f(\theta) = (y_0(\theta), y_1(\theta), y_2(\theta), t_0(\theta), t_1(\theta), t_1(\theta), t_2(\theta))$ denote the social choice function for a first-price auction. We assume that in the case of a tie, agent 1 receives the item. We have, $y_0(\theta) = 0, \forall\theta$. Furthermore,

$$y_1(\theta) = \begin{cases} 1 & \theta_1 \geq \theta_2 \\ 0 & otherwise \end{cases}$$

$$y_2(\theta) = \begin{cases} 1 & \theta_2 > \theta_1 \\ 0 & otherwise \end{cases}$$

Also, $t_1(\theta) = -y_1(\theta) \cdot \theta_1$, $t_2(\theta) = -y_2(\theta) \cdot \theta_2$ and $t_0(\theta) = -(t_1(\theta) + t_2(\theta))$.

We can also write the SCF for a *second-price* auction, i.e., an auction where the highest bidder receives the item but pays the second-highest bid. One can observe that only the payments of the agents will change in this case.

Example 2.6: SCF for Second Price Auction

For the mechanism defined in Example 2.4, let $f(\theta) = (y_0(\theta), y_1(\theta), y_2(\theta), t_0(\theta), t_1(\theta), t_1(\theta), t_2(\theta))$ denote the social choice function for a second-price auction. We assume that in the case of a tie, agent 1 receives the item. We have, $y_0(\theta) = 0, \forall\theta$. Furthermore,

$$y_1(\theta) = \begin{cases} 1 & \theta_1 \geq \theta_2 \\ 0 & otherwise \end{cases}$$

$$y_2(\theta) = \begin{cases} 1 & \theta_2 > \theta_1 \\ 0 & otherwise \end{cases}$$

In this case, we now have $t_1(\theta) = -y_1(\theta) \cdot \theta_2$, $t_2(\theta) = -y_2(\theta) \cdot \theta_1$ and $t_0(\theta) = -(t_1(\theta) + t_2(\theta))$.

The second-price auction from Example 2.6 is also called a *Vickrey* auction, after the Novel prize-winning work of Vickrey [284]. Lastly, the process of computing $f(\theta)$ is also referred to as the *preferrence aggregation* problem. The problem usually comes in the form of an optimization problem.'

2.2.3.1 SCF: Important Properties

We now discuss two crucial properties of a SCF, namely, Pareto Optimality and Dictatorship.

2.2.3.1.1 Pareto Optimality Our equilibrium discussion briefly focused on Pareto optimal equilibrium, i.e., no agent can move away from the equilibrium strategy and receive a strictly higher utility while the other agents receive the same utility. Pareto Optimality for a SCF is similar, defined formally as follows.

Definition 2.7: Pareto Optimality or Ex-post Efficiency [205]

A social choice function $f : \Theta \to \mathcal{X}$ is said to be *ex-post efficient (or paretian)* if for every $\theta \in \Theta$, $f(\theta)$ is Pareto optimal. That is $\nexists x \in \mathcal{X}$ such that,

$$u_i(x, \theta) \geq u_i\big(f(\theta), \theta\big) \quad \forall i \in \mathcal{A}$$

with a strict inequality for at least one agent.

Intuitively, it is an economic state where resources (items) are allocated most efficiently, and it is obtained when a distribution strategy exists where one party's situation cannot be improved without worsening another party's situation.

Example. Consider the single-item, multi-unit auction setting where the auctioneer wishes to allocate a single item (available in multiple but fixed units) among n agents. Any allocation for this will always be Pareto Optimal; the only way to make someone better off than before would be to give them more units of the item, implying that at least one other agent gets less than their previous share since the quantity is fixed.

2.2.3.1.2 Dictatorship If a mechanism's outcome reflects a single agent's preferences, we say that it is a dictatorship mechanism. More formally,

Definition 2.8: Dictatorship [205]

A SCF $f : \Theta \to \mathcal{X}$ is dictatorial if there exists an agent $d \in \mathcal{A}$, such that $\forall \theta \in \Theta$,

$$u_d\big(f(\theta), \theta\big) \geq u_d(x, \theta) \quad \forall x \in \mathcal{A}.$$

If a SCF does not contain a dictator, we say it is *non-dictatorial*.

Non-dictatorship is a property of the SCF, which requires that the results of the voting cannot simply mirror that of any single agent's preferences without considering other agents.

Example. The *Dictator Game* [148], is a degenerate game in Game Theory, where the first agent, the dictator, determines how to split an endowment (such as a cash prize) between themselves and the second agent. The second agent, the recipient, simply receives the remainder of the endowment left by the dictator. The recipient's role is entirely passive and has no input into the game's outcome.

2.2.4 Preference Elicitation

Preference Elicitation is how agents are made to state (or "elicit") their preferences. This is also called *information revelation* problem. As we will see, this is achieved either by a *direct mechanism* or an *indirect mechanism*, and involves equilibrium in Bayesian Games, which can be of the type (very weakly) *dominant strategy* equilibrium and *Bayesian Nash* equilibrium.

2.2.4.1 Direct and Indirect Problems

Recall that the agent's valuation is private to itself. To solve the preference aggregation problem, $f(\theta)$, the first step is to find a way by which to get the agent to elicit their type information. Mechanism design literature relies on either *direct* or *indirect* mechanisms for such a truthful elicitation.

Here, we define these mechanisms formally. From Definition 2.5, we know that the set of outcomes $\mathcal{X}$, agents $\mathcal{A}$, types $\Theta_i, \forall i$, the distribution Φ and utility functions $u_i, \forall i$ are common knowledge among the agents. Let us first define a mechanism in this context. The set S_i for each agent $i \in \mathcal{A}$ is the set of actions available. The agent i uses its type θ_i to choose an action $s_i \in S_i$.

Definition 2.9: Mechanism [205]

A mechanism $\mathfrak{M} = ((S_i)_{i \in \mathcal{A}}, g(\cdot))$ is a collection of action sets $(S_1, \ldots, S_n)$ and an outcome function $g : S \to \mathcal{X}$.

Given this definition, the case where the agents are asked to reveal their types becomes its special case, referred to as the *direct revelation mechanism*.

Definition 2.10: Direct Revelation Mechanism (DRM) [205]

Given the SCF $f : \Theta \to \mathcal{X}$, the mechanism $\mathfrak{D} = ((\Theta_i)_{i \in \mathcal{A}, f(\cdot)})$ is referred to as the DRM.

One can immediately observe that the DRM is a special case of the mechanism from Definition 2.9, i.e., $\mathfrak{M} = ((S_i)_{i \in \mathcal{A}}, g(\cdot))$ with $S_i = \Theta_i, \forall i \in \mathcal{A}$ and $g = f$. A mechanism that is not a DRM is called an *indirect* mechanism.

Indirect mechanisms aim to provide agents with a choice of actions and specify an outcome for each action profile. This induces a game among the agents such that the strategies played by the agents in the game's equilibrium will indirectly reflect their original types. More formally, indirect mechanisms induce a Bayesian game.

Definition 2.11: Bayesian Game

We are given the set of agents $\mathcal{A}$, with types $(\Theta_1, \ldots, \Theta_n)$ having a common prior $\phi \sim \Delta(\Theta)$ and a set of outcomes $\mathcal{X}$. If each agent $i \in \mathcal{A}$ has the utility $u_i : X \times \Theta_i \to \mathbb{R}$, then a mechanism $\mathfrak{M} = (S_1, \ldots, S_n, g(.))$ induces the Bayesian game $\Gamma^b : (N, (\Theta_i), (S_i), (p_i), (U_i))$ among the agents with each agent $i \in \mathcal{A}$ utility as $u_i(g(s_1, \ldots, s_n), \theta_i)$.

Examples. An example of a DRM is Borda count, wherein each agent must explicitly give its complete preference order of all possible types. An example of an indirect mechanism is majority voting, where private information related to the actual ordering of candidates by an agent is not required, but the agent is required to play a strategy based on its preferred preference type. Additionally, the sealed-bid first-price or second-price auctions seen earlier are indirect mechanisms.

2.2.4.2 Generalized Vickrey Auction

We wrap up our discussion on preference aggregation and preference elicitation by looking at the famous generalized Vickrey auction.

Example 2.7: Generalized Vickrey Auction (GVA)

We have the set of $\mathcal{A} = \{1, \ldots, n\}$ bidding for the items $\{1, \ldots, m\}$ with M as the power set for $[m]$. We get an indirect mechanism $\mathfrak{M} = ((S_i)_{i \in \mathcal{A}}, g(\cdot)$, where $S_i \subset (\mathbb{R}^+)^{2^m - 1}$ is the set of bids that agent i can submit to the auctioneer. The function $g(\cdot)$ is the outcome rule given by $g(b) = ((y_i^\star(S, b))_{i \in \mathcal{A}, S \subset M}, t_1(b), \ldots, t_n(b))$ where $b = (b_1, \ldots, b_n)$ is the bidding profile with b_i as agent i's bid.

The functions $y_i^\star(\cdot, \cdot)$ are referred to as the *winner determination rules*, and the function $t_i(\cdot)$ are the *payment* rules.

The winner determination problem in Example 2.7 is the solution to the following optimization problem:

$$
\begin{aligned}
\textbf{Maximize} \quad & \sum_{i=1}^{n} \sum_{S \subset M} b_i(S) \cdot y_i(S, b) \\
\textbf{subject to} \quad & \text{(i)} \quad \sum_{S \subset M} y_i(S, b) \leq 1, \forall i \in \mathcal{A} \\
& \text{(ii)} \quad \sum_{S \subset M | j \in S} \sum_{i=1}^{n} y_i(S, b) \leq 1, \forall j \in M \\
& \text{(iii)} \quad y_i(S, b) \in \{0, 1\} \forall i \in \mathcal{A}, S \subset M
\end{aligned}
$$

Furthermore, the payment rule is as follows,

$$
t_i(b) = \sum_{j \neq i} v_j(k^\star(b), b_j) - \sum_{j \neq i} v_j(k_{-1}^\star(b_{-i}), b_j), \tag{2.4}
$$

where $v_j(k^\star(b), b_j) = \sum_{S \subset M} b_j(S) \cdot y_j^\star(S, b)$ is the total value of the bundle which is allocated to the agent j. The term $v_j(k_{-1}^\star(b_{-i}), b_j) = \sum_{S \subset M} b_j(S) \cdot y_j^\star(S, b_{-i})$ is the total value of

the bundle that agent $j \neq i$ will get if agent i is not present in the system. Eq. 2.4 is a useful payment structure (for reasons apparent in Chapter 2.2.5.1) and is often referred to as Vickrey, Clark, Groves (VCG) payments.

The interested reader can also observe that if the set M consists of just one item, then the winner determination rule $y_i^\star$ will be the same as the one presented for the Vickrey auction (Example 2.6). Thus, the name Generalized Vickrey Auction.

2.2.5 Properties of a Mechanism

Our discussion on mechanism design highlights preference revelation (or elicitation) and aggregation as the primary challenges. We saw different types of mechanisms (direct/indirect) that aim to elicit the agent's private information to aggregate them using an SCF. However, we desire that the elicitation is *truthful*. As such, we must ensure that an agent's true revelation is its best response, consistent with our rationality and intelligence assumptions. One popular approach towards this is to offer appropriate incentives. This leads to the notion of *incentive compatibility*, first introduced by Hurwicz [142], offering appropriate incentives so that agents prefer to elicit their private information truthfully.

2.2.5.1 Incentive Compatibility

In Chapter 2.1, we saw two types of equilibrium, namely dominant strategy and Nash. The first states gave a strategy that is an agent's best response irrespective of the others' strategy, whereas the second states that the strategy is an agent's best response only when the others' also choose the same strategy. Analogously, we also have two types of incentive compatibility: firstly, when truthful elicitation is the best response for each agent irrespective of other agents' reports, and secondly, when truthful elicitation is the best response for each agent when the other agents report truthfully. The first definition is referred to as *dominant strategy incentive compatibility* (DSIC) and the second as *Nash*

incentive compatibility (NIC). Note that, as truthful elicitation is always with respect to types, only direct mechanisms are relevant when defining incentive compatibility.

Before formally defining DSIC and NIC, we first define the general notion of incentive compatibility.

Definition 2.12: Incentive Compatibility [205]

A SCF $f : \Theta_1 \times \ldots \times \Theta_n \to \mathcal{X}$ is incentive compatible (or truthfully implementable) if the Bayesian game induced by the DRM $\mathfrak{D} = ((\Theta_i)_{i\in\mathcal{A}}, f(\cdot))$ has a pure strategy equilibrium $s^*(\cdot) = (s_1^*(\cdot), \ldots, s_n^*(\cdot))$ in which $s_i^*(\theta_i) = \theta_i, \forall\theta_i \in \Theta_i, \forall i \in \mathcal{A}$.

Informally, Definition 2.12 states that truth revelation by each agent comprises an equilibrium of the game induced by $\mathfrak{D}$. One can observe that if an SCF $f(\cdot)$ is incentive compatible then the DRM $\mathfrak{D} = ((\Theta_i)_{i\in\mathcal{A}}, f(\cdot))$ can implement it. In other words, directly asking the agents to report their types and using this information in $f(\cdot)$ to get the social outcome will solve both our mechanism design challenges, namely, preference elicitation and preference aggregation.

2.2.5.1.1 Dominant Strategy Incentive Compatibility (DSIC) We say that the game induced by a DRM $\mathfrak{D}$ is DSIC if truth revelation by each agent constitutes a dominant strategy equilibrium of $\mathfrak{D}$. More formally,

Definition 2.13: DSIC [205]

A SCF $f : \Theta_1 \times \ldots \times \Theta_n \to \mathcal{X}$ is dominant strategy incentive compatible (or truthfully implementable in dominant strategies) if the DRM $\mathfrak{D} = ((\Theta_i)_{i\in\mathcal{A}}, f(\cdot))$ has a *weakly dominant strategy equilibrium* $s^*(\cdot) = (s_1^*(\cdot), \ldots, s_n^*(\cdot))$ in which $s_i^*(\theta_i) = \theta_i, \forall\theta_i \in \Theta_i, \forall i \in \mathcal{A}$.

We said earlier that Vickrey auctions (Example 2.6) are of particular interest. This is because these are DSIC, as shown next.

Claim 2.1. *In a second-price auction (Example 2.6), every agent $i \in \mathcal{A}$ has a dominant strategy: set its bid b_i equal to its private valuation θ_i. That is, this strategy maximizes the utility of the agent i, no matter what the other agents bid.*

Proof. Choose an arbitrary agent, say agent $i \in \mathcal{A}$. Agent i has valuation θ_i, and it knows the other agents' bids b_{-i}. To prove the claim, we must show that agent i's utility is maximized if $b_i = \theta_i$. Let $B = \max_{j \not= i} b_j$ be the highest bid among all agents except i. Since this is a second-price auction, if $b_i < B$, then i is not the winner, and its utility is 0. If $b_i \geq B$[4], then i wins and pays the value B such that its utility is $\theta_i - B$.

Consider the following two cases. In Case 1, we have $\theta_i < B$. The highest utility i can get is $\max\{0, \theta_i - B\} = 0$, and it achieves this by bidding truthfully. In Case 2, we have $\theta_i \geq B$. The highest utility i can now get is $\max\{0, \theta_i - B\} = \theta_i - B$. It achieves this utility by bidding truthfully. □

2.2.5.1.2 Nash Incentive Compatibility

We say that the game induced by a DRM $\mathfrak{D}$ is NIC if truth revelation by each agent constitutes a Nash equilibrium of $\mathfrak{D}$. More formally,

Definition 2.14: NIC [205]

A SCF $f : \Theta_1 \times \ldots \times \Theta_n \to X$ is Nash incentive compatible (or truthfully implementable in Nash equilibrium) if the DRM $\mathfrak{D} = ((\Theta_i)_{i \in N}, f(\cdot))$ has a *Nash equilibrium* $s^*(\cdot) = (s_1^*(\cdot), \ldots, s_n^*(\cdot))$ in which $s_i^*(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i, \forall i \in \mathcal{A}$.

2.2.5.2 Individual Rationality

The incentive compatibility property allows a social planner to resolve the primary challenges while designing a mechanism. In contrast, individual rationality is a property of

[4]For completeness, note that the assumption is that the tie gets broken in agent i's favor. However, the claim is independent of how the tie gets broken.

the participating agents, which ensures that the agents are not worse off by participating in the mechanism. More concretely, it ensures non-negative utility for the agents. The precise definition of individual rationality depends on the precise stage of the mechanism, either (i) post every agent's participation, (ii) post the agent's participation, and (iii) before any agent's participation. For the definitions, let $\overline{u_i}(\theta_i)$ be the utility that agent i receives by withdrawing from the mechanism, given θ_i as its type.

Definition 2.15: Ex-Post Individual Rationality (Ex-post IR) [205]

When the agent knows the types of all the agents. Then, the SCF f satisfies ex-post participation (or individual rationality) constraints when,

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geq \overline{u_i}(\theta_i), \ \forall \ (\theta_i, \theta_{-i}) \in \Theta.$$

Definition 2.16: Interim Individual Rationality (Interim IR) [205]

For this, each agent $i \in \mathcal{A}$ only knows its own type θ_i. Then, the SCF f satisfies interim participation (or individual rationality) constraints when,

$$\mathbb{E}_{\theta_{-i}}[u_i(f(\theta_i, \theta_{-i}), \theta_i)|\theta_i] \geq \overline{u_i}(\theta_i), \ \forall \ (\theta_i) \in \Theta_i.$$

Definition 2.17: Ex-Ante Individual Rationality (Ex-Ante IR) [205]

Each agent $i \in \mathcal{A}$ is unaware of its own type. Then, the SCF f satisfies ex-ante participation (or individual rationality) constraints when,

$$\mathbb{E}_{\theta}[u_i(f(\theta_i, \theta_{-i}), \theta_i)] \geq \mathbb{E}_{\theta_i}[\overline{u_i}(\theta_i)].$$

The three variants of IR are related as follows.

Proposition 2.1. *Given a SCF $f : \Theta \to \mathcal{X}$, we have the relation:*

$$f(\cdot) \textit{ is ex-post IR} \implies f(\cdot) \textit{ is interim IR} \implies f(\cdot) \textit{ is ex-ante IR}$$

We see from Proposition 2.1 that ex-post IR is the strongest property among the three IR properties. That is, ensuring ex-post IR ensures interim and ex-ante IR.

The wonderful Vickrey auction (Example 2.6) is not only DSIC but also ex-post IR, as shown next.

Claim 2.2. *In a second-price auction (Example 2.6), every truthful agent is guaranteed non-negative utility.*

Proof. By construction, all losing agents receive utility 0. If an agent i wins, its utility is $\theta_i - p$, where p is the second-highest bid. Since i won and is truthful (that is, it bid $b_i = \theta_i$) and p is the second-highest bid, we have $p \leq \theta_i \implies \theta_i - p \geq 0$. □

2.2.5.3 Gibbard-Satterwaite Impossibility Theorem

Intuitively, DSIC is a desirable property for an SCF. After all, DSIC will ensure that it is beneficial for an agent to report its type truthfully, irrespective of the strategy employed by the other agents. Unfortunately, the desirability of DSIC often comes at the cost of certain other desirable properties.

More concretely, the Gibbard-Satterwaite (G-S) Impossibility Theorem, proved independently by Gibbard [117] and Satterthwaite [250], shows that the DSIC property will force an SCF to be dictatorial if the utility environment is an unrestricted one[5].

Preference Relations [205]. Before formally presenting the theorem, we first step the following notations. Recall that for an agent $i \in \mathcal{A}$, its preference over the set $\mathcal{X}$ is described by a utility function $u_i : \mathcal{X} \times \Theta_i \to \mathbb{R}$. That is, for every possible type $\theta_i \in \Theta_i$, we can define a utility function $u_i(\cdot, \theta_i)$ over the set $\mathcal{X}$. Let this utility function induce a unique preference relation $\succsim_i (\theta_i)$ over $\mathcal{X}$. For instance,

$$x \succsim y \iff u_i(x, \theta_i) \geq u_i(y, \theta_i).$$

[5]The GS theorem is a brilliant reinterpretation of the famous Arrow's impossibility theorem [13].

The above preference relation is often called a *rational* preference relation. More concretely, the relation $\succsim$ is rational if it posses: (i) *Reflexivity*: $\forall x \in \mathcal{X}$, $x \succsim x$, (ii) *Completeness*: $\forall x, y \in \mathcal{X}$, $x \succsim y$ or $y \succsim x$ or both, and (iii) *Transitivity*: $\forall x, y, z \in \mathcal{X}$, $x \succsim y$ and $y \succsim z$, then $x \succsim z$.

1. **Strict-total Preference Relation.** The relation $\succsim$ is strict-total if it possesses Reflexivity, Completeness, and Transitivity. Moreover, it also satisfies the antisymmetry property, i.e., for any $\forall x, y \in \mathcal{X}$ such that $x \neq y$; we have either $x \succsim y$ or $y \succsim x$ but not both.

   The curious reader may notice that strict total satisfies the property of a "greater than or equal to" relationship on the real line. As such, it is also referred to as the *linear order relation*. Denote $\mathfrak{R}$ as all rational preference relations and $\mathfrak{P}$ as the strict-total preference relations on $\mathcal{X}$. Trivially, $\mathfrak{P} \subset \mathfrak{R}$.

2. **Ordinal Preference Relation.** We have, for every possible type $\theta_i \in \Theta_i, \forall i \in \mathcal{A}$, defined a utility function $u_i(\cdot, \theta_i)$ over the set $\mathcal{X}$. Let this utility function induce a unique preference relation $\succsim_i (\theta_i)$ over $\mathcal{X}$. The set $\mathfrak{R}_i = \{\succsim : \succsim = \succsim_i (\theta_i)$ for some $\theta_i \in \Theta_i\}$ is known as the ordinal preference relations for agent i. Trivially, $\mathfrak{R}_i \subset \mathfrak{R}, \forall i \in \mathcal{A}$.

With all the groundwork in place, we now formally present the G-S theorem next.

Theorem 2.1 (Gibbard-Satterwaite (G-S) Impossibility Theorem [117, 250])**.** *Given a SCF $f : \Theta \to \mathcal{X}$, if*

1. *(Condition 1). The outcome set $\mathcal{X}$ is such that $3 \leq |\mathcal{X}| < \infty$*
2. *(Condition 2). $\mathfrak{R}_i = \mathfrak{P}, \forall i \in \mathcal{A}$*
3. *(Condition 3). The SCF $f(\cdot)$ is onto, $f(\Theta) = \mathcal{X}$, that is, the image of SCF $f(\cdot)$ is the set $\mathcal{X}$.*

Then, the SCF $f(\cdot)$ is truthfully implementable in dominated strategies (DSIC) if and only if it is dictatorial.

We refer the reader to [187, Proposition 23.C.3] for the proof of the theorem. We summarize some immediate implications of G-S theorem next.

1. Condition (1) asserts that $|\mathcal{X}| = 3$. $|\mathcal{X}| = 1$ corresponds to a trivial solution. When $|\mathcal{X}| = 2$, it is of more interest but still restricted. This may correspond to the crowdfunding of public projects (e.g., a public park) where the decision is yes/no (either construct the park or not).

2. Condition (2) asserts that $\mathfrak{R}_i = \mathfrak{P}, \forall i \in \mathcal{A}$. That is, an agent's preferences cover the entire space of strict-total preference relations on $\mathcal{X}$. Suppose, if $|X| = 4$, then the total number of preference ordering possible would equal 4!, i.e., the number of distinct permutations. Then, each agent i maintains $|R_i| = 24$ preferences.

3. Condition (3) along with f being monotonic, implies that f is an onto function; that is, for every $x \in X$ there is a preference profile $[\succsim]$ such that $f([\succsim]) = x$.

To construct DSIC mechanisms, that is, avoid the G-S impossibility, we must relax at least one of the three conditions outlined in Theorem 2.1. Here, we see that Condition (2) enforces that each agent's preferences span the entire space. Restricting these preferences can violate Condition (2) but still ensure that agents' preferences are sufficiently covered. For this, we move to *quasilinear settings*, where monetary payments are added to valuation functions, which enables us to change the order of preferences.

2.2.5.4 Quasi-linear Environment

As mentioned above, the quasi-linear setting allows us to violate the conditions for the G-S theorem and, in turn, allows the social planner to construct useful mechanisms. In

this environment, we overload the alternative/outcome $x \in \mathcal{X}$ with a vector of the form $x = (k, t_1, \ldots, t_n)$. Here, $k \in \mathcal{K}$ where $\mathcal{K}$ is the set of project choices or *allocations* and is assumed to be finite, in general. As before, $t_i \in \mathbb{R}, \forall i$ corresponds to the monetary transfer to agent i. If $t_i > 0$, agent i receives the money and pays the money if $t_i < 0$. The general setup assumes that there is no external funding source, i.e., $\sum_{i \in \mathcal{A}} t_i \leq 0$. To summarize, the alternative $\mathcal{X}$ is,

$$\mathcal{X} = \left\{ (k, t_1, \ldots, t_n) : k \in \mathcal{K};\ t_i \in \mathbb{R},\ \ \forall\, i \in \mathcal{A};\ \ \sum_{i \in \mathcal{A}} t_i \leq 0 \right\}.$$

SCF in Quasi-linear Setting. The social choice function (SCF) in this setting takes the form $f(\theta) = (k(\theta), t_1(\theta), \ldots, t_n(\theta))$ where, for every $\theta \in \Theta$, we have $k(\theta) \in \mathcal{K}$ and $\sum_i t_i(\theta) \leq 0$[6].

Agent's Utility in Quasi-linear Setting. Given a direct revelation mechanism (DRM) $\mathfrak{D} = ((\Theta_i)_{i \in \mathcal{A}}, f(\cdot))$, an agent i's utility function takes the following quasi-linear form:

$$u_i(x, \theta_i) = u_i((k, t_1, \ldots, t_n), \theta_i) = v_i(k, \theta_i) + m_i + t_i.$$

Here, m_i denotes agent i's initial endowment of the money, and the function $v_i(\cdot, \theta_i)$ is i's *valuation* function.

From our definition of the mechanism design environment (Definition 2.5), we know that the utility functions are $u_i(\cdot)$ are common knowledge. In the quasi-linear environment, this implies that each agent $j \not= i$ (and the social planner) has knowledge about $v_i(\cdot, \theta_i)$. Furthermore, in many scenarios, for a DRM $\mathfrak{D} = ((\Theta_i)_{i \in \mathcal{A}}, f(\cdot))$, the set Θ_i is actually the set of feasible $v_i(\cdot, \theta_i)$ for agent i. Each possible function represents the possible types of agent i, implying that reporting a type is equivalent to reporting a valuation function.

While we formally introduce the quasi-linear setting here, we remark that typical examples of such mechanisms were previously discussed, including the first-price (Example 2.5), second-price (Example 2.6), and the GVA (Example 2.7).

[6] To remain consistent with the literature, we use the notation k both as an element of $\mathcal{K}$ and as the function $k : \Theta \to \mathcal{K}$. The underlying context will make clear the role of k.

In the quasi-linear environment, we also have two important properties of an SCF, namely, allocation efficiency and strong budget balance.

2.2.5.4.1 Allocative Efficiency The allocative efficiency of an SCF implies that the allocation $k(\theta)$ is such that it maximizes the sum of the values of all the agents. In other words, the items are allocated to the agents who value them the most.

Definition 2.18: Allocative Efficiency (AE) [205]

We say that an SCF $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is allocatively efficient if for each $\theta \in \Theta$, $k(\theta)$ satisfies the following condition:

$$k(\theta) \in \underset{k \in \mathcal{K}}{\arg\max} \sum_{i=1}^{n} v_i(k, \theta_i). \tag{2.5}$$

Equivalently,

$$\sum_{i=1}^{n} v_i(k(\theta), \theta_i) = \max_{k \in \mathcal{K}} \sum_{i=1}^{n} v_i(k, \theta_i).$$

Definition 2.18 implicitly assumes that given any θ, $\sum_{i=1}^{n} v_i(., \theta_i) : \mathcal{K} \to \mathbb{R}$ attains a maximum over the set $\mathcal{K}$. Moving forward, we will use $k^*(\cdot)$ for a function $k(\cdot)$ that satisfies Eq. (2.5). As the definition shows, AE is a desirable property for any SCF.

2.2.5.4.2 Strong Budget Balance Consider the following definition.

Definition 2.19: Budget Balance [205]

We say that an SCF $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is budget balanced, if for each $\theta \in \Theta$, $t_1(\theta), \ldots, t_n(\theta)$ satisfy the following condition:

$$\sum_{i=1}^{n} t_i(\theta) = 0. \tag{2.6}$$

Definition 2.19 is also commonly referred to as *strong* budget balance with *weak* budget balance used for the condition $\sum_{i=1}^{n} t_i(\theta) \leq 0$. For the rest of the thesis, with budget balance, we refer to the strong variant.

The budget balance property ensures that the total receipts equal the payments made, implying a "closed" system with no surplus or deficit. Likewise, a weak budget balance implies that the total payments are either greater or equal to the total receipts.

AE + BB $\implies$ (ex-post) Efficient. We now show an SCF is ex-post efficient (Definition 2.7) if and only if it is AE and BB.

Lemma 2.1. *An SCF $f(\cdot) = (k(\cdot), t_1(\cdot), \ldots, t_n(\cdot))$ is ex-post efficient in the quasi-linear environment if and only if it is allocative efficient and budget balance.*

Proof. (SKETCH.) The proof consists of three parts: (i) showing that AE + BB $\implies$ (ex-post) efficiency, (ii) not AE $\implies$ not (ex-post) efficient, and (iii) not BB $\implies$ not (ex-post) efficient. Each part can be individually shown with algebraic manipulations. We refer the reader to [205] for the formal proof. □

All SCFs are non-dictatorial. This result connects nicely to the G-S theorem in that the quasi-linear setting implies that SCFs are non-dictatorial. In light of this, the social planner can look for SCFs that are ex-post-efficient and DSIC. Or perhaps, using Lemma 2.1, look for SCFs that are AE, BB and DSIC.

Lemma 2.2. *All social choice functions in quasi-linear environments are non-dictatorial.*

Proof. If possible, assume that an SCF $f(\cdot)$ is dictatorial in the quasi-linear environment. That is, there exists an agent called the dictator, say $d \in \mathcal{A}$, such that for each $\theta \in \Theta$, we have

$$u_d(f(\theta), \theta_d) \geq u_d(x, \theta_d) \; \forall \; x \in \mathcal{X}.$$

Recall the utility of an agent in the quasi-linear environment. We have, $u_d(f(\theta), \theta_d) = v_d(k(\theta), \theta_d) + t_d(\theta)$. We define the following alternative $x \in \mathcal{X}$:

$$x = \begin{cases} (k(\theta), (t_i = t_i(\theta))_{i \neq d}, t_d = t_d(\theta) - \sum_{i=1}^{n} t_i(\theta)) & : \quad \sum_{i=1}^{n} t_i(\theta) < 0 \\ (k(\theta), (t_i = t_i(\theta))_{i \neq d,j}, t_d = t_d(\theta) + \epsilon, t_j = t_j(\theta) - \epsilon) & : \quad \sum_{i=1}^{n} t_i(\theta) = 0 \end{cases}$$

where $\epsilon > 0$ is any arbitrary number, and j is an agent such that $j \neq d$. One can verify that for this outcome x, we have $u_d(x, \theta_d) > u_d(f(\theta), \theta_d)$, which contradicts the fact that d is a dictator. □

In summary, Chapter 2.2.5.4 shows that for a desirable mechanism, the social planner must look for an SCF that satisfies AE, BB, and DSIC. Next, we investigate the existence of such mechanisms.

2.2.5.5 Auction Theory

With Chapter 2.2.5.5, we introduce the famous VCG mechanism, which shows the existence of a social choice function (SCF) that is both allocative efficient (AE) and DSIC in the quasi-linear setting.

2.2.5.5.1 VCG Mechanisms VCG mechanism is a landmark result in mechanism design and is named after its inventors, William Vickrey, Edward Clarke, and Theodore Groves. As mentioned before, Vickrey [284] proposed the famous second-price or Vickrey auction (Example 2.6). Clarke [63] and Groves [129] generalized the mechanism to present a broad class of DSIC mechanisms for the quasi-linear environment (recall that we have already seen the extension for auctions with Example 2.7). VCG mechanisms are the most extensively studied among the set of quasi-linear mechanisms. Their popularity stems from the strong properties they satisfy and their mathematical elegance. Let us now take a detailed look at this class of mechanisms.

Groves' Theorem [129]. We now discuss a sufficient condition for an allocative efficient SCF to be DSIC in the quasi-linear environment.

Theorem 2.2 (Groves' Theorem [129]). *Consider an SCF $f(\theta) = (k(\theta), t_1(\theta), \ldots, t_n(\theta))$ that is allocative efficient (AE). Then $f(\cdot)$ is also DSIC if it satisfies the following payment structure*[a]

$$t_i(\theta) = \left[\sum_{j \neq i} v_j(k^*(\theta), \theta_j)\right] + h_i(\theta_{-i}) \;\; \forall\, i = 1, \ldots, n \tag{2.7}$$

Here, $h_i : \Theta_{-i} \to \mathbb{R}$ is an arbitrary function that honors the feasibility condition $\sum_i t_i(\theta) \leq 0 \; \forall \; \theta \in \Theta$.

[a] In mechanism design literature, Eq. 2.7 is also referred to as the Groves payment/incentive scheme.

Proof. The proof uses proof by contradiction. We assume that $f(\cdot)$ is AE and satisfies Eq. 2.7 but is not DSIC. That is, $f(\cdot)$ does not satisfy the following necessary and sufficient condition for DSIC: $\forall i \in \mathcal{A}, \;\; \forall \theta \in \Theta$,

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geq u_i(f(\theta_i^{'}, \theta_{-i}), \theta_i) \; \forall \theta_i^{'} \in \Theta_i, \forall \theta_{-i} \in \Theta_{-i}.$$

From the above, for agent i, we have,

$$u_i(f(\theta_i^{'}, \theta_{-i}), \theta_i) > u_i(f(\theta_i, \theta_{-i}), \theta_i)$$

for some $\theta_i \in \Theta_i$, for some $\theta_{-i} \in \Theta_{-i}$, and for some $\theta_i^{'} \in \Theta_i$. For agent i, there exists $\theta_i \in \Theta_i, \theta_i^{'} \in \Theta_i, \theta_{-i} \in \Theta_{-i}$ such that

$$v_i(k^*(\theta_i^{'}, \theta_{-i}), \theta_i) + t_i(\theta_i^{'}, \theta_{-i}) + m_i > v_i(k^*(\theta_i, \theta_{-i}), \theta_i) + t_i(\theta_i, \theta_{-i}) + m_i.$$

Recall that

$$t_i(\theta_i, \theta_{-i}) = h_i(\theta_{-i}) + \sum_{j \neq i}(k^*(\theta_i, \theta_{-i}), \theta_j)$$
$$t_i(\theta_i^{'}, \theta_{-i}) = h_i(\theta_{-i}) + \sum_{j \neq i}(k^*(\theta_i^{'}, \theta_{-i}), \theta_j).$$

Substituting these, we get

$$v_i(k^*(\theta_i^{'}, \theta_{-i}), \theta_i) + \sum_{j \neq i} v_i(k^*(\theta_i^{'}, \theta_{-i}), \theta_j) > v_i(k^*(\theta_i, \theta_{-i}), \theta_i) + \sum_{j \neq i} v_i(k^*(\theta_i, \theta_{-i}), \theta_j),$$

which implies

$$\sum_{i=1}^{n} v_i(k^*(\theta_i^{'}, \theta_{-i}), \theta_i) > \sum_{i=1}^{n} v_i(k^*(\theta_i, \theta_{-i}), \theta_i).$$

We see that the above inequality contradicts the assumption that $f(\cdot)$ is AE, completing the proof. □

We can use the Groves payment structure from Eq. 2.7 to design a direct revelation mechanism (DRM) whose SCF is AE and satisfies Eq. 2.7. We refer to such a DRM as the Groves mechanism.

Definition 2.20: Groves Mechanism [205]

A DRM $\mathfrak{D} = ((\Theta_i)_{i \in N}, f(\cdot))$ wherein the SCF $f(\theta) = (k(\theta), t_1(\theta), \ldots, t_n(\theta))$ satisfies AE and Eq. 2.7 is known as the Groves mechanism.

The Groves mechanism is also popularly referred to as the Vickrey, Clarke, Groves (VCG) mechanism.

Eq. 2.7 is also Necessary. Theorem 2.2 presents a sufficient condition under which an AE SCF is also DSIC. Green et al. [123] provides a set of conditions under which the Groves payment structure is also necessary for an AE SCF to be DSIC.

Theorem 2.3 (Green-Laffont's First Characterization Theorem [123])**.** *Let $\mathfrak{F}$ denote the set of all possible functions $f : K \to \mathbb{R}$. Assume that for each agent $i \in \mathcal{A}$, we have $\{v_i(\cdot, \theta_i) : \theta_i \in \Theta_i\} = \mathfrak{F}$, i.e., every possible valuation function from K to $\mathbb{R}$ arises for some $\theta_i \in \Theta_i$. Then, any SCF $f(\cdot)$ will be DSIC if and only if it satisfies Eq. 2.7.*

In Theorem 2.3, every possible valuation function from K to $\mathbb{R}$ exists for any $\theta_i \in \Theta_i$. However, depending upon $\mathfrak{K}$'s structure, it is quite possible that for some type profile $\theta = (\theta_1, \ldots, \theta_n)$, the maximum of the function $\sum_{i=1}^{n} v_i(\cdot, \theta_i)$ over the set $\mathfrak{K}$ may not exist. In such cases, the set of AE SCFs is empty. One way to overcome this is to assume that the set $\mathfrak{K}$ is finite. Another is to restrict the allowable valuation functions to the class

of continuous functions. Our next characterization, again from Green et al. [123], $\mathfrak{F}$ is replaced with $\mathfrak{F}_c$ where $\mathfrak{F}_c$ denotes the set of all possible continuous functions $f : K \to \mathbb{R}$.

Theorem 2.4 (Green-Laffont's First Characterization Theorem [123]). *Let $\mathfrak{F}_c$ denote the set of all possible but continuous functions $f : K \to \mathbb{R}$. Assume that for each agent $i \in \mathcal{A}$, we have $\{v_i(\cdot, \theta_i) : \theta_i \in \Theta_i\} = \mathfrak{F}_c$, i.e., every possible valuation function from K to $\mathbb{R}$ arises for some $\theta_i \in \Theta_i$. Then, any SCF $f(\cdot)$ will be DSIC if and only if it satisfies Eq. 2.7.*

2.2.5.5.2 Clarke (Pivotal) Mechanism. Clarke [63] independently proposed a special case of Groves mechanism, referred to as *Clarke*, or the *pivotal* mechanism. It is a special case as it uses a natural form of $h_i(\cdot)$ (refer to Theorem 2.2). In Clarke's mechanism, we have,

$$h_i(\theta_{-i}) = -\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j) \;\; \forall\, \theta_{-i} \in \Theta_{-i}, \forall\, i = 1, \ldots, n \tag{2.8}$$

Here, $k^*_{-i}(\theta_{-i}) \in K_{-i}$ is the choice of an allocation (e.g., item) that is AE if there were only the $n-1$ agents $j \neq i$. Formally, $k^*_{-i}(\theta_{-i})$ must satisfy the following condition:

$$\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j) \geq \sum_{j \neq i} v_j(k, \theta_j) \;\forall\, k \in K_{-i} \tag{2.9}$$

where the set K_{-i} is the set of allocation (e.g., items distributed) choices available when agent i is absent. Substituting the value of $h_i(\cdot)$ from Eq. 2.8 in Eq. 2.7, we get the following expression for agent i's transfer in Clarke's mechanism,

$$t_i(\theta) = \left[\sum_{j \neq i} v_j(k^*(\theta), \theta_j)\right] - \left[\sum_{j \neq i} v_j(k^*_{-i}(\theta_{-i}), \theta_j)\right]. \tag{2.10}$$

A careful reader will observe that these transfers are identical to the payments proposed for the Generalized Vickery Auction (GVA) in Eq. 2.4. Indeed, this payment rule is appealing. Given a type profile $\theta = (\theta_1, \ldots, \theta_n)$, the monetary transfer to agent i in Eq. 2.10

is given by the total value of all agents other than i under an efficient allocation when agent i is present in the system minus the total value of all agents other than i under an efficient allocation when agent i is absent in the system.

We summarize this discussion with this: Groves mechanisms are referred to as VCG mechanisms, as the Clarke mechanism is a special case of the Groves mechanism, and the Vickrey mechanism is a special case of the Clarke mechanism.

2.2.5.5.3 Impossibility of an SCF Satisfying DSIC and Ex-post-Efficiency. Recall that our original goal was to construct a DSIC and ex-post-efficient SCF (the allocations are Pareto-optimal). Equivalently, from Lemma 2.1, we want an SCF that is AE, BB, and DSIC. We were off to a good start with the Groves mechanism being AE and DSIC. If the Groves mechanism is also budget-balanced (BB), we arrive at an ideal SCF.

To this end, we re-look at the $h_i(\cdot)$ functions introduced with Groves' theorem. Unfortunately, Green et al. [123] show that in the quasi-linear environment, if the set of possible agent types is sufficiently rich, then we cannot have an SCF that is AE, BB, and DSIC. More formally,

Theorem 2.5 (Green-Laffont Impossibility Theorem [123])**.** *Suppose for each agent $i \in \mathcal{A}$ we have $\mathfrak{F} = \{v_i(\cdot, \theta_i) : \theta_i \in \Theta_i\}$, i.e., every possible valuation function from K to $\mathbb{R}$ arises for some $\theta_i \in \Theta_i$. Then, there exists no SCF that is ex-post efficient and DSIC.*

In other words, the Green-Laffont Impossibility Theorem states that if the agent types are sufficiently rich, then we cannot find a way to define $h_i(\cdot)$'s in Eq. 2.7 such that $\sum_{i=1}^{n} t_i(\theta) = 0$.

2.2.5.6 Combinatorial Auctions

The Generalized Vickrey Auction from Example 2.7 is actually the Clarke mechanism applied to a *combinatorial* auction. Combinatorial auctions are simply auctions where the

Agents	$\{A\}$	$\{B\}$	$\{A, B\}$
1	-	-	10
2	5	-	-
3	-	5	-

Table 2.3: Agent Valuations [205]

bids correspond to *bundles* or combinations of different items. This thesis focuses on the "forward" combinatorial auction[7], where a seller auctions different types of goods/items and buyers are interested in purchasing certain subsets of these items. We refer the reader to [68] and [211, Chapter 11] for a comprehensive discussion on combinatorial auctions.

2.2.5.6.1 VCG Mechanisms: GVA Re-visited. For illustration, from [205], consider a seller who auctions two items, A and B. We have three interested agents $\{1, 2, 3\}$. The subsets (or bundles) available to buyers are $\{A\}$, $\{B\}$ and $\{A, B\}$. Table 2.3 presents the agent valuations for these bundles, with "-" denoting the fact that an agent is not interested in that bundle.

Applying Clarke's mechanism here, firstly, we know that agents will bid their valuation since the mechanism is DSIC. Next, we have two allocative efficient (AE) allocations: (i) allocate $\{A, B\}$ to agent 1 and (ii) allocate bundle $\{A\}$ to agent 2 and bundle $\{B\}$ to agent 3. Both the allocations give a value of 10. Suppose the seller chose the second allocation. Now, payments to agents 2 and 3 are computed using the Clarke payment rule (Eq. 2.10).

For the payments, let us see what happens when agents 2 and 3 are removed separately from the mechanism. If agent 2 were not present, the seller would give agent 1 the bundle $\{A, B\}$, resulting in a total value of 10. The Vickrey discount to agent 2 is $10 - 10 = 0$, resulting in agent 2's payment of $5 + 0 = 5$. Likewise, for the agent 3 also, the payment is

[7] A reverse combinatorial auction is one in which a buyer requires different types of goods, and several sellers are interested in selling different subsets to it.

5. This results in a net revenue of 10 for the seller (i.e., it has captured the entire consumer surplus).

However, the VCG mechanism for the combinatorial auction need not always generate the optimal revenue. Replacing the valuation of 5 to 10 in Table 2.3 for agents 2 and 3 will lead to zero revenue for the seller. That is, guaranteeing a mechanism that is AE and DSIC came at a cost to the seller.

2.2.5.6.2 Combinatorial Auctions: Standard Model For ease of discussion, moving forward in Chapter 2.2.5.6, we use the following definition to denote a combinatorial auction.

Definition 2.21: Combinatorial Auction [211]

Model: We have a seller/auctioneer AU interested in selling $\mathcal{M} = \{1, \ldots, m\}$ indivisible items. There are $\mathcal{A} = \{1, \ldots, n\}$ interested, and rational, buyers. For each $i \in \mathcal{A}$, the valuation function ϑ_i describes its preferences. That is, for each possible subset $S_i \in 2^{\mathcal{M}}$, ϑ_i is a real-valued function such that $\vartheta_i(S_i)$ denotes agent i's valuation if it gets the bundle S_i. Moreover, given the allocations $S = (S_1, \ldots, S_n)$, let $p_i(S)$ denote the payment for an agent i if it gets the bundle S_i. Each agent $i \in \mathcal{A}$ quasi-linear utility is: $u_i(S, \{\vartheta_i\}_{i \in \mathcal{A}}) = \vartheta_i(S_i) - p_i(S, \{\vartheta_i\}_{i \in \mathcal{A}})$.

Allocation Rule: As before, let K denote the allocation function, which takes the valuations as inputs and outputs the list of bundles allocated to each agent.

Payment Rule: Let $p = (p_1(\cdot), \ldots, p_n(\cdot)$ denote the payments for the agents in $\mathcal{A}$. That is, the combinatorial auction is characterized by the tuple (K, p).

Given Definition 2.21, we can define two useful properties:

- **Feasible Allocation.** An allocation $S = (S_1, \ldots, S_n)$ is feasible if $S_i \cap S_j = \emptyset$ for every $i \neq j$ in $\mathcal{A}$.

- **Social Welfare.** We define the social welfare of an allocation S as $\sum_{i \in \mathcal{A}} \vartheta_i(S_i)$. A socially efficient allocation maximizes the social welfare among all allocations.

Computational Complexity. Unfortunately, computing allocation and payments for a combinatorial auction is computationally hard. In fact, the hardness remains for even simple special cases. We study one such simple case next.

2.2.5.6.3 The Single-minded Case The single-minded case is a restriction of combinatorial auctions wherein agents are interested in a single specific bundle of items and get a scalar value if they get this whole bundle (or any super-set) and get zero value for any other bundle. Formally,

Definition 2.22: Single-minded Valuation Function [211]

A single-minded valuation function is a function in which there exists a bundle of items $S^\star$ and a value $\vartheta^\star \in \mathbb{R}^+$ such that $\vartheta(S) = \vartheta^\star$, $\forall S \supseteq S^\star$ and $\vartheta(S) = 0$ for all other S. A single-minded bid is the pair $(S^\star, \vartheta^\star)$.

As obvious, it is easy to represent single-minded valuations since agents are required only to hold a single parameter for valuation and a single bundle of information. This is unlike the general combinatorial auction case where the bidders must maintain their valuation functions, which are exponential in the number of items m. Despite their simplicity in terms of representation, the single-minded case is also computationally hard.

Computational Complexity of Allocation. From Definition 2.21, the general allocation problem gives disjoint sets of items S_i to each $i \in \mathcal{A}$. In the case of single-minded bidders whose bids are $(S_i^\star, \vartheta_i^\star)$, an allocation to agent i is either the bundle it wants $(S_i = S_i^\star)$ or nothing at all $(S_i = \emptyset)$. More formally,

Definition 2.23: Algorithmic Allocation for the Single-minded Case [211]

Input: $(S_i^\star, \vartheta_i^\star)$ for each agent $i = 1, \ldots, n$

Output: A subset of winning bids $W \subseteq \{1, \ldots, n\}$ such that $S_i^\star \cap S_j^\star = \emptyset$ for every $i \neq j$ in $\mathcal{A}$ (i.e., the winning allocation is feasible). We also want the social welfare to be maximized $\sum_{i \in W} \vartheta_i^\star$.

The algorithmic allocation is a "weighted-packing" problem. We next show that this is NP-hard by reducing from the known NP-complete INDEPENDENT-SET problem.

Theorem 2.6 (Allocation Problem in Definition 2.23 in NP-Hard [211])**.** *The decision problem of whether the optimal allocation has social welfare at least k (where k is an additional part of the input) is NP-complete.*

Proof. Independent of this result, to show that any problem X is NP-hard, we must take a known NP-complete problem and reduce it to X. The computational hardness follows the direction of the reduction, implying that X is also NP-hard.

For the allocation problem specifically, we make the reduction from the well-known NP-complete INDEPENDENT-SET problem [211]. In this problem, we have an undirected graph $G = (V, E)$ and a number k. The problem is determining whether G has an independent set of size k. Recall that an independent set is a subset of vertices with no edge between them.

To reduce from the INDEPENDENT-SET problem, we build an allocation problem as follows:

- The set of items is E (i.e., the set of edges)
- Let each vertex of G denote an agent. For vertex $i \in V$, let the desired bundle of i be the set of adjacent edges $S_i^\star = \{e \in E \mid i \in e\}$ and the value be $\vartheta_i^\star = 1$.

Now, the set W is feasible ($S_i^\star \cap S_j^\star = \emptyset$) if and only if the set of vertices corresponding to W is an independent set in G. The social welfare obtained by W is exactly the size of this

set, i.e., the size of the independent set. That is, an independent set of size at least k exists if and only if the social welfare of the optimal allocation is at least k. This concludes the NP-hardness proof. Furthermore, the decision problem of whether the optimal allocation has social welfare is at least k is in NP is trivial. We can guess the optimal allocation, and then the social welfare can be guessed routinely. □

Usually, if any computational problem is shown to be NP-complete, there are three immediate approaches: (i) approximation, special cases, and heuristics. We focus on finding an approximately equal allocation.

Algorithm 1 ICA-SM Algorithm [169]

1. *Initialization:*
   - Sort the agents according to the order : $\vartheta_1^\star/\sqrt{|S_1^\star|} \geq \vartheta_2^\star/\sqrt{|S_2^\star|} \geq \cdots \geq \vartheta_n^\star/\sqrt{|S_n^\star|}$
   - $W \leftarrow \emptyset$
2. *For* $i : 1 \rightarrow n$, if $S_i^\star \cap (\cup_{j \in W} S_j^\star) = \emptyset$ then $W \leftarrow W \cup \{i\}$
3. *Output:*
   - *Allocation:* The set of winners is W.
   - *Payments:* $\forall i \in W, p_i = \vartheta_j^\star/\sqrt{|S_j^\star|/|S_i^\star|}$ where j is the smallest index such that $S_i^\star \cap S_j^\star \neq \emptyset$, and for all $k < j$, $k \neq i$, $S_k^* \cap S_j^* = \emptyset$. If no such j exists then $p_i = 0$.

Incentive-comptabile, Approximate Algorithm for the Single-minded Case (ICA-SM) [169]. We say that an allocation $S = (S_1, \ldots, S_n)$ is c-approximate of the optimal allocation if for every other allocation $T = (T_1, \ldots, T_n)$, including the optimal one, we have that $\frac{\sum_i \vartheta(T_i)}{\sum_i \vartheta(S_i)} \leq c$. Algorithm 1 describes ICA-SM, which is a *greedy* algorithm that solves

the allocation problem for single-minded case with n agents, m items, ϑ_i and S_i as agent i's bid valuation and preferred bundle of items, with W as the set of winners approximately.

ICA-SM is *computationally efficient*, *incentive compatible* and is $\sqrt{m}$-approximate [169]. Computational efficiency is trivial as the algorithm is greedy and, thus, polynomial in the number of agents n. The algorithm is also DSIC, whose proof[8] is available at [211, Lemma 11.9]. Claim 2.3 proves the $\sqrt{m}$ bound.

Claim 2.3. *Let OPT be an allocation (i.e., set of winners) with maximum social welfare, $\sum_{i \in OPT} \vartheta_i^\star$, and let W be the output of ICA-SM. We have $\sum_{i \in OPT} \vartheta_i^\star \leq \sqrt{m} \sum_{i \in W} \vartheta_i^\star$.*

Proof. For each $i \in \mathcal{A}$, let $OPT_i = \{j \in OPT, j \geq i \mid S_i^\star \cap S_j^\star = \emptyset\}$ denote the set of agents in OPT that were not able to be part of W because of i. Clearly, $OPT \subseteq \cup_{i \in W} OPT_i$. To show the bound, we need to show that for every $i \in W$, $\sum_{i \in OPT} \vartheta_i^\star \leq \sqrt{m} v_i^\star$.

Each $j \in OPT_i$ appears after i in the algorithm's greedy order, thus, $\vartheta_j^\star \leq \frac{\vartheta_i^\star \sqrt{|S_j^\star|}}{|\sqrt{|S_i^\star||}}$. Summing over all j, we get,

$$\sum_{j \in OPT_i} \vartheta_j^\star \leq \frac{\vartheta_i^\star}{\sqrt{|S_i^\star|}} \sum_{j \in OPT_i} \sqrt{|S_j^\star|}.$$

From the Cauchy-Schwartz inequality, we get,

$$\sum_{j \in OPT_i} \sqrt{|S_j^\star|} \leq \sqrt{|OPT_i|} \sqrt{\sum_{j \in OPT_i} |S_j|}$$

By construction, every $S_j^\star$ for $j \in OPT_i$ intersects $S_i^\star$. Since OPT is an allocation, these intersections must all be disjoint, and thus, $|OPT_i| \leq |S_i^\star|$. Since OPT is a valid allocation, $\sum_{j \in OPT_i} |S_j| \leq m$. We get $\sum_{j \in OPT_i} \sqrt{|S_j^\star|} \leq \sqrt{|S_i^\star|} \sqrt{m}$, and substituting this into the first inequality gives the claim $\sum_{i \in OPT} \vartheta_i^\star \leq \sqrt{m} \sum_{i \in W} \vartheta_i^\star$. □

[8] For completeness, the proof follows from the famous Myerson's lemma [201] as the allocation is monotone and payments satisfy the unique payment structure defined in the lemma.

Recall the Green-Laffont impossibility theorem (Theorem 2.3), which states that if the space of agent types is sufficiently rich, we cannot design a mechanism that is ex-post efficient, DSIC, and budget balanced (i.e., $\sum_i t_i = 0$). To this end, in Chapter 2.2.5.7, we look at Redistribution mechanisms, which are a class of mechanisms that aim to satisfy ex-post efficiency, DSIC, and minimizing $\sum_i t_i$ by redistributing the collected payments back to the agents.

2.2.5.7 Redistribution Mechanisms

For our discussion on combinatorial auctions, we consider a set of items $\mathcal{M} := [m]$ being auctioned with $\mathcal{A} := [n]$ as the set of interested agents. For redistribution mechanisms (RMs), we have the same setting with a slight caveat: the social planner (instead of an auctioneer) wishes to allocate $\mathcal{M}$ items to $\mathcal{A}$ agents. The social planner is *not* interested in maximizing its revenue. Instead, it prefers to allocate the items (i) as cheaply as possible and (ii) to agents who value them the most. Naturally, the social planner can try using VCG mechanisms as they are allocatively efficient (AE) and DSIC. But, from Theorem 2.3, we know that the VCG mechanisms are only weakly budget balanced, i.e., the transfer of money in the system is non-zero. This implies that the social planner will derive revenue from allocating the items, contrary to its desire to allocate them for free.

RMs tackle scenarios where the surplus money is not desired. E.g., allocation of public resources such as public housing, road access, national park permits, and health care. The Green-Laffont impossibility theorem (Theorem 2.3) necessitates collecting payments to ensure DSIC and AE. To this end, Maskin et al. [188] proposes first to collect the payments from the agents (similar to VCG) and then *redistribute* the surplus among them. The total payment redistributed to the agents is referred to *rebates* to the agents. More formally,

Redistribution Mechanism [188]. We say a Groves mechanism is a *Groves redistribution mechanism* or simply *redistribution mechanism* (RM) if it allocates items to the

agents in an AE manner and redistributes the Clarkes surplus in the system in the form of rebates to the agents such that the net payment made by each agent still follows the Groves payment structure.

2.2.5.7.1 Rebate Function Sticking to the notations introduced for the VCG mechanism (Chapter 2.2.5.5). Let the bid vector be $b = (b_1, b_2, \ldots, b_n)$ where $b_i = (b_{i1}, \ldots, b_{im})$ is the bid submitted by agent i for the m items. For a bid profile b, the rebate to an agent $i \in \mathcal{A}$ is denoted by $r_i(b)$. Furthermore, $t_i(b)$ is the payment made by i in Clarke pivotal mechanism, i.e., $t_i(b) = v_i(k^\star(b)) - (v(k^*(b)) - v(k^\star_{-i}(b)))$, where $k^\star(b)$ is an allocatively efficient allocation and $k^\star_{-i}(b)$ is allocatively efficient allocation without agent i. Refer to Eq. 2.10 for details. An RM is defined by the rebate function. Gujar et al. [130] provide the following characterization for rebate functions for designing DSIC RMs.

Theorem 2.7 ([130]). *In an RM, any deterministic, anonymous rebate function $g(\cdot)$ is DSIC iff the rebate for an agent $i \in \mathcal{A}$ is defined as $r_i := g(b_1, b_2, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n)$, $\forall i \in [n]$, where $b_1 \geq b_2 \geq \ldots \geq b_n$.*

The rebate function is DSIC if the rebate for an agent i is independent of its own bid. In general, the rebate function could take any form. E.g., the linear rebate function is defined as,

Definition 2.24: Linear Rebate Function

The rebate to an agent in a Redistribution Mechanisms (RM) is *linear* if it is a linear combination of the bid vectors of all the remaining agents. Furthermore, if the RM uses linear rebate functions for all the agents, we say that the RM is a linear RM.

In this thesis, we will focus only on linear rebate functions. We refer to the reader to Prakash [232] for an extended discussion on RMs.

Redistribution Index. Notice that RMs collect payments from the agents and redistribute the surplus using a rebate function. An RM's utility is quantified by the amount of surplus it redistributes. To measure this amount, we now define the redistribution index. For this, let $r_i(b)$ denote agent i's rebate given the bid vector $b = (b_1, b_2, \ldots, b_n)$ as input.

Definition 2.25: Redistribution Index [132]

We define the redistribution index as the worst-case or average-case fraction of the Clarke surplus that gets redistributed among the agents. That is,

$$e_{\mathsf{wc}} := \inf_{b:t(b)\neq 0} \frac{\sum_{i\in\mathcal{A}} r_i(b)}{t(b)} \text{ and } e_{\mathsf{avg}} := \mathbb{E}_{b:t(b)\neq 0}\left[\frac{\sum_{i\in\mathcal{A}} r_i(b)}{t(b)}\right]$$

Homogeneous vs. Heterogeneous Items. The RM literature explores both the homogeneous (i.e., items are identical) and heterogeneous (i.e., items are non-identical). E.g., Guo et al. [133] propose *Worst-case Optimal (WCO)*, a mechanism that uniquely maximizes the worst-case RI (among all RMs that are deterministic, anonymous, and satisfy DSIC, AE, and IR) when the items are homogeneous with unit demand [133, Theorem 1]. For the heterogeneous case, Gujar et al. [130] shows the impossibility of the existence of a linear rebate function with a non-zero redistribution index. This thesis focuses on the homogeneous RM setting. To this end, we first discuss the Bailey-Cavallo RM [49].

2.2.5.7.2 Bailey-Cavallo RM To reiterate, we have the set of agents $\mathcal{A} = [n]$ and $\mathcal{M} = [m]$ as the items with unit demand. That is, each agent requires one unit among the m available. The case with $m < n$ is non-trivial. The linear rebate function (Definition 2.24) can be expressed as for any agent $i \in \mathcal{A}$,

$$r_i = c_0 + c_1 \cdot b_1 + \ldots + c_{i-1} \cdot b_{i-1} + c_i \cdot b_{i+1} + \ldots + c_{n-1} \cdot b_n.$$

Here, $c_j \in \mathbb{R}, \forall j$ and w.l.o.g. we assume that $b_1 \geq b_2 \ldots \geq b_n$. We next define the Bailey-Cavallo RM.

Bailey-Cavallo Mechanism [49]. In this mechanism, $c_{m+1} = m/n$, $c_i = 0$ for all other i. The rebate r_i is thus,

$$r_i = \frac{m}{n} \cdot b_{m+2} \quad i \leq m+1, \quad r_i = \frac{m}{n} b_{m+1} \quad i > m+1$$

The total money redistributed is $(m+1)\frac{m}{n}b_{m+2} + (n-m-1)\frac{m}{n}b_{m+1}$. Since $r_i \geq 0$ for all agents, i.e., it satisfies IR. Furthermore, as $(m+1)\frac{m}{n}b_{m+2} + (n-m-1)\frac{m}{n}b_{m+1} \leq n\frac{m}{n}b_{m+1} = mb_{m+1}$, hence it is also feasible.

The Bailey-Cavallo mechanism offers the best-case rebate when $b_{m+1} = b_{m+2}$. That is, it returns $m \cdot b_{m+1}$. Significantly, this is the total VCG payments collected, implying that in the best case, the mechanism redistributes 100% of the revenue. However, the worst case occurs when $b_{m+2} = 0$, with the amount being redistributed being $\frac{n-m-1}{n}$ of the total VCG payment collected. Next, we look at an RM that achieves optimal worst-case redistribution.

2.2.5.7.3 Worst-case Optimal (WCO) [133]. Since the items are identical, each agent $i \in \mathcal{A}$ has the same value for each item in $\mathcal{M}$, i.e., θ_i. W.l.o.g., we assume that $\theta_1 \geq \theta_2 \ldots \geq \theta_n$. In Clarke's pivotal mechanism (Chapter 2.2.5.5.2), the first m agents will receive the items, and each of these m agents will pay the highest losing bid, i.e., θ_{m+1}. So the VCG surplus is $m \cdot \theta_{m+1}$.

WCO maximizes the worst-case fraction of this surplus that gets redistributed. In WCO, (again) the first m agents receive the items and pay $p_i := t_i - r_i$, where r_i is their rebate. More formally,

$$
\begin{aligned}
r_i^{\mathsf{WCO}} &= c_{m+1}\theta_{m+2} + c_{m+2}\theta_{m+3} + \ldots + c_{n-1}\theta_n && i = 1, \ldots m+1 \\
r_i^{\mathsf{WCO}} &= c_{m+1}\theta_{m+1} + \ldots + c_{i-1}\theta_{i-1} + c_i\theta_{i+1} + \ldots + c_{n-1}\theta_n && i = m+2, \ldots n
\end{aligned}
\tag{2.11}
$$

where,

$$
c_i = \frac{(-1)^{i+m-1}\,(n-m)\,\binom{n-1}{m-1}}{i\binom{n-1}{i}\sum_{j=m}^{n-1}\binom{n-1}{j}} \left\{ \sum_{j=i}^{n-1} \binom{n-1}{j} \right\}; \quad i = m+1, \ldots, n-1 \tag{2.12}
$$

If $y_1 \geq y_2 \geq \ldots \geq y_{n-1}$ were the bids of the $(n-1)$ agents excluding agent i, then equivalently the rebate to agent i would be,

$$
r_i^{\mathsf{WCO}} = \sum_{j=m+1, j\neq i}^{n-1} c_j \cdot y_j \tag{2.13}
$$

The (worst-case) redistribution index of WCO, say e^{WCO}, is

$$
e^{\mathsf{WCO}} = 1 - \frac{\binom{n-1}{m}}{\sum_{j=m}^{n-1}\binom{n-1}{j}}
$$

WCO is optimal as there exists no other RM that can guarantee more than e^{WCO} fraction redistribution in the worst-case [133, Theorem 1].

Chapter 2.2.5.6 presents an interesting set of direct revelation mechanisms (DRMs) for the general and special cases of combinatorial auctions, whereas Chapter 2.2.5.7 presents DRMs for redistribution mechanisms. Next, we look at civic crowdfunding, which is an interesting application for the general class of indirect revelation mechanisms.

2.3 Civic Crowdfunding

Crowdfunding is a popular method to raise funds for various projects. These projects include for-profit start-ups or ventures and community well-being projects such as medical/academic bills, public bridges, libraries, or parks. This thesis focuses on crowdfunding

for public projects commonly referred to as *civic* crowdfunding [19, 312, 52, 54]. Bagnoli et al. [19] present the seminal work in civic crowdfunding, namely the Provision Point Mechanism (PPM).

2.3.1 Provision Point Mechanism (PPM)

Before describing PPM, we first state the general civic crowdfunding model used throughout this thesis.

Definition 2.26: The General Civic Crowdfunding Model

- Consider the crowdfunding of a set of projects $\mathcal{P}_m = \{1, \ldots, m\}$ with $\mathcal{A} = \{1, \ldots, n\}$ as the sets of interested agents. Denote $C_m = \{c_1, \ldots, c_m\}$ as the target costs (also called the "provision point") for the projects. The target costs absorb the required funds for a project's construction. Throughout this thesis, we assume that each project is non-rivalrous and the crowdfunding process has a *binary* outcome, i.e., each project either gets constructed or does not.

- We use $\theta_{ij} \in \mathbb{R}_{\geq 0}$ to denote agent i's valuation for the project j. The valuation quantifies an agent's interest in funding the project. Denote $\vartheta_j = \sum_{i \in \mathcal{A}} \theta_{ij}$ as the overall valuation for the project j. Throughout this thesis, we assume that $\vartheta_j \geq c_j$, $\forall j \in \mathcal{P}_m$. The case $\vartheta_j < c_j$ represents a lack of interest of the community towards funding the project and is typically not of interest.

- Each agent i contributes $x_{ij} \in \mathbb{R}_{\geq 0}$ to a project $j \in \mathcal{P}_m$ at any time t_{ij}. Denote $\mathbf{x}_j = \sum_{i \in \mathcal{A}} x_{ij}$ as the total contribution towards a project j.

- The crowdfunding process continues until a publicly-known deadline τ. At the end of τ, if $\mathbf{x}_j \geq c_j$, we say that the project j is funded (or "provisioned"). Likewise, if $\mathbf{x}_j < c_j$, the project j remains unfunded.

Furthermore, unless stated otherwise, the civic crowdfunding literature [312, 52] assumes the following:

1. Agents contribute at most once to each project $j \in \mathcal{P}_1$.
2. Agent utilities are quasi-linear.
3. Agents do not have any external information regarding any project funding except the total contribution made at any time and the time remaining till the deadline. That is, they are *symmetric* in their belief regarding the project's funding.
4. Agents are interested in the project's funding, i.e., their valuation towards the project's funding is positive.

Provision Point Mechanism (PPM) [19]. Bagnoli et al. [19] presented the first civic crowdfunding mechanisms, namely, PPM. The mechanism is for a single project, i.e., $m = 1$ in Definition 2.26. In PPM, the agents contribute to the project's funding until the deadline τ. If the project's total contribution crosses the target cost by τ, the agents receive a quasi-linear utility, which is the difference between their valuation and their contribution. Otherwise, the project remains unfunded, and the agents receive their contributions back. More formally, in PPM, each agent's i utility $u_i(\cdot)$ is, $\forall j \in \mathcal{P}_1$,

$$u_i(\theta_{ij}, x_{ij}; \mathbf{x}_j, c_j) := \underbrace{\mathbb{1}_{\mathbf{x}_j \geq c_j} \cdot (\theta_{ij} - x_{ij})}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x}_j < c_j} \cdot 0}_{\text{Unfunded Utility}} \tag{2.14}$$

In Eq. 2.14, $\mathbb{1}_X$ is an indicator variable that takes the value of 1 when X is true and zero otherwise.

2.3.1.1 PPM: Equilibrium Behavior

The solution concept used to analyze the equilibrium behavior of PPM is (pure-strategy) Nash equilibrium (Definition 2.3). Unfortunately, the game induced in PPM comprises both

(i) efficient equilibrium where the project is funded and (ii) inefficient equilibrium where the project is not funded. To illustrate these, consider the following examples.

Example 2.8: PPM: Efficient Equilibrium [19]

Given $\mathcal{A}, \mathcal{P}_1, \mathcal{C}_1$ (refer to Definition 2.26), an efficient (pure-strategy) Nash equilibrium in PPM is where $\forall j \in \mathcal{P}_1$ we have $0 \leq x_{ij} \leq \theta_{ij}\ \forall i \in \mathcal{A}$ such that $\sum_{i \in \mathcal{A}} x_{ij} = c_j$.

Example 2.9: PPM: Inefficient Equilibrium [19]

Given $\mathcal{A}, \mathcal{P}_1, \mathcal{C}_1$ (refer to Definition 2.26), an efficient (pure-strategy) Nash equilibrium in PPM is where $\forall j \in \mathcal{P}_1$ we have $\{x_{ij} \geq 0\}_{i \in \mathcal{A}}$ such that $\sum_{i \in \mathcal{A}} x_{ij} < c_j$ and $\sum_{k \in \mathcal{A}, k \neq i} x_{kj} + \theta_{ij} \leq c_j, \forall i \in \mathcal{A}$.

The inefficiency in Example 2.9 is apparent as the total contributions are short of the target cost. Furthermore, the contributions satisfy Nash equilibrium as no agent in Example 2.9 will prefer to change its contribution such that $\sum_{i \in \mathcal{A}} x_{ij} = c_j$ since it alone does not have a sufficient valuation for it.

2.3.1.2 PPM: Free-riding

In addition to the existence of inefficient equilibria in PPM, the civic crowdfunding model in general also suffers from *free-riding*. As mentioned in Definition 2.26, we consider public (i.e., non-excludable) and non-rivalrous projects. As such, any agent can consume the project without contributing towards its funding. E.g., if an agent does not contribute towards funding a public bridge, it can still avail the bridge for its commute. This results in a lack of incentive for rational agents to contribute actively towards a project's funding – leading to agents "free-riding" on others to contribute and fund the project while they only avail its benefits.

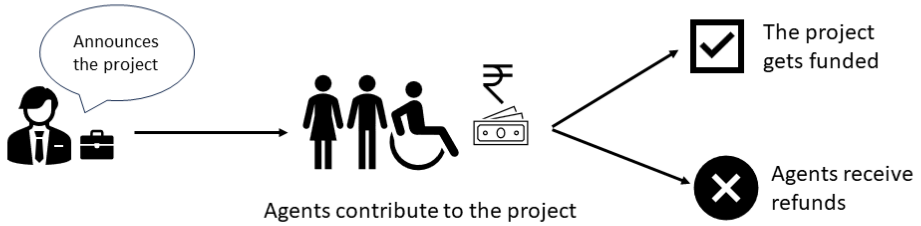


Figure 2.1: An overview of the civic crowdfunding process in PPR [312].

2.3.2 Provision Point Mechanism with Refunds (PPR)

Zubrickas [312] introduces a novel mechanism, namely, PPR, to address the two shortcomings of PPM: (i) the existence of inefficient equilibrium and (ii) agents free-riding. The key idea behind PPR is the introduction of *refunds* to the agents. More concretely, unlike PPM, in PPR, if the project is unfunded, the agents receive a refund – in proportion to their contributions – in addition to their contributions being returned. Similar to PPM, PPR is also for a single project, i.e., $m = 1$ in PPR. Figure 2.1 presents an overview of the civic crowdfunding process in PPR.

More formally, in PPR, each agent's i utility $u_i(\cdot)$ is, $\forall j \in \mathcal{P}_1$,

$$u_i(\theta_{ij}, x_{ij}; \mathbf{x}_j, c_j, B) := \underbrace{\mathbb{1}_{\mathbf{x}_j \geq c_j} \cdot (\theta_{ij} - x_{ij})}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x}_j < c_j} \cdot \left(\frac{x_{ij}}{\mathbf{x}_j} \cdot B \right)}_{\text{Unfunded Utility}} \tag{2.15}$$

In Eq. 2.15, $B > 0$ denotes the bonus budget. The social planner must set aside a publicly-known B at the start of the crowdfunding process. In a follow-up work, Cason et al. [48] conduct real-world experiments which suggest an ideal value of B to be around ten percent of the project's target cost.

Free-riding. The introduction of refunds to contributing agents overcomes the problem of free-riding from PPM. Agents are not only incentivized to contribute due to the allure of receiving refunds but also increase their contribution as the refund in Eq. 2.15 is proportional to the contribution made.

2.3.2.1 PPR: Project Status at Equilibrium

In PPR, the game-induced has the equilibrium wherein the project is funded. More formally,

Theorem 2.8 ([312]). *Given $\mathcal{A}, \mathcal{P}_1, \mathcal{C}_1$ (refer to Definition 2.26) with $0 < B \leq \vartheta_j - c_j,\ \forall j \in \mathcal{P}_1$ and agents utility structure as defined in Eq. 2.15, then at equilibrium (i) $\mathbf{x}_j = c_j,\ \forall j \in \mathcal{P}_1$ and (ii) the set of (pure-strategy) Nash equilibria are $\{x^\star_{ij} \mid x^\star_{ij} \leq \frac{c_j}{B+c_j}\theta_{ij},\ \forall i \in \mathcal{A},\ \forall j \in \mathcal{P}_1\}$.*

Proof. We provide the proof also available at [312]. For part (i), we have to show that $\mathbf{x}_j = c_j,\ \forall j \in \mathcal{P}_1$. Note that $\mathbf{x}_j < c_j$ cannot hold at equilibrium as any agent can obtain a higher refund by marginally increasing its contribution as $B > 0$. Likewise, if $\mathbf{x}_j > c_j$, then any agent can decrease its contribution to receive a higher utility. That is, at equilibrium, $\mathbf{x}_j = c_j$.

For part (ii), for any agent i contributing to project $j \in \mathcal{P}_1$, $x^\star_{ij}$ is an equilibrium if for each agent i its funded utility (refer to Eq. 2.15) exceeds the highest possible refund, i.e.,

$$x^\star_{ij} \leq \frac{c_j}{B + c_j}\theta_{ij}. \tag{2.16}$$

Summing up this inequality $\forall i \in \mathcal{A}$ also results in the upper bound on the bonus budget $0 < B \leq \vartheta_j - c_j$. This completes the proof of the theorem. □

Theorem 2.8 Discussion. The condition $0 < B \leq \vartheta_j - c_j$ is the condition required for the existence of Theorem 2.8 as it ensures that the bound on $x^\star_{ij}$ from Eq. 2.16 is satisfied $\forall i \in \mathcal{A}$. In case $B > \vartheta_j - c_j$, then $\mathbf{x}_j = c_j$ cannot hold as Eq. 2.16 will be violated for at least an agent i.

Figure 2.2 presents the intuition behind the proof presented in Theorem 2.8. We see that the promise of a refund incentivizes agents with insufficient contributions to increase their contributions. The agents increase their contribution until they reach the upper limit given in Eq. 2.16.

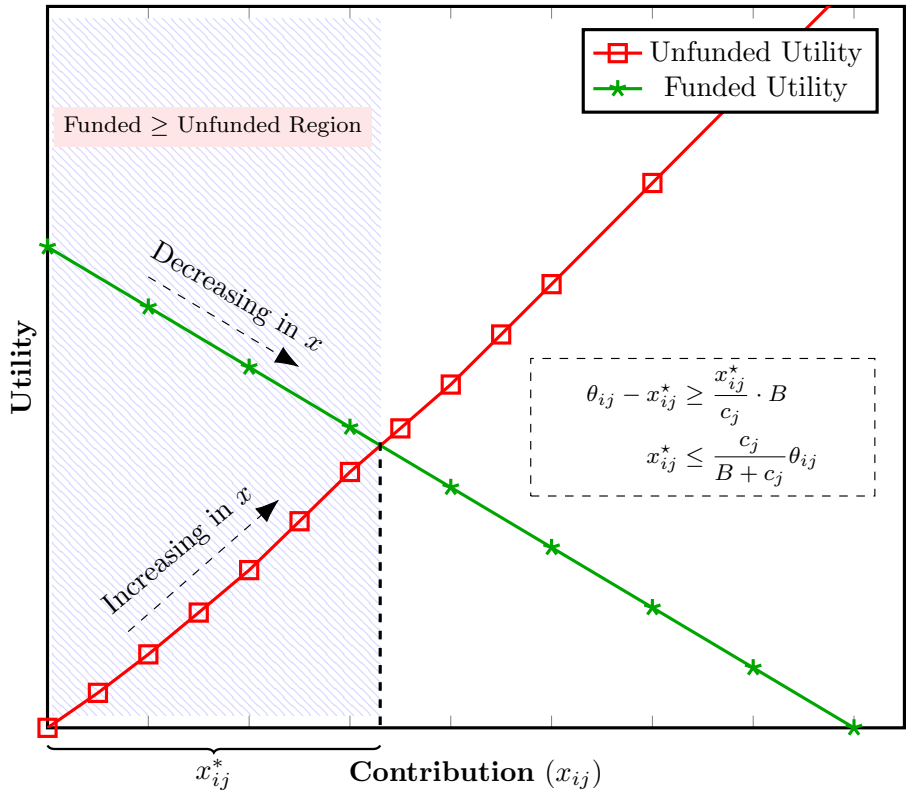


Figure 2.2: Funded vs. Unfunded Utility for Agent i. This figure illustrates the proof presented in Theorem 2.8.

Role of Bonus. Trivially, for another budget $B' > B$, we see that the set of equilibrium contributions that satisfy Eq. 2.16 are less than than for B. Intuitively, increasing the bonus implies an increase in refunds, and the agents must also receive a higher utility from the project's funding at equilibrium, reducing the set of equilibria. Furthermore, if $B = \vartheta_j - c_j$, PPR funds the public project as the *unique* equilibrium. This equilibrium is special: all agents contribute the same proportion to their valuations, i.e., $x_{ij} = \frac{c_j}{\vartheta_j} \cdot \theta_{ij}$.

Sources of Bonus [312]. Assuming that the general interest in the public project's funding is more than the target cost, $\vartheta > c$, the social planner only needs to provide credibility of its capacity to raise the bonus. Since at equilibrium, the public project is funded (Theorem 2.8), and thus, the bonus is *not* paid out. There may exist several ways to raise the refund bonus. For instance, from [312], some individual donors may provide initial seed money. Another method may be through insurance funds raised from premiums paid by the planners.

2.3.2.2 PPR: Race Condition

So far, our discussion around PPR highlights how the mechanism overcomes the challenges in civic crowdfunding due to free-riding and the existence of inefficient equilibria. PPR's impressive properties are due to the introduction of refunds to contributing agents. However, the mechanism's analysis assumes that agents decide their contributions *simultaneously* without knowledge of contributions made by the other agents. In practice, *online* crowdfunding platforms create a *sequential* setting where agents can observe contributions over time[9]. As such, each agent's strategy also comprises the time of contribution t_{ij} in addition to x_{ij}.

[9]Throughout this thesis, we refer to *online* CC as crowdfunding over online platforms such as Kickstarter [156], where the agents can observe real-time information. In contrast, *offline CC* is the case where the agents cannot observe any real-time information. Thus, agent strategies do not depend on time in offline CC.

Claim 2.4 ([52]). *Given $\mathcal{A}, \mathcal{P}_1, \mathcal{C}_1$ (refer to Definition 2.26) with $0 < B \leq \vartheta_j - c_j,\ \forall j \in \mathcal{P}_1$ and agents utility structure as defined in Eq. 2.15, then at equilibrium (i) $\mathbf{x}_j = c_j,\ \forall j \in \mathcal{P}_1$ and (ii) the set of (pure-strategy) Nash equilibria are $\{(x^\star_{ij}, \tau) \mid x^\star_{ij} \leq \frac{c_j}{B+c_j}\theta_{ij},\ \forall i \in \mathcal{A},\ \forall j \in \mathcal{P}_1\}$.*

Proof. As the refund is independent of time in PPR, agents are not incentivized to contribute before the deadline τ if the remaining agents follow the same strategy. Effectively, PPR collapses to a simultaneous-move game at the deadline τ. Theorem 2.8 defines the $x^\star_{ij}$s for this, which completes the claim. □

Claim 2.4 implies that the agents are incentivized to delay their contributions as close to the deadline. Near the deadline, we may see a "race" among the agents to grab as much refund as possible at the end. Such a race may lead to the equilibrium not being observed in practice due to PPR not offering any advantage to early contributors. Cason et al. [47] conduct real-world experiments to study the impact of early refunds. The authors observe both (i) a higher success probability of projects that offer early refunds and (ii) a "race" among the agents near the deadline to grab refunds.

2.3.3 Provision Point Mechanism with Securities (PPS)

Intuitively, an early refund bonus may provide the required advantage to early contributors, leading to the race condition being avoided. To this end, Chandra et al. [52] introduce the Provision Point Mechanism with Securities (PPS), a mechanism with temporal refunds. More concretely, the refunds in PPS decrease with an increase in time. PPS, similar to PPR and PPM, is for a single project, i.e., $m = 1$. In PPS, each agent's i utility $u_i(\cdot)$ is, $\forall j \in \mathcal{P}_1$,

$$u_i(\theta_{ij}, x_{ij}, t_{ij}; \mathbf{x}_j, c_j, B) := \underbrace{\mathbb{1}_{\mathbf{x}_j \geq c_j} \cdot (\theta_{ij} - x_{ij})}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x}_j < c_j} \cdot \left(s_{ij}^{t_{ij}} - x_{ij}\right)}_{\text{Unfunded Utility}} \tag{2.17}$$

In Eq. 2.17, $s_{ij}^{t_{ij}}$[10] are the number of securities allocated to the agent i. The securities $s_{ij}^{t_{ij}}$ depend on x_{ij}, t_{ij} and the total number of securities issued in the market at t_{ij}, denoted by $q_{t_{ij}}$. We have, from [52, Eq. 6],

$$s_{ij}^{t_{ij}} = C_0^{-1}\left(x_{ij} + C_0(q_{t_{ij}})\right) - q_{t_{ij}}, \tag{2.18}$$

where C_0 is the *cost function* governing the underlying prediction market in PPS obtained from the general cost function C by fixing the number of positive outcome securities. To be used in PPS, a cost function must satisfy [52, CONDITIONS 1-4, 6]. The properties of the cost function, C_0, relevant to our discussion are,

Property 2.1 (Condition 7, [52])**.** *The securities allocated to an agent $i \in \mathcal{A}$ contributing x_{ij} at time t_{ij} to a project $j \in \mathcal{P}_1$, $s_{ij}^{t_{ij}}$, is a monotonically increasing function of x_{ij}. In particular, we have $\frac{\partial s_{ij}^{t_{ij}}}{\partial x_{ij}} > 1$, $\forall \theta_{ij} < c_j$, $\forall i \in \mathcal{A}$ and $\forall j \in \mathcal{P}_1$.*

Property 2.2 (Step-2 (Theorem 3), [52])**.** *The refund to an agent $i \in \mathcal{A}$ contributing x_{ij} at time t_{ij} to a project $j \in \mathcal{P}_1$, $s_{ij}^{t_{ij}} - x_{ij}$, is a decreasing function of t_{ij}* and *$q_{t_{ij}}$ as $q_{t_{ij}}$ is non-decreasing function of t_{ij}, $\forall i \in \mathcal{A}$ and $\forall j \in \mathcal{P}_1$.*

Next, we illustrate the role of the cost function C_0 using the Logarithmic Market Scoring Rule (LMSR).

[10] Even though $s_{ij}^{t_{ij}}$ also depends on x_{ij}, we omit the dependency for simplicity of notation.

Example 2.10: Logarithmic Market Scoring Rule (LMSR) [59]

or a binary outcome event like civic crowdfunding, the LMSR cost function [59] is a popular cost function. Given the parameter b (used to control the speed at which the price changes), we can write the LMSR cost function as,

$$C_0(q_t) := b \cdot \ln(1 + \exp q_t/b)$$

$$\begin{aligned}\mathsf{Cost}(s^t|q_t) &:= C_0(q_t + s^t) - C_0(q_t) \\ &= b \cdot \ln\left(\frac{1 + \exp\left(\frac{q_t+s^t}{b}\right)}{1 + \exp\left(\frac{q_t}{b}\right)}\right)\end{aligned}$$

Now, an agent i contributing x_{ij} at time t_{ij} to project j receives $s_{ij}^{t_{ij}}$ securities where:

$$x_{ij} = C_0(q_{t_{ij}} + s_{ij}^{t_{ij}}) - C_0(q_{t_{ij}}) \tag{2.19}$$

$$s_{ij}^{t_{ij}} = C_0^{-1}(x_i + C_0(q_{t_{ij}})) - q_{t_{ij}} \tag{2.20}$$

$$s_{ij}^{t_{ij}} = b \cdot \ln\left(\exp\left(\frac{x_{ij}}{b} + \ln\left(\ln(1 + \exp\left(\frac{q_{t_{ij}}}{b}\right)\right)\right) - 1\right) - q_{t_{ij}}.$$

By algebraic manipulations, we can see that this satisfies Property 2.1 as $\frac{\partial s_{ij}^{t_{ij}}}{\partial x_{ij}} > 1$.

2.3.3.1 PPS: Project Status at Equilibrium

So far, we have shown that PPR suffers from the race condition and introduced temporal refunds through PPS. To show that PPS does not suffer from the race condition, we have to show that an agent's strategy in PPS is (x, t) for any $t \in \tau$. That is, the agents prefer to contribute at some t and do not participate in a race to contribute at the deadline τ. Theorem 2.9 shows that this is indeed the case in PPS. Furthermore, the agents actually prefer to contribute *as soon as they arrive* – instead of waiting for the deadline – due to the refunds being a decreasing function of t. For the theorem, denote a_{ij} as the time at which an agent $i \in \mathcal{A}$ arrives at the crowdfunding platform for project $j \in \mathcal{P}_1$.

Theorem 2.9 ([52]). *Consider the civic crowdfunding setup with $\mathcal{A}, \mathcal{P}_1, \mathcal{C}_1$ (refer to Definition 2.26) and each agent i's utility structure defined by Eq. 2.17. Let $C : \mathbb{R}^2 \to \mathbb{R}$ be a cost function satisfying [52, CONDITIONS 1-4, 6] and $C_0 : \mathbb{R} \to \mathbb{R}$ as the cost function obtained from C by fixing the number of positive outcome securities. If C_0 satisfies Property 2.1 and $\vartheta_j > C_0^{-1}(c_j + C_0(0)),\ \forall j \in \mathcal{P}_1$, then we have (i) $\mathbf{x}_j = c_j,\ \forall j \in \mathcal{P}_1$ and (ii) the set of sub-game Perfect Nash equilibria are $\{(x^\star_{ij}, a_{ij}) \mid x^\star_{ij} \leq C_0(\theta_{ij} + q_{a_{ij}} - C_0(q_{a_{ij}}),\ \forall i \in \mathcal{A},\ \forall j \in \mathcal{P}_1\}$.*

Proof. The proof similar ideas from Theorem 2.8. The first part, $\mathbf{x}_j = c_j,\ \forall j \in \mathcal{P}_1$, follows using the same argument from Theorem 2.8. For the second part, firstly, we know that at equilibrium, an agent's funded utility must be at least its unfunded utility. That is, if an agent's equilibrium strategy is $(x^\star_{ij}, t^\star_{ij})$, then $s_{ij}^{t^\star_{ij}} - x^\star_{ij} \leq \theta_{ij} - x^\star_{ij} \implies s_{ij}^{t^\star_{ij}} \leq \theta_{ij}$. Expressing $s_{ij}^{t^\star_{ij}}$ in terms of $x^\star_{ij}$ and $t^\star_{ij}$ using Eq. 2.19, we get:

$$
\begin{aligned}
C_0^{-1}(x^\star_{ij} + C_0(q_{tij})) - q_{t_{ij}} &\leq \theta_{ij} \\
\implies x^\star_{ij} &\leq C_0(\theta_{ij} + q_{t_{ij}})) - C_0(q_{tij})
\end{aligned}
$$

Here, the RHS of the last inequality is a monotonically decreasing function of $q_{t_{ij}}$ due to Property 2.2. That is, each agent $i \in \mathcal{A}$ minimizes the RHS at $t^\star_{ij} = a_{ij}$, resulting in its equilibrium strategy as $(x^\star_{ij}, a_{ij})$.

To complete the theorem's proof, we have to show that the strategy $(x^\star_{ij}, a_{ij})$ is also sub-game perfect. Here, we use backward induction. Essentially, the following possibilities are there for any agent i arriving at a_{ij},

- If the remaining amount left to be funded is zero, then $x^\star_{ij} = 0$.
- If the remaining amount is non-zero, then irrespective of the remaining amount, the agent i's equilibrium strategy is still defined by $(x^\star_{ij}, a_{ij})$. With backward induction and similar reasoning, every agent's best response is to follow their equilibrium strategy irrespective of history.

This completes the proof of the theorem. □

In summary, we saw PPR and PPS, two indirect mechanisms for civic crowdfunding that ensure the project is funded at equilibrium. Significantly, as the project gets funded, the social planner does not have to pay the refunds.

2.4 Fairness

In this chapter, we have extensively looked at what can be clubbed as "incentive" properties, e.g., DSIC, IR, allocative efficiency, among others. For a mechanism designer, these properties are (typically) crucial for a mechanism to satisfy. Recently, the proliferation of AI systems has also resulted in mechanism designers looking at mechanisms from a "fairness" perspective.

While discussing combinatorial auctions in Chapter 2.2.5.5, we said that we desired an allocation that maximizes social welfare. One can also look at this requirement as a fairness constraint since this implies that the items are allocated to those who value them the most! Researchers have recently highlighted the need for fairness in algorithms [65], particularly fair rewards in human-centered crowdsourcing [253]. Fair crowdsourcing platforms are necessary to ensure the participation of the crowd. As a result, recent research has focused on mechanisms with the fair provision of rewards, which will be discussed in detail next.

2.4.1 Fair Reward Mechanisms

This thesis focuses on what we refer to as "fair incentives". That is, designing incentives so that the mechanism satisfies certain fairness constraints. Looking at maximizing social welfare as a fairness constraint, we can say that the VCG payments are fair as they result in a DSIC mechanism allowing the mechanism designer to allocate the items to those who value them the most.

We look at recent literature on fair incentives [119, 200, 150]. This line of literature focuses on fair rewards for *crowdsourcing*. Crowdsourcing is the process where a *requester* publishes a set of tasks and solicits the help of a set of agents (sometimes called workers) to gather data/information. For their efforts, agents receive rewards. There are two popular approaches to deciding agent rewards in the crowdsourcing literature: (i) using gold-standard tasks and (ii) using the consistency of an agent's report with that of a random peer (Peer-based Mechanisms (PBMs)).

2.4.1.1 Deep Bayesian Trust (DBT)

One of the earliest fair-reward mechanisms in crowdsourcing, Goel et al. [119] propose DBT, which uses gold-standard tasks (i.e., tasks whose ground-trust is known)[11] to determine agent rewards. More concretely, DBT assigns gold-standard tasks to a select subset of agents and uses transitivity to derive the accuracy of the remaining agents from their peers' accuracy. DBT is DSIC and provides fair rewards to the agents.

Model. In DBT, each task has a discrete answer space $[K] := \{1, \ldots, K-1\}$. For any task, there are three random variables: (i) unknown *ground truth* g, (ii) each agent $i \in \mathcal{A}$ *observed signal* x_i which is private to it, and (iii) each agent $i \in \mathcal{A}$ *reported answer* y_i. Here, we have $g, x_i, y_i \in [K]$, $\forall i \in \mathcal{A}$. Each agent i's exerted effort is denoted by $e_i \in \{0, 1\}$, i.e., efforts are binary, representing either high or low effort. If $e_i = 1$, the agent i incurs a strictly positive finite cost. Agent utility is the quasi-linear utility comprising the difference of the reward it receives with its effort.

2.4.1.1.1 Agent's Strategy

We now discuss the agent's possible strategies:

Reporting Strategy. There are two possibilities based on the efforts exerted by an agent $i \in \mathcal{A}$.

[11] A popular example of such a task is that of recognizing the number of occurrences of a particular object/entity in a photo.

1. If $e_i = 1$, the reporting strategy S_i is a $K \times K$ right stochastic matrix where $S_i[x, y]$, $\forall x, y \in [K]$, is the probability of its reported answer on a task being y given that the observed signal was x.

2. If $e_0 = 0$, the reporting strategy $\vec{S}_i$ is a K dimensional probability vector where $\vec{S}_i[y]$, $\forall y \in [K]$, is the probability of its reported answer being y.

Truthful Strategy. An agent i's strategy is called truthful if $e_i = 1$ and S_i is an identity matrix. The agent judiciously solves the task and reports the answer obtained.

Heuristic Strategy. An agent i's strategy is called heuristic either if $e_i = 0$ or if $e_i = 1$ and all rows of S_i are identical. That is, the agent either does not solve the task ($e_i = 0$) or solves the tasks ($e_i = 1$) but reports its answer independently of the observed signal.

2.4.1.1.2 Proficiency and Trustworthiness Matrix

Consider the following definitions.

Definition 2.27: Proficiency Matrix [119]

For an agent $i \in \mathcal{A}$, we denote $A_i \in \mathbb{R}^{K \times K}$ as its proficiency matrix. Here, each $A_i[g, x]$, $\forall g, x \in [K]$, is the probability of the agent obtaining the answer x given that the ground truth of the task is g.

The idea behind the proficiency matrix is to help model the ability of an agent to obtain correct answers, given that it has exerted efforts. Note that each agent can have a different proficiency matrix.

Definition 2.28: Trustworthiness Matrix [119]

For an agent $i \in \mathcal{A}$, we denote $T_i \in \mathbb{R}^{K \times K}$ as its Trustworthiness matrix. Here, each $T_i[g, x]$, $\forall g, y \in [K]$, is the probability of the agent's reported answer being y given that the ground truth of the task is g.

While the proficiency matrix models an agent's ability, the trustworthiness matrix is a function of its ability and honesty.

2.4.1.1.3 Finding Trustworthiness Transitively The main ingenuity in DBT is to derive the trustworthiness of an agent, given the trustworthiness of another agent, using the joint distribution of their answers on shared tasks. For this, Goel et al. [119] define a *peer* for an agent i as another worker j so that their assigned set of tasks (say) Q_i and Q_j have a large overlap, i.e., $|Q_i \cap Q_j| \gg 0$. Given that these agents are solving a shared set of tasks, DBT uses their reported answers to create a joint probability distribution of their answers. Let $\omega(Y_i = y_i|Y_j|y_j)$ denote the conditional empirical distribution and $\omega(Y_j = y_j)$ to denote the empirical distribution of answers of peer j only. Also, let $P(g)$ denote the prior probability of a ground truth answer of any randomly selected task being g. The following lemma provides a closed-form solution to construct agent i's empirical distribution, given agent j's distribution and their joint distribution.

Lemma 2.3 ([119]). *As $|Q_i \cap Q_j| \to \infty$, with high probability, we have,*

$$\omega(Y_i = y_i|Y_j = y_j) = \sum_{g \in [K]} T_i[g, y_i] \cdot \left(\frac{T_j[g, y_j] \cdot P(g)}{\omega(Y_j = y_j)} \right),$$

$y_i, y_j \in [K]$ and $\omega(Y_j = y_j) \neq 0$.

Algorithm 2 Deep Bayesian Trust (DBT) [119]

1: Assign task set to an Oracle o and obtain its answers. Any agent o is called the Oracle if its trustworthiness matrix T_0 is known and T_0 does not have identical rows.

2: Initialize an *Informative answer pool* (IAP) with the answers given by the oracle.

3: Select some tasks from IAP.

4: Prepare a set of batches of tasks such that each contains tasks selected in the previous step and some fresh tasks.

5: Publish the batches on the platform and let agents select a batch they solve.

6: For an agent $i \in \mathcal{A}$ who submits its batch, find T_i according to Lemma 2.3. Reward agent i with an amount equal to $\beta \cdot \left(\sum_{g \in [K]} T_i[g, g]\right) - 1$, where β is a scaling constant.

7: If the agent's answers i satisfy informative criteria, add the answers to IAP and assign trustworthiness T_i as obtained in Step 6.

8: Asynchronously repeat steps 3, 4, 5, 6, and 7 until the desired numbers of answers are collected for all tasks.

DBT. Algorithm 2 presents the formal mechanism. The algorithm uses an *information criterion* as defined next.

Definition 2.29: Informativeness Criterion [119]

If $\omega(Y_j = y_j) \neq 0$ and the coefficient matrix $\frac{T_j[g, y_j] \cdot P(g)}{\omega(Y_j = y_j)}$ is full rank, the informative criterion is said to be satisfied.

The criterion's purpose is to check whether the answers provided by agent i can be used to estimate the trustworthiness of another agent. The criterion merely checks whether the

closed-form solution from Lemma 2.3 is solvable or not. As shown next, DBT is DSIC and (ex-post) IR in expectation.

Theorem 2.10 ([119])**.** *Given C^E as the cost of effort required to solve a given batch of tasks such that $\beta > \frac{C^E}{(\sum_{g\in[K]} A_i[g,g])-1}$ and $A_i[g,g] > A_i[g',g], \forall g' = g$, then the Deep Bayesian Trust Mechanism (Algorithm 2)) is DSIC $\forall i \in \mathcal{A}$ and ensures strictly positive expected reward in the truthful strategy.*

We can also show that employing the heuristic strategy does not yield any reward for the agent [119, Theorem 2]. These two results show that DBT satisfies the desirable incentive properties. Goel et al. [119] additionally, also introduce the following notion of a fair incentive mechanism.

Definition 2.30: Fair Incentive Mechanism [119]

An incentive mechanism is called *fair* if the expected reward of any agent is directly proportional to the accuracy of the answers reported by her and independent of the strategy and proficiency of her random peer.

Theorem 2.11. *([119]) DBT is a fair incentive mechanism.*

DBT has desirable properties but assumes access to gold-standard tasks. However, such tasks may not always be available.

2.4.1.2 FaRM: Fair Reward Mechanism for Information Aggregation in Spontaneous Localised Settings

Moti et al. [200] focuses on spontaneous localized settings where tasks are location-specific and must be answered quickly. In such settings, information aggregation is challenging as nearby agents can only collect the task answers. Moreover, prior knowledge

about the answer might not be readily available. Moti et al. [200] propose FaRM, a fair reward mechanism curated for these settings.

Model. FaRM is similar to DBT in that it also considers tasks with a discrete space, denoted by $\mathcal{X}$. Given a task, let x_i denote the observed signal of agent $i \in \mathcal{A}$, with y_i as its submitted answer. The utility structure of the agent comprises three functions. Details follow.

1. **Report Strength** $(\Phi(y_i))$**.** This is the count of the number of agents that reported the same answer as agent i, i.e.,

$$\Phi(y_i) = \sum_{j \in \mathcal{A}} \mathbb{1}_{y_i = y_j}.$$

   By definition, this quantity is always positive. Moreover, if all agents report truthfully, then it is agent i's best response (the response that maximizes $(\Phi(y_i))$ to report truthfully [200, Lemma 4.4].

2. **Consistency Score** (α)**.** This score keeps track of an agent's reputation in the mechanism. It increases with accurate reporting. Since the ground truth answers are unavailable, the consistency score uses the most frequent answer as a proxy. More formally, we have

$$\alpha_i^t = \begin{cases} \alpha_i^{t-1} - \frac{\alpha_i^{t-1}}{k} \times \frac{(\phi_1 - \Phi(y_i))}{|\mathcal{A}|} & \text{if } \Phi(y_i) < \phi_1 \\ \alpha_i^{t-1} + \frac{1-\alpha_i^{t-1}}{k} \times \frac{(\phi_1 - \phi_2)}{|\mathcal{A}|} & \text{if } \Phi(y_i) = \phi_1 \end{cases}$$

   where $k \geq 1$ and ϕ is defined as follows,

$$\phi_1 = \max_{s \in \mathcal{X}} (\Phi(s))$$

$$\phi_2 = \begin{cases} \max 2_{s \in \mathcal{X}} (\Phi(s)) & \text{if } \max 2_{s \in \mathcal{X}} > 0 \\ \frac{\psi_1^2 - 1}{\psi_1} & \text{if } \max 2_{s \in \mathcal{X}} = 0 \end{cases}$$

Here, we follow Moti et al. [200] and define $\max 2$ to denote the second highest value. Further, $\phi_1 \geq \phi_2 \geq 0$. By definition, consistency scores are within the range $[0, 1]$. Also, if all other agents report truthfully, then the best response for agent i to maximize its consistency score is to report truthfully [200, Lemma 4.8].

3. **Reliability Score (β).** The last component of an agent's utility is its reliability score. These aim to incentive any agent i to not collude with its set of neighboring agents (denoted $\mathcal{I}_i$). We also use $\mathcal{E}_i$ to define the set of agents not in i's neighborhood (i.e., $\mathcal{E}_i = \mathcal{A} \backslash \mathcal{I}_i$. Formally, it is the ratio of external agreement to internal agreement.

$$\beta_i = \frac{\frac{\sum_{j \in \mathcal{E}_i} \mathbb{1}_{y_i = y_j}}{|\mathcal{E}_i|}}{\frac{\sum_{j \in \mathcal{I}_i} \mathbb{1}_{y_i = y_j}}{|\mathcal{I}_i|} + 1}$$

Reliability Scores also map to the range $[0, 1]$. Similar to the previous two scores, if all other agents report truthfully, then the best response for agent i to maximize its reliability score is to report truthfully [200, Lemma 4.13].

Fairness Notions. FaRM introduces two notions of fairness: selective and cumulative.

Definition 2.31: Selective Fairness [200]

Consider any two *distinct* agents $i, j \in \mathcal{A}$ who submit two identical reports y_i and y_j such that $y_i = y_j$. Any reward scheme admits selective fairness if the reward is the same for both i and j.

Definition 2.32: Cumulative Fairness [200]

Consider any two *distinct* agents $i, j \in \mathcal{A}$ who submit two identical reports y_i and y_j such that $y_i = y_j$. Any reward scheme admits cumulative fairness if the reward is more for the agent who is consistently reporting the truth.

The authors show report strength (Φ) is selectively fair [200, Claim 4.6] and consistency score (α) is cumulatively fair [200, Claim 4.10]. With this, the authors show that FaRM admits a Nash equilibrium where reporting truthfully is the best response.

Theorem 2.12. *([200, Theorem 4.15].) FaRM is Nash incentive compatible with guaranteed non-negative utility and is weak budget balanced.*

Proposition 2.2. *([200, Proposition 4.16].) FaRM admits selective fairness and cumulative fairness and, hence, is a fair mechanism.*

2.4.1.3 REFORM

Goel et al. [119] propose DBT, a fair reward mechanism for crowdsourcing when the tasks have ground truth available. Kanaparthy et al. [151] present REFORM, a Reputation Based Fair and Temporal Reward Framework for Crowdsourcing. The key novelty in REFORM is that if an agent and its peer's report do not match, the agent receives additional chance(s) of pairing if its reputation exceeds that of its peer. REFORM uses a reputation mechanism to keep track of an agent's reputation across its historical performance. The reputation mechanism ensures that only trustworthy agents receive these additional chances to pair.

The authors argue that these additional chances are fairer for the agents since they may receive penalties due to unfair pairings (which may occur due to malicious or random behavior). To quantify fairness in REFORM, the authors propose the following notions of fairness.

Definition 2.33: γ-Fairness [151]

Any peer-based crowdsourcing mechanism is γ-Fair if the expected difference in its optimal and the expected reward taken over all possible reports equals γ.

Definition 2.34: Qualitative Fairness [151]

Any peer-based crowdsourcing mechanism with a reputation mechanism should ensure that an agent with a higher reputation score has a greater expected reward than agents with the same report but a lower reputation.

The authors show that REFORM satisfies both these fairness notions while also satisfying Nash IC [151].

Chapter 3

Preliminaries: Privacy and Blockchain

This thesis chapter provides a comprehensive overview of foundational concepts in cryptography, zero-knowledge proofs, and differential privacy. First, we discuss several popular cryptographic primitives, including hash functions, symmetric and asymmetric encryptions, and digital signatures. Second, we introduce zero-knowledge proofs, elucidating the significance of cryptographic protocols that enable one party to prove the validity of a statement to another without revealing any information about the statement itself. Third, we look at differential privacy (DP), a crucial paradigm in privacy-preserving data analysis. We formally define DP and explore methods for incorporating noise into computations to safeguard individual data. Last, the chapter looks at blockchains, specifically Bitcoin and Ethereum, explaining how these decentralized, distributed ledger technologies leverage cryptographic techniques to ensure transaction security and data integrity. In essence, this chapter lays the groundwork for the privacy-preserving applications discussed later as part of this thesis contributions.

⋆ ⋆ ⋆ ⋆ ⋆

3.1 Privacy

We begin by summarizing the standard cryptographic primitives in Chapter 3.1.1, followed by covering zero-knowledge proofs in Chapter 3.1.2. Chapter 3.1.3 introduces differential privacy (DP) and summarizes its relevant literature.

More concretely, Chapter 3.1.1 first presents the primitives (e.g., hash functions, encryption, and commitment) as abstract definitions. We then instantiate these to introduce several popular schemes, e.g., RSA and ElGamal encryptions and Pedersen commitments. With Chapter 3.1.2, we define ZKPs and their properties. Furthermore, we look at the two popular classes of ZKPs and conclude the section with a brief overview of zk-SNARKs. In Chapter 3.1.3, we define DP, introduce mechanisms to achieve DP and look at state-of-the-art composability methods.

3.1.1 Cryptographic Primitives

Cryptographic primitives are broadly aimed to allow the *private* transfer of information from a party (say "Alice") to another party (say "Bob"). Different primitives exist, effectively allowing the same in different settings in the presence of different adversaries. We first look at the most common of such primitives – *hash functions*.

For our analysis, denote λ as the security parameter and $\mathsf{negl}(\lambda)$ as a function negligible in λ. For completeness, a negligible function is defined as $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ such that for every positive polynomial $\mathsf{poly}(\cdot)$ there exists an integer $N_{\mathsf{poly}} > 0$ such that $\forall \lambda > N_{\mathsf{poly}}$ we have:

$$|\mathsf{negl}(\lambda)| < \frac{1}{\mathsf{poly}(\lambda)}.$$

3.1.1.1 Cryptographic Hash Functions

Hash functions are one-way functions that can input data of arbitrary size and map it to a fixed-size output. Formally,

Definition 3.1: Cryptographic Hash Function [69]

Given the security parameter λ, these are functions with fixed-size output, that is, $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$.

The function $H(\cdot)$ is said to be desirable for cryptographic applications if it satisfies the following three properties.

1. **Pre-image Resistance.** H satisfies pre-image resistance, if given a hash value h the probability of finding a message m such that $H(m) = h$ is negligible, i.e., $\Pr[H(m) = h|h] \leq \mathsf{negl}(\lambda)$.

2. **Second Pre-image Resistance.** H satisfies second-image resistance, if given an input message m_1 the probability of finding a distinct message m_2 such that $H(m_1) = H(m_2)$ is negligible, i.e., $\Pr[H(m_1) = H(m_2)|m_1] \leq \mathsf{negl}(\lambda)$.

3. **Collision Resistance.** H satisfies collision-resistance if the probability of finding two distinct messages m_1 and m_2 such that $H(m_1) = H(m_2)$ is negligible, i.e., $\Pr[H(m_1) = H(m_2)] \leq \mathsf{negl}(\lambda)$.

We say that the pair m_1 and m_2 is a *hash collision* if $H(m_1) = H(m_2)$. Trivially, collision resistance implies second pre-image resistance[1]. However, it does not imply pre-image resistance.

Illlustration. Intuitively, these properties mean that a malicious adversary cannot replace or modify the input message m without changing the digest $H(m)$. If two messages have the same digest, we can confidently assume that the strings are identical. E.g., let us say that Alice poses a difficult math problem to Bob and claims that she has solved it. Bob wishes to try the problem and also wants to ensure that Alice is not bluffing. Alice

[1]That is, second pre-image resistance is a weaker property, and hash functions that only satisfy it are considered to be insecure.

decides to write down her solution, compute its hash, and reveal only the hash value to Bob.

A few days go by, and Bob computes the solution to the problem himself and writes it down. Alice tells Bob that she had this solution earlier by revealing it and asking Bob to compute the hash and check that it matches the hash given to him before.

So far, we have provided the abstract definition of H and looked at an example showing its usefulness. Next, we look at some popular instances of hash functions.

SHA Family The Secure Hash Family (SHA) family of cryptographic hash functions was introduced by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS). The SHA family consists of the following algorithms:

1. **SHA-1.** First introduced in 1995 by the National Security Agency (NSA), these have an output size of 160 bits. The SHA-1 algorithm is secure against collision attacks up to 63 bits. Unfortunately, SHA-1 is not used in practice anymore due to known hash collisions.

2. **SHA-2.** The successor to SHA-1, SHA-2 has an output size of either 256 bits (SHA-256) or 512 bits (SHA-512). These are also designed by the NSA. Bitcoin [203] uses SHA-256.

3. **SHA-3.** Initially referred to as *Keccak*, the SHA-3 algorithm was chosen in 2012 after a public competition among non-NSA designers. It supports both 256 and 512 output sizes. Ethereum [42] uses the 256-bit variant.

3.1.1.2 Public-key Cryptography

Hash functions provide a useful way of "committing" to a message: an adversary observing $H(m)$ will not be able to infer the value of m. However, a drawback is that there is

no way for Alice to communicate the value of m to Bob privately. To enable such private information exchange, we need a method by which Alice can 'encrypt' the plaintext m to a ciphertext and communicate the ciphertext to Bob, which Bob can then 'decrypt' to learn m. This encryption-decryption paradigm can be achieved in the following two popular ways.

1. **Symmetric-key Cryptography.** This method requires Alice and Bob to agree on a *private* (or *secret*) key, which will used for both encryption and decryption. That is,

$$\text{Encrypt}_{\text{private key}}(\text{message}) = \text{ciphertext}$$

$$\text{Decrypt}_{\text{private key}}(\text{ciphertext}) = \text{message}$$

   While such an approach is useful in many applications, e.g., payment interfaces and data storage, it is also limited since generating and communicating the secret key is often challenging.

2. **Public-key Cryptography.** Asymmetric or Public-key Cryptography refers to using *both* a public key (for encryption) and a private key (for decryption). That is,

$$\text{Encrypt}_{\text{public key}}(\text{message}) = \text{ciphertext}$$

$$\text{Decrypt}_{\text{private key}}(\text{ciphertext}) = \text{message}$$

   E.g., if Bob wants to learn the message m privately held by Alice, he first communicates its public key to Alice. Alice uses the public key to encrypt m and then communicates it to Bob. Bob can safely use its private key to decrypt and learn m. Moreover, since the public key is safe to publish, anyone can send any private message to Bob using it. Public-key cryptography must follow the *correctness* property:

$$\text{Decrypt}_{\text{private key}}(\text{Encrypt}_{\text{public key}}(\text{message})) = \text{message}.$$

Next, we look at several useful cryptographic primitives based on public-key cryptography.

3.1.1.2.1 Encryption Schemes As discussed above, encryption schemes are cornerstones of cryptography and useful for private communication. We now formally define them.

Definition 3.2: Public-Key Encryption (PKE) Scheme

A PKE scheme consists of the following algorithms:

1. $\mathsf{KeyGen}(\lambda) \rightarrow (sk, pk)$. Given the security parameter λ, KeyGen samples a secret key $sk \leftarrow\!\!\$\ \{0,1\}^{\lambda}$, computes the public key pk using sk and outputs the pair (sk, pk).

2. $\mathsf{Enc}(pk, m; r) \rightarrow E(pk, m; r)$. To encrypt a message m, Enc takes input randomness r and outputs the ciphertext $E(pk, m; r)$.

3. $\mathsf{Dec}(E(pk, m; r), sk) \rightarrow m$. To decrypt m from $E(pk, m; r)$, the algorithm uses the secret key sk for decryption.

We remark that different instances of the algorithms outlined in Definition 3.2 result in different encryption schemes. We provide popular encryption schemes in Chapter 3.1.1.3

3.1.1.2.2 Digital Signatures A digital signature is a security technique that utilizes public key cryptography to verify the authenticity of a message. It creates a unique value linked to the message, which can be validated to confirm two crucial aspects: first, only the individual possessing the corresponding private/secret key could have generated a valid signature, and second, the message has remained unaltered since the signing process. Formally,

Definition 3.3: Digital Signature Scheme

The scheme consists of the following algorithms:

1. $\mathsf{KeyGen}(\lambda) \rightarrow (sk, pk)$. Given the security parameter λ, KeyGen samples a secret key $sk \leftarrow\!\!\$\ \{0,1\}^{\lambda}$, computes the public key pk using sk and outputs the pair (sk, pk).

2. $\mathsf{Sign}(sk, m) \rightarrow \mathsf{sign}$. The signing algorithm takes the secret key and the message to be signed m and produces the digital signature sign.

3. $\mathsf{SignVer}(pk, m, \mathsf{sign})$. The signing verification algorithm either accepts or rejects the message m's claim of authenticity based on the inputs: the public key, m, and the signature sign.

Digital signature schemes must satisfy two main properties. Firstly, the *correctness* property states that a digital signature generated using some private key should be correctly verified using the corresponding public key. This allows anyone with the public key to verify the authenticity of a signed message. Formally, we have

$$\mathsf{SignVer}(pk, m, \mathsf{Sign}(sk, m)) = \text{accept}.$$

The second property, namely *unforgeability*, requires that the probability with which an adversary can generate a valid signature for a message m – that verifies with the public key – without knowing the corresponding private key is negligible in λ. We refer the reader to [154, Chapter 12] for the formal definitions of the two properties.

3.1.1.2.3 Commitment Schemes A commitment scheme is a cryptographic protocol that allows a Sender (say Alice) to commit a chosen message m while keeping it hidden from the Receiver (say Bob). The Receiver has the ability to reveal the committed value later. Formally,

Definition 3.4: Commitment Schemes

These schemes consist of the following two phases:

1. **Commit Phase:** Given a security parameter λ, Alice samples a random value $r \xleftarrow{\$} \{0,1\}^\lambda$ and uses the commit function Commit and a private key $sk \xleftarrow{\$} \{0,1\}^\lambda$ to generate the commitment $c := \mathsf{Commit}(m; r, sk)$. Next, Alice sends the value c to Bob.

2. **Reveal Phase:** Here, Alice first sends the secret key sk to Bob. Bob uses the Open algorithm to open the commitment, i.e., $\mathsf{Open}(sk, c) \to (m', r')$. To check if m' was the original committed message, Bob runs $\mathsf{Check}(sk, m', r', c) \to \{0,1\}$ which outputs 0 if m' was the original message and 1 otherwise.

Any commitment scheme is useful for a cryptographic application if it satisfies the following two security properties.

- **Hiding.** Bob, upon receiving the commitment c, must learn no information about the underlying committed message m.

- **Binding.** Alice, after selecting the private key sk in the Commit phase, cannot send another key $sk' \neq sk$ to Bob in the Reveal phase and still pass the check. Moreover, given a secret key sk, the probability of generating another pair (m', r') that outputs the same commitment c must be negligible. That is,

$$\Pr[\mathsf{Commit}(m; r, sk) = \mathsf{Commit}(m'; r', sk) | m \neq m'] \leq \mathsf{negl}(\lambda).$$

So far, we have looked at the abstractions of popular cryptographic primitives. To instantiate these, we next introduce elliptic-curve and non-elliptic curve cryptography.

3.1.1.3 Elliptic-curve (EC) and non-EC Cryptography

Cryptography researchers rely on the known hardness of problems such as *interger factorization* and the *discrete-logarithm* problem (DLP) to instantiate the cryptographic primitives mentioned above.

3.1.1.3.1 The Discrete Logarithm Problem We now present the DLP for both Elliptic-curve (EC) and non-EC cryptosystems[2]. These are the two most popular cryptosystems. We begin by defining the non-EC variant.

Definition 3.5: Discrete Logarithm Problem (DLP)

Let G be any group with multiplication as the group operation. Let G's identity element be 1. Consider $g \in G$ and a positive integer r such that g^r denotes the product of g with itself r times. Likewise, let g^{-r} denote the product of g^{-1} with itself r times. Now, let $h \in G$. The DLP is to find the discrete logarithm to the base g of h, i.e., to solve $g^r = a$ or $r = \log_g h$.

For e.g., for the group $\mathbb{Z}_5^*$, we have the generator $g = 2$. Now the discrete logarithm of $h = 1$ is 4 because $2^4 = 1 \pmod 5$.

Elliptic Curves (ECs). Before defining DLP for ECs, we formally define them.

Definition 3.6: Elliptic Curves (ECs)

An elliptic curve E over a field $\mathbb{K}$ is the set

$$\{(x, y) \in \mathbb{K}^2 : y^2 = x^3 + ax + b, a, b, \in \mathbb{K}\} \cup \{\infty\},$$

with the restriction that $4a^3 + 27b^2 \neq 0$.

[2]Sometimes also referred to as the RSA cryptosystem

The restriction $4a^3+27b^2 \neq 0$ is imposed to ensure that the elliptic curve is non-singular. A non-singular elliptic curve does not have any singular points. These are points where the tangent line is not well-defined. Singular points can cause issues in the group law operations on the elliptic curve.

The DLP (Definition 3.5) for ECs is defined as follows.

Definition 3.7: Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given a curve E, let $p, q \in E$ be two points on the curve such that $q = k \cdot p$, for some positive integer k. Here, $q = k \cdot p$ represents the point p on the elliptic curve E added to itself k times. Then ECDLP is to determine k given p and q

Hardness of DLP. Both variants of DLP are computationally intractable. That is, it is computationally easy to calculate q given k and p, but it is computationally infeasible to determine k given q and p for a large curve E.

Next, we present the RSA cryptosystem, which is the first PKE scheme for non-Eliptic curves. It relies on the hardness of the discrete logarithm problem and integer factorization.

3.1.1.3.2 RSA Cryptosystem The RSA (Rivest, Shamir, and Adleman) algorithm was proposed with the patent [241]. Notably, the authors received the ACM Turing Award for their contribution. While most cryptosystems rely on the hardness of DLP, the security of RSA relies on the hardness of integer factorization. Formally,

Definition 3.8: RSA PKE Scheme [241]

The scheme consists of the following algorithms:

1. $\mathsf{KeyGen}(\lambda) \to (e, d, n)$: The key generation involves several computations, as follows.
   (a) $p \leftarrow\$ \{0,1\}^\lambda$ and $q \leftarrow\$ \{0,1\}^\lambda$ such that p and q are both primes and $p \neq q$
   (b) $n := p \times q$
   (c) $\phi(n) := (p-1)(q-1)$ where ϕ is the Euler phi function
   (d) Select an integer e such that $\gcd(e, \phi(n)) = 1$ and $1 < e < \phi(n)$
   (e) $d := e^{-1} \pmod{\phi(n)}$
   (f) Denote (e, n) as the public key and (d, n) as the private key
2. $\mathsf{Enc}(e, n, m) \to \mathsf{enc}$: For any $m < n$, compute the encryption as $\mathsf{enc} := m^e \pmod{n}$
3. $\mathsf{Dec}(d, n, \mathsf{enc}) \to m$: Decrypt the ciphertext as $m := \mathsf{enc}^d \pmod{n}$

Correctness. It is trivial to see that the decryption step returns the message m as $\mathsf{enc}^d \pmod{n} = (m^e)^d \pmod{n} = m \pmod{n} = m$. This is because $e \cdot d \equiv 1 \pmod{\phi(n)}$ by construction.

Security. An attacker can try to break the RSA PKE scheme through various attacks. We discuss some of them here.

- **Brute Force.** The simplest attack is trying all possible private keys. However, as the key space is exponential, RSA PKE easily thwarts this attack.
- **Using Integer Factorization.** Note that in RSA PKE, the integer modulus n is public. To generate d, the attacker must follow the key generation steps. It needs

to know the value $\phi(n)$. Thus, RSA is only secure if the integer factorization of $n = p \times q$ is hard. Luckily, carefully choosing p and q (e.g., equal-sized primes) makes this problem computationally infeasible. In modern usage, we prefer n to be greater than 2048 bits to avoid currently known factoring algorithms.

3.1.1.3.3 Paillier Encryption We now present the Paillier PKE scheme, which is also quite popular. Formally,

Definition 3.9: Paillier PKE Scheme [220]

The scheme consists of the following algorithms.

1. $\mathsf{KeyGen}(\lambda) \to (n, g, \alpha, \mu)$. The key generation involves the following computations.
   (a) Select $p \leftarrow\$ \{0,1\}^\lambda$ and $q \leftarrow\$ \{0,1\}^\lambda$ such that p and q are primes and $\gcd(pq, (p-1)(q-1)) = 1$
   (b) $n := pq$ and $\alpha := \text{lcm}(p-1, q-1)$
   (c) $g \leftarrow\$ \mathbb{Z}^*_{n^2}$
   (d) Ensure that n divides the order of g by the existence of the following modular multiplicative inverse: $\mu := (L(g^\alpha \bmod n^2))^{-1} \bmod n$. Here, $L(x) = \frac{x-1}{n}$ where $\frac{a}{b}$ denotes the quotient of a divided by b.
   (e) The public key is the pair (n, g) and the private key is the pair (α, μ)
2. $\mathsf{Enc}(n, g, m) \to \mathsf{enc}$. Given the message $0 \leq m < n$, sample a random r such that $0 < r < n$ and $\gcd(r, n) = 1$. The encryption is $\mathsf{enc} := g^m \cdot r^n \bmod n^2$.
3. $\mathsf{Dec}(n, \alpha, \mu, \mathsf{enc})$. To decrypt the message, perform the computation $m = L(\mathsf{enc}^\alpha \bmod n^2) \cdot \mu \bmod n$

Correctness. Similar to the RSA PKE scheme, we can see that the decryption step returns the message m. That is,

$$L(\mathsf{enc}^{\alpha} \mod n^2) \cdot \mu \mod n = L((g^m \cdot r^n \mod n^2)^{\alpha} \mod n^2) \cdot \mu \mod n$$
$$= L((g^m \cdot r^n \mod n^2)^{\alpha} \mod n^2) \cdot (L(g^{\alpha} \mod n^2))^{-1} \mod n$$

Here, we use the fact that $r^{n\alpha} \equiv 1 \mod n^2$ (from Carmichael's theorem [45]). Now,

$$L(\mathsf{enc}^{\alpha} \mod n^2) \cdot \mu \mod n = L((g^{m\alpha} \mod n^2) \mod n^2) \cdot (L(g^{\alpha} \mod n^2))^{-1} \mod n$$

Using the binomial theorem, we can also show that (refer to [111] for the formal proof):

$$L((1+n)^x \mod n^2) \equiv x \mod n$$

Now,

$$L(\mathsf{enc}^{\alpha} \mod n^2) \cdot \mu \mod n = (m\alpha) \cdot \alpha^{-1} \mod n = m \quad \square$$

Homomorphic Properties. An important feature of the Paillier PKE scheme is its *homomorphic* properties. More concretely, the scheme is additively homomorphic[3], as follows.

- **Homomorphic addition of plaintexts.** The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts, i.e.,

$$\mathsf{Dec}(n, \alpha, \mu, \mathsf{Enc}(n, g, m_1) \cdot \mathsf{Enc}(n, g, m_2) \mod n^2) = (m_1 + m_2) \mod n$$

- **Homomorphic multiplication of plaintexts.** A ciphertext raised to the power of a plaintext will decrypt to the product of the two plaintexts, i.e.,

$$\mathsf{Dec}(n, \alpha, \mu, \mathsf{Enc}(n, g, m_1)^{m_2} \mod n^2) = (m_1 \cdot m_2) \mod n$$

$$\mathsf{Dec}(n, \alpha, \mu, \mathsf{Enc}(n, g, m_2)^{m_1} \mod n^2) = (m_1 \cdot m_2) \mod n$$

[3] In the cryptography literature, such functions are also referred to as *partially* homomorphic.

3.1.1.3.4 ElGamal Encryption

We now present another popular partially homomorphic scheme, namely the ElGamal PKE scheme.

Definition 3.10: ElGamal PKE Scheme [95]

The PKE consists of the following algorithms.

1. $\mathsf{KeyGen}(\lambda) \rightarrow (pk, sk)$. The key generation algorithm takes the security parameter λ and performs the following computations.
   (a) Choose a prime $p \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
   (b) A primitive element $\alpha \mod p$
   (c) Randomly select the private key sk such that $2 \leq sk \leq p-2$
   (d) Compute $\beta := \alpha^{sk} \mod p$
   (e) The public key is tuple $pk := (p, \alpha, \beta)$ while the private key is sk
2. $\mathsf{Enc}(pk, m) \rightarrow \mathsf{enc}$. To encrypt the message $m < p$, first select the randomness $k \leftarrow\!\!\$\ \mathbb{N}$. Next, compute $r \equiv \alpha^{k} \mod p$ and $t = \beta^{k} \cdot m \mod p$. The encryption enc is the pair (r, t).
3. $\mathsf{Dec}(sk, \mathsf{enc}) \rightarrow m$. We decrypt the ciphertext as $(t \cdot r)^{-sk} = m \mod p$.

To better understand the ElGamal PKE scheme, let us now look at an illustrative example.

Example 3.1: ElGamal PKE Illustration

Alice chooses $p = 107, \alpha = 2$ and $sk = 67$. She computes $\beta := 2^{67} \equiv 94 \mod 107$. That is, the public key is the tuple $(p, \alpha, \beta) = (107, 2, 94)$ and her private key is $sk = 67$. She communicates her public key with Bob.

Now, Bob wishes to send the message "B" ($m = 66$ in ASCII) to Alice. He chooses the randomness $k = 45$. He generates the encryption $\mathsf{enc} = (r, t) := (\alpha^k, \beta^k \cdot m) \equiv (2^{45}, 94^{45} \cdot 66) \equiv (28, 9) \mod 107$. He communities enc to Alice.

Alice learns about m by decrypting enc as follows: $t \cdot r^{-sk} = 9 \cdot 28^{-67} = 9 \cdot 28^{106-67} \equiv 9 \cdot 43 \equiv 66 \mod 107$.

We can also define the ElGamal PKE for EC as follows.

Definition 3.11: ElGamal PKE Scheme for EC [95]

The PKE consists of the following algorithms. Let $\mathbb{E}$ be an elliptic curve defined over a large prime field $\mathbb{F}_p$ with $G, H \in \mathbb{E}$ as publicly known generators.

1. $\mathsf{KeyGen}(\lambda) \to (sk, pk)$. Given the security parameter λ, KeyGen samples a secret key $sk \leftarrow\!\!\$\ \{0,1\}^\lambda$, computes the public key $pk = sk \cdot G$. It outputs the key-pair (sk, pk).

2. $\mathsf{Enc}(pk, m; r) \to \mathsf{enc}$. To encrypt a message m, the algorithm inputs a randomness r and outputs the curve point $(r \cdot G, P_m + \cdot r(sk \cdot G))$. Here, P_m is a publicly-known mapping of a value m to a curve point in $\mathbb{E}$.

3. $\mathsf{Dec}(\mathsf{enc}, sk) \to m$. To decrypt m from enc, the algorithms computes $m := P_m + \cdot r(sk \cdot G) - r \cdot sk \cdot G$.

Corrrectness. The correctness of the decryption process of ElGamal PKE can be trivially shown as follows.

$$\begin{aligned}(t \cdot r)^{-sk} &\equiv \beta^k \cdot m(\alpha^k)^{-sk} \mod p \\ &\equiv (\alpha^s k)^k \cdot m \cdot (\alpha^k)^{-sk} \mod p \\ &\equiv m \mod p \quad \square\end{aligned}$$

Security. Unlike the RSA PKE scheme, whose security relies on the hardness of factoring large integers, ElGamal's security relies on the DLP's hardness for a large prime modulus. In addition, compared to RSA, ElGamal has the disadvantage that its ciphertext is twice as long as its plaintext. However, it has the advantage that for the same plaintext, it can output, with a high probability, different ciphertexts.

3.1.1.3.5 EdDSA Digital Signature. A digital signature scheme allows verification of the authenticity of a message. Earlier, we looked at the RSA digital signature scheme. We now present the Edwards-curve Digital Signature Algorithm (EdDSA) [30], a signature scheme for elliptic-curve cryptography. EdDSA is a Schnorr-based signature[4] scheme defined over $\mathbb{E}$. In EdDSA, given G, one derives its public key pk by sampling $sk \leftarrow\$ \mathbb{F}_p$. A party i signs the value $H(m_i)$ for a secret message m_i denoted as its signature $cert_i = (R_i, S_i)$. Here, $R_i = r \cdot G$ s.t. $r \leftarrow\$ \mathbb{F}_p$, and $S_i = r + H(m_i) \cdot sk$. A verifier accepts the signature iff $S_i \cdot G = R_i + H(m_i) \cdot pk$ holds.

3.1.1.3.6 Pedersen Commitment We now discuss both the EC and non-EC variants of the Pedersen Commitment scheme.

[4]This is a digital signature produced by the Schnorr signature algorithm and is one of the oldest and simplest signature schemes based on prime-order groups. We refer the reader to [254] for further details.

Definition 3.12: Pedersen Commitment Scheme [229]

This schemes consists of the following two phases:

1. **Commit Phase:** Given a security parameter λ, let $p \leftarrow\$ \{0,1\}^\lambda$ and $q \leftarrow\$ \{0,1\}^\lambda$ denote large primes such that q divides $p-1$, G_q as the unique subgroup of $\mathbb{Z}_p^*$ of order q, and g as a generator of G_q. Also, let g and $h = g^a$ be elements of G_q such that $\log_g h$ is intractable, where $a \in \mathbb{Z}_q$ is the secret key.

   To commit a message $m \in \mathbb{Z}_q$, Alice first samples a random value $r \leftarrow\$\in \mathbb{Z}_q$. Alice then uses the commit function Commit to generate the commitment $c := \mathsf{Commit}(m; r, g, h, p) = g^x \cdot h^r \mod p$. Next, Alice sends the value c to Bob.

2. **Reveal Phase:** Here, Alice first sends the secret key a to Bob. Bob uses the Open algorithm to open the commitment, i.e., $\mathsf{Open}(a, c) \rightarrow (m', r')$. To check if m' was the original committed message, Bob runs $\mathsf{Check}(a, m', r', c) \rightarrow \{0,1\}$ which outputs 0 if m' was the original message and 1 otherwise.

Likewise, the scheme for elliptic curves is defined as follows.

Definition 3.13: Pedersen Commitment Schemes for EC [229]

The schemes consist of the following two phases:

1. **Commit Phase:** Let $\mathbb{E}$ be an elliptic curve defined over a large prime field $\mathbb{F}_p$ with $G, H \in \mathbb{E}$ as publicly known generators. Here, $G = a \cdot H$, where a is the secret key such that $a \leftarrow\!\!\$\, \mathbb{F}_p$.

   To commit a message $m \in \mathbb{F}_p$, Alice first samples a random value $r \leftarrow\!\!\$\!\in \mathbb{F}_p$. Alice then uses the commit function Commit to generate the commitment $c := \mathsf{Commit}(m; r) = m \cdot G + r \cdot H$. Next, Alice sends the value c to Bob.

2. **Reveal Phase:** Here, Alice first sends the secret key a to Bob. Bob uses the Open algorithm to open the commitment, i.e., $\mathsf{Open}(a, c) \rightarrow (m', r')$. To check if m' was the original committed message, Bob runs $\mathsf{Check}(a, m', r', c) \rightarrow \{0, 1\}$ which outputs 0 if m' was the original message and 1 otherwise.

Security. Similar to the ElGamal PKE scheme, the Pedersen commitment scheme's security relies on the DLP's hardness for both EC and non-EC. Pedersen commitments are *computationally binding* (it is not feasible to "change one's mind" after committing), and *perfectly hiding* (they reveal nothing about the committed data).

Homomorphic Properties. Similar to ElGamal and PKE schemes, Pedersen Commitments are also additively homomorphic. That is, $\mathsf{Commit}(m_1, r_1) \circ \mathsf{Commit}(m_2, r_2) = \mathsf{Commit}(m_1 + m_2, r_1 + r_2)$.

3.1.2 Zero-knowledge Proofs (ZKP)

Zero-knowledge proof (ZKP) is a method by which a party, called a *Prover* ($\mathcal{P}$), is able to convince another party, called a *Verifier* ($\mathcal{V}$), that it knows some information ω, without revealing ω (or any other information related to ω). Formally, $\mathcal{P}$ must convince $\mathcal{V}$ that

$\exists\omega : \mathcal{R}(l, \omega) = 1$ for a relation $\mathcal{R}$, an input l (from $\mathcal{V}$) and a witness ω from $\mathcal{P}$. A ZKP must satisfy:

- *Completeness:* If $\exists\omega : \mathcal{R}(l, \omega) = 1$, then an honest $\mathcal{P}$ convinces $\mathcal{V}$ except with negligible probability, i.e., with probability at-most $\ll 1/2$.
- *Soundness:* If $\nexists\omega : \mathcal{R}(l, \omega) = 1$, a dishonest $\mathcal{P}'$ convinces $\mathcal{V}$ with negligible probability, i.e., with probability at-most $\ll 1/2$.
- *Zero-knowledge.* If $\exists\omega : \mathcal{R}(l, \omega) = 1$, then $\mathcal{V}$ does not learn any information about ω except with negligible probability, i.e., with probability at-most $\ll 1/2$.

3.1.2.0.1 Interactive and Non-Interactive ZKPs Each ZKP is a tussle between the Prover $\mathcal{P}$ and the verifier $\mathcal{V}$: $\mathcal{P}$ wants to convince $\mathcal{V}$ that it knows a secret via a proof which must not reveal any information about the secret to $\mathcal{V}$. In general, these proofs can be (i) *interactive* (involve multiple rounds of communication between $\mathcal{P}$ and $\mathcal{V}$) and (ii) *non-interactive*, which do not involve any communication.

Interactive ZKP. We illustrate an interactive ZKP through the following example wherein Alice (acting as $\mathcal{P}$) wishes to convince Bob (acting as $\mathcal{V}$) that she knows the discrete log of a given value, given a group (refer to Definition 3.5 for the definition of the DLP).

More formally, Alice wants to prove to Bob that she knows a witness ω such that $y = g^{\omega} \mod p$. Here, the values y, g, and p are public knowledge. Alice first picks a randomness r_1 from the group, computes $t := g_1^r \mod p$, and sends t to Bob. In turn, Bob picks another randomness r_2 from the group and sends it to Alice. Alice computes $c := (r_1 - r_2 \cdot \omega) \mod p$ and sends c to Bob. Bob accepts that Alice knows ω iff $t = g^c \cdot y^{r_2} \mod p$. This is because, by trivial substitution, $g^{r_1-(r_1-r_2\cdot\omega) \mod p} \cdot g^{(r_1-r_2\cdot\omega) \mod p} = g^{r_1} = t$.

While interactive ZKPs find applicability in many systems (e.g., authentication systems), they also have certain drawbacks. For instance, these have limited transferability,

i.e., for Alice to prove that she knows ω to another party, the entire interactive process must be repeated. These ZKPs are also unscalable since they require both $\mathcal{P}$ and $\mathcal{V}$ to be active at the same time.

Non-interactive ZKP. Introduced by Amos Fiat and Adi Shamir in 1986, the "Fiat-Shamir" heuristic [104] is the first mechanism that converts an interactive ZKP to a non-interactive ZKP. The core idea is to create a digital signature of the proof. This process allows any $\mathcal{V}$ to verify the existence of ω without requiring $\mathcal{P}$ to be active simultaneously. We illustrate this by converting the interactive ZKP mentioned above into a non-interactive one.

As before, Alice wants to prove to Bob that she knows a witness ω such that $y = g^{\omega} \mod p$. She picks a randomness r from the group and computes $t := g^{r} \mod p$. Given a publicly known hash function $H(\cdot)$, Alice computes $h := H(g||y||t)$, where $||$ is the concatenation operator. Next, Alice computes $d := (r - h \cdot \omega) \mod p$. Bob accepts the proof iff $t := g^{d} \cdot y^{h} \mod p$.

In essence, the interactive step where Bob picks a randomness and sends it to Alice is replaced by the use of the Hash function, whose value depends on y. In contrast to their interactive counterpart, non-interactive ZKPs are scalable and transferrable. They form the backbone of more succinct (smaller proofs) ZKPs such as bulletproofs [40] (range proofs) and zk-SNARKs [249].

3.1.2.1 zk-SNARKs

A zero-knowledge succinct, non-interactive argument of knowledge (zk-SNARK) [249] allows a prover to convince a verifier about the correctness of a computation on private input through a protocol. The verifier-available information is referred to as *statement* $\vec{x}$ and the private input of the prover as *witness* $\vec{\omega}$. The protocol execution takes place in a non-interactive manner, with succinct communication.

A zk-SNARK consists of: *(i)* a *Setup* algorithm which outputs the public parameters PP for a NP-complete language $\mathcal{L}_{\mathcal{S}} = \{\vec{x} \mid \exists\ \vec{\omega} \text{ s.t. } \mathcal{S}(\vec{x}, \vec{\omega}) = 1\}$, where $\mathcal{S} : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$ is the *arithmetic circuit satisfiability problem* of an $\mathbb{F}$-arithmetic circuit; *(ii)* A *Prover* algorithm that outputs a constant size proof π, attesting to the correctness of $\vec{x} \in \mathcal{L}_S$ with witness $\vec{\omega}$; and *(iii)* A *Verifier* algorithm which efficiently checks the proof.

As part of Chapter 3.1.1, we looked at popular cryptographic primitives, including hashes, encryptions, commitments, and ZKPs, that are important ingredients to creating privacy-preserving applications. While cryptography has achieved widespread acceptance as the default privacy toolbox, certain applications in statistics or ML are more suited for another privacy measure, namely, differential privacy (DP). We next provide a comprehensive background on the DP literature.

3.1.3 Differential Privacy (DP)

Differential privacy [91, 92] is a crucial concept in the realm of data privacy and security, aiming to balance the need for accurate data analysis with the protection of individual privacy. At its core, differential privacy provides a rigorous framework for designing algorithms that allow the extraction of valuable insights from sensitive datasets while minimizing the risk of identifying specific individuals within the data.

Chapter 3.1.3 summarizes the existing DP literature, which DP practitioners may find useful. More concretely, we (i) formally define DP, (ii) introduce popular additive-noise mechanisms for DP, (iii) summarize several important properties, and (iv) summarize state-of-the-art composability results.

3.1.3.1 DP: Pure and Approximate

Consider a scenario with n individuals, denoted as X_1 through X_n, each possessing a distinct data point. These individuals communicate their respective data points to a "trusted curator," a singular entity trusted by all participants solely with their raw data

points. The curator employs an algorithm[5], denoted as $\mathcal{M}$, based on the received data and publicly discloses the computed result. The concept of differential privacy pertains to $\mathcal{M}$, asserting that the inclusion or exclusion of any individual's data does not substantially influence the algorithm's output. Datasets that defer in a single record are called *adjacent* databases. In essence, it ensures that the privacy of individual contributions is maintained in the face of publicized results. More formally,

Definition 3.14: Differential Privacy (DP) [93]

For a set of databases $\mathcal{X}$ and the set of noisy outputs $\mathcal{Y}$, a randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is said to be (ϵ, δ)-DP if $\forall D, D' \in \mathcal{X}$, s.t. D and D' are adjacent ($|D - D'| \leq 1$), and $\forall S \subseteq \mathcal{Y}$ the following holds,

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in S] + \delta. \tag{3.1}$$

Intuition. As stated, DP provides a statistical guarantee against an inference the adversary can make based on $\mathcal{M}$'s output. This guarantee is upper-bounded by ϵ, often referred to as the *privacy budget*[6]. The privacy budget, ϵ, controls the trade-off between quality (or, in the case of ML, the accuracy) of the output vis-a-vis the privacy guarantee. Researchers observe that an $\epsilon \geq 1$ implies that the chance of a privacy leak is around 99.9%. As such, $\epsilon < 1$ are preferred, with greater ϵs providing virtually no privacy protection.

Pure and Approximate DP. The role of δ in Eq. 3.1 can be seen as an "error" probability: the probability by which the ϵ bound is violated. E.g., $(1, 0.1)$-DP can be seen as the $\epsilon = 1$ bound holding with probability $1 - 0.1 = 0.9$.

Given this, if $\delta > 0$, we get an "approximate" worst-case guarantee while $\delta = 0$ provides a perfect bound. The DP literature often refers to (ϵ, δ)-DP where $\delta > 0$ as approximate-DP and $(\epsilon, 0)$-DP as pure-DP. The pure DP version is strictly more private than the

[5]The DP literature often refers to algorithms as "mechanisms". As such, we use algorithm/mechanism interchangeably in the DP context.

[6]The privacy budget term is more popularly used when there is more than one query.

approximate version for the same ϵ. DP practitioners typically set δ to be of the order $\mathcal{O}(1/n)$ or $\mathcal{O}(1/n^2)$ [93], where n is the number of data entries in the dataset D.

Privacy Loss Random Variable (PRLV). ϵ is a metric of *privacy loss* defined as,

$$L^{y}_{\mathcal{M}(x)||\mathcal{M}(x')} = \ln\left(\frac{\mathcal{M}(x)=y}{\mathcal{M}(x')=y}\right) \leq \epsilon \text{ w.p. } 1-\delta. \tag{3.2}$$

The PRLV is a useful notion for visualizing (and deriving) DP. However, it is non-trivial to show that Eq. 3.1 and Eq. 3.2 are equivalent (i.e., Eq. 3.1 $\iff$ Eq. 3.2). We refer the reader to [93, Lemma 3.17] for the proof of the same[7].

3.1.3.2 Additive Noise Mechanisms: Laplace and Gaussian

So far, we have introduced the notion of DP and provided some intuitive understanding regarding its usefulness as a metric for privacy. We now discuss methods by which one can create differentially private mechanisms. The most popular approach is adding calibrated noise to the output of the non-private mechanism. These class of DP mechanisms are referred to as *additive-noise* mechanisms.

3.1.3.2.1 Laplace Mechanism Before presenting the Laplace additive-noise mechanism [93], let us recollect the PDF of the Laplace distribution. We have,

$$\mathcal{L}(\mu, b) = \frac{1}{2b} \cdot \exp\left(-\frac{|x-\mu|}{b}\right), \tag{3.3}$$

where μ is the mean and b the scaling factor. When compared to the Gaussian distribution, the Laplace mechanism has a sharper peak and heavier tail. Before defining the Laplace mechanism, we will first define *sensitivity* of the function $f(\cdot)$, which we wish to make private as follows.

[7] We remark that it is relatively straightforward to show that Eq. 3.2 implies Eq. 3.1. E.g., using the Gaussian mechanism as the argument.

Definition 3.15: Sensitivity [93]

Let $\mathcal{D} = \{D_1, \ldots, D_m\}$ be the universal set of all adjacent datasets, i.e., $|D - D'| \leq 1$ for $D, D' \in \mathcal{D}$. For $f : \mathcal{D} \to \mathbb{R}^k$, we define its sensitivity Δ_f as:

$$\Delta_f = \max_{D,D'} ||f(D) - f(D')||. \tag{3.4}$$

When $k > 1$, the maximum usually corresponds to the L_1 or L_2 norm. With this, we are now ready to define the Laplace mechanism. Recall that $\mathcal{M}(f, D)$ is our randomized mechanism which ensures that the input function $f(\cdot)$ becomes (ϵ, δ)-DP such that $\delta \geq 0$.

Theorem 3.1 (Laplace Mechanism [92]). *A mechanism $\mathcal{M}(f, D, \epsilon)$ is said to be ϵ-DP if it adds noise drawn from $\mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right)$ to the output of $f(D)$. Formally,*

$$\mathcal{M}(f, D, \epsilon) = f(D) + \mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right). \tag{3.5}$$

Proof. From Eq. 3.2 and for $|D - D'| \leq 1$ for $D, D' \in \mathcal{D}$,

$$\begin{aligned}
\frac{\Pr(\mathcal{M}(f, D, \epsilon) = \vec{o})}{\Pr(\mathcal{M}(f, D', \epsilon) = \vec{o})} &= \frac{\Pr\left(f(D) + \mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right) = \vec{o}\right)}{\Pr\left(f(D') + \mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right) = \vec{o}\right)} = \frac{\prod_{i\in[k]} \Pr\left(f_i(D) + \mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right) = o_i\right)}{\prod_{i\in[k]} \Pr\left(f_i(D') + \mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right) = o_i\right)} \\
&= \frac{\prod_{i\in[k]} \Pr\left(\mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right) = o_i - f_i(D)\right)}{\prod_{i\in[k]} \Pr\left(\mathcal{L}\left(0, \frac{\Delta_f}{\epsilon}\right) = o_i - f_i(D')\right)} \\
&= \frac{\prod_{i\in[k]} \frac{1}{2b} \cdot \exp\left(-\frac{|o_i - f_i(D)|}{b}\right)}{\prod_{i\in[k]} \frac{1}{2b} \cdot \exp\left(-\frac{|o_i - f_i(D')|}{b}\right)} \text{ From Eq.3.3 and } \mu = 0 \\
&= \exp\left(\sum_{i\in[k]} \frac{|o_i - f_i(D')| - |o_i - f_i(D)|}{b}\right) \\
&\leq \exp\left(\sum_{i\in[k]} \frac{|f(D) - f(D')|}{b}\right) = \exp\left(\frac{\Delta_f}{b}\right) = \exp\epsilon.
\end{aligned}$$

The second last inequality follows from the triangle inequality and the last equality from Definition 3.15. This completes the proof. □

While the Laplace mechanism is used in many data science and AI applications, it can provide negative outputs for queries that must always be positive. Researchers propose certain workarounds in the standard Laplace mechanism to avoid discrepancies.

For a case where the output of a statistical query over database D is the sum of the relevant entries of total count C, i.e., $D.sum()$, these workarounds *may* be the following mechanisms. To study these, let the non-private output be in the interval $[a_{min}, a_{max}]$.

- **Mechanism 1.**
  1. Consider the value: $S = D.sum() + \mathcal{L}(0, \frac{a_{max}-a_{min}}{b})$
  2. If $S/C > a_{max}$ or $S/C < a_{min}$ re-compute $S := D.sum() + \mathcal{L}(0, \frac{a_{max}-a_{min}}{b})$
  3. Repeat Step 2 until $S/C \in [a_{min}, a_{max}]$
  4. Output S/C
- **Mechanism 2.**
  1. Consider the value: $S = D.sum() + \mathcal{L}(0, \frac{a_{max}-a_{min}}{b})$
  2. If $S/C > a_{max}$ return a_{max}
  3. Else If $S/C < a_{min}$ return a_{min}
  4. Else return S/C
- **Mechanism 3.**
  1. Consider the value: $S = D.sum() - C \cdot \frac{a_{max}+a_{min}}{2} + \mathcal{L}(0, \frac{a_{max}-a_{min}}{2})$
  2. Let $\bar{C} = C + \mathcal{L}(0, \frac{2}{\epsilon})$
  3. Return $\frac{S}{\bar{C}} + \frac{a_{max}+a_{min}}{2}$

- **Discussion.** While Mechanism 1 does *not* satisfy (ϵ)-DP, Mechanism 2 and 3 do. Mechanism 2 is widely used among the two as it guarantees smaller ϵ values and is also commonly referred to as "clipping" in the broad DP-ML literature.

3.1.3.2.2 Gaussian Mechanism With the Laplace mechanism, we added calibrated noise from the Laplace mechanism and proved that the resulting additive noise mechanism satisfies pure-DP. For an approximate DP guarantee, we now discuss the Gaussian Noise mechanism. Formally, a randomized mechanism $\mathcal{M}(x)$ satisfies (ϵ, δ)-DP if the agent communicates $\mathcal{M}(x) \triangleq x + \mathcal{N}\left(0, \frac{2\Delta(x)^2 \ln(1.25/\delta)}{\epsilon^2}\right)$. Here, x is the private value to be communicated with *sensitivity* $\Delta(x)$, and $\mathcal{N}(0, \sigma^2)$ the Gaussian distribution with mean zero and variance σ^2.

So far, we have looked at specific mechanisms that achieve (ϵ, δ)-DP. Recall that DP allows us to quantify the extent to which individual privacy in a statistical database is preserved. By ensuring indistinguishability between two adjacent databases, DP neutralizes linkage attacks. In the next section, we look at other fundamental properties of DP.

3.1.3.3 DP: Important Properties

3.1.3.3.1 Composability Our first property answers the question: when we run multiple algorithms, each of which have privacy guarantees on their own, what is the privacy guarantee on the union of their outputs? How do the privacy parameters degrade?

The cumulative privacy guarantee, i.e., privacy guarantee over different set of queries, is referred to as *composition*. The following property gives the resultant guarantee across k such queries. For the property, consider an adversary $\mathcal{A}$ with a view over k queries as $V^b = (R, Y_{1,b}, \ldots, Y_{k,b})$. Here, Y is the queries output and R $\mathcal{A}$'s internal randomness with $b \in \{0, 1\}$ as a binary parameter.

Property 3.1 (Composability)**.** *The class of ϵ-differentially private mechanisms $\mathcal{M}$ satisfies $k\epsilon$-DP under k-fold adaptive composition for an adversary $\mathcal{A}$. We assume that each query's randomness is independent of the other.*

Proof. A view of the adversary is the tuple $v = (r, y_1, \ldots, y_k)$. That is,

$$\begin{aligned}\frac{\Pr[V = v]}{\Pr[V' = v]} &= \left(\frac{\Pr[R = r]}{\Pr[R' = r]}\right) \cdot \prod_{i=1}^{i=k} \frac{\Pr[V_i = v_i | V_1 = v_1, \ldots, V_{i-1} = v_{i-1}]}{\Pr[V'_i = v_i | V'_1 = v_1, \ldots, V'_{i-1} = v_{i-1}]} \\ &\leq \prod_{i=1}^{i=k} \exp(\epsilon) \qquad (\text{since } M \in \mathcal{M} \text{ is } (\epsilon, \delta)\text{-DP}) \\ &= \exp(k\epsilon).\end{aligned}$$

□

3.1.3.3.2 Closure under Post-processing What is more, a differentially-private mechanism is also immune to post-processing. This implies that irrespective of any operation an adversary performs over the output of a DP mechanism, the privacy guarantees w.r.t. the indistinguishability of the databases does not change.

Property 3.2 (Closure under Post-processing)**.** *Let $\mathcal{M} : \mathbb{Z}_+^R \to R$ be a randomized mechanism that satisfies (ϵ, δ)-DP. Let $f : \mathbb{R} \to R'$ be an arbitrary function. Then, $foM : \mathbb{Z}_+^R \to R'$ is also (ϵ, δ)-DP.*

Proof. Given two adjacent databases $|D - D'| \leq 1$ and the output space $S \subseteq R'$, consider the following mapping: $T = \{r \in S | f(r) \in S\}$. Now, we have,

$$\begin{aligned}\Pr(foM(D) \in S) &= \Pr(\mathcal{M}(D) \in T) \\ &\leq \exp(\epsilon) \cdot \Pr(\mathcal{M}(D') \in T) \qquad (\text{since } \mathcal{M} \text{ is } (\epsilon, \delta)\text{-DP}) \\ &\exp(\epsilon) \cdot \Pr(foM(D') \in S).\end{aligned}$$

That is, foM is also (ϵ, δ)-DP. □

This property is quite useful for designing and the adaptivity of a DP mechanism. For e.g., consider generating a DP ML model. As these models are generally released for public use, post-processing property implies that the underlying privacy guarantees will not change irrespective of how one uses the ML model.

3.1.3.3.3 Group Privacy Next, we consider the case when the distance between the adjacent databases is greater than one row. In particular, consider databases differing in c rows. This amounts to the fact that an adversary with arbitrary auxiliary information can know if c particular participants submitted their information. We capture the privacy guarantee for such a case with the next property.

Property 3.3 (Group Privacy)**.** *For a set of databases $\mathcal{X}$ and the set of noisy outputs $\mathcal{Y}$, a randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is said to be (ϵ, δ)-LDP if $\forall D, D' \in \mathcal{X}$, s.t. $|D - D'| \leq c$, and $\forall S \subseteq \mathcal{Y}$ the following holds,*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon \cdot c) \Pr[\mathcal{M}(D') \in S] + \delta. \tag{3.6}$$

The proof for this follows directly from Definition 3.14. Crucially, the property highlights that the strength of the privacy guarantee drops linearly with the group size.

3.1.3.4 DP: Composition

The composition result presented with Property 3.1 can also be seamlessly extended for the approximate-DP case. That is, the class (ϵ, δ)-DP mechanisms satisfy $(k \cdot \epsilon, k \cdot \delta)$-DP given a k-fold adaptive adversary.

While we cannot do better than $k \cdot \epsilon$ for the pure-DP, notice that for the approximate-DP case, we may be able to derive a tighter bound than $k \cdot \epsilon$! This is because of the error probability δ – we can look at tighter bounds for ϵ by comprising (marginally) the error probability. Proposed by Dwork, Rothblum, and Vadhan [94], the celebrated *Advanced Composition* theorem uses this idea to reduce the overall privacy budget from $k \cdot \epsilon$ to (very coarsely) $\sqrt{k} \cdot \epsilon$.

3.1.3.4.1 Advanced Composition We start by looking at the main advanced composition theorem.

Theorem 3.2 (Advanced Composition [94]). *For all $\epsilon, \delta, \delta' > 0$, let $\mathcal{M} = (M_1, \ldots, M_k)$ be a sequence of (ϵ, δ)-DP mechanisms, where the M_i's are potentially chosen sequentially and adaptively, i.e., the adversary $\mathcal{A}$ is k-fold adaptive. Then $\mathcal{M}$ is $(\tilde{\epsilon}, \tilde{\delta})$-DP where $\tilde{\epsilon} = \epsilon\sqrt{2k\log(1/\tilde{\delta}} + k\epsilon\frac{e^\epsilon - 1}{e^\epsilon + 1}$ and $\tilde{\delta} = k\delta + \delta'$.*

Proof. (SKETCH). The proof of the theorem is involved and available at [94]. Here, we provide an overview of it. For the proof, we first show that any possible DP mechanism can be reduced to a binary mechanism. That is, mechanisms that either output $\{0, 1\}$ or blatantly violate privacy. Next, we show that the composition of such mechanisms gives the bounds as stated in the theorem.

To this end, we first look at the case where the privacy budget is not bounded (which gives the error probability, $k\delta + \delta'$). To calculate the privacy budget, we condition the complement of this event. The term $k\epsilon\frac{e^\epsilon - 1}{e^\epsilon + 1}$ gives the expected value of the k-fold privacy budget and is derived using Chernoff bound. The second term $\epsilon\sqrt{2k\log(1/\tilde{\delta}}$ shows how far away the expected value of the privacy budget can one be and is again calculated using the Chernoff bound. □

It may be more important for DP practitioners to know the usefulness of the Advanced composition theorem. To this end, consider the case where we require high privacy, i.e., small ϵ, then $\frac{e^\epsilon - 1}{e^\epsilon + 1} = \epsilon/2$. This implies that the second term in $\tilde{\epsilon}$ is $\approx \epsilon^2/2$. This term is an order lower than $k\epsilon$ for small ϵ

Moreover, the privacy guarantee also has a hyperparameter δ' – increasing δ' decreases the multiplicative factor in $\tilde{\epsilon}$. Combining these arguments, we get an improvement of $\sqrt{k}$ in the privacy budget, which is quite useful in many settings.

3.1.3.4.2 Moments Accountant While the Advanced composition theorem is a significant result in the DP literature, the $\sqrt{k}$ gain in the privacy budget is often insufficient for practical applications such as differentially-private ML. For applications that are often stochastic (e.g., ML with Stochastic Gradient Descent (SGD)), Abadi et al. [3] introduce the *moments accountant* which uses privacy amplification through *subsampling* to provide tighter composition bounds than advanced composition.

Why Moments Accountant? To double down on the "loose" accounting of the standard composition theorem, we provide the following running example (from [3]). Let us consider that we use the Gaussian mechanism (refer to Chapter 3.1.3.2), such that $\sigma = 4$ and $\delta = 10^{-5}$. That is, for each query, we have $(\epsilon, \delta) = (1.2, 10^{-5})$. Using the (naive) Basic Composition for $k = 10,000$ queries gives us a final privacy budget of $(k \cdot \epsilon, k \cdot \delta) = (12,000, 0.1)$. Certainly, this privacy guarantee provides virtually no protection. If we were a bit savvy and used the Advanced composition (Theorem 3.2), our privacy guarantees would evolve as follows: $(\tilde{\epsilon}, \tilde{\delta}) = (\epsilon\sqrt{k\log(1/\delta)}, k \cdot \delta) = (360, 0.1)$. This is an improvement, but it still offers virtually no protection.

While DP composes seamlessly, the increase in the privacy budget with the number of queries seems to lead to very weak privacy guarantees. However, training an ML model with SGD often involves several training epochs. Abadi et al. [3] introduce the moments' accountant that provides tighter privacy bound, particularly for ML training paradigms. Their accountant relies on privacy amplification by subsampling, as explained next.

Privacy Amplification by Subsampling. Recall the DP definition presented with Definition 3.14. The (ϵ, δ) privacy guarantee is the worst-case guarantee for two adjacent datasets D and D'. Consider training an ML model with SGD. At each step, we select a random subset of samples from the dataset, say, the 0.1 fraction. Now, the probability that the record in which D and D' differ being part of the subset also drops from 1 to 0.1. Thus, subsampling reduces the privacy budget by a factor of the subsampling probability (0.1 in our SGD example).

For instance, combining amplification by subsampling and the advanced composition [152] in our running example, with "q" as the subsampling probability, the privacy budget becomes: $(\tilde{\epsilon}, \tilde{\delta}) = (2q\epsilon\sqrt{k\log(1/\delta)}, qk\delta) = (10, 0.1)$. Here, we take $q = 0.01$. This is a couple of orders of magnitude improved privacy protection from the naive approach we started with. Unfortunately, the privacy guarantee is still higher for practical purposes.

Algorithm 3 DP-SGD [29]

Require: Training data $\{x_1, \ldots, x_n\}$, model parameters θ, loss function $\mathcal{L}(\theta) = \frac{1}{n}\sum_i \mathcal{L}(\theta, x_i)$, and hyperparameters including, learning rate η_t, noise scale σ, group/batch size L, gradient norm bound C, number of training epochs k

Initialize θ_0 randomly
for $t \in [k]$ **do**
  Take a random sample L_t with sampling probability L/n
  Compute gradient: for each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta, x_i)$
  Clip gradient: $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{||\mathbf{g}_t(x_i)||_2}{C}\right)$ ▷ bounds the sensitivity!
  Add noise: $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$
  Descent: $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$
end for
return θ_k

Moments Accountant. For tighter bound on the privacy budget, Abadi et al. [3] propose the moments accountant along with their differentially private variant of SGD, namely DP-SGD.

Theorem 3.3 (Moments Accountant for the Gaussian Mechanism [3])**.** *There exist constants c_1 and c_2 so that given the subsampling probability $q = L/n$ and the number*

of training steps k, Algorithm 3 is (ϵ, δ)-DP for any $\delta > 0$, if

$$\sigma \geq c_2 \frac{q\sqrt{k \cdot \log(1/\delta)}}{\epsilon}. \tag{3.7}$$

In particular, the moments accountant from Theorem 3.3 saves a factor of $\sqrt{\log(k/\delta)}$ compared to the Advanced composition. To illustrate the significance further, let us re-look our running example. Using the moments account we have: $(\tilde{\epsilon}, \tilde{\delta}) = (2q\epsilon\sqrt{k}, \delta) = (1.25, 10^{-5})$. Here, we again take $q = 0.01$. Indeed, we see that the accountant provides significant privacy protection!

3.2 Blockchain

In Chapter 3.1.1 and Chapter 3.1.2, we presented several popular cryptographic preliminaries and summarized zero-knowledge proofs. Whereas Chapter 3.1.3 provided a summary of another popular privacy variant, namely differential privacy. Towards designing privacy-preserving applications, *blockchains* have also emerged as the useful block. This is because a blockchain (with smart contract support) acts as a medium for pseudo-anonymous communication and supports pseudo-anonymous payments. As such, in Chapter 3.2, we introduce blockchain technology and summarize popular protocols such as Bitcoin [203] and Ethereum [42].

Blockchain is an interesting application of three fundamental fields in the computer science literature, namely, cryptography (Chapter 3.1), game theory (Chapter 2.1), and distributed systems. Since we have already discussed the former two, we begin our discussion on blockchains by first discussing distributed systems.

3.2.1 Distributed Systems

Distributed systems, e.g., Amazon Web Services (AWS), Google Drive, and many others, are ubiquitous today. Due to parallelism, these systems can be distributed geographically

for reliability and availability. They provide several benefits, including increased storage, computational power, interconnection across spatial locations, and no single point of failure. However, such a distribution also introduces the challenge of *coordination*.

3.2.1.1 Consensus

To understand the challenge of arriving *consensus* in a distributed system, let us look at the following example [287]. Imagine that two friends, Alice and Bob, wish to coordinate and arrange a dinner. Alice sends a text message to Bob to meet for dinner at 6 PM. Since her text may not have reached Bob, Alice cannot be sure that Bob read the message, so she will only go to the meeting point if she receives a confirmation from Bob. But Bob cannot be sure that his confirmation message was received; if his text got lost, Alice could not know if Bob received her suggestion or if Bob's confirmation was lost. So Bob will require Alice's confirmation to ensure she will be there. But Alice's confirmation can also be lost... the message exchange will continue forever!

This coordination problem is referred to as *consensus*, defined next.

Definition 3.16: Consensus [287]

There are n nodes, among which, at most, f might crash. That is, $n - f$ nodes are correct. Let node i start with an input value of v_i. We say that the system has achieved consensus if the n nodes decide on one of the v_i values and satisfy the following properties.

1. **Agreement.** All correct nodes decide on the same value.
2. **Termination.** All correct nodes terminate in finite time.
3. **Validity.** The agreed value must be the input value of a node.

Typically, we assume that all n nodes can send messages to every other node. Moreover, we do not consider a broadcast medium. If a node wishes to send a message to multiple nodes, it must send multiple messages individually.

3.2.1.1.1 Impossibility of Consensus

Consider the following definition.

Definition 3.17: Asynchronous Model [287]

The asynchronous model is event-based, i.e., "upon receiving message ..., do ..". Nodes do not have access to a synchronized wall clock. A message sent from one node to another will arrive in a finite but unbounded time.

From Definintion 3.17, we see that the Asynchronous model is a challenging scenario to achieve consensus, compared to the synchronous model where the nodes agree on a synchronized clock. In particular, a node's failure can be catastrophic in the Asynchronous model. Theorem 3.4 proves that there is no deterministic fault-tolerant consensus algorithm in the asynchronous model, not even for binary input.

Theorem 3.4 (Fischer, Lynch, and Paterson (FLP) Impossibility Theorem [106])**.** *There is no deterministic algorithm that always achieves consensus in the asynchronous model with* $f > 0$.

Theorem 3.4's proof is highly involved and requires setting up various properties, which we omit. For the interested reader, the proof is available at [287, Theorem 3.14]. The FLP impossibility is a landmark result in distributed systems. This result was awarded the 2001 PODC Influential Paper Award. From the theorem, we see that if $f = 0$, then each node can send its value to all others, wait for all values, and choose the minimum. However, if a single node crashes, there is no deterministic solution to consensus in the asynchronous model.

Researchers overcome the above impossibility by introducing randomness to each node. For instance, the Randomized consensus algorithm [287, Algorithm 3.15] achieves consensus in the asynchronous model when $f < n/2$ and with an expected runtime $\mathcal{O}(n^2)$ [287, Theorem 3.20][8].

3.2.1.2 Consensus Algorithms: BFT and its Variants

3.2.1.2.1 Byzantine Agreement. So far, we have assumed that nodes are either correct or they crash. We can make the system more complex by introducing *byzatine* nodes, i.e., nodes with arbitrary behavior, including collusion[9], as follows.

Definition 3.18: Byzantine [287]

A node that can have arbitrary behavior is called byzantine. This includes "anything imaginable", e.g., not sending any messages at all, sending different and wrong messages to different neighbors, or lying about the input value.

We also tweak the consensus definition introduced in Definition 3.16 to incorporate byzantine nodes.

Definition 3.19: Byzantine Agreement [287]

Achieving consensus as in Definition 3.16 in a system with byzantine nodes is called byzantine agreement.

The validity property introduced with Definition 3.16 also needs updating since the byzantine nodes can lie about their inputs. Thus, we need a definition of validity that differentiates between correct and byzantine inputs. This is called the **all-same validity**: if all correct nodes start with the same input v, the decision value must be v.

[8] We remark that *no* algorithm, deterministic or otherwise, can achieve consensus in the asynchronous model with $f \geq n/2$ node failures [287, Theorem 3.21].

[9] An important distinction is that a byzantine node cannot forge an incorrect sender address. This ensures that a single byzantine node cannot impersonate all nodes.

3.2.1.2.2 Synchronous Model The last piece of business before we discuss algorithms for byzantine agreement is defining the synchronous model. Recall the FLP impossibility (Theorem 3.4), which already states that if $f > 0$, the consensus is not possible in the asynchronous model. As such, we now look at the synchronous case but with the added complexity of nodes being byzantine.

Definition 3.20: Synchronous Model [287]

In the synchronous model, nodes operate in synchronous rounds. In each round, each node may send a message to the other nodes, receive the messages sent by the other nodes, and do some local computation.

3.2.1.2.3 Byzantine Fault Tolerance (BFT) BFT refers to the ability of a distributed system to achieve byzantine agreement in the presence of byzantine nodes. Pease, Shostak, and Lamport [228] present a fundamental result which shows that there exists *no* BFT algorithm for $n \leq 3f$. More formally,

Theorem 3.5 (BFT Impossibility [228])**.** *A distributed system with n nodes cannot reach byzantine agreement with $f \geq n/3$ byzantine nodes.*

Proof. We present the proof when $n = 3$. The general case follows from similar reasoning and is available at [287, Theorem 4.12]. Consider the distributed system with nodes u, v, w. For the nodes to achieve the all-same validity, a correct node must decide on its own value if another node supports that value. The third node might disagree, but that node could be byzantine.

If correct node u has input 0 and correct node v has input 1, the byzantine node w can lie and inform u that its input value is 0 but inform v of its values as 1. This leads to u and v deciding on their own values, which results in violating the agreement condition. The nodes u and v can communicate among themselves and notice that they have different w's values, but in this case, u cannot distinguish whether w or v is byzantine. □

Theorem 3.5 shows that when $f < n/3$, we can hope to create BFT algorithms. We present one such algorithm next.

Algorithm 4 King's Algorithm (for $f < n/3$) [29]

```
1: x = my input value
2: for phase = 1 to f + 1 do
3:     Broadcast value(x)                                   ▷ Round 1
4:     if some value(y) at least n − f times then           ▷ Round 2
5:         Broadcast propose(y)
6:     end if
7:     if some propose(z) received more than f times then
8:         x = z
9:     end if
10:    Let node v_i be the predefined king of this phase i  ▷ Round 3
11:    The king v_i broadcasts its current value w
12:    if received strictly less than n − f propose(x) then
13:        x = w
14:    end if
15: end for
```

3.2.1.2.4 King's Algorithm. Algorithm 4 presents the formal algorithm (from [29]). The algorithm is used to reach a byzantine agreement on a single value. The algorithm has the following characteristics: (i) Phases: There are $f + 1$ phases, with two rounds per phase; (ii) Broadcasts: Each process broadcasts its preferred value to all other processes in the first round of each phase and (iii) Faulty nodes: Faulty nodes are not required to broadcast and can send conflicting messages to different nodes. The algorithm reaches agreement if: (i) All correct nodes start with the same value and (ii) Correct nodes agree

on the same value after the phase where a correct node was king (refer to [287, Theorem 4.19] for the formal proof).

Unfortunately, King's algorithm requires $f+1$ predefined kings (a Byzantine agreement task itself). The algorithm is also for the restrictive synchronous model.

3.2.2 Blockchain

So far, we have seen several interesting consensus algorithms for the asynchronous and synchronous settings. However, they are restricted either by design (e.g., synchronous model) or the number of byzantine/faulty nodes. Bitcoin [203] introduced the *blockchain*, a groundbreaking method to create a decentralized ledger maintained asynchronously by (potential) byzantine nodes. The original blockchain proposed in [203] used a novel *Proof-of-Work* (PoW) based consensus algorithm to ensure that each node in the system maintains the same copy of the ledger.

Chapter 3.2.2 introduces and summarizes popular blockchains in literature. We end our discussion on the required preliminaries with blockchains, as these are one of the most popular applications of Game theory (Chapter 2.1), Cryptography (Chapter 3.1.1) and Distributed Systems (Chapter 3.2.1).

3.2.2.1 Blockchain as a Data Structure

At its core, Blockchain is a distributed and decentralized data structure that serves as a secure and transparent ledger for recording transactions across a network of computers. It consists of a chain of blocks, where each block contains a list of transactions. Figure 3.1 provides an overview of the transaction in a typical blockchain network. The key features that define blockchain as a data structure include:

1. **Decentralization:** Unlike traditional databases that are often centralized, a blockchain operates on a decentralized network of nodes. Each node maintains a copy of the en-

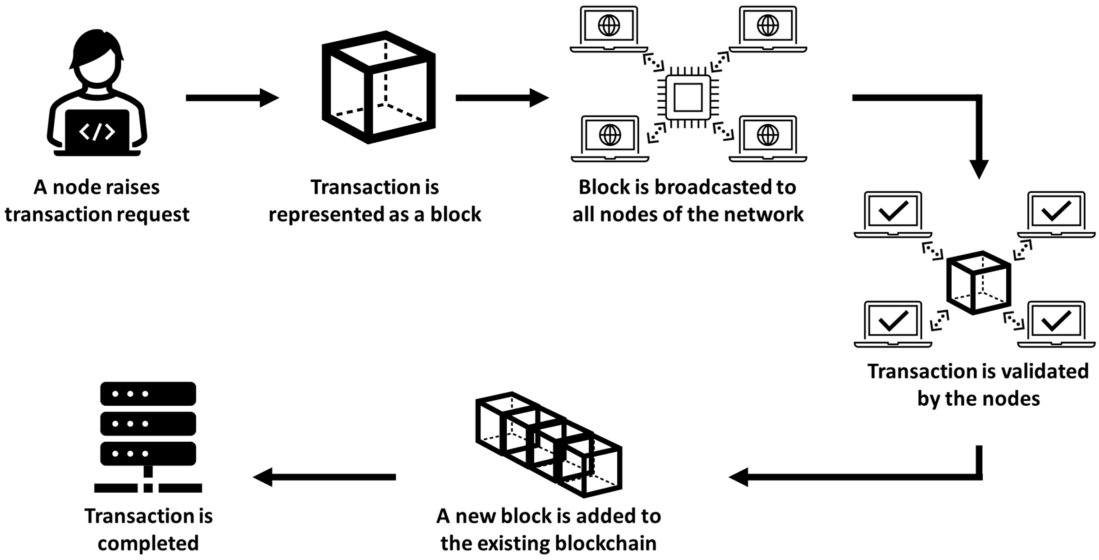


Figure 3.1: Transaction flow in a typical blockchain network (image credit: [236])

tire blockchain, and there is no single point of control. This decentralization enhances security and resilience.

2. **Immutability:** Once a block is added to the blockchain, it is extremely challenging to alter its content. Each block contains a cryptographic hash of the previous block, creating a chain of interlinked blocks. Modifying information in one block would necessitate changing all subsequent blocks, a task computationally infeasible due to the distributed nature of the network.

3. **Consensus Mechanism:** Blockchain relies on a consensus mechanism among network nodes to agree on the validity of transactions. This ensures that all nodes have a consistent view of the blockchain. Common consensus mechanisms include Proof of Work (used in Bitcoin, Chapter 3.2.2.2) and Proof of Stake (used in Ethereum 2.0, Chapter 3.2.2.3), among others.

4. **Transparency:** The entire transaction history is visible to all participants in the network. While the identities of users may be pseudonymous, the details of transactions, including amounts and timestamps, are often open and accessible.

5. **Security through Cryptography:** Cryptographic techniques, such as digital signatures and hash functions, play a pivotal role in securing transactions and maintaining the integrity of the blockchain. Transactions are signed using private keys, and the hash functions contribute to the immutability of the data.

3.2.2.2 Bitcoin

Introduced in 2008 by Satoshi Nakamoto [203], Bitcoin was the first successful blockchain application. Bitcoin is a peer-to-peer cryptocurrency offering pseudo-anonymous payments and resistance to double-spending as long as the majority of users in the system are honest. In Bitcoin, *miners* add transactions from the set of outstanding transactions (referred to as *mempool*). Each block is connected to the previous block using through a *hash chain*. For Bitcoin to work, we must (i) find a way to select a miner for a block and (ii) find a way to achieve consensus, i.e., all users in the system agree on a single copy of the blockchain ledger. In Bitcoin, these two processes are achieved using the Proof-of-Work (PoW) consensus protocol.

3.2.2.2.1 PoW-based Consensus Selecting a Miner. In a nutshell, a PoW is a piece of data that is computationally difficult for a user to produce but easy for others to verify that it meets certain requirements. In Bitcoin, a miner 'mines' a block by computing the SHA-256 hash of the block's header so that the hash is lower than or equal to the *target* (publicly decided on the network) of the block. More concretely, Bitcoin requires that the hash of a block must contain a certain number of zeros. To compute a hash that starts with several zero is low, so there is a race among the miners to be the first to do so. E.g., computing the hash where the last p bits are zero is $\frac{1}{2^p}$, i.e., negligible in p. For their

efforts, the miners receive a block reward (currently 6 BTC) and also earn transaction fees from the transactions they add to their block.

Reaching Consensus. Miners *append* their block, which contains the set of new transactions to the existing blockchain. We can expect that at a given time, multiple chains may exist either through malicious activity (e.g., miners appending their blocks at random heights) or through chance (e.g., multiple miners solving the hash puzzle simultaneously). In such an event, Bitcoin is said to have a *fork*. To resolve this fork, i.e., to converge to a single agreed chain of blocks, the Bitcoin protocol requires the miners to select the chain with the most number of blocks. That is, an honest miner will append its block to the chain with the most valid blocks over another chain with fewer.

Since its early adoption, blockchain technology has transcended its use in cryptocurrency applications, especially with the introduction of smart contracts.

3.2.2.3 Ethereum and Smart Contracts

The first and currently most popular blockchain network with smart-contract capabilities is Ethereum [42]. A smart contract is a computer program that can be run on-chain. In Ethereum, it is essentially bytecode executed over the Ethereum Virtual Network (EVM). Performing any smart-contract computation over the blockchain requires *gas*. The amount of gas charged depends on the type of computation. More computationally extensive operations require higher gas to be executed on-chain. The total charge for the computation is referred to as *gas cost*. Any computation that alters the *state* of the contract, i.e., alters any contract method or variables, consumes gas. On the contrary, reading data from the contract is free. To submit state-altering transactions, users need to specify a *gas price* that they are willing to pay per unit of gas. This choice affects the verification time for each such transaction. Generally, higher gas prices result in faster verification times.

3.2.2.4 Transaction Fee Mechanism Design

TFM design for public blockchains such as Bitcoin [203] and Ethereum [42] considers the following model. The blockchain's public ledger maintains the *state* and orders the sequence of *transactions* $t_1, t_2, \ldots, t_n$ that update the state. Let s_t be the size[10] of a transaction t. Each agent i broadcasts its transaction t_i with a bid $b_i \geq 0$. The bid represents the amount agent i is willing to pay for t_i, given its valuation $\theta_i \geq 0$. For security and practical reasons, each block has a *finite* size (denoted by C). Miners create blocks, maintain a *mempool* of outstanding transactions ($M = [t_n]$), and add a subset of these transactions to their blocks. Generally, the set of outstanding transactions is larger than the block size.

3.2.2.4.1 Transaction Fee Mechanism (TFM). Consider $\mathcal{H} = B_1, \ldots, B_{k-1}$ as the sequence of blocks denoting the on-chain history, current block B_k and mempool M. Designing a TFM involves defining (i) an *allocation* rule, which decides the transactions that get added to B_k, (ii) a *payment* rule describing the fraction of each transaction's bid that gets paid to the miner, and (iii) a *burning* rule, that is, the fraction of the amount that is removed from the supply, forever. An idiosyncrasy of blockchain involves *randomization* in transaction allocation. More concretely, with a "deterministic" TFM, we imply that a miner can include transactions in its block using any deterministic function. Whereas a "randomized" TFM implies that the miner selects the transactions to include through a random function[11].

[10] E.g., Ethereum transactions may be token transfers (smaller size) or sophisticated smart contract calls (larger size).

[11] TFMs may also use trusted on-chain randomness for transaction inclusion [62].

Definition 3.21: Transaction Fee Mechanism (TFM) [245]

For a given on-chain history $\mathcal{H}$, the mempool M and the current block B_k with size C, a TFM is the tuple $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ in which,

1. $\mathbf{x}$ is a feasible block allocation rule, i.e., $\sum_{t \in M} s_t \cdot x_t(\mathcal{H}, M) \leq C$ where $x_t(\cdot) \in \{0, 1\}$, $\forall t \in M$.
2. $\mathbf{p}$ is the payment rule with the payment for each transaction $t \in B_k$ denoted by $p_t(\mathcal{H}, B_k) \geq 0$.
3. $\mathbf{q}$ is the burning rule with the amount of burned coins for each transaction $t \in B_k$ denoted by $q_t(\mathcal{H}, B_k) \geq 0$.
4. $\tau \in \{\tau_D, \tau_R\}$ is the mechanism's type – either deterministic (τ_D) or randomized (τ_R).

3.2.3 User Model and Properties

We now define the relevant incentive properties introduced in [245] for a TFM. Recall that we assume that the miners and bidding agents are myopic [245, 103, 310, 62] – they are only concerned with their utility from the next block. Agent i's (per unit size) valuation is denoted by θ_i and its (per unit size) bid b_i. That is agent i bids $s_i \cdot b_i$. Let the vector $\mathbf{b}$ comprise all bids with $\mathbf{b}_{-i}$ representing all bids without agent i. Given $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ with $\mathcal{H}, M,$ and B_k, an agent i's *quasi-linear* utility u_i is,

$$u_i(\theta_i, B_k) = \begin{cases} (\theta_i - p_i - q_i) \cdot s_i & \text{if} \quad x_i = 1 \text{ (i.e., } t_i \in B_k), \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

Dominant-strategy Incentive Compatibility (DSIC). A strategic agent i will select b_i such that it maximizes its utility defined in Eq. 3.8. As such, we now define DSIC for a TFM.

Definition 3.22: DSIC [245]

We say any $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ with $\mathcal{H}$, M, and B_k is dominant-strategy incentive compatible if, assuming the miner follows the allocation rule $\mathbf{x}$, bidding θ_i for each agent i maximizes $u_i(\cdot)$ (Eq. 3.8) – irrespective of the remaining bids. That is, $\forall i, \forall b_i$ and $\forall \mathbf{b}_{-i}$ we have: $u_i(b_i^\star = \theta_i, \mathbf{b}_{-i}) \geq u_i(b_i, \mathbf{b}_{-i})$.

Informally, DSIC states that it is the best response for an agent to submit its valuation as its transaction fee.

Myopic Miner Incentive Compatibility (MIC). In TFMs, the miner of block B_k has complete control over the set of transactions to add to B_k (i.e., implement alternate allocation rule over the specified one). To deviate from the intended rule $\mathbf{x}$, a miner typically adds "fake" transactions to the mempool. For the set of fake transactions F (i.e., $F \subset M$) and for any $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ with $\mathcal{H}$, M, and B_k we can write miner's utility u_m as follows [245].

$$u_m(B_k, F) = \sum_{t \in B_k \cap M \setminus F} s_t \cdot p_t(\cdot) - \sum_{t \in B_k \cap F} s_t \cdot q_t(\cdot). \tag{3.9}$$

The first term represents the miner's revenue, and the second term represents the fee burned from the miner's fake transactions. The miner performs the following optimization to maximize its utility: $B_k = \{t \in M \mid x_t = 1\}$.

$$\begin{cases} \max_{\mathbf{x}'} & \sum_{t \in B_k \cap M \setminus F} x_t \cdot s_t \cdot p_t(\cdot) - \sum_{t \in B_k \cap F} x_t \cdot s_t \cdot q_t(\cdot) \\ \text{s.t.} & \sum_{t \in M} s_t \cdot x_t \leq C \text{ and } x_t(\mathcal{H}, M) \in \{0, 1\}, \forall t \end{cases} \tag{3.10}$$

Given the possibility of a miner's strategic deviation, Roughgarden [245] introduces MIC, as defined next.

Definition 3.23: MIC [245]

We say any $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ with $\mathcal{H}$, M, and B_k is MIC, if a miner maximizes u_m (Eq. 3.10) by not creating any fake transactions, $F = \emptyset$ and following the rule $\mathbf{x}$.

We denote OPT as the miner's utility from Eq. 3.10 with $p_t = b_t$ and $q_t = 0,\ \forall t \in B_k$. Note that computing the optimal feasible set, say $\mathbf{x}^\star$, in Eq. 3.10 is NP-Hard since it reduces to KNAPSACK auctions [6]. Miners may instead adopt a greedy-based approach [245].

Off-chain Agreement Proof (OCAP). Another important property in the TFM literature is OCAP, which deals with the off-chain collusion with a miner and the bidding agents. More formally,

Definition 3.24: OCAP [245]

Consider a miner m and the set of agents $A = \{i \mid t_i \in M\}$. We say that $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ is OCAP if no collusion between m and any subset of A improves the canonical on-chain outcome.

3.2.4 Popular TFMs and Their Properties

We now summarize some popular TFMs in literature.

First-price (FPA) and Second-price (SPA) Mechanisms. Bitcoin employs a first-price TFM and can be expressed in the language of $\mathcal{T}^{1st} = (\mathbf{x}^{1st}, \mathbf{p}^{1st}, \mathbf{q}^{1st}, \tau^{1st})$. Here, $\mathbf{x}^{1st}$ follows Eq. 3.10. For each $t_i \in B_k$ we have, $p_i^{1st} = b_i$, $q_i^{1st} = 0$ and $\tau^{1st} = \tau_D$. FPA does not satisfy DSIC but satisfies MIC and OCAP [245].

Likewise, w.l.o.g., let $b_1 \geq b_2 \geq b_3, \ldots$. We denote the second-price TFM with $\mathcal{T}^{2nd} = (\mathbf{x}^{2nd}, \mathbf{p}^{2nd}, \mathbf{q}^{2nd}, \tau^{2nd})$. Here, $\mathbf{x}^{2nd}$ follows Eq. 3.10. With $b_{n'}$ as the lowest bid transaction

in B_k, for each $t_i \in B_k$ we have, $p_i^{2nd} = b_{n'}$[12], $q_i^{2nd} = 0$ and $\tau^{2nd} = \tau_D$. SPA approximately satisfies DSIC but does not satisfy MIC and OCAP [245].

EIP-1559 [43]. Denoted with $\mathcal{T}^{1559} = (\mathbf{x}^{1559}, \mathbf{p}^{1559}, \mathbf{q}^{1559}, \tau^{1559})$, in EIP-1559, for each $t_i \in B_k$, we have $p_i^{1559}(\mathcal{H}, B_k) = b_i - \lambda$ where λ is the (dynamic) base fee[13], $q_i^{1559} = \lambda$ and $\tau^{1559} = \tau_D$. The miner maximizes its utility such that $\mathbf{x}^{1559}$ follows Eq. 3.10.

EIP-1559 satisfies DSIC *only* if λ is *not* "excessively low" [246, Def. 5.6]. The base fee λ is excessively low if λ is large enough so that the number of transactions with a valuation greater than λ does not exceed the block size. EIP-1559 also satisfies MIC and OCAP.

BitcoinF [258]. We denote BitcoinF as $\mathcal{T}^B = (\mathbf{x}^B, \mathbf{p}^B, \mathbf{q}^B, \tau^B)$. Each agent i creates two transactions offering δ and $\delta + \hat{b}_i, \hat{b}_i > 0$ as fees. If one gets added, the other is nullified. The allocation rule $\mathbf{x}^B$ splits the block into $\alpha \in (0, 1]$ and $1 - \alpha$ fractions. The miner must first fill the $1 - \alpha$ section through FIFO collecting transactions with δ, after which it can greedily fill the α section. Let C_α and $C_{1-\alpha}$ denote the capacity of the α and $1 - \alpha$ sections, i.e., $C_\alpha + C_{1-\alpha} = C$. For each t_i in the α section, we have $p_i^B = \hat{b}_i + \delta$ and $q_i^B = 0$. Likewise, for each i in the $1 - \alpha$ section, we have $p_i^B = \delta$ and $q_i^B = 0$. Lastly, $\tau^B = \tau_D$.

$$\left.\begin{array}{ll} \max_{\mathbf{x}^B} & \sum_{i \in M} x_i^B \cdot p_i^B(\mathcal{H}, B_k) \cdot s_i \\ \text{s.t.} & \sum_{t \in M, b_t \neq \delta} s_t \cdot x_t^B(\mathcal{H}, M) \leq C_\alpha \\ & \sum_{t \in M, b_t = \delta} s_t \cdot x_t^B(\mathcal{H}, M) = C_{1-\alpha} \text{ and} \\ & x_t^B(\mathcal{H}, M) \in \{0, 1\}, \forall t \in M. \end{array}\right\} \tag{3.11}$$

As a warm-up result to further understand TFMs, we show that strategic miners in $\mathcal{T}^B$ may deviate, i.e., miners may include fake transactions in the $1 - \alpha$ section of the block to increase their utility from the α section. Remark 3.1 captures this result.

[12] Generally, SPAs require agents to pay the highest losing bid. As payments cannot depend on transactions not part of a block, Roughgarden [245] suggests using the lowest winning bid as a proxy.

[13] λ is dynamic and depends on the network congestion. If the block size $> C$, then the congestion is higher, and λ is incremented by 12.5%. If the block size is $\leq C$, λ is decremented by 12.5% [245].

Remark 3.1. BitcoinF ($\mathcal{T}^B$, Eq. 3.11) does not satisfy MIC.

Proof. Consider the following example, where each transaction is of the same size. Let $n = 5$ such that the current block B_k can hold up to 8 transaction. Further, we have $\alpha = 3/4$. The miner must first add (any) 2 transactions to the $1-\alpha$ section before greedily adding transactions to the α section. Whichever transactions from M the miner adds to the $1-\alpha$ section, it can strictly increase its utility by adding 2 fake transactions instead. That is, by adding these fake transactions, the miner can add the real transactions of M to the α section. Thus, BitcoinF's allocation rule does not satisfy MIC. □

PART A: Game-theoretically Sound and Fair Mechanism Design

PART A *of this thesis primarily focuses on game theory and mechanism design and its application in (i) Civic Crowdfunding (CC) and (ii) Transaction Fee Mechanisms (TFMs). In particular, we extend the CC literature by first presenting mechanisms that relax the assumptions placed on the agent's information with [75], resulting in inclusive CC mechanisms. Second, in [76] we look at the efficiency of conducting CC mechanisms over a blockchain that have the potential of eliminating the middle-person and result in a transparent and fair CC platform for the participating agents. Third, we introduce combinatorial CC [77] that studies the simultaneous crowdfunding of multiple public projects under budget-constrained agents. Fourth, we shift our focus on designing game-theoretically sound TFMs that are fair to the agents. More concretely, with [79], we argue that for a TFM to be fair, it must support the inclusion of transactions with zero fees while preserving the monotonicity of bid inclusion. Last, in [78], we introduce Transaction Fee Redistribution Mechanisms (TFRMs), a new class of TFMs using Redistribution Mechanisms (RMs) that look at reducing agents' transaction fees by providing calibrated rebates. In summary, the overarching theme of* PART A *is to contribute to the evolving landscape of e-commerce/e-*

governance applications with a focus on addressing the challenges faced by the participating agents.

Chapter 4

Civic Crowdfunding for Agents with Negative Valuation and Agents with Asymmetric Beliefs

The existing Civic Crowdfunding (CC) literature focuses only on agents with positive valuation and symmetric belief toward the public project's funding (or provision). In this chapter, we present novel mechanisms that break these two barriers, i.e., mechanisms that independently incorporate negative valuation and asymmetric belief. For negative valuation, we present a methodology for converting existing CC mechanisms into mechanisms incorporating agents with negative valuations. Particularly, we adapt existing PPR and PPS mechanisms to present novel PPRN and PPSN mechanisms that incentivize strategic agents to contribute to the project based on their true preferences. With respect to asymmetric belief, we propose a reward scheme Belief Based Reward (BBR) based on Robust Bayesian Truth Serum mechanism. With BBR, we propose a general CC mechanism incorporating asymmetric agents. Again, we leverage PPR and PPS to present PPRx and PPSx. We prove that in PPRx and PPSx, agents with greater belief in the project's provision contribute more than agents with lesser belief in equilibrium. Further,

we also show that the contributions are such that the project is provisioned at equilibrium.

⋆ ⋆ ⋆ ⋆ ⋆

4.1 Introduction

In Chapter 2.3, we introduced Civic Crowdfunding as an effective method for generating funds from agents for public projects (e.g., public parks and libraries, among others). We also discussed PPR (Chapter 2.3.2) and PPS (Chapter 2.3.3), two mechanisms that propose refund schemes to incentivize strategic agents to contribute to the project's funding.

Negative Valuations. Note that in mechanisms like PPR and PPS, only those agents with positive valuations towards the project contribute to its funding (or provision). However, several agents may *prefer* the project not to be provisioned, i.e., their valuations may be negative for the project getting provisioned. For instance, consider the construction of a garbage dump yard in a locality. While the project may be welcomed by several agents, a certain set of agents may wish to relocate the project from its current location to another. In other words, these agents may not *prefer* the construction of the dump yard – in the locality proposed. In such a scenario, the construction of the dump yard (as well as the locality in which it is constructed) must depend on the majority's opinion of it. The CC literature does not address such negative valuations. CC with agents with negative valuations can provide a natural way for *preference aggregation* if addressed.

Asymmetric Belief. Apart from only considering agents with positive valuations, PPR and PPS also assume that apart from knowing the history of contributions, agents do not have any information regarding the provision of the project, i.e., every agent's belief is *symmetric* towards the project's provision. These mechanisms assume that agent belief regarding the project's provision is $\frac{1}{2}$. This chapter relaxes this assumption to incorporate

agents with *asymmetric* beliefs. That is, the agent's belief regarding the project's provision can be $\frac{1}{2} \pm \epsilon$, where $\epsilon > 0$.

Information Structure and Asymmetric Belief. We define an agent's *information structure* as consisting of its valuation and its belief towards the project's provision. Based on their valuation, we categorize these agents as follows: *positive (negative) agents* i.e., agents with positive (negative) valuations or *positive (negative) preferences* towards the project's provision. We also let an agent's belief be asymmetric toward the public project's provision.

4.1.1 Chapter's Contributions

Motivated to break these barriers on an agent's information structure in existing literature for CC, in this chapter, we address these two limitations by (i) handling symmetric agents with negative preferences and (ii) handling positive agents with *asymmetric* belief towards the project's provision, independently. Relaxing both assumptions in one mechanism remains elusive.

CC for Agents with Negative Valuations. To incorporate CC for agents with negative valuations, we require mechanisms that integrate negative agents. For this, we set up two parallel markets, with two different targets – one for the provision, i.e., *provision point* and one against the provision, i.e., *rejection point*, for the project. The project is provisioned (not provisioned) if the provision (rejection) point is reached first. A strategic agent may choose to contribute to a market against its preference. Thus, the challenge in such a setting remains to ingeniously design a refund scheme such that the agents are incentivized to contribute based on their preferences. For this, we propose a methodology through which existing mechanisms for positive preferences also allow for agents with negative preferences, such that agents contribute to the market based on their true preferences. In particular, we adapt existing PPR and PPS mechanisms to design PPRN and PPSN mechanisms.

We prove that in these mechanisms at equilibrium, either the provision or rejection points hold.

CC for Agents with Asymmetric Beliefs. Further, designing mechanisms for CC for agents with *asymmetric* beliefs is not trivial. For instance, a rational agent with significant belief towards the project's provision may choose to free-ride, as it believes that the project will be provisioned regardless of its contribution. Such asymmetric agents need to be *further* incentivized to contribute towards the project's provision. For this, we propose a novel reward scheme *Belief Based Reward* (BBR) that rewards an agent based on their belief towards the project's provision. We deploy a *peer prediction* mechanism for information aggregation of each agent's belief. With BBR, we propose a novel class of mechanisms for civic crowdfunding that incentivizes agents with asymmetric beliefs to contribute towards the provision, such that the project is provisioned at equilibrium.

4.1.2 Chapter Notation and Solution Concepts

Throughout this chapter, we use the notations introduced during the description of CC in Chapter 2.3[1]. We focus on the single-project case, i.e., $m = 1$ in Definition 2.26. Thus, we drop the subscript "j" from this chapter. For instance, agent i contributing x_{ij} to project $j \in [m]$ is simply agent i contributing x_i as there is only one project. Table 4.1 tabulates the notations used in this chapter.

We also build our mechanisms on the existing PPR (Chapter 2.3.2) and PPS (Chapter 2.3.3) ones. Lastly, the solution concepts used are Nash equilibrium (Definition 2.3) and sub-game perfect Nash equilibrium (Definition 2.4).

4.1.2.1 Additional Preliminaries

4.1.2.1.1 Notations To incorporate an agent's negative valuation and asymmetric belief, we introduce additional notations to those described in Chapter 2.3. First, based on

[1]We refer the reader to Chapter 2.3 for a primer on CC with refunds, PPR [312], and PPS [52], including the role of refund schemes and the source of the refund bonus.

their valuations, we categorize agents as follows: positive (negative) agent, i.e., Agent i with $\theta_i \geq 0$ ($\theta_i < 0$) or with a positive (negative) preference towards the public project's provision. Let $\mathcal{P}$ ($\mathcal{N}$) denote the set of all positive (negative) agents, such that $\mathcal{A} = \mathcal{P} \cup \mathcal{N}$. Further, let $\vartheta^1 = \sum_i \theta_i \, \forall i \in \mathcal{P}$ as the total valuation for the public project getting provisioned and $\vartheta^2 = \sum_i (-\theta_i) \; \forall i \in \mathcal{N}$ as the total valuation for the public project not getting provisioned, i.e., $\vartheta = \vartheta^1 - \vartheta^2$.

Second, we also consider agents with asymmetric beliefs towards the project's provision, i.e., agents may believe that the project may be provisioned with probability (belief) $1/2 \pm \epsilon$ or may not be provisioned with probability $1/2 \mp \epsilon$ for some $\epsilon \geq 0$. Let $k_i^1 = (1/2 + \epsilon_i)$ and $k_i^2 = (1/2 - \epsilon_i)$ for some $\epsilon_i \geq 0$ such that $k_i^1 + k_i^2 = 1$, $\forall i \in \mathcal{A}$. Let $\mathcal{A}^+$ ($\mathcal{A}^-$) be the set of agents which believe that the project will (will not) be provisioned, i.e., every agent $i \in \mathcal{A}^+$ ($i \in \mathcal{A}^-$) has belief k_i^1 (k_i^2) that project will be provisioned, such that $\mathcal{A} = \mathcal{A}^+ \cup \mathcal{A}^-$.

4.1.2.1.2 Robust Bayesian Truth Serum We require each agent to truthfully elicit its belief regarding the provision of the public project. Since an agent's opinion (belief) is its private information, we look for mechanisms that incentivize it to elicit its true opinion, i.e., are incentive compatible. Towards this, we make use of peer prediction mechanisms.

Peer prediction mechanisms (PPM) allow for eliciting and aggregating subjective opinions from a set of agents. These are generally deployed when there is no method of verifying an agent's honesty (of their opinion) or ability. In the literature, there are several existing peer prediction mechanisms [196, 146, 163, 82, 235]. For illustrative purposes, in this chapter, we focus on the Robust Bayesian Truth Serum (RBTS) mechanism [296].

While RBTS' properties hold for an arbitrary number of signals, we present the binary version of the mechanism, as we are interested in eliciting an agent's belief towards the provision of the project. In RBTS, each agent is required to submit, from [296]:

1. **Information Report:** Let $f_i = \{0, 1\}$ be Agent i's reported signal.

2. **Prediction Report:** Let $g_i \in [0,1]$ be Agent i's report about the frequency of high signals among the citizens.

Based on the information and prediction reports, RBTS assigns each agent a score. The mechanism, for each Agent i, selects a *reference* agent $j = i + 1$ (modulo n) and a *peer* agent $k = i + 2$ (modulo n) and calculates,

$$g_i' = \begin{cases} g_j + \delta & \text{if } f_i = 1 \\ g_j - \delta & \text{if } f_i = 0 \end{cases}$$

where $\delta = \min(g_j, 1 - g_j)$. Then, the RBTS score for Agent i is given by,

$$RBTS_i = \underbrace{G_m(g_i', f_k)}_{\text{information score}} + \underbrace{G_m(g_i, f_k)}_{\text{prediction score}} . \tag{4.1}$$

$G_m(\cdot)$ is the binary quadratic scoring rule [255] normalized to give scores between 0 and 1 and is a strictly proper scoring rule.

RBTS is IC and ex-post ID ([296, Theorem 10]) for the elicitation of binary information for all $n \geq 3$, without relying on knowledge of the *common prior*. Note that many other peer prediction mechanisms are IC only when $n \to \infty$, so we chose RBTS.

Notation	Definition
$\mathcal{P}_m, m = 1$	Set of Projects to be crowdfunded
$\mathcal{A} = \{1, \ldots, n\}$	Set of Agents
$\theta_i \in \mathbb{R}$	Agent i's valuation for the project as $m = 1$
$\vartheta = \sum_{i \in \mathcal{A}} \theta_i$	Overall valuation of the project
$c \in \mathbb{R}_{\geq 0}$	Target cost of the project
$\tau \in \mathbb{R}_{\geq 0}$	Deadline of the project
$x_i \in \mathbb{R}_{\geq 0}$	Agent i's contribution to the project
$t_i \in \mathbb{R}_{\geq 0}$	Agent i's time of contribution
$\mathbf{x} = \sum_{i \in \mathcal{A}} x_i$	Total contribution to the project
$B > 0$	Refund bonus
$u_i(\theta_i, x_i; \mathbf{x}, c, B)$	Agent i's utility
P	Set of Agents with $\theta \geq 0$
N	Set of Agents with $\theta < 0$
$\vartheta^1 = \sum_{i \in P} (\theta_i)$	Overall positive valuation
$\vartheta^2 = \sum_{i \in N} (-\theta_i)$	Overall negative valuation
k_i^1	Agent i's belief regarding project's funding
k_i^2	Agent i's belief regarding project not being funded
$\mathcal{A}^+$	Set of agents s.t. $\forall i$, $k_i^1 \geq 1/2$
$\mathcal{A}^-$	Set of agents s.t. $\forall i$, $k_i^1 < 1/2$
b_i	Belief Based Reward for Agent i

Table 4.1: Chapter Notations

4.2 Civic Crowdfunding for Agents With Negative Valuations

We now introduce our methodology through which CC mechanisms can incorporate *symmetric* agents with negative preference (valuation) towards the public project's provision. For this, the PM sets up two separate markets, i.e., one for the provision and one against the provision of the project. Thus, agents now have a greater scope for manipulation. In such a setting, a strategic agent may choose to contribute to a market against its preference if its expected utility for contributing in that market is more than if it contributes to the market based on its preference. Therefore, to incorporate agents with negative preferences, we must ingeniously construct the refund scheme to incentivize agents to contribute to the market based on their true preferences.

To illustrate this methodology, we provide two mechanisms for the same by adopting existing mechanisms in PPR and PPS, namely, PPRN and PPSN. The reason for introducing PPRN and PPSN is that these mechanisms work in different settings. PPRN (based on PPR (Chapter 2.3.2)) is for the offline CC setting, whereas PPSN (based on PPS (Chapter 2.3.3)) is for the online CC setting. For these mechanisms, let c_1 (c_2) be the target for provision (rejection) of the public project with $\mathbf{x}_1$ ($\mathbf{x}_2$) as the total funding received towards (against) its provision.

4.2.1 Provision Point Mechanism with Refunds with Negative Preference (PPRN)

We now propose a mechanism for civic crowdfunding for agents with negative valuation by leveraging PPR, namely PPRN.

4.2.1.1 Protocol

In PPRN, Agent i not only contributes its contribution x_i but also specifies its preference, i.e., whether it wants to contribute towards the project getting provisioned or not getting provisioned. Let $\beta_i \in \{1, 2\}$, $\forall i \in \mathcal{A}$ be a private preference variable for Agent i, such that $\beta_i = 1, \forall i \in \mathcal{P}$ and $\beta_i = 2, \forall i \in \mathcal{N}$. Further, let Agent i's reported preference be $\tilde{\beta}_i$. The social planner adds Agent i's contribution towards the project's provision or rejection based on its reported preference $\tilde{\beta}_i$. The project is provisioned or not based on whichever target is first reached[2].

4.2.1.2 Agent Utility

The utility for Agent $i \in \mathcal{A}$ with $\tilde{\beta}_i = 1$ in PPRN is as follows. The utility is similar to the PPR utility (Eq. 2.15) with the addition of the two markets.

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x_1} \geq c_1} \cdot (\theta_i - x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x_1} < c_1} \cdot \left(\frac{x_i}{\mathbf{x}_1 + \mathbf{x}_2} \right) B}_{\text{Unfunded Utility}} \tag{4.2}$$

Similarly, the utility for Agent $i \in \mathcal{A}$ with $\tilde{\beta}_i = 2$ in PPRN is as follows,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x_2} \geq c_2} \cdot (-x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x_2} < c_2} \cdot \left(\theta_i + \left(\frac{x_i}{\mathbf{x}_1 + \mathbf{x}_2} \right) B \right)}_{\text{Unfunded Utility}} \tag{4.3}$$

4.2.1.3 PPRN: Equilibrium Analysis

The equilibrium analysis for PPRN follows similarly to that of PPR (refer to Chapter 2.3.2.1). Consider the following theorem.

[2] As our goal is to fund *one* of the two projects under crowdfunding, choosing to stop the process as soon as either the provision or the rejection point is reached seems a natural choice. Moreover, in practice, it is likely that the point with more aggregate valuation will reach faster, and thus, this choice will also mimic the majority vote.

Theorem 4.1. *Given* $\mathcal{A}, m = 1, c_1$ *and* c_2 *(refer to Definition 2.26) with* $0 < B \leq \frac{(c_1+c_2)(\vartheta^1-c_1)}{c_1}$, $0 < B \leq \frac{(c_1+c_2)(\vartheta^2-c_2)}{c_2}$ *and agent utility structure as defined in Eq. 4.2 and Eq. 4.3, then at equilibrium (i) either* $\mathbf{x}_1 = c_1$ *or* $\mathbf{x}_2 = c_2$ *holds, (ii)* $\tilde{\beta}_i = \beta_i$, $\forall i \in \mathcal{A}$ *and (iii) the set of (pure-strategy) Nash equilibria are* $\{x_i^\star \mid x_i^\star \leq \left(\frac{c_1+c_2}{B+c_1+c_2}\right)|\theta_{ij}|,\ \forall i \in \mathcal{A}\}$.

Proof. In Step 1, we show that the equilibrium contributions are such that at equilibrium, either $\mathbf{x}_1 = c_1$ or $\mathbf{x}_2 = c_2$ holds. Step 2 shows that each agent delays its contribution till the deadline τ. We prove that every agent contributes based on its true preference in Step 3. Step 4 calculates the equilibrium contribution of every agent. Finally, in Step 5, we give the conditions for the existence of Nash Equilibrium.

Step 1: As $\vartheta^1 > c_1$ and $\vartheta^2 > c_2$, at equilibrium $\mathbf{x}_1 < c_1$ and $\mathbf{x}_2 < c_2$ cannot hold, as $\exists i \in \mathcal{P}$ and $\exists j \in \mathcal{N}$ with $x_i < \theta_i$ and $x_j < |\theta_j|$, at least, that could obtain a higher refund bonus by marginally increasing its contribution because of $B > 0$. Likewise, any agent with a positive contribution could gain in utility by marginally decreasing its contribution if $\mathbf{x}_1 > c_1$ and $\mathbf{x}_2 > c_2$. Thus, at equilibrium, either $\mathbf{x}_1 = c_1$ or $\mathbf{x}_2 = c_2$ holds.

Step 2: As the refund in PPRN is independent of time, no agent has any incentive to contribute early. Thus, each agent delays its contribution till the deadline, τ.

Step 3: Since every Agent i is symmetric in its belief towards the project's provision, its expected utility is given by $\frac{1}{2}\left(\theta_i - x_i + \frac{x_i}{\mathbf{x}_1+\mathbf{x}_2}B\right)$ – irrespective of which of the two project it contributes to. Thus, each Agent i has no incentive to deviate from its preference. That is, $\tilde{\beta}_i = \beta_i, \forall i \in \mathcal{A}$.

Step 4: We split this into categories depending on an agent's valuation type.

1. *For Positive agents:* At equilibrium, the *best response* for an Agent $i \in \mathcal{P}$ is that contribution $x_i^\star$ such that its *provisioned utility is not less than not provisioned utility* since it prefers the project to be provisioned and is symmetric in its belief, i.e., $\forall i \in \mathcal{P}$,

$$\theta_i - x_i^\star \geq \left(\frac{x_i^\star}{\mathbf{x}_1 + \mathbf{x}_2}\right) B$$

$$\Rightarrow x_i^\star \leq \left(\frac{(\mathbf{x}_1+\mathbf{x}_2)}{B+\mathbf{x}_1+\mathbf{x}_2}\right)\theta_i < \left(\frac{c_1+c_2}{B+c_1+c_2}\right)\theta_i,$$

The last inequality follows from the fact that $\mathbf{x}_1+\mathbf{x}_2 < c_1+c_2$.

2. *For Negative agents:* The *best response* for an Agent $i \in \mathcal{N}$ is that equilibrium contribution $x_i^\star$ such that its *not provisioned utility is not less than provisioned utility* since the agent prefers the project not to be provisioned and is symmetric in its belief, i.e., $\forall i \in \mathcal{N}$,

$$|\theta_i| + \left(\frac{x_i^\star}{\mathbf{x}_1+\mathbf{x}_2}\right)B \leq -x_i^*$$

$$\Rightarrow x_i^\star \leq \left(\frac{(\mathbf{x}_1+\mathbf{x}_2)}{B+\mathbf{x}_1+\mathbf{x}_2}\right)|\theta_i| < \left(\frac{c_1+c_2}{B+c_1+c_2}\right)|\theta_i|$$

Step 5: Summing up $x_i^\star, \forall i \in \mathcal{P}$ gives the condition for existence of Nash Equilibrium as,

$$0 < B < \frac{(c_1+c_2)(\vartheta^1-c_1)}{c_1}.$$

Similarly, summing up $x_i^\star, \forall i \in \mathcal{N}$,

$$0 < B < \frac{(c_1+c_2)(\vartheta^2-c_2)}{c_2}.$$

gives the condition for the existence of Nash Equilibrium. □

Next, leveraging PPS, we present *Provision Point Mechanism for Securities with Negative Preference* (PPSN).

4.2.2 Provision Point Mechanism for Securities with Negative Preference (PPSN)

We now propose a mechanism for civic crowdfunding for agents with negative valuation by leveraging PPS, namely PPSN.

4.2.2.1 Protocol

In PPSN, we consider a mechanism with two *independent* PPS prediction markets - PPS1 and PPS2. In PPS1, agents contribute for the project to be provisioned (and buy negative securities); in PPS2, agents contribute for the project not to be provisioned (and buy positive securities). Note that the markets are independent, and the prices in both markets are also independent of the other. The provision point for PPS1 is reached when its total contribution reaches c_1, and the rejection point for PPS2 when the total contribution reaches c_2. Let $\mathbf{x}_1$ be the total contribution received by the project in PPS1 and $\mathbf{x}_3$ be the total contribution received by the project in PPS2. The project is provisioned or not based on whichever target is first reached. Let $\beta_i \in \{1, 2\}$, be a private preference variable for Agent i, such that $\beta_i = 1, \forall i \in \mathcal{P}$ and $\beta_i = 2, \forall i \in \mathcal{N}$.

4.2.2.2 Common Refund Scheme

An agent may not contribute to the market based on its preference if its expected refund is higher in case it contributes to the other market. To prevent this, we present a *common refund scheme* that ensures that the agent obtains the same refund despite which market it chooses to contribute. In this, Agent i contributes x_i in any market based on a refund that depends on the minimum of the issued securities present in both the markets[3], i.e., $Q^{t_i} = \min(q^{t_i}_{PPS1}, q^{t_i}_{PPS2})$. Based on this, Agent i is issued securities ($s^{t_i}_i$) for a contribution x_i given by

$$s^{t_i}_i = C_0^{-1}(x_i + C_0(Q^{t_i})) - Q^{t_i},$$

from [52, Eq. 6]. Thus, Agent i's refund in this scheme is $s^{t_i}_i - x_i$. However, the number of issued securities only changes for the market in which the agent contributes x_i to, i.e.,

$$C_0^{-1}(x_i + C_0(q^t_{PPS(\beta_i)})) - q^t_{PPS(\beta_i)},$$

will be the change in the total number of issued securities in the market PPS(β_i).

[3]For more details on PPS, please refer to Chapter 2.3.3.

Now, consider the following propositions based on this setup.

Proposition 4.1. *The securities allotted to an agent with total issued securities as Q^t is always greater than or equal to those it would have received with securities q^t_{PPS1} or q^t_{PPS2} for the same contribution and the same cost function C_0.*

Proof. The statement follows directly from the fact that the number of securities allotted for the same contribution is a decreasing function of the total issued securities [52, Step-2 (Theorem 3)]. □

Proposition 4.2. *The refund given by $s_i^{t_i} - x_i$ for Agent i is a decreasing function with respect to time t_i.*

Proof. The securities allotted to Agent i, $s_i^{t_i}$, decreases as Q^{t_i} increases (Proposition 4.1). Further, since $Q^t = \min(q^t_{PPS1}, q^t_{PPS2})$ and q^t_{PPS1} and q^t_{PPS2} are non-decreasing with respect to time t; Q^t is a non-decreasing function of time. Thus, $s_i^{t_i} - x_i$ for Agent i is a decreasing function with respect to time t_i. □

4.2.2.3 Agent Utility

For simplicity of notations, let us call PPS1 as p_1 and PPS2 as p_2. Thus, $p_{\beta_i} = p_1, \forall i \in \mathcal{P}$ and $p_{\beta_i} = p_2, \forall i \in \mathcal{N}$. Further, let the market in which Agent i contributes be $\tilde{p}_{\beta_i}$. The utility for Agent $i \in \mathcal{A}$ with $\tilde{p}_{\beta_i} = p_1$, in PPSN is as follows, where x_i is the contribution at time t_i,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x_1} \geq c_1} \cdot (\theta_i - x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x_1} < c_1} \cdot \left(s_i^{t_i} - x_i\right)}_{\text{Unfunded Utility}} \tag{4.4}$$

The utility for Agent $i \in \mathcal{A}$ with $\tilde{p}_{\beta_i} = p_2$, in PPSN is as follows, where x_i is the contribution at time t_i,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x_2} \geq c_2} \cdot (-x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x_2} < c_2} \cdot (\theta_i + s_i^{t_i} - x_i)}_{\text{Unfunded Utility}} \tag{4.5}$$

4.2.2.4 PPSN: Equilibrium Analysis

The equilibrium analysis for PPS follows similarly to that of PPS (refer to Chapter 2.3.3). Consider the following theorem.

Theorem 4.2. *Consider the civic crowdfunding setup with $\mathcal{A}, m = 1, c_1$ and c_2 (refer to Definition 2.26) and each agent i's utility structure defined by Eq. 4.4 and Eq. 4.5. Let $C : \mathbb{R}^2 \to \mathbb{R}$ be a cost function satisfying [52, CONDITIONS 1-4, 6]. Let $C_0^1 : \mathbb{R} \to \mathbb{R}$ be the cost function obtained from C by fixing the number of positive outcome securities, satisfying Property 2.1, $\vartheta^1 > (C_0^1)^{-1}(c_1 + C_0(0))$ and used in the market p_1. Likewise, $C_0^2 : \mathbb{R} \to \mathbb{R}$ be the cost function obtained from C by fixing the number of negative outcome securities, satisfying Property 2.1, $\vartheta^2 > (C_0^2)^{-1}(c_2 + C_0(0))$ and used in the market p_2. Then (i) either $\mathbf{x}_1 = c_1$ or $\mathbf{x}_2 = c_2$ holds, (ii) $\tilde{\beta}_i = \beta_i$, $\forall i \in \mathcal{A}$ and (iii) the set of sub-game Perfect Nash equilibria are $\{(x_i^\star, a_i) \mid x_i^\star \leq C_0^{\beta_i}(|\theta_i| + q_{a_i} - C_0^{\beta_i}(q_{a_i}),\ \forall i \in \mathcal{A}\}$.*

Proof. In Step 1, we show that the equilibrium contributions are such that at equilibrium, either $\mathbf{x}_1 = c_1$ or $\mathbf{x}_2 = c_2$ holds. Step 2 shows that each agent delays its contribution till the deadline τ. We prove that every agent contributes based on its true preference in Step 3. Step 4 calculates the equilibrium contribution of every agent. Finally, in Step 5, we give the conditions for the existence of Nash Equilibrium.

Step 1: As $\vartheta^1 > c_1$ and $\vartheta^2 > c_2$, at equilibrium $\mathbf{x}_1 < c_1$ and $\mathbf{x}_2 < c_2$ cannot hold, as $\exists i \in \mathcal{P}$ and $\exists j \in \mathcal{N}$ with $x_i < \theta_i$ and $x_j < |\theta_j|$, at least, that could obtain a higher refund bonus by marginally increasing its contribution since $\frac{\partial s_i^{t_i}}{\partial x_i} > 1$ [52, CONDITION 7]. Likewise, any agent with a positive contribution could gain in utility by marginally decreasing its contribution if $\mathbf{x}_1 > c_1$ or $\mathbf{x}_2 > c_2$. Thus, at equilibrium, $\mathbf{x}_1 = c_1$ or $\mathbf{x}_2 = c_2$ holds.

Step 2: Since every Agent i is symmetric in its belief towards the project's provision, its expected utility is given by $1/2(\theta_i + s_i) - x_i$ for both the markets. Thus, every Agent i has

no incentive to deviate from its preference. Therefore, $\tilde{p}_{s_i} = p_{s_i}, \forall i \in \mathcal{A}$.

Step 3: As the refund scheme is decreasing with respect to time t (Proposition 4.2), Agent i contributes as soon as it arrives, i.e., at time a_i.

Step 4: Let $q_{p_1}^{a_i}$ be the number of total issued securities at market p_1 at time a_i, and $q_{p_2}^{a_i}$ be the number of total issued securities at market p_2 at time a_i, with $Q^{a_i} = \min(q_{p_1}^{a_i}, q_{p_2}^{a_i})$ for Agent i. Now,

1. *For Positive agents:* At equilibrium, the *best response* for an Agent $i \in \mathcal{P}$ is that contribution $x_i^\star$ in market p_1 at time a_i such that its *provisioned utility is not less than not provisioned utility* since it prefers the project to be provisioned and is symmetric in its belief, i.e.,

$$\theta_i - x_i^\star \geq s_i^\star - x_i^\star$$

$$\theta_i \geq s_i^\star$$

$$\Rightarrow x_i^\star \leq C_0(\theta_i + Q^{a_i}) - C_0(Q^{a_i}) \ \forall i \in \mathcal{P}$$

   The result follows from [52, Eq. 6]. Based on this $x_i^\star$, the number of issued securities changes by $C_0^{-1}(x_i^\star + C_0(q_{p_1}^{a_i})) - q_{p_1}^{a_i}$ in p_1, since a positive agent always contributes in p_1.

2. *For Negative agents:* At equilibrium, the *best response* for an Agent $i \in \mathcal{N}$ is that contribution $x_i^\star$ at time a_i in market p_2 such that its *not provisioned utility is not less than provisioned utility* since it prefers the project not to be provisioned and is symmetric in its belief i.e.,

$$s_i^\star - x_i^\star + |\theta_i| \leq -x_i^\star$$

$$s_i^\star \leq |\theta_i|$$

$$\Rightarrow x_i^\star \leq C_0(|\theta_i| + Q^{a_i}) - C_0(Q^{a_i}) \ \forall i \in \mathcal{N}$$

   The result follows from [52, Eq. 6]. Based on this $x_i^\star$, the number of issued securities changes by $C_0^{-1}(x_i^\star + C_0(q_{p_2}^{a_i})) - q_{p_2}^{a_i}$ in p_2, since a negative agent always contributes in p_2.

Step 4: From Proposition 4.1, $\theta_i \geq s_i^\star$ can be written as $\theta_i \geq s_i^\star \geq C_0^{-1}(x_i^\star + C_0(q_{p_1}^{a_i})) - q_{p_1}^{a_i}$. Summing up $\forall i \in \mathcal{P}$, we get the condition for the existence of Nash Equilibrium here as, from [52, Eq. 7],

$$C_0^{-1}(c_1 + C_0(0)) < \vartheta^1.$$

Similarly for p_2, $\forall i \in \mathcal{N}$ we have,

$$C_0^{-1}(c_2 + C_0(0)) < \vartheta^2.$$

Step 5: For Agent j entering last, if $\mathbf{x}_1 = c_1$ or $\mathbf{x}_2 = c_2$, then its best response is contributing 0. If $\mathbf{x}_1 < c_1$ and $\mathbf{x}_2 < c_2$, irrespective of the total contribution, its provision, and the not provisioned utility is the same at $x_j^\star$, defined in the theorem, and it is the best response for Agent j to follow the equilibrium strategy. With backward induction, by similar reasoning, it is the best response for every agent to follow the equilibrium strategy irrespective of the history of the contributions.

For Agent $j \in \mathcal{P}(j \in \mathcal{N})$ entering the market, in the first case, when the amount left to fund the project is less than its equilibrium upper bound, then it will fund the remaining contribution. This is because, for this contribution, its provisioned utility will be greater than its not provisioned utility. Agent j will also contribute the upper bound if it can since its not provisioned utility increases as its contribution increases. Therefore, contributing an amount less than the bound will result in a lesser, not-provisioned utility for the agent. Thus, these strategies also form a set of sub-game perfect equilibria. □

Discussion: For PPSN, it can be seen that $\vartheta = \vartheta^1 - \vartheta^2$. The project is always provisioned if $\vartheta^1 > c_1$ and $\vartheta \geq 0$ or is never provisioned if $\vartheta^2 > c_2$ and $\vartheta < 0$. Here, it must be noted that $\vartheta^1 > c_1$ and $\vartheta^2 > c_2$ can be simultaneously satisfied. In that case, if $\vartheta^1 > \vartheta^2$, the project attains the provision point faster than the rejection point and vice-versa.

The significance of this result is that in PPSN (and PPRN), at equilibrium, the project is provisioned if the majority *prefers* it, i.e., only when $\vartheta \geq 0$. Thus, this methodology

allows for truthful aggregation of the private preferences of each agent with respect to public projects.

4.3 Civic Crowdfunding for Agents with Asymmetric Beliefs

We now present a *General Mechanism* which incentivizes agents with asymmetric beliefs towards the public project's provision, i.e., contributing towards it. We *restrict* our attention to the case where every agent has a *positive* valuation towards the project's provision.

The General Mechanism involves two phases: a *Belief Phase* (BP) and a *Contribution Phase* (CP). In BP, each Agent i submits its belief for the provision of the project for which it is allocated some share (denoted by b_i) of the reward calculated through the Belief Based Reward (BBR) scheme (described shortly). In CP, each Agent i submits its contribution (x_i) to the project, which depends on the refund obtained in the BP and the provision point mechanism deployed for civic crowdfunding.

The mechanism requires two separate bonuses for both phases, which the social planner announces at the start of the project. Let $B^B(B^C)$ be the bonus allocated for the BP and the CP, respectively. Further, let $a_i^1(a_i^2)$ be the time at which Agent i arrives at the mechanism for the BP (CP) with $t_i^1(t_i^2)$ as the time at which it reports its belief (contribution). Let the BP (CP) deadline be $\tau^B(\tau^C)$ announced at the start of the project.

Unlike in the case of civic crowdfunding for agents with symmetric beliefs, an asymmetric agent with significant belief towards the project getting provisioned or not may choose to free-ride and not contribute. Therefore, we introduce a reward scheme that further incentivizes such agents to contribute towards the project.

4.3.1 Belief Based Reward (BBR)

To quantitatively measure the reward share to be distributed to every contributing agent in the BP, we use a peer-prediction mechanism, say $\mathcal{M}$. We consider PPMs that incentivize truthful elicitation of an agent's belief, i.e., PPMs that are IC.

Let the score of Agent i dependent on its belief $(1/2 \pm \epsilon_i)$ be M_i. Further, let S^{t_i} be the set consisting of all the agents that have reported their belief, including the Agent i, who reports its belief at time t_i. For τ^B as the deadline, S^{τ^B} consists of all the agents that have reported their belief. Let $M_i, \forall i \in S^{\tau^B}$ be the agent scores calculated after the deadline. For,

$$w_i = \frac{M_i}{\sum_j M_j} \ \forall j \in S^{t_i},$$

Agent i's reward in the scheme is,

$$b_i = \begin{cases} \frac{w_i}{\sum_j w_j} \times B^B & \forall j \in \mathcal{A}^+; \quad \forall i \in \mathcal{A}^+ \\ \frac{w_i}{\sum_j w_j} \times B^B & \forall j \in \mathcal{A}^-; \quad \forall i \in \mathcal{A}^- \end{cases} \tag{4.6}$$

We refer to the reward scheme given by Eq. 4.6 as Belief-Based Reward (BBR). With this, we show the following proposition:

Proposition 4.3. *BBR is a decreasing function of time.*

Proof. From Eq. 4.6, BBR is inversely proportional to the order in which agents report their beliefs. Since agents' arrival is non-decreasing w.r.t. time, BBR is a decreasing function of time. □

In addition, BBR is also *budget balanced*, i.e., in BBR, the entire budget is utilized. At the end of the mechanism, only one set of agents, either $\mathcal{A}^+$ or $\mathcal{A}^-$, are rewarded.

RBTS Reward Scheme: For illustration, we use RBTS as our peer-prediction mechanism to calculate M_i for each Agent i. For this, every agent submits its prediction and information report as described earlier. In this reward scheme, let $f_i = 0$ denote that Agent

i believes that the project will be provisioned and $f_i = 1$ denote that Agent i believes that the project will not be provisioned. Thus, through each agent's prediction report, the PM knows whether an agent belongs to the set $\mathcal{A}^+$ or the set $\mathcal{A}^-$.

We now present *Provision Point Mechanism with Refunds for Agents with Asymmetric Beliefs* (PPRx) by plugging the PPR refund bonus scheme for the CP.

4.3.2 Provision Point Mechanism with Refunds for Agents with Asymmetric Beliefs (PPRx)

We plugin the PPR refund bonus scheme for the Contribution Phase in this mechanism.

4.3.2.1 Agent Utility

The utility for Agent $i \in \mathcal{A}^+$, in PPRx is as follows,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x} \geq c} \cdot (\theta_i - x_i + b_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x} < c} \cdot \left(\left(\frac{x_i}{\mathbf{x}} \right) B^C \right)}_{\text{Unfunded Utility}} \tag{4.7}$$

Similarly, the utility for Agent $i \in \mathcal{A}^-$, in PPRx are as follows,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x} \geq c} \cdot (\theta_i - x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x} < c} \cdot \left(\left(\frac{x_i}{\mathbf{x}} \right) B^C + b_i \right)}_{\text{Unfunded Utility}} \tag{4.8}$$

4.3.3 PPRx: Equilibrium Analysis

Consider the following theorem.

Theorem 4.3. *Given $\mathcal{A}, m = 1$ and c as the target cost (refer to Definition 2.26) with $0 < B^B \leq \vartheta - c$, $B^B, B^C > 0$ and agents utility structure as defined in Eq. 4.7 and Eq. 4.8, then at equilibrium (i) $\mathbf{x} = c$ and (ii) the set of (pure-strategy) Nash equilib-*

ria are $\{(x_i^\star, a_i^1, a_i^2) \mid x_i^\star \leq \left(\frac{k_i^1\theta_i + k_i^1 b_i}{k_i^2 B^C + k_i^1 c}\right) c,\ \forall i \in \mathcal{A}^+\}$ *and the set* $\{(x_i^\star, a_i^1, a_i^2) \mid x_i^\star \leq \left(\frac{k_i^2\theta_i - k_i^1 b_i}{k_i^1 B^C + k_i^2 c}\right) c,\ \forall i \in \mathcal{A}^-\}$.

Proof. In Step 1, we show that the equilibrium contributions are such that at equilibrium $\mathbf{x} = c$ holds. We prove that every agent reports its belief as soon as it arrives at the Belief Phase in Step 2. Step 3 calculates the equilibrium contributions of every agent. Finally, in Step 4, we give the conditions for the existence of Nash Equilibrium.

Step 1: At equilibrium, $\mathbf{x} = c$, since the social planner stops the protocol as soon as the provision point is reached. Further, if $\mathbf{x} < c$, then an agent can increase its utility by contributing more to the project and receiving a higher utility since $B^C > 0$. Therefore, the contributions are such that the market is provisioned at equilibrium.

Step 2: Since the reward scheme for the Belief Phase is a decreasing function of time (Proposition 4.3), a rational Agent i would report its belief as soon as it arrives at the phase, i.e., at time a_i^1.

Step 3: The equilibrium strategy for each agent $i \in \mathcal{A}$ is such that its *provisioned utility is not less than its not provisioned utility.* Now,

1. For agent $i \in \mathcal{A}^+$:

$$k_i^1(\theta_i - x_i + b_i) \geq k_i^2 \left(\frac{x_i}{\mathbf{x}} B^C\right)$$

$$\Rightarrow x_i^\star \leq \left(\frac{k_i^1\theta_i + k_i^1 b_i}{k_i^2 B^C + k_i^1 c}\right) c.$$

   Since at equilibrium $\mathbf{x} = c$.

2. For agent $i \in \mathcal{A}^-$:

$$k_i^2(\theta_i - x_i) \geq k_i^1 \left(\frac{x_i}{\mathbf{x}} B^C + b_i\right)$$

$$\Rightarrow x_i^\star \leq \left(\frac{k_i^2\theta_i - k_i^1 b_i}{k_i^1 B^C + k_i^2 c}\right) c.$$

   Since at equilibrium $\mathbf{x} = c$.

Step 4:Note that from Eq. 4.7, $x_i^\star \leq \theta_i + b_i \ \forall i \in \mathcal{A}^+$ and from Eq. 4.8, $x_i^\star \leq \theta_i \ \forall i \in \mathcal{A}^-$. Thus,

$$\sum_{i \in \mathcal{A}} x_i^* \leq \sum_{i \in \mathcal{A}^+} (\theta_i + b_i) + \sum_{i \in \mathcal{A}^-} (\theta_i)$$

$$\sum_{i \in \mathcal{A}} x_i^* \leq \vartheta + B^B.$$

At equilibrium we have $\sum_{i \in \mathcal{A}} x_i^\star = c$. Therefore,

$$c \leq \vartheta + B^B,$$

is the condition for the existence of Nash Equilibrium. □

4.3.4 Provision Point Mechanism with Securities for Agents with Asymmetric Beliefs (PPSx)

We now present *Provision Point Mechanism with Securities for Agents with Asymmetric Beliefs* (PPSx) by plugging the PPS refund bonus scheme for the CP.

4.3.4.1 Agent Utility

The utility for Agent $i \in \mathcal{A}^+$, in PPSx is as follows,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x} \geq c} \cdot (\theta_i - x_i + b_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x} < c} \cdot \left(s_i^{t_i^2} - x_i \right)}_{\text{Unfunded Utility}} \tag{4.9}$$

Similarly, the utility for Agent $i \in \mathcal{A}^-$, in PPSx is as follows,

$$u_i(\cdot) = \underbrace{\mathbb{1}_{\mathbf{x} \geq c} \cdot (\theta_i - x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x} < c} \cdot \left(s_i^{t_i^2} - x_i + b_i \right)}_{\text{Unfunded Utility}} \tag{4.10}$$

4.3.5 PPSx: Equilibrium Analysis

Consider the following theorem.

Theorem 4.4. *Consider the civic crowdfunding setup with $\mathcal{A}, m = 1$ and c as the target cost (refer to Definition 2.26) and each agent i's utility structure defined by Eq. 4.9 and Eq. 4.10. Let $C : \mathbb{R}^2 \to \mathbb{R}$ be a cost function satisfying [52, CONDITIONS 1-4, 6]. Let $C_0 : \mathbb{R} \to \mathbb{R}$ be the cost function obtained from C by fixing the number of positive outcome securities, satisfying Property 2.1, $\vartheta > C_0^{-1}(c_1 + C_0(0))$. Then (i) $\mathbf{x} = c$ holds and (ii) the set of sub-game perfect Nash equilibria are $\{(x_i^\star, a_i^1, a_i^2) \mid x_i^\star \leq C_0(\theta_i + b_i + q^{a_i^2}) - C_0(q^{a_i^2}),\ \forall i \in \mathcal{A}^+\}$ and the set $\{(x_i^\star, a_i^1, a_i^2) \mid x_i^\star \leq C_0(\theta_i - b_i + q^{a_i^2}) - C_0(q^{a_i^2}),\ \forall i \in \mathcal{A}^-\}$.*

Proof. In Step 1, we show that the equilibrium contributions are such that at equilibrium $\mathbf{x} = c$ holds. We prove that every agent reports its belief as soon as it arrives to the Belief Phase as well as contributes as soon as it arrives at the Contribution Phase in Step 2. Step 3 calculates the equilibrium contributions of every agent. In Step 4, we give the conditions for the existence of Nash Equilibrium. We show that this set of strategies is sub-game perfect in Step 5.

Step 1: At equilibrium $\mathbf{x} < c$ cannot hold, as $\exists i \in \mathcal{A}^+$ with $x_i < \theta_i + b_i$ or $\exists i \in \mathcal{A}^-$ with $x_i < \theta_i$, at least, that could obtain a higher refund bonus by marginally increasing its contribution, since $\frac{\partial s_i^{t_i}}{\partial x_i} > 1$ [52, CONDITION 7]. Likewise, any agent with a positive contribution could gain in utility by marginally decreasing its contribution if $\mathbf{x} > c$. Thus, at equilibrium $\mathbf{x} = c$.

Step 2: Since the reward scheme for the Belief Phase is a decreasing function of time (Proposition 4.3), a rational Agent i would report its belief as soon as it arrives at the BP, i.e., at time a_i^1. Also, since the CP is the PPS mechanism, the best response for any agent is also to contribute as soon as it arrives, i.e., at time a_i^2.

Step 3: From the utility of Agent i in PPSx (Eq. 4.9 and Eq. 4.10), it is clear to see that $\forall i \in \mathcal{A}^+$, $x_i \leq \theta_i + b_i$ (as a strategic agent will not contribute greater than its valuation and the reward it received) and $\forall i \in \mathcal{A}^-$, $x_i \leq \theta_i$. Also, $x_i \geq 0$. Further, the equilibrium

strategy for each Agent $i \in \mathcal{A}$ is such that its *provisioned utility is not less than its not provisioned utility*. Now,

1. *For Agent $i \in \mathcal{A}^+$:*

$$k_i^1(\theta_i + b_i - x_i^*) \geq k_i^2(s_i^\star - x_i^\star)$$
$$\Rightarrow x_i^\star \leq \frac{k_i^1(\theta_i + b_i) - k_i^2(s_i^\star)}{k_i^1 - k_i^2}.$$

   Thus, the maximum value of $x_i^\star$ is,

$$\hat{x}_i^\star = \frac{k_i^1(\theta_i + b_i) - k_i^2(s_i^\star)}{k_i^1 - k_i^2}$$

   We also have $x_i \leq \theta_i + b_i, \forall i \in \mathcal{A}^+$. Therefore, the maximum value should also be less than $\theta_i + b_i$, i.e.,

$$\hat{x}_i^\star \leq \theta_i + b_i \Rightarrow s_i^\star \leq \theta_i + b_i.$$

   This follows from the value of $\hat{x}_i^*$ defined above. To obtain $s_i^\star$ securities at equilibrium, an Agent i's contribution $x_i^\star$ must be

$$\Rightarrow x_i^* \leq C_0(\theta_i + b_i + q^{a_i^2}) - C_0(q^{a_i^2}) \quad \forall i \in \mathcal{A}^+.$$

   The result follows from [52, Eq. 6], i.e., the securities obtained ($s_i^\star$) are monotonic function of the contribution ($x_i^\star$).

2. *For Agent $i \in \mathcal{A}^-$:*

$$k_i^1(s_i^\star - x_i^\star + b_i) \leq k_i^2(\theta_i - x_i^\star).$$

   Similar to (1) of this step, the equilibrium contribution $x_i^\star$ becomes

$$\Rightarrow x_i^\star \leq C_0(\theta_i - b_i + q^{a_i^2}) - C_0(q^{a_i^2}) \quad \forall i \in \mathcal{A}^-.$$

   This follows from $x_i^\star \leq \theta_i, \forall i \in \mathcal{A}^-$ and [52, Eq. 6].

Step 4: We have $x_i^\star \leq \theta_i + b_i \ \forall i \in \mathcal{A}^+$ and $x_i^\star \leq \theta_i \ \forall i \in \mathcal{A}^-$. Thus,

$$\sum_{i \in \mathcal{A}} x_i^\star \leq \sum_{i \in \mathcal{A}^+} (\theta_i + b_i) + \sum_{i \in \mathcal{A}^-} (\theta_i)$$

At equilibrium we have $\sum_{i \in \mathcal{A}} x_i^\star = c$. Therefore,

$$c \leq \vartheta + B^B,$$

is the condition for the existence of Nash Equilibrium.

Step 5: For Agent j entering last, if $\mathbf{x} = c$, then its best response is contributing 0. If $\mathbf{x} < c$, irrespective of the total contribution, its provisioned and not provisioned utility is the same at $x_j^\star$, defined in the theorem, and it is the best response for Agent j to follow the equilibrium strategy. With backward induction, by similar reasoning, it is the best response for every agent to follow the equilibrium strategy irrespective of the history of the contributions. □

4.4 Discussion and Conclusion

4.4.1 PPRx and PPSx: Equilibrium Contribution Analysis

We now compare the equilibrium contribution of both sets of agents in PPRx and PPSx. Towards this, let Agent $i \in \mathcal{A}^+$ and Agent $j \in \mathcal{A}^-$ such that $\theta_i = \theta_j$, $b_i = b_j$, $a_i^2 = a_j^2$ and $k_i^1 = k_j^1$ ($k_i^1 + k_i^2 = 1$). Agent i's belief about the project getting provisioned is k_i^1 and Agent j's k_j^2.

1. *For PPRx*: The difference in the equilibrium contribution of Agent $i \in \mathcal{A}^+$ and Agent $j \in \mathcal{A}^-$ as defined above in PPRx now becomes,

$$x_i^\star - x_j^\star = \left(\frac{k_i^1 \theta_i + k_i^1 b_i}{k_i^2 B^C + k_i^1 c} \right) c - \left(\frac{k_i^2 \theta_i - k_i^1 b_i}{k_i^1 B^C + k_i^2 c} \right) c$$

As the denominator is always positive, we only consider the numerator in the RHS of the above equation. Observe,

$$c(k_i^1 B^C + k_i^2 c)(k_i^1 \theta_i + k_i^1 b_i) -$$
$$c(k_i^2 B^C + k_i^1 c)(k_i^2 \theta_i - k_i^1 b_i) > 0$$

$$\text{as, } B^C \theta_i((k_i^1)^2 - (k_i^2)^2) + B^C b_i((k_i^1)^2 + k_i^1 k_i^2) +$$
$$c b_i((k_i^1)^2 + k_i^1 k_i^2) > 0,$$

since $k_i^1 \geq k_i^2$. Thus, the upper bound of each Agent $i \in \mathcal{A}^+$ is always greater than for an Agent $j \in \mathcal{A}^-$ with the same valuation and belief.

2. *For PPSx*: For Agent $i \in \mathcal{A}^+$ and Agent $j \in \mathcal{A}^-$ as defined above, the equilibrium contribution of Agent i will always be greater than that of Agent j since $C_0(\theta_i + b_i + q^{a_i^2}) > C_0(\theta_i - b_i + q^{a_i^2})$ as $b_i > 0$ and $\frac{\partial s_i^{t_i}}{\partial x_i} > 1$ [52, CONDITION 7].

Thus, for both PPRx and PPSx, the upper bound on the equilibrium contributions $\forall i \in \mathcal{A}^+$ (with k_i^1 as the belief towards project's provision) is greater than the upper bound on the equilibrium contributions $\forall i \in \mathcal{A}^-$ (with k_i^2 as the belief towards project's provision; $k_i^1 + k_i^2 = 1$), for the same valuation and belief.

This implies that agents with greater belief in the project's provision contribute more than agents with lesser belief in it. Thus, BBR (Eq. 4.6 and the utility structure as given by Eqs. 4.7, 4.8 for PPRx and Eqs. 4.9, 4.10 for PPSx, provides a natural way for civic crowdfunding with asymmetric agents such that the project is provisioned at equilibrium.

4.4.2 Setting Up the Markets

For PPSN (PPRN), the social planner must set up two independent PPS (PPR) markets. The provision points for these projects are determined based on the economics of their construction. The rejection point can be similarly determined. For instance, the rejection point for our garbage dump yard example could be the cost of constructing the dump yard

at a different locality. Another method for determining the rejection point could be the cost incurred by the government as a result of the public project not getting provisioned. An instance of this could be the construction of dams. The cost of not setting up the dam, i.e., the rejection point for the project, could be the cost incurred by the government in providing electricity or water to the nearby areas, which they could have achieved through the dam's construction. Note that the amount collected if the project is rejected is at the government's discretion.

The social planner should allocate a reasonable budget for all these mechanisms. Allocating huge budgets may not guarantee the provision/rejection of the projects. In such a case, the agents may prefer to contribute just enough to get substantial refunds. Likewise, allocating insignificant budgets may not incentivize agents to contribute to the market. In PPSN, the cost function, C_0, used to allocate the securities must be the same for both markets. Additional details for setting up the prediction markets and the budget can be found at [52, 53].

4.4.3 Designing Mechanisms for Asymmetric Agents with Negative Valuation

Civic crowdfunding for agents with an information structure consisting of both – their preference and their belief towards the provision of the project is not trivial, as it provides an extra dimension for the agents to manipulate the mechanism. For instance, combining PPSN and PPSx (PPRN and PPRx) will not suffice. An Agent $i \in \mathcal{A}^+$ with $\theta_i \geq 0$, will *always* choose to contribute towards the project not getting provisioned, as it believes that the project will be provisioned anyways, making it eligible for the additional refund bonus. Likewise, an Agent $i \in \mathcal{A}^-$ with $\theta_i < 0$ will always contribute towards the provision of the project. However, an Agent $i \in \mathcal{A}^+$ with $\theta_i < 0$ and an Agent $i \in \mathcal{A}^-$ with $\theta_i \geq 0$ will always contribute as per their true preference.

For instance, this intuitive result can be shown as follows, from Eq. 4.4 and Eq. 4.5 in PPSN with BBR as defined in Eq. 4.6, for an Agent $i \in \mathcal{A}^+$ with $\theta_i \geq 0$, the difference in its expected utility in contributing in both the markets can be given as,

$$k_i^1(\theta_i - x_i + b_i) + k_i^2(s_i - x_i) - k_i^1(\theta_i + s_i - x_i + b_i) - k_i^2(-x_i)$$

$$\Rightarrow (k_i^2 - k_i^1) \cdot (s_i) \leq 0,$$

as $\forall i \in \mathcal{A}^+$, $k_i^1 \geq k_i^2$ and $s_i \geq 0$. Thus, a strategic Agent $i \in \mathcal{A}^+$ with $\theta_i \geq 0$ will always lie about its preference by contributing against the provision of the project. Likewise, an Agent $i \in \mathcal{A}^-$ with $\theta_i < 0$ will always contribute towards the provision of the project. However, an Agent $i \in \mathcal{A}^+$ with $\theta_i < 0$ and an Agent $i \in \mathcal{A}^-$ with $\theta_i \geq 0$ will always contribute as per their true preference.

Thus, the general method and mechanism proposed in this chapter for civic crowdfunding for agents with negative valuation and agents with asymmetric belief, respectively, are insufficient to incentivize every asymmetric agent to contribute as per their true preference. This can be further explored in future work.

4.4.4 Conclusion

This chapter explores the limitations of existing literature on civic crowdfunding. The CC literature restricts the information structure of agents, as it only allows for positive and symmetric agents. We break this barrier on the information structure of an agent by proposing (i) a general methodology for addressing symmetric agents with negative preferences based on which we proposed two mechanisms, PPRN and PPSN, and (ii) a general mechanism for positive agents with asymmetric beliefs based on which we proposed two mechanisms, PPRx and PPSx. Future work can explore the feasibility of combining negative preferences and asymmetric beliefs into one framework for civic crowdfunding.

Chapter 5

Civic Crowdfunding over Blockchain

With Blockchains gaining traction, in this chapter, we look at implementing Civic Crowdfunding (CC) mechanisms in a reliable, transparent, and secure manner with smart contracts (SCs). As discussed, PPR [312] resolves the free-riding problem in CC by giving a refund bonus to the contributing agents if the project is not provisioned. However, Chandra et al. [52] shows that PPR faces a challenge wherein the agents defer their contribution until the deadline. This chapter defines this delaying of contributions as a race condition. To address this, PPS [52] considers the temporal aspects of a contribution. However, PPS is computationally complex and expensive to implement as an SC, and it is sophisticated and difficult to explain to a layperson. This chapter aims to identify all essential properties a refund bonus scheme must satisfy to curb free-riding while avoiding the race condition. We prove Contribution Monotonicity and Time Monotonicity are sufficient conditions for this. We propose three elegant refund bonus schemes satisfying these two conditions, leading to three novel mechanisms for CC - PPRG, PPRE, and PPRP. We show that PPRG is the most cost-effective mechanism when deployed as an SC. We show that under certain modest

assumptions on the valuations of the agents, in PPRG, the project is funded at equilibrium.

⋆ ⋆ ⋆ ⋆ ⋆

5.1 Introduction

Typically, online Civic Crowdfunding (CC) projects are conducted using digital platforms like Kickstarter [156] and GoFundMe [120]. With the advancement of the *blockchain* technology, *smart contracts* (SC) (refer to Chapter 3.2.2.3) now allow for the deployment of Civic Crowdfunding (CC) projects on-chain. Recall that a smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Since a crowdfunding project as an SC is on a trusted, publicly distributed ledger, it is open and auditable. This property makes the agents' contributions and the execution of the payments transparent and anonymous. Besides, there is no need for any centralized, trusted third party, which reduces the cost of setting up the project. WeiFund [290] and Starbase [264] are examples of decentralized crowdfunding platforms on public blockchains like *Ethereum*.

We know that the introduction of the refund bonus is *vital* in mechanisms such as PPR [312] and PPS [52] as it incentivizes agents to contribute, thus avoiding free-riding. Consequently, in this chapter, we aim to abstract out conditions that *refund bonus schemes* should satisfy to avoid free-riding and the race condition. We believe such a characterization would make it easier to explore simpler and computationally efficient CC mechanisms.

5.1.1 Chapter Contributions

Towards this, we introduce *Contribution Monotonicity* (CM) and *Time Monotonicity* (TM). Contribution monotonicity states that an agent's refund should increase with an increase in its contribution. Further, time monotonicity states that an agent's refund

Notation	Definition
$\mathcal{P}_m, m = 1$	Set of Projects to be crowdfunded
$\mathcal{A} = \{1, \ldots, n\}$	Set of Agents
$\theta_i \in \mathbb{R}$	Agent i's valuation for the project as $m = 1$
$\vartheta = \sum_{i \in \mathcal{A}} \theta_i$	Overall valuation of the project
$c \in \mathbb{R}_{\geq 0}$	Target cost of the project
$\tau \in \mathbb{R}_{\geq 0}$	Deadline of the project
$x_i \in \mathbb{R}_{\geq 0}$	Agent i's contribution to the project
$t_i \in \mathbb{R}_{\geq 0}$	Agent i's time of contribution
$\mathbf{x} = \sum_{i \in \mathcal{A}} x_i$	Total contribution to the project
$B > 0$	Refund bonus
$R_i(x_i, t_i; \mathbf{x}, c, B)$	Refund bonus to Agent i
$u_i(\theta_i, x_i; \mathbf{x}, c, B)$	Agent i's utility

Table 5.1: Chapter Notations

should decrease if it delays its contribution. We prove these two conditions are *sufficient* to provision a public project via crowdfunding in a sequential setting at equilibrium and avoid the race condition (Theorem 5.1). We also prove that TM and *weak* CM are also *necessary*, under certain assumptions on equilibrium behavior (Theorem 5.2).

With these theoretical results on CM and TM, we propose three elegant refund bonus schemes that satisfy CM and TM. These schemes are straightforward to explain to a layperson and are computationally efficient to implement as an SC. With these three schemes, we design novel mechanisms for CC, namely *Provision Point mechanism with Refund through Geometric Progression* (PPRG); *Provision Point mechanism with Refund based on Exponential function* (PPRE), and *Provision Point mechanism with Refund based on Polynomial function* (PPRP). We analyze the cost-effectiveness of these mechanisms and PPS

when deployed as SCs and show that PPRG is significantly more cost-effective, i.e., PPRG requires the least amount of capital to set up.

5.1.2 Chapter Notations and Solution Concepts

Throughout this chapter, we use the notations introduced during the description of CC in Chapter 2.3. We focus on the single-project case, i.e., $m = 1$ in Definition 2.26. Thus, we drop the subscript "j" from this chapter. For instance, agent i contributing x_{ij} to project $j \in [m]$ is simply agent i contributing x_i as there is only one project. Table 5.1 tabulates the notations used in this chapter.

We also build our mechanisms on the existing PPR (Chapter 2.3.2) and PPS (Chapter 2.3.3) ones. Lastly, the solution concepts used are Nash equilibrium (Definition 2.3) and sub-game perfect Nash equilibrium (Definition 2.4). We begin our analysis by formally defining the race condition.

5.2 Race Condition in Provision Point Mechanisms

Recall that the refund scheme in PPR, i.e., $\frac{x_i}{\mathbf{x}} \cdot B$ for an agent $i \in \mathcal{A}$, is non-temporal. That is, the refund is independent of the time of contribution. Chandra et al. [52] show that this results in the agents deferring their contributions till the deadline τ (Claim 2.4). We now formally define this "race" to contribute at the deadline.

Definition 5.1: Race Condition

A strategy profile $\sigma^\star = (\sigma_1^\star, \ldots, \sigma_n^\star)$ is said to have a race condition if $\exists S \subseteq \mathcal{A}$ with $|S| > 1$, for which $\forall i \in S$ the strategy $\sigma_i^\star = (x_i^\star, t)$, with $x_i^\star$ as the equilibrium contribution, is the pure-strategy NE (PSNE) of the induced game i.e., $\forall \sigma_i, \forall i \in S$,

$$u_i(\sigma_i^\star, \sigma_{-i}^\star; \theta_i) \geq u_i(\sigma_i, \sigma_{-i}^\star; \theta_i) \text{ where } t \in [\bar{y}, \tau] \text{ s.t. } \bar{y} = \max_{j \in S} a_j.$$

Here, $\sigma_i = (x_i, t_i)\ \forall t_i \in [a_i, \tau]$. One can see that Definition 5.1 gives back the race condition identified in Claim 2.4 for PPR with $S = \mathcal{A}$ and $t = \tau$. Cason et al. [47] conduct real-world experiments and show that the race condition is observable in practice. Moreover, this race to contribute at the deadline is an undesirable property for a refund bonus scheme to possess.

5.3 Desirable Properties of Refund Bonus Schemes

Motivated by the theoretical guarantees of PPR [312] and PPS [52], we look for CC mechanisms with refund bonus schemes in this chapter. In this context, a *desirable* refund bonus scheme should not just restrict the set of strategies so that the project is provisioned at equilibrium. Still, it should also incentivize *greater* and *early* contributions to avoid the race condition from all interested agents. A refund bonus scheme without these would fail in a sequential (web-based) setting, similar to PPR; hence, these are essential for the online provision point mechanism's implementation. We formalize these desirable properties as the following two *conditions* for a refund bonus scheme $R(\sigma)$ where $\sigma = ((x_i, t_i) \mid \forall i \in \mathcal{A})$ such that $x_i \in (0, c]$, $t_i \in [a_i, T]\ \forall i \in \mathcal{A}$ and with budget B.

Condition 5.1 (Contribution Monotonicity)**.** *The refund must always increase with the increase in contribution to incentivize greater contribution, i.e.,* $\forall i \in \mathcal{A}$, $R_i(\sigma) \uparrow$ *as* $x_i \uparrow$. *Further, if* $R_i(\cdot)$ *is a differential in* $x_i\ \forall i$, *then,*

$$\frac{\partial R_i(\sigma)}{\partial x_i} > 0\ \forall t_i. \tag{5.1}$$

Note. If the strict inequality is replaced with $\geq$ in Eq. 5.1, we call it "*weak*" CM.

Condition 5.2 (Time Monotonicity)**.** *The refund must always decrease with the increase in the duration of the project to incentivize early contribution, i.e.,* $R(\sigma)$ *must*

be a monotonically decreasing *function with respect to time* $t_i \in (0, \tau), \forall x_i,\ \forall i \in \mathcal{A}$ *or*

$$\begin{aligned} &R_i(\sigma) \downarrow \ \text{ as } t_i \uparrow \ \text{ and } \exists\ t_i < \tau, \text{and } \Delta t_i \ \text{ s.t.,} \\ &\frac{R_i\left((x_i, t_i + \Delta t_i), \sigma_{-i}\right) - R_i\left((x_i, t_i), \sigma_{-i}\right)}{\Delta t_i} < 0 \end{aligned} \tag{5.2}$$

Note that, with Condition 5.2, we impose that $\nexists t \in [0, \tau]$ such that there is a race among the agents to contribute at t. We now analyze the consequence of such a refund bonus scheme on the game's characteristics induced by it.

5.3.1 Sufficiency of the Refund Bonus Scheme

We show that a refund bonus scheme satisfying Conditions 5.1 and 5.2 is sufficient to implement civic crowdfunding projects in sequential settings. For this, let G be the game induced by the refund bonus scheme $R(\cdot)$, for the payoff structure given by Eq. 5.3.

$$u_i(\sigma; \theta_i) = \underbrace{\mathbb{1}_{\mathbf{x} \geq c} \cdot (\theta_i - x_i)}_{\text{Funded Utility}} + \underbrace{\mathbb{1}_{\mathbf{x} < c} \cdot (R_i(\sigma))}_{\text{Unfunded Utility}} \tag{5.3}$$

We require G to satisfy the following properties.

Property 5.1. *In G, the total contribution equals the provision point at equilibrium, i.e., $\mathbf{x} = c$.*

Property 5.2. *G must avoid the race condition.*

Property 5.3. *G is a sequential game.*

With these, consider the following theorem.

Theorem 5.1 (Sufficiency Conditions)**.** *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$ and the utility structure for each agent $i \in \mathcal{A}$ given by Eq. 5.3. If $R(\cdot)$ satisfies Conditions 5.1 and 5.2, Properties 5.1, 5.2 and 5.3 hold.*

Proof. In Steps 1, 2, and 3, we show that $R(\cdot)$ satisfying Condition 5.1 is sufficient to satisfy Property 5.1 and Condition 5.2 is sufficient to satisfy Properties 5.2 and 5.3.

Step 1: As $\vartheta > c$, from Eq. 5.3, at equilibrium $\mathbf{x} < c$ cannot hold, as $\exists i \in \mathcal{A}$ with $x_i < \theta_i$, at least. Such an Agent i could obtain a higher refund bonus by marginally increasing its contribution since $R(\cdot)$ satisfies Condition 5.1 and $B > 0$. For $\mathbf{x} > c$, any agent with a positive contribution could gain in utility by marginally decreasing its contribution. Thus, at equilibrium $\mathbf{x} = c$ or G satisfies Property 5.1.

Step 2: Every Agent i contributes as soon as it arrives, since $R(\cdot)$ satisfies Condition 5.2 i.e., $\forall i \in \mathcal{A}$,

$$u_i\left((x_i, a_i), \sigma_{-i}\right) > u_i\left((x_i, t), \sigma_{-i}\right) \ \forall t \in (a_i, \tau].$$

In other words, the best response $\forall i \in \mathcal{A}$ is the strategy $\sigma_i = (x_i, a_i)$. Thus, as per Definition 5.1, G avoids the race condition or G satisfies Property 5.2.

Step 3: Since G satisfies Property 5.2, it avoids the race condition. Hence, it can be implemented in a sequential setting, or G is a sequential game. □

5.3.2 Necessity of the Refund Bonus Schemes

Theorem 5.1 shows that Condition 5.1 is sufficient to satisfy Property 5.1 and Condition 5.2 is sufficient to satisfy Properties 5.2 and 5.3. With Theorem 5.2, we further prove that Condition 5.2 is necessary for Properties 5.2 and 5.3; while *weak* Condition 5.1 is necessary for Property 5.1. However, we remark that Theorem 5.2 does not characterize G completely. For the theorem to hold, *unlike* in the case of Theorem 5.1, we assume there exists a unique equilibrium defined by the strategy $(x_i^\star, t_i^\star),\ \forall i \in \mathcal{A}$.

Theorem 5.2 (Necessary Conditions)**.** *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$ and the utility structure for each agent $i \in \mathcal{A}$ given by Eq. 5.3. If $R(\cdot)$ satisfies Properties 5.1, 5.2 and 5.3 and there is unique equlibrium, then "weak" Condition 5.1 and Condition 5.2 hold.*

Proof. Observe that,

- Property 5.1 $\implies$ weak Condition 5.1. Assume weak Condition 5.1 does not hold. This implies that $\exists i \in \mathcal{A}$ for whom $R_i(x_i, \cdot) > R_i(x_i + \epsilon, \cdot)$ for some $\epsilon > 0$. Now consider a case, w.l.o.g., that the agent i is the last agent. Further, the project will be funded if agent i contributes $x_i + \epsilon$, i.e., where its funded utility equals its unfunded payoff [312]. Since $R_i(x_i, \cdot) > R_i(x_i + \epsilon, \cdot)$, agent i will prefer to contribute x_i and at equilibrium, $\mathbf{x} \neq c$. This is a contradiction as it is given that Property 5.1 holds.

- Properties 5.2 and 5.3 $\implies$ Condition 5.2. Property 5.2 implies that G avoids the race condition. That is, $\nexists\ i \in \mathcal{A}$ for whom $u_i(x_i, a_i) > u_i(x_i, a_i + \epsilon)$ for any $\epsilon > 0$ which in turn implies Condition 5.2. This is because, for the same x_i, u_i and R_i are both decreasing with respect to t_i.

These complete the proof of the theorem. □

Theorem 5.1 shows that a refund bonus scheme satisfying Conditions 5.1 and 5.2 avoids the race condition (Property 5.2) and induces a sequential game (Property 5.3). Thus, a mechanism deploying such a refund bonus scheme can be *implemented sequentially*, i.e., over web-based (or online) platforms. Refund bonus schemes should also be clear to explain to a layperson. Moreover, when deployed as a smart contract, these should be computationally efficient and cost-effective. We show the following proposition through this generalized result on refund bonus schemes.

Proposition 5.1. *PPS [52] satisfies Condition 5.1 and Condition 5.2.*

Proof. Since every cost function used in PPS for crowdfunding must satisfy $\frac{\partial(s_i^{t_i} - x_i)}{\partial x_i} > 0$, $\forall i$ [52, CONDITION-7], PPS satisfies Condition 5.1. For Condition 5.2, observe that $\forall i$, from [52, Eq. 6]

$$(s_i^{t_i} - x_i) = C_0^{-1}(x_i + C_0(q^{t_i})) - q^{t_i} - x_i. \tag{5.4}$$

	Refund Scheme	Parameters	Covergence of Sum	Based On
PPRG	$R_i^{PPRG}(\cdot) = \left(\frac{x_i + a \times (1/\gamma)^{i-1}}{C + K_1}\right) B$	$a > 0, 1/\gamma < 1, K_1 = \frac{a\gamma}{\gamma - 1}$	$\sum_{i=1}^{\infty} \left(x_i + a(1/\gamma)^{i-1}\right) = C + K_1$	Geometric Progression (GP)
PPRE	$R_i^{PPRE}(\cdot) = \left(\frac{x_i + K_2 \times e^{-t_i}}{C + K_2}\right) B$	$K_2 > 0$	$\sum_{i=1}^{\infty}(x_i) + \int_{t=t_1}^{\infty}(K_2 e^{-t} dt) \leq C + K_2$	Exponential Function (EF)
PPRP	$R_i^{PPRP}(\cdot) = \left(\frac{x_i + K_3 \times \frac{1}{i(i+1)}}{C + K_3}\right) B$	$K_3 > 0$	$\sum_{i=1}^{\infty} \left(x_i + K_3 \frac{1}{i(i+1)}\right) = C + K_3$	Polynomial Function (PF)

Table 5.2: Various Refund schemes satisfying Condition 5.1 and Condition 5.2 for an Agent i. Note that, in R^{PPRG} and R^{PPRP}, the subscript i denotes the order of the contribution.

In Eq. 5.4, as $t_i \uparrow$, $q^{t_i} \uparrow$ as it is a monotonically non-decreasing function of t and thus R.H.S. of Eq. 5.4 decreases since R.H.S. of Eq. 5.4 is a monotonically decreasing function of q^{t_i} [52, Theorem 3 (Step 2)]. Thus, PPS also satisfies Condition 5.2. □

The following corollary immediately follows the above propositions.

Corollary 5.1. *PPS avoids the race condition and thus can be implemented sequentially.*

We next present three novel refund schemes satisfying Conditions 5.1 and 5.2 and the novel provision point mechanisms based on them.

5.3.3 Refund Bonus Schemes

Table 5.2 presents three novel refund schemes for an Agent $i \in \mathcal{A}$ contributing x_i at time t_i and the mechanisms that deploy them. Note that we require all the refund bonus schemes to converge to a particular sum that can be pre-computed. This convergence allows these schemes to be *budget balanced*. The parameters a, γ, K_1, K_2, K_3 and B are mechanism parameters (for their respective mechanisms) that the social planner must announce at the start. Additionally, the refund schemes presented deploy three mathematical functions: geometrical, exponential, and polynomial decay. $R^{PPRG}(\cdot)$ and $R^{PPRP}(\cdot)$ refunds the contributing agents based on the sequence of their arrivals (similar to PPS), while the refund scheme $R^{PPRE}(\cdot)$ refunds them based on their time of contribution.

Sufficiency Conditions. We now show that PPRG satisfies Conditions 5.1 and 5.2.

Claim 5.1. *$R^{PPRG}(\sigma)$ satisfies Condition 5.1 $\forall i \in \mathcal{A}$.*

Proof. Observe that $\forall i \in \mathcal{A}$,

$$\frac{\partial R_i^{PPRG}(\sigma)}{\partial x_i} = \frac{B}{C + K_1} > 0 \ \forall t_i.$$

Therefore, $R^{PPRG}(\cdot)$ satisfies Condition 5.1 $\forall i$. □ □

Claim 5.2. *$R^{PPRG}(\sigma)$ satisfies Condition 5.2.*

Proof. For every Agent $i \in \mathcal{A}$ arriving at time a_i, its share of the refund bonus given by $R^{PPRG}(\cdot)$ will only decrease from that point in time, since its position in the sequence of contributing agents can only go down, making it liable for a lesser share of the bonus, for the same contribution. Let $\tilde{t_i}$ be the position of the agent arriving at time a_i, when it contributes at time t_i. While $\tilde{t_i}$ will take discrete values corresponding to the position of the agents, for the purpose of differentiation, let $\tilde{t_i} \in \mathbb{R}$. We can argue that at every epoch of time t_i, Agent $\tilde{t_i}$ will contribute to the project. With this, $R^{PPRG}(\cdot)$ can be written as,

$$R_i^{PPRG}(\sigma) = \left(\frac{x_i + a \times (1/\gamma)^{\tilde{t_i}-1}}{C + K_1} \right) B.$$

Further observe that $\forall i \in \mathcal{A}$,

$$\frac{\partial R_i^{PPRG}(\sigma)}{\partial \tilde{t_i}} = -\left(\frac{a \times (1/\gamma)^{\tilde{t_i}}}{C + K_1} \right) B < 0 \ \forall x_i.$$

Therefore, $R^{PPRG}(\cdot)$ satisfies Condition 5.2. □ □

We can similarly prove that R^{PPRE} and R^{PPRP} satisfy Conditions 5.1 and 5.2. We omit the proofs for their simplicity.

5.3.4 Gas Comparisons

As aforementioned, CC is now being deployed as smart contracts (SCs) over public blockchains such as Ethereum [42]. Thus, CC mechanisms deployed as SCs must be efficient, i.e., result in less *gas* consumption. Gas is a unit of fees that the Ethereum protocol charges per computational step executed in a contract or transaction. This fee prevents deliberate attacks and abuse on the Ethereum network [42].

We show a hypothetical cost comparison between PPS, PPRG, PPRE, and PPRP based on the Gas usage statistics from [42, 297]. For the relevant operations, the cost in Gas units is: ADD: 3, SUB: 3, MUL: 5, DIV: 5, EXP(x): $10 + 10 * log(x)$ and LOG(x): $365 + 8 *$ size of x in bytes. Table 5.3 presents the comparison[1]. We remark that the only difference in the induced CC game will be the computation of the refund bonus for each contributing agent. This refund will depend on the underlying refund bonus scheme. Thus, we focus only on the gas cost because of the said schemes.

From Table 5.3, for every agent, PPRG takes 21 gas units, PPRP takes 31 gas units, PPRE takes at least 31 gas units, and PPS takes at least 407 gas units. When implemented on smart contracts, PPS is expensive because of its logarithmic scoring rule for calculating payment rewards. On the other hand, PPRG, PPRP, and PPRE use simpler operations and therefore have minimal operational costs.

Inference from Table 5.3. Note that the average gas price per unit varies. At the time of writing this thesis, we have a (generous) average gas price ≈ 200 GWei, i.e., 2×10^{-7} ETH; and also 1 ETH ≈ 2396 USD. As a result, the cost incurred by a crowdfunding platform, assuming when $n = 100$, is (approximately) (i) PPS: 20 USD (at least); (ii) PPRG: 1 USD; (iii) PPRE: 1.48 USD (at least); and (iv) PPRP: 1.48 USD. Further, in December 2019, Kickstarter had 3524 active projects [156]. The data implies the total project cost

[1]We do not require any exponential calculation in PPRG – by storing the last GP term in a temporary variable.

Operation	PPS		PPRG		PPRE		PPRP	
	Operations	Gas Consumed	Operations	Gas Consumed	Operations	Gas Consumed	Operations	Gas Consumed
ADD	2	6	2	6	2	6	2	6
SUB	2	6	0	0	0	0	0	0
MUL	2	10	2	10	2	10	3	15
DIV	2	10	1	5	1	5	2	10
EXP(x)	2	$10 + 10 \times (log(x))$	0	0	1	$10 + 10 \times (log(x))$	0	0
LOG(x)	2	$365 + 8 \times$ (bytes logged)	0	0	0	0	0	0
	Total Gas:	407 (at least)	**Total Gas:**	21	**Total Gas:**	31 (at least)	**Total Gas:**	31

Table 5.3: Gas Consumption comparison between PPS, PPRG, PPRE and PPRP for an agent. All values are in Gas units.

for (i) PPS: 70480 USD and (ii) PPRG: 3524 USD. PPRG reduces the cost incurred by the platform by (at least) ≈ 20 times.

5.4 PPRG

We now describe the mechanism *Provision Point mechanism with Refund through Geometric Progression* (PPRG) for crowdfunding a public project. PPRG incentivizes an interested agent to contribute as soon as it arrives at the crowdfunding platform. In PPRG, for the exact contribution of Agent i and Agent j, i.e., $x_i = x_j$, the one who contributed earlier obtains a higher share of the refund bonus. These share differences are allocated using an infinite geometric progression series (GP) with a common ratio of < 1.

Refund Bonus Scheme. The sum of an infinite GP with $a > 0$ as the first term and $0 < 1/\gamma < 1$ as the common ratio is: $K_1 = a \times \sum_{i=0}^{\infty}(1/\gamma)^i = \frac{a\gamma}{\gamma - 1}$. With this, we propose a novel refund bonus scheme,

$$R_i^{PPRG}(\sigma) = p_i = \left(\frac{x_i + a \times (1/\gamma)^{i-1}}{C + K_1}\right) B \tag{5.5}$$

for every Agent $i \in \mathcal{A}$, $B > 0$ as the total bonus budget allocated for the project by the SP and where $\sigma = ((x_i, t_i) \mid \forall i \in N)$. The values a and γ are mechanism parameters that the social planner must announce at the project's start.

Equilibrium Analysis of PPRG. The analysis follows from Theorem 5.1.

Theorem 5.3. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$ and the utility structure for each agent $i \in \mathcal{A}$ given by Eq. 5.3. Let the refund scheme be given by Eq. 5.5, $\forall i \in \mathcal{A}$. Then at equilibrium, we have (i) $\mathbf{x} = c$ and (ii) the set of strategies $\left\{\sigma_i^\star = (x_i^\star, a_i) \mid x_i^\star \leq \frac{\theta_i(c+K_1) - aB \times (1/\gamma)^{i-1}}{c+K_1+B}\right\} \forall i \in \mathcal{A}$ are sub-game perfect Nash equilibria.*

Proof. We prove the theorem with the following steps.

Step 1: Since $R^{PPRG}(\cdot)$ satisfies Condition 5.1 (Claim 5.1) and Condition 5.2 (Claim 5.2) and has a payoff structure as given by Eq. 5.3, from Theorem 5.1 we get the result that PPRG induces a sequential move game and thus, can be implemented in a sequential setting.

Step 2: From Claim 2, the best response for any agent is to contribute as soon as it arrives i.e., at time a_i.

Step 3: We assume that each agent is symmetric in its belief for the provision of the project. Moreover, from Theorem 5.1, agents know that the project will be provisioned at equilibrium. Therefore, for any agent, its equilibrium contribution becomes that $x_i^\star$ for which its provisioned payoff is *greater than or equal to* its not provisioned payoff. Now, with $\mathbf{x} = c$ at equilibrium,

$$\theta_i - x_i^\star \geq \left(\frac{x_i^\star + a \times (1/\gamma)^{i-1}}{\mathbf{x} + K_1}\right) B$$

$$\Rightarrow x_i^\star \leq \frac{\theta_i(c + K_1) - aB \times (1/\gamma)^{i-1}}{c + K_1 + B}$$

Step 4: Summing over $x_i^\star$, $\forall i$ we get,

$$B \leq \frac{(c + K_1)\vartheta - c^2 - cK_1}{c + K_1}.$$

as $\sum_{i \in \mathcal{A}} x_i^\star = c$. From the above equation, we get

$$0 < B \leq \frac{(c + K_1)\vartheta - c^2 - cK_1}{c + K_1} = \vartheta - c$$

as a sufficient condition for the existence of Nash Equilibrium for PPRG.

Step 5: The following scenarios prove the strategies are sub-game perfect.

- For an Agent i entering the project such that $\mathbf{x} = c$, its best response is contributing 0.

- For an Agent i entering the project such that $c - \mathbf{x} > 0$ with $x_i^* > c - \mathbf{x}$, its best response is contributing $c - \mathbf{x}$. Observe that Agent i will contribute the maximum contribution required, $c - \mathbf{x}$, since its not provisioned payoff increases as its contribution increases (Claim 5.1). Therefore, for a contribution less than $c - \mathbf{x}$, Agent i will receive *lesser* payoff in comparison for the contribution $c - \mathbf{x}$.

- Lastly, for an Agent i entering the project such that $c - \mathbf{x} > 0$ with $x_i^\star \leq c - \mathbf{x}$, its best response is contributing $x_i^\star$ (as defined in Theorem 5.3). This is because, for the contribution x_i^*, its provisioned payoff is *equal* to its not provisioned payoff. For this scenario, with backward induction, it is the best response for every Agent i to follow the same strategy in which their provisioned payoffs are equal to their not provisioned payoffs, irrespective of $c - \mathbf{x}$.

This proves the theorem. □

Discussion. Observe that, as the refund bonus decreases with time (Claim 5.2), each agent in PPRG is better off contributing once instead of breaking up its contribution. This result follows as we assume that each agent's belief for the project's provision is symmetric and does not vary.

With Theorem 5.3, we identify a set of pure-SPE at which the project is provisioned. However, we do not claim that these are the only set of pure-SPE possible. We leave it for future work to explore other possible pure-SPE at which the project gets provisioned. Also, the equilibrium analysis of PPRE and PPRP is similar to Theorem 5.3.

5.5 Conclusion

This chapter looked for provision point mechanisms for CC with refund bonus schemes. Towards it, we introduced Contribution Monotonicity and Time Monotonicity for refund bonus schemes in CC mechanisms. We proved that these two conditions are sufficient to implement provision point mechanisms with refund bonuses to possess an equilibrium that avoids free-riding and the race condition (Theorem 5.1). We then proposed three simple refund bonus schemes and designed novel mechanisms that deploy them: PPRG, PPRE, and PPRP. We showed that PPRG has much less cost when implemented as a smart contract over the Ethereum framework. We identified a set of sub-game perfect equilibria for PPRG in which it provisions the project at equilibrium (Theorem 5.3).

Chapter 6

Combinatorial Civic Crowdfunding with Budgeted Agents

For single-project CC, we saw that refunds incentivize agents to contribute, thereby guaranteeing the project's funding. Unfortunately, these funding guarantees only apply when agents have an unlimited budget. This chapter focuses on a combinatorial setting, where multiple projects are available for CC and agents have a limited budget. We study certain specific conditions where funding can be guaranteed. Further, funding the optimal social welfare subset of projects is desirable when every available project cannot be funded due to budget restrictions. We prove the impossibility of achieving optimal welfare at equilibrium for any monotone refund scheme. We study different heuristics that the agents can use to contribute to the projects in practice. We demonstrate the heuristics' performance as the average-case trade-off between welfare obtained and agent utility through simulations.

⋆ ⋆ ⋆ ⋆ ⋆

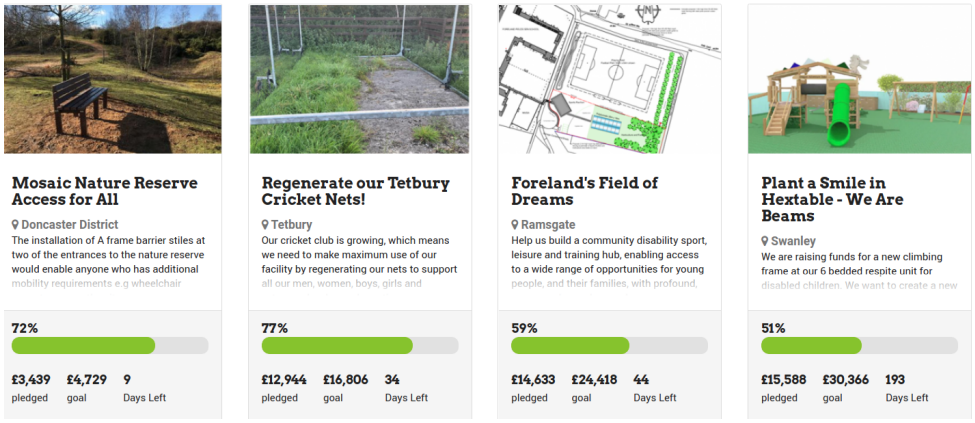


Figure 6.1: Example instance of Combinatorial Civic Crowdfunding (CC) [294]. Notice that agents may be interested in contributing to more than one project (especially if they are similar in type).

Property	**Socially Efficient Equilibrium**
Budget Surplus	✗ (Corollary 6.1)
Budget Surplus + Subset Feasibility	✓ (Theorem 6.2)
Budget Deficit + Subset Feasibility	✗ (Theorem 6.3)

Table 6.1: Overview of our theoretical results.

6.1 Introduction

In Chapter 4, we proposed new mechanisms that enrich the agent's information structure by allowing negative valuations and agents with asymmetric beliefs. Chapter 5 focused only on agents with positive valuation and symmetric beliefs and presented necessary and sufficient conditions for the refund bonus scheme. However, the mechanisms proposed in Chapter 4, Chapter 5, and the seminal works introduced in Chapter 2.3 – all (i) assume that agents have an unlimited budget (i.e., can contribute their equilibrium contributions) and (ii) are only for the single-project case.

Combinatorial CC. Typically, multiple projects are simultaneously available for CC; refer to Figure 6.1 for a concrete example. We refer to the crowdfunding of multiple CC projects simultaneously as *combinatorial* CC. This chapter aims to lay the theoretical foundation for combinatorial CC with budgeted agents. Table 6.1 presents the overview of the results presented in this chapter, described in detail next.

6.1.1 Chapter Contributions

Budget Surplus (BS). We first study the seemingly straightforward case of *Budget Surplus*, i.e., the overall budget across the agents is more than the projects' total cost. For this, it is welfare optimal to fund all the projects. Despite the surplus budget, we show that the projects' funding cannot be guaranteed at equilibrium (Theorem 6.1 and Corollary 6.1).

Subset Feasibility (SF). We observe that the budget distribution among the agents plays a significant role in deciding the funding status of the projects. Conditioning on the budget distribution, we introduce *Subset Feasibility* of a given subset of projects. We prove that Subset Feasibility coupled with Budget Surplus guarantees funding of every available project at equilibrium (Theorem 6.2), thereby generating the maximum possible social welfare.

Budget Deficit (BD). Trivially, in the case of *Budget Deficit* – when there is no Budget Surplus – one can only fund a subset of projects. It may be desirable that such a subset is welfare-maximizing within the budget. We refer to the funding of the social welfare optimal subset at equilibrium as *socially efficient equilibrium*. For this case, we present the following results.

First, we show that, in general, achieving socially efficient equilibrium is impossible for any refund scheme (Example 6.1). Next, we prove that even with the stronger assumption of Subset Feasibility, it is still impossible to achieve socially efficient equilibrium (Theorem 6.3). Specifically, we prove that strategic deviations may exist for agents such that the optimal welfare subset remains unfunded. We then show that it is NP-Hard for an agent

to find its optimal deviation, given the contributions of all the other agents (Theorem 6.5 and Corollary 6.2). Due to Theorem 6.3 and hardness of optimal deviation (Theorem 6.5), we construct five heuristics for the agent's contributions and empirically study their social welfare and agent utility through simulations (Section 6.4).

6.1.2 Chapter Notations and Solution Concepts

Throughout this chapter, we use the notations introduced during the description of CC in Chapter 2.3. We focus on the general multi-project CC model, as introduced in Definition 2.26. Table 6.2 tabulates the notations used in this chapter.

We also build our mechanisms on the existing PPR mechanism (Chapter 2.3.2). Lastly, the solution concepts used are Nash equilibrium (Definition 2.3) and sub-game perfect Nash equilibrium (Definition 2.4).

6.1.2.1 Additional Preliminaries

Here, we provide additional definitions required for the results presented in this chapter. Firstly, let γ_i denote an agent $i \in \mathcal{A}$ budget. That is, an agent i contributes $x_{ij} \in \mathbb{R}_+$ to project j, s.t., $\sum_{j \in \mathcal{P}_m} x_{ij} \leq \gamma_i$. With this, consider the following definitions.

6.1.2.1.1 Welfare Optimal Ideally, when there is a limited budget, it may be desirable to fund the welfare optimal subset defined as follows. Note that, the welfare obtained from project j if funded is $\vartheta_j - c_j$ and zero otherwise [34, 51][1].

[1] All the results presented in this chapter also hold if $P^\star \in \arg\max_{M \subseteq \mathcal{P}_m} \sum_{j \in M} \vartheta_j$ s.t. $\sum_{j \in M} c_j \leq \sum_{i \in \mathcal{A}} \gamma_i$.

Definition 6.1: Welfare Optimal

A set of projects $P^\star \subseteq \mathcal{P}_m$ is welfare optimal if it maximizes social welfare under the available budget, i.e.,

$$P^\star \in \arg\max_{M \subseteq \mathcal{P}_m} \sum_{j \in M} (\vartheta_j - c_j) \text{ s.t. } \sum_{j \in M} c_j \leq \sum_{i \in \mathcal{A}} \gamma_i. \tag{6.1}$$

We make the following observations based on Definition 6.1.

- Finding $P^\star$ requires public knowledge of ϑs, cs and the value $\sum_{i \in \mathcal{A}} \gamma_i$. Recall that ϑ is assumed to be public knowledge in CC (Section 2.3). Similarly, we also assume that the overall budget in the system $\sum_{i \in \mathcal{A}} \gamma_i$ is public knowledge. This may be done by deriving the overall budget by aggregating citizen interest [8, 141].
- Computing $P^\star$ is NP-Hard as it can be trivially reduced from the KNAPSACK problem. However, note that our primary results focus on $P^\star$'s funding guarantees at equilibrium (and not actually computing it). Moreover, computing $P^\star$ may also not be a deal breaker as the number of simultaneous projects available will not be arbitrarily large. One may also employ FPTAS [166].

6.1.2.1.2 Refund Scheme Civic crowdfunding mechanisms employ refunds to overcome the free-riding challenge. We define the refund scheme for each project $j \in \mathcal{P}_m$ as $R_j(B_j, x_{ij}, \mathbf{x}_j) : \mathbb{R}^3_+ \to \mathbb{R}_+$ s.t. $r_{ij} = R_j(B_j, x_{ij}, \mathbf{x}_j)$ is agent i's refund share for contributing x_{ij} to project j. The overall budget for the refund bonus $B_j > 0$ is public knowledge. Typically, if a project is unfunded, the agents receive r_{ij}, and zero refund otherwise. The total refunds distributed for project j can be such that $\sum_{i \in \mathcal{A}} r_{ij} = B_j$ (e.g., [312]) or $\sum_{i \in \mathcal{A}} r_{ij} < B_j$ (e.g., [52]). Throughout the chapter, we assume that $\sum_{i \in \mathcal{A}} r_{ij} = B_j \ \forall j$.

The CC literature also assumes that R is anonymous, i.e., refund share is independent of agent identity. Furthermore, we also assume that R satisfies Contribution Monotonicity (CM) (Condition 5.1).

6.1.2.1.3 Agent Utilities and Important Definitions

Let $\mathcal{M}_{CC} = \langle \mathcal{P}_m, \mathcal{A}, \gamma, (\vartheta_j)_{j\in\mathcal{P}_m}, (R_j)_{j\in\mathcal{P}_m}, (B_j)_{j\in\mathcal{P}_m} \rangle$ define a general combinatorial CC game. In this, the overall agent utility can be defined as.

Definition 6.2: Agent Utility

Given an instance of $\mathcal{M}_{CC}$, with agents having valuations $[\theta_{ij}]$ and contributions $[x_{ij}]$, the utility of an agent i for each project $j \in \mathcal{P}_m$ is given by $u_{ij}(\theta_{ij}, x_{ij}, r_{ij}, \mathbf{x}_j, c_j) : \mathbb{R}^5_+ \to \mathbb{R}$

$$u_{ij}(\cdot) = \mathbb{1}_{\mathbf{x}_j \geq c_j} \cdot \underbrace{(\theta_{ij} - x_{ij})}_{\text{Funded utility } u^F_{ij}} + \mathbb{1}_{\mathbf{x}_j < c_j} \cdot \underbrace{r_{ij}}_{\text{Unfunded utility } u^U_{ij}}$$

An agent i's utility for project j is either $u^F_{ij} = \theta_{ij} - x_{ij}$ when j is funded, and $u^U_{ij} = r_{ij}$ otherwise. Let $U_i(\cdot)$ denote the total utility an agent i derives, i.e., $U_i(\cdot) = \sum_{j\in\mathcal{P}_m} u_{ij}$[2]. This incentive structure induces a game among the agents. As the agents are strategic, each agent aims to provide contributions that maximize its utility. As such, we focus on contributions that follow the pure strategy Nash equilibrium (Definition 2.3).

Socially Efficient Equilibrium (SEE). Given the contributions $[x^*_{ij}]$, we can compute the set of the projects that are funded and unfunded at equilibrium. Throughout the chapter, we refer to the funding of $P^\star$ at equilibrium as *socially efficient equilibrium*. We next define budget surplus.

Definition 6.3: Budget Surplus (BS)

There is enough overall budget to fund each project $j \in \mathcal{P}_m$, i.e., $\sum_{i\in\mathcal{A}} \gamma_i \geq \sum_{j\in\mathcal{P}_m} c_j$.

[2]This chapter assumes that the agent utilities are *additive*. We leave the study of other valuation functions as future work.

We refer to the scenario $\sum_{i\in\mathcal{A}}\gamma_i < \sum_{j\in\mathcal{P}_m} c_j$ as Budget Deficit (BD). Recall that from Definition 2.26, we also know that $\vartheta_j > c_j, \forall j \in \mathcal{P}_m$. That is, there is sufficient interest in each available project's funding. Hence, when there is a surplus budget, it is optimal to fund all the projects, i.e., $P^\star = \mathcal{P}_m$. Further, we assume that agents do not have any additional information about the funding of the public projects. This assumption implies that their belief in the projects' funding is symmetric.

Notation	Definition
$\mathcal{P}_m, m \geq 2$	Set of Projects to be crowdfunded
$\mathcal{A} = \{1, \ldots, n\}$	Set of Agents
$\theta_{ij} \in \mathbb{R}$	Agent i's valuation for project $j \in \mathcal{P}_m$
$\vartheta_j = \sum_{i \in \mathcal{A}} \theta_{ij}$	Overall valuation of the project j
$c_j \in \mathbb{R}_{\geq 0}$	Target cost of the project j
$\tau_j \in \mathbb{R}_{\geq 0}$	Deadline of the project j
$x_{ij} \in \mathbb{R}_{\geq 0}$	Agent i's contribution to the project j
$\mathbf{x}_j = \sum_{i \in \mathcal{A}} x_i$	Total contribution to the project
$B_j > 0$	Refund bonus
R_j	Refund Bonus Scheme
$r_{ij} = R_j(x_{ij}; \mathbf{x}_j, c_j, B_j, \cdot)$	Refund bonus to Agent i to project j
$u_{ij}(\theta_{ij}, x_{ij}; \mathbf{x}_j, c_j, B_j)$	Agent i's utility to project j
$\gamma_i \in \mathbb{R}_{\geq 0}$	Agent i individual budget
$\Gamma \in \sum_{i \in \mathcal{A}} \gamma_i$	Total budget in the system
$P^\star$	Welfare optimal subset of projects
u_{ij}^F	'Funded' utility of Agent i from project j
u_{ij}^F	'Funded' utility of Agent i from project j

Table 6.2: Chapter Notations

6.2 Funding Guarantees for Combinatorial CC under Budget Surplus

For CC under Budget Surplus (Definition 6.3), a sufficient overall budget exists to fund all the projects. Theorem 6.1 shows that despite the sufficient budget, projects may not get funded as the set of equilibrium contributions for an agent may not exist. Unlike the analysis of PPR [312] and PPS [52], agents may not have well-defined contributions satisfying PSNE due to limited budgets. The non-existence results due to the uneven distribution of budget among the agents. Hence, agents with higher budgets exploit the mechanism to obtain higher refunds while ensuring the projects remain unfunded.

Theorem 6.1. *Given $(R_j)_{j\in\mathcal{P}_m}$ which satisfy Condition 5.1, there are Budget Surplus (Definition 6.3) game instances of $\mathcal{M}_{CC}$ with $B_j = \vartheta_j - c_j, \forall j$ such that there is no equilibrium. That is, the set of equilibrium contributions may be empty.*

Proof. Consider $\mathcal{P}_m$ projects and $\mathcal{A}$ agents s.t. Def. 6.3 is satisfied, i.e., $\sum_{i\in\mathcal{A}}\gamma_i \geq \sum_{j\in\mathcal{P}_m} c_j$. We can easily construct game instances where there exists non-empty $N_1 \subset \mathcal{A}$ s.t. $\sum_{i\in N_1}\gamma_i < \min_j c_j$. To satisfy Budget Surplus (Def. 6.3), $N_2 = \mathcal{A}\setminus N_1$ must have enough budget so that the agents in $N_1 + N_2$ can fund all the projects.

Each agent i receives a funded utility $u_{ij}^F = \theta_{ij} - x_{ij}$ for contributing x_{ij} towards project j. That is, as $x_{ij}\uparrow \implies u_{ij}^F \downarrow$. The agent may also receive an unfunded utility of $u_{ij}^U = r_{ij} = R_j(x_{ij}, B_j, \cdot)$ for project j. Since R_j is monotonically increasing (Condition 5.1), $x_{ij}\uparrow \implies u_{ij}^U \uparrow$. Observe that u_{ij}^U and u_{ij}^F intersect at the upper-bound of the equilibrium contribution, $\bar{x}_{ij}$ (e.g., Theorem 2.8, Theorem 5.3), where $u_{ij}^F = u_{ij}^U$. For any contribution $x_{ij} > \bar{x}_{ij}$, the agent receives a greater unfunded utility.

Consider a scenario where the agents in N_1 do not have sufficient budget and thus contribute less than $\bar{x}_{ij}$. Let $C' = \sum_{i\in N_1}\left(\sum_{j\in\mathcal{P}_m} x_{ij}\right)$. Now, we have $x_r = \sum_{j\in\mathcal{P}_m} c_j - C'$ as the remaining amount to fund the set P. In order to fund $\mathcal{P}_m$, the remaining agents in N_2 may need to contribute $\hat{x}_{ij} > \bar{x}_{ij}$ where $i \in N_2$. The unfunded utility is more at $\hat{x}_{ij}$.

Hence, the agents contribute $x_{ij} = \bar{x}_{ij} - \epsilon$, s.t. $\epsilon \to 0$ and $\epsilon > 0$, i.e., the contributions are not well-defined. □

Observe that if any project j is funded at equilibrium then the equilibrium set $(x^*_{i1}, \ldots, x^*_{im})_{i \in \mathcal{A}}$ can not be empty, contradicting Theorem 6.1. Corollary 6.1 captures this observation.

Corollary 6.1. *Given $(R_j)_{j \in \mathcal{P}_m}$ satisfying CM (Condition 5.1), there are game instances of $\mathcal{M}_{CC}$ s.t. even with Budget Surplus (Def. 6.3), no project in $\mathcal{P}_m$ may be funded at equilibrium.*

With Corollary 6.1, we prove that Budget Surplus is not sufficient to fund every project at equilibrium. To this end, we next identify the sufficient condition to ensure the funding of every project, under Budget Surplus.

Subset Feasibility

With N_1 and N_2 in Theorem 6.1's proof, we assume a specific distribution on agents' budget. To resolve this, we introduce *Subset Feasibility*, which assumes a restriction on each agent's budget distribution. Informally, if each agent i has enough budget to contribute $\bar{x}_{ij}$ (see proof of Theorem 6.1) for $j \in M$, $M \subseteq \mathcal{P}_m$, then Subset Feasibility is satisfied for M. Formally,

Definition 6.4: Subset Feasibility for M (SF_M)

Given an instance of $\mathcal{M}_{CC}$ with $(R_j)_{j \in \mathcal{P}_m}$ satisfying Condition 5.1, SF_M, $M \subseteq \mathcal{P}_m$, is satisfied if, $\forall i \in \mathcal{A}$ we have $\gamma_i \geq \sum_{j \in M} \bar{x}_{ij}$. Here, $\theta_{ij} - \bar{x}_{ij} = R_j(\bar{x}_{ij}, B_j, \cdot)$ (refer proof of Theorem 6.1).

Claim 6.1. *Given any $(R_j)_{j\in\mathcal{P}_m}$ whose equilibrium contributions satisfy $\sum_i \bar{x}_{ij} \geq c_j$, we have $SF_{\mathcal{P}_m} \implies$ BS.*

Proof. From the definition of $\text{SF}_{\mathcal{P}_m}$, $\gamma_i \geq \sum_{j\in\mathcal{P}_m} \bar{x}_{ij},\ \forall i \in \mathcal{A}$. As $\sum_i \bar{x}_{ij} \geq c_j$, we have $\sum_i \gamma_i \geq \sum_{j\in\mathcal{P}_m} \sum_i \bar{x}_{ij} \geq \sum_{j\in M} c_j$, i.e., Budget Surplus is also satisfied. □

Similarly, we have BS $\not\implies$ $SF_{\mathcal{P}_m}$. From Claim 6.1, it is welfare optimal to fund every available project under $SF_{\mathcal{P}_m}$. Theorem 6.2 indeed proves that under $SF_{\mathcal{P}_m}$ each project $j \in \mathcal{P}_m$ gets funded at equilibrium, thereby generating optimal social welfare.

Theorem 6.2. *Given $\mathcal{M}_{CC}$ and $(R_j)_{j\in\mathcal{P}_m}$ satisfying CM (Condition 5.1) such that $SF_{\mathcal{P}_m}$ is satisfied, at equilibrium all the projects are funded, i.e., $\mathbf{x}_j = c_j,\ \forall j \in \mathcal{P}_m$ if $B_j \leq \vartheta_j - c_j,\ \forall j \in \mathcal{P}_m$. Further, the set of PSNEs are:*
$\Big\{(x^*_{ij})_{j\in\mathcal{P}_m} \mid u^F_{ij}(x^*_{ij};\cdot) \geq u^U_{ij}(x^*_{ij};\cdot), \forall j \in \mathcal{P}_m,\ \forall i \in \mathcal{A}\Big\}$.

Proof. For a given refund scheme R_j and corresponding $\bar{x}_{ij}$, each agent i's budget γ_i satisfies, using $SF_{\mathcal{P}_m}$, $\gamma_i \geq \sum_{j\in\mathcal{P}_m} \bar{x}_{ij}$. Let each agent i's final contribution $x^*_{ij} \leq \bar{x}_{ij}$; where $u^F_{ij}(\bar{x}_{ij}) = u^U_{ij}(\bar{x}_{ij})$. Since $\sum_{i\in\mathcal{A}} \bar{x}_{ij} \geq c_j$, the project gets funded, and each agent obtains a utility of $u_{ij}(x^*_{ij}, x^*_{-ij};\cdot) = \theta_{ij} - x^*_{ij}$.

W.l.o.g., let us assume that agent n contributes $x^*_{nj} = c_j - \sum_{i\in\mathcal{A}\setminus\{n\}} x^*_{ij} \leq \bar{x}_{nj}$ s.t. $\sum_{i\in\mathcal{A}} x^*_{ij} = c_j$. Then $\forall j \in \mathcal{P}_m$, we can construct the following two deviations for i:

1. $\hat{x}_{ij} < x^*_{ij}$. For any $\hat{x}_{ij}$ less than x^*_{ij}, the project j is not funded and $R_j(\hat{x}_{ij}) = u_{ij}(\hat{x}_{ij}, x^*_{-ij};\cdot) < u_{ij}(x^*_{ij}, x^*_{-ij};\cdot)$.
2. $\hat{x}_{ij} > x^*_{ij}$. For any $\hat{x}_{ij} > x^*_{ij}$, the project is funded but the funded utility reduces, i.e., $u_{ij}(\hat{x}_{ij}, x^*_{-ij};\cdot) = \theta_{ij} - \hat{x}_{ij} < u_{ij}(x^*_{ij}, x^*_{-ij};\cdot) = \theta_{ij} - x^*_{ij}$.

In both cases, we obtain the following,

$$u_{ij}(x^*_{ij}, x^*_{-ij};\cdot) \geq u_{ij}(\hat{x}_{ij}, x^*_{-ij};\cdot), \forall \hat{x}_{ij}, \forall j.$$

Therefore,

$$\sum_{j \in \mathcal{P}_m} u_{ij}(x^*_{ij}, x^*_{-ij}; \cdot) \geq \sum_{j \in \mathcal{P}_m} u_{ij}(\hat{x}_{ij}, x^*_{-ij}; \cdot), \forall \hat{x}_{ij}$$

From Definition 2.3, x^*_{ij} are PSNE, and every project gets funded. □

Theorem 6.2 implies that, under $SF_{\mathcal{P}_m}$, the socially efficient equilibrium (with $P^\star = \mathcal{P}_m$) is achieved. Intuitively, under $SF_{\mathcal{P}_m}$, combinatorial CC collapses to simultaneous single projects, and thus, we can provide closed-form equilibrium contributions. However, $SF_{\mathcal{P}_m}$ is a strong assumption and, in general, may not be satisfied. In fact, the weaker notion of Budget Surplus itself may not always apply. Therefore, we next study combinatorial CC with Budget Deficit.

6.3 Impossibility of Achieving Socially Efficient Equilibrium for Combinatorial CC under Budget Deficit

We now focus on the scenario when there is *Budget Deficit*, i.e., $\sum_{i \in \mathcal{A}} \gamma_i < \sum_{j \in \mathcal{P}_m} c_j$. In this scenario, only a subset of projects can be funded. Unfortunately, identifying the subset of projects funded at equilibrium is challenging. In CC, the agents decide which projects to contribute to based on their private valuations and available refunds. This circular dependence of the equilibrium contributions and the set of funded projects makes providing analytical guarantees challenging on the funded set. To analyze agents' equilibrium behavior and funding guarantees, we fix our focus on the subset of projects that maximize social welfare, i.e., $P^\star$ (Definition 6.1).

Section Outline. In this section, we first show that funding $P^\star \subset \mathcal{P}_m$ at equilibrium is, in general, not possible for any $R(\cdot)$ satisfying Condition 5.1. Second, we prove that even with the stronger assumption of Subset Feasibility of the optimal welfare set, i.e., $\text{SF}_{P^\star}$, we may not achieve socially efficient equilibrium due to agents' strategic deviations. Last, we

show that computing an agent i''s optimal deviation, given the contributions of the other agents $\mathcal{A} \setminus \{i'\}$, is NP-Hard.

Algorithm 5 Instance with $\mathcal{P}_m = \mathcal{A} = \{1, 2\}$ and fixed $R(\cdot)$

procedure GENERATEVALUES($R(\cdot)$)
  $c_1 \leftarrow \mathbb{R}_+$
  Choose θ_{11} s.t. $\bar{x}_{11} < c_1 < \theta_{11}$ based on $R_1(\cdot)$
  Choose θ_{21} s.t. $\bar{x}_{21} := c_1 - \bar{x}_{11}$
  Set $c_2 = \bar{x}_{21}$, $\theta_{12} = 0$ and choose θ_{22} s.t.
    $\theta_{21} < \theta_{22} < \theta_{11} + \theta_{21} - x^*_{11}$ ▷ $P^\star = \{1\}$
  Set $\gamma_1 := \bar{x}_{11}$ and $\gamma_2 := \bar{x}_{21}$ ▷ *Satisfying* $SF_{P^\star}$
  return θ's, γ's, and c's ▷ s.t. Agent 2 deviates
end procedure

6.3.1 Welfare Optimality at Equilibrium

Consider the following example instance.

Example 6.1: Combinatorial CC Game Instance

Let $\mathcal{P}_2 = \{1, 2\}$ and $\mathcal{A} = \{1, 2\}$. Let $\gamma_1 = 1, \gamma_2 = 0, \theta_{11} = 1, \theta_{12} = 2$ and $\theta_{22} = 1$ with $\theta_{21} = 10$.

In Example 6.1, the maximum funded utility agent 1 can receive from project 1 is 0 and unfunded utility $r_{11} < \theta_{11} = 1$. On the other hand, the agent obtains a utility of 1 when contributing to project 2. Hence, project 2 gets funded at equilibrium, although it is welfare-optimal to fund project 1. Thus, socially efficient equilibrium is not achieved.

Example 6.1 is one pathological case where the agent with high valuation has zero budget, leading to the sub-optimal outcome at equilibrium. Hence, we next strengthen

the assumption on the budgets of the agents. Let $P^\star$ be the non-trivial welfare optimal subset, and we assume that Subset Feasibility is satisfied for $P^\star$, i.e., $\text{SF}_{P^\star}$. Recall that with $\text{SF}_{P^\star}$, we assume that every agent has enough budget to contribute $\bar{x}_{ij}$ in $P^\star$. Theorem 6.3 shows that achieving socially efficient equilibrium may not be possible despite this strong assumption.

Theorem 6.3. *Given an instance of $\mathcal{M}_{CC}$, a unique non-trivial $P^\star \subset \mathcal{P}_m$ may not be funded at equilibrium even with Subset Feasibility for $P^\star$, $SF_{P^\star}$, for any set of $(R_j)_{j\in P}$ satisfying Condition 5.1.*

Proof. We provide a proof by construction for $n = 2$ and $m = 2$ where one agent is incentivized to deviate when $P^\star$ is funded. Procedure 5 presents the general steps to construct the instance. Informally, we first select the target cost of the project 1, i.e., c_1, to be any positive real value. Given $(R_j)_{j\in\mathcal{P}_m}$ under Condition 5.1, we can always find an $\bar{x}_{11}$ for agent 1. At $\bar{x}_{11}$, funded utility equals unfunded utility. Trivially, $\bar{x}_{11} < \theta_{11}$. We now prove that the construction defined in Procedure 5 is always possible.

Line 3. We select θ_{11} such that $x^*_{11} < c_1 < \theta_{11}$. We are required to prove that such a θ_{11} will always exist. When $\theta_{11} = c_1$, it must be true that, $x^*_{11} < \theta_{11} = c_1$. If $x^*_{11} = c_1$, then $R(x^*_{11}) = 0$ (i.e., funded utility equals unfunded utility), since $R(0) = 0$ by construct, contradicting Condition 5.1. Hence $x^*_{11} < c_1$, when $\theta_{11} = c_1$. As x^*_{11} is continuous w.r.t. θ_{11}, we can always chose $\theta_{11} > c_1$ s.t. $x^*_{11} < c_1$.

Line 4. We choose θ_{21} s.t., $x^*_{21} = c_1 - x^*_{11}$. It is possible since $c_1 - x^*_{11} > 0$.

Line 5. We let $c_2 = x^*_{21}$ and $\theta_{12} = 0$. Then, select $\theta_{22} < \theta_{11} + \theta_{21} - x^*_{11}$. Observe that, we can always select $\theta_{22} > \theta_{21}$ since $\theta_{11} - x^*_{11} > 0$.

Given the above, and the values $c_2 = c_1 - x^*_{11}$ and $\theta_{12} = 0$; we arrive at $\theta_{11} + \theta_{21} - c_1 > \theta_{12} + \theta_{22} - T_2$. Hence, $P^\star = \{1\}$, i.e., the welfare optimal subset is $\{1\}$ and is unique.

Line 6. To ensure $SF_{P^\star}$, i.e., Subset Feasibility of $P^\star$, we set $\gamma_1 = x^*_{11}$ and $\gamma_2 = x^*_{12}$. Note that, $\gamma_1 + \gamma_2 = c_1 < c_1 + c_2$, Budget Deficit is also satisfied. Thus, Procedure 5 returns valid instance, say $\mathcal{M}'$, of $\mathcal{M}_{CC}$.

Next, note that, in $\mathcal{M}'$, agent 1 contributes $\bar{x}_{11}$ and agent 2 must contribute $\bar{x}_{21}$ to fund project 1 and to achieve socially efficient equilibrium. If agent 2 does not deviate, it will receive a utility $u_2 = \theta_{21} - \bar{x}_{21}$. However, let agent 2 unilaterally deviate and contribute $\bar{x}_{21}$ to project 2. As $c_2 = \bar{x}_{21}$, project 2 gets funded, and agent 2 obtains a utility of $u'_2 = \theta_{22} - \bar{x}_{21}$. From Line 5 in Procedure 5, we know that $u'_2 > u_2$, hence agent 2 will deviate, implying $P^\star = \{1\}$ is not funded at equilibrium for the $\mathcal{M}'$ game. □

Illustration of Procedure 5 for PPR To further clarify the impossibility presented in Theorem 6.3, we demonstrate Procedure 5 when $(R_j)_{j\in\mathcal{P}_m}$ is the PPR refund scheme (Eq. 2.15). As required, consider a setting with $\mathcal{P}_m = \{1, 2\}$ and $\mathcal{A} = \{1, 2\}$. Let $c_1 = 10$ and $B_1 = 1$. We choose $\theta_{11} = 10.9$ which implies $\bar{x}_{11} = 9.91$ using Theorem 2.8. That is, $\bar{x}_{11} < c_1 < \theta_{11}$. Now, fix $\bar{x}_{21} = c_1 - \bar{x}_{11} = 0.99$. This also implies $c_2 = 0.99$. The upper bound on the equilibrium contribution, $\bar{x}_{21} = 0.99$, is possible for the value $\theta_{21} = 1.089$. Select $\theta_{22} = 1.9$, to get $\theta_{22} > \theta_{21}$. This also implies $P^\star = \{1\}$, i.e., $\theta_{11} + \theta_{21} - c_1 = 1.989$ and $\theta_{12} + \theta_{22} - c_2 = 0.91$. Despite this, agent 2 will deviate since $\theta_{22} - c_2 > \theta_{21} - \bar{x}_{21}$.

6.3.2 CCC with Budget Deficit: Optimal Strategy

Theorem 6.3 implies that P^* may not be funded at equilibrium even when P^* satisfies Subset Feasibility. In other words, w.l.o.g., an agent i' may be incentivized to deviate from any strategy that funds P^*. Motivated by such a deviation, we now address the question: *Given the total contribution by $N \setminus \{i'\}$ agents towards each project j, can the agent i' compute its optimal strategy?* We answer this question by (i) showing that such an optimal strategy may not exist if an agent's contribution space is continuous, i.e., $x \in \mathbb{R}_+$, and (ii) if contributions are discretized, then computing the optimal strategy is NP-Hard.

$$\max_{(x_{i'j})_{j\in\mathcal{P}_m}} \sum_{j\in\mathcal{P}_m} z_{i'j}\cdot(\theta_{i'j}-x_{i'j})+(1-z_{i'j})\cdot R(x_{i'j},\cdot)$$

$$\textbf{s.t.} \sum_{j\in\mathcal{P}_m} x_{i'j} \le \gamma_{i'} \qquad \text{// Budget Constraint}$$

$$x_{i'j} \le c_j - \mathbf{x}_{j-i'}, \forall j \qquad \text{// Remaining Contribution}$$

$$\left.\begin{array}{l}(x_{i'j}-c_j+\mathbf{x}_{j-i'})\cdot z_{i'j}\ge 0,\forall j\\ x_{i'j}-c_j+\mathbf{x}_{j-i'} < z_{i'j},\forall j\\ z_{i'j}\in\{0,1\},\forall j\end{array}\right\} \qquad \text{// Defining Indicator Variable}$$

Figure 6.2: MIP-CC: Mixed Integer Program to calculate Agent i''s optimal strategy given the contributions of the remaining agents $\mathcal{A}\setminus\{i'\}$.

6.3.2.1 MIP-CC: Mixed Integer Program for CC

We first describe the general optimization for an agent i' to compute its optimal strategy (i.e., contribution). Assume that the agents $\in \mathcal{A}\setminus\{i'\}$ have contributed denoted by $\mathbf{x}_{j-i'}$. For agent i''s optimal strategy, we need to maximize its utility given $xc_j - \mathbf{x}_{j-i'}$, $\forall j \in P$ and other variables such as the refund scheme R_j and bonus budget B_j. Figure 6.2 presents the formal MIP, namely *MIP-CC*, which follows directly from agent i' utility (Definition 6.2).

6.3.2.1.1 MIP-CC: Optimal Strategy May Not Exist.

We now show that MIP-CC (Figure 6.2) may not always admit well-defined contributions.

Example 6.2: Combinatorial CC Game Instance

Let $\mathcal{P}_3=\{1,2,3\}$ and $\mathcal{A}=\{1,2\}$ s.t. both agents are *identical*, i.e., each $i\in\mathcal{A}$ has the same value θ for each $j\in\mathcal{P}_m$ and $\gamma_1=\gamma_2$. Additionally $\forall j\in\mathcal{P}_m$, $c_j=c$ and $B_j=\vartheta-c$,. Let the agents have budget s.t. $\gamma_i=\bar{x}_{i1}$ $\forall i\in\mathcal{A}$, where $\bar{x}_{i1}$ is the upper bound equilibrium contribution for the single project case (see Theorem 2.8).

Theorem 6.4. *Given an instance of $\mathcal{M}_{CC}$ and for any set of $(R_j)_{j\in\mathcal{P}_m}$ satisfying Condition 5.1, an agent i''s optimal strategy may not exist.*

Proof. From Example 6.2, let agent 1 contribute its entire budget to project 1, i.e., $c_1 - \mathbf{x}_1 = \bar{x}_{11}$ and $\mathbf{x}_2 = 0$. Now we solve MIP-CC for agent 2. Since $\gamma_i = \bar{x}_{i1}$, $\forall i$ and $c_j = \bar{x}_{1j} + \bar{x}_{2j}$, $\forall j$ (as $B_j = \vartheta_j - c_j$), the overall budget is enough to fund only a single project. That is, as agent 1 has contributed to project 1, agent 2 can (at best) fund only project 1. So the set of indicator variable zs in MIP-CC can be either $z_1 = \{1, 0, 0\}$ or $z_2 = \{0, 0, 0\}$.

Computing Optimal Strategy. We have the following cases:

- $z = \{1, 0, 0\}$: This is possible when agent 2 contributes $x_{21} = \gamma_2$ to project 1, resulting in project 1's funding. The utility agent 2 gets is $\theta - \bar{x}_{21}$.
- $z = \{0, 0, 0\}$: Agent 2 may opt to contribute to all the 3 projects to grab the maximum refund. As the refund shares from $R_j(\cdot)$ sum exactly to B_j, agent 2 can grab the entire bonus from projects 2 and 3 by contributing any arbitrary positive value, $\epsilon > 0$. However, agent 2 must contribute ϵ less in project 1 since, $\gamma_2 = \bar{x}_{21}$. Hence, project 1 becomes unfunded. For optimal strategy, agent 2 must maximize the refund from project 1, i.e., $\max_{x_{21}} R_1(x_{21}, \cdot)$ s.t. $x_{21} < c_1 - \mathbf{x}_1$. Therefore, x_{21} is not well-defined as $R_1(x_{21}, \cdot)$ satisfies CM. So the overall utility by deviating for agent 2 becomes $2B + R(x_{21} - \epsilon, \cdot)$ where an optimal ϵ is not defined.

This proves the theorem. □

MIP-CC-D. To overcome the above non-existence, we discretize the contribution space. More concretely, an agent i can contribute $\kappa \cdot \delta$ where $\kappa \in \mathbb{N}^+$ and δ the smallest unit of contribution. With this restriction on an agent's contribution, the search space in MIP-CC (Figure 6.2) becomes discrete and finite. Consequently, agent i''s optimal strategy always exists. To distinguish MIP-CC with a discrete contribution space, we refer to it as *MIP-CC-D*.

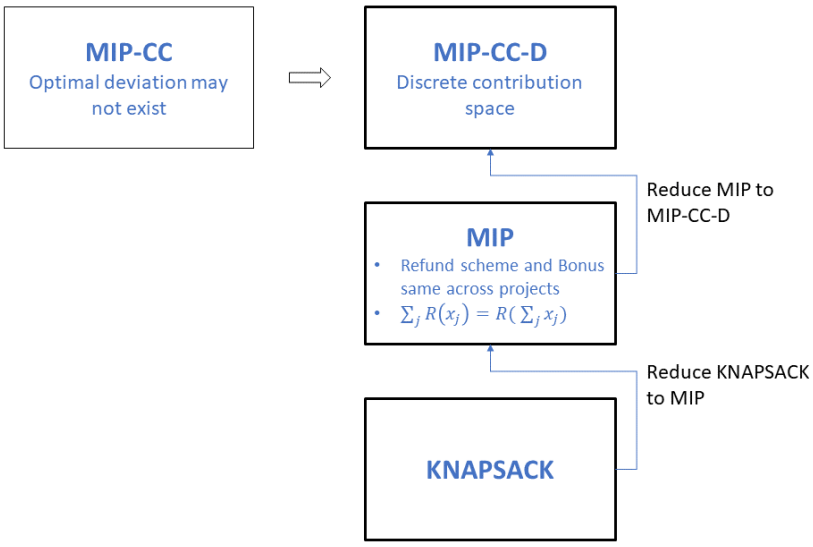


Figure 6.3: Overview of the Proof of Theorem 6.5

6.3.2.2 MIP-CC-D: Finding Optimal Strategy is NP-Hard

We now show that solving MIP-CC for discrete contributions (i.e., MIP-CC-D) is NP-hard. Before that, we first state the decision version of the problem.

Decision Version: For MIP-CC-DD, w.l.o.g. dropping the agent subscript i', the decision version is as follows: given $m = |\mathcal{P}_m|$ decision variables $(z_1, \ldots, z_m)$ and contributions $(x_1, \ldots, x_m)$, the agent budget γ, the social welfare value V, is there a subset $Q = \{j | z_j = 1,\ \forall j \in \mathcal{P}_m\}$ such that (i) $\sum_{j \in Q} \vartheta_j \geq V$, (ii) $\sum_{j \in Q} z_j \cdot x_j \geq \gamma$, the remaining contribution condition (iii) $x_j \leq c_j - \mathbf{x}_j,\ \forall j \in \mathcal{P}_m$, and the indicator variable conditions (iv) $(x_j - c_j + \mathbf{x}_j) \cdot z_j \geq 0,\ \forall j \in \mathcal{P}_m$ and (v) $x_j - c_j + \mathbf{x}_j < z_j,\ \forall j \in \mathcal{P}_m$.

Claim 6.2. *MIP-CC-D is in NP.*

Proof. From the decision version of MIP-CC-D, we can trivially see that the problem is in NP. The proof is the set of decision variables $(z_1, \ldots, z_m)$ and contributions $(x_1, \ldots, x_m)$

chosen post which computing the conditions (i) to (v) outlined in the decision variable takes polynomial time. □

For the hardness proof, the main point to note is that the reduction and the computational hardness follow from a known NP-Hard problem to the problem in consideration. That is, to show that MIP-CC-D is NP-Hard, we reduce the known KNAPSACK problem to an instance of MIP-CC-D. Figure 6.3 depicts an overview of the proof.

Theorem 6.5. *Given an instance of $\mathcal{M}_{CC}$ with discrete contributions and for any set of $(R_j)_{j\in\mathcal{P}_m}$ satisfying Condition 5.1, computing optimal strategy for agent i', given the contributions of $\mathcal{A}\setminus\{i'\}$, is NP-Hard.*

Proof. For completeness, we first re-write MIP-CC-D (see Figure 6.2) for discrete contributions. Let $D = \{\kappa\cdot\delta \mid \kappa\in\mathbb{N}\}$ denote the discrete contribution set where δ is the smallest unit of contribution. Now, the following optimization defines MIP-CC-D for agent i' without the sub-script "i'". Note that Eq. 6.2 is merely the optimization presented in Figure 6.2 with the added constraint on the contribution set.

$$\left.\begin{array}{ll} \max_{(x_j)_{j\in\mathcal{P}_m}} & \sum_{j\in P} z_j\cdot(\theta_j - x_j) + (1-z_j)\cdot R(x_j,\cdot) \\ \textbf{s.t.} & \sum_{j\in\mathcal{P}_m} x_j \le \gamma \\ & x_j \le c_j - \mathbf{x}_{j-i'}, \forall j \\ & (x_j - c_j + \mathbf{x}_{j-i'})\cdot z_j \ge 0,\ \forall j \\ & x_j - c_j + \mathbf{x}_{j-i'} < z_j, \forall j \\ & z_j\in\{0,1\}, x_j\in D, \forall j \end{array}\right\} \tag{6.2}$$

We first construct an optimization problem MIP, which we show is equivalent to the KNAPSACK problem in Part A. In Part B, we show that the MIP reduces to MIP-CC-D, implying MIP-CC-D is also NP-Hard.

Part A: Designing MIP (Eq. 6.3). For our proof, we consider a refund scheme $R_j(\cdot) = R(\cdot)$ and bonus budget $B_j = B$, which is the same for each project j. $R(\cdot)$ satisfies CM (Condition 5.1) such that the refund share satisfies $\sum_j R(x_j, \cdot) = R(\sum_j x_j, \cdot)$. [3] Now, we define the MIP as follows.

$$\left.\begin{array}{ll} & \max_{z=(z_1,\ldots,z_m)} \sum_{j\in\mathcal{P}_m} z_j(\theta_j - r_j) + R(\gamma - \sum_{j\in P} z_j r_j, \cdot) \\ \textbf{s.t.:} & \sum_j z_j r_j \leq \gamma \text{ and } z_j \in \{0,1\} \end{array}\right\} \tag{6.3}$$

We now prove that the MIP defined in Eq. 6.3 is NP-Hard. The MIP can be re-written as,

$$\begin{aligned} & \max_{z=(z_1,\ldots,z_m)} \sum_{j\in\mathcal{P}_m} z_j(\theta_j - r_j) + R(\gamma - \sum_{j\in\mathcal{P}_m} z_j r_j, \cdot) \\ = & \max_{z=(z_1,\ldots,z_m)} \sum_{j\in\mathcal{P}_m} z_j(\theta_j - r_j) - \sum_{j\in P} z_j R(r_j) + R(\gamma) \\ & \qquad \left(\text{By } \sum_j R(x_j,\cdot) = R(\sum_j x_j, \cdot),\ z_j \in \{0,1\}.\right) \\ = & \max_{z=(z_1,\ldots,z_m)} \sum_{j\in\mathcal{P}_m} z_j(\theta_j - r_j - R(r_j)) + R(\gamma) \\ = & \max_{z=(z_1,\ldots,z_m)} \sum_{j\in\mathcal{P}_m} z_j(\theta_j - r_j - R(r_j)) \end{aligned}$$

We reduce the MIP problem from the NP-complete KNAPSACK problem: given m items with weights $w_1, \cdots, w_m$ and value $s_1, \cdots, s_m$, capacity B and value V, does there exist a subset $Q \subseteq \{1, \cdots, m\}$ such that $\sum_{j\in Q} w_j \leq B$ and $\sum_{j\in Q} s_j \geq V$? Given an instance of the KNAPSACK problem, we build an instance of the above MIP as follows:

- The set of projects is the set of items. The amount left for funding the project is $r_j = w_j$. The budget of the agent i' is $\gamma = B$.

- The value of each project to the agent is, $\theta_j - r_j - R(r_j) = s_j$, i.e., $\theta_j - R(r_j) = s_j + w_j$.

[3] An example of such a refund scheme can be $R(\cdot) = \frac{x \cdot B_{\min}}{\vartheta_{\max}}$ where $B_{\min} = \min_j(\vartheta_j - c_j)$ and $\vartheta_{\max} = \max_j \vartheta_j$.

We can see that the utility obtained by choosing a set of projects $Q = \{j|z_j = 1\}$ is exactly equal to the value of choosing set of items Q in the KNAPSACK problem i.e, $\sum_{j\in Q} \theta_j - r_j - R(r_j) = \sum_{j\in Q} s_j$. Also note that the budget constraint is satisfied if and only if the capacity constraint is satisfied. It follows that a solution with value at least V exists in the KNAPSACK problem if and only if there exists a set of projects whose social welfare in the above instance is at least V.

Thus, we reduce MIP from KNAPSACK, implying MIP is NP-Hard. We next show that MIP in Eq. 6.3 also reduces to MIP-CC-D.

Part B: Reducing MIP to MIP-CC-D. Before we show the reduction, we define the following handy notations for the objectives of MIP-CC-D and MIP.

$$F(z,x) = \sum_{j\in\mathcal{P}_m} z_j \cdot (\theta_j - x_j) + \sum_{j\in\mathcal{P}_m} (1 - z_j) \cdot R(x_j, \cdot) \qquad \text{"MIP-CC-D"}$$

$$G(z) = \sum_{j\in\mathcal{P}_m} z_j \cdot (\theta_j - r_j) + R(\gamma - \sum_{j\in\mathcal{P}_m} z_j r_j, \cdot) \qquad \text{"MIP in Eq. 6.3"}$$

Further, let (Z^F, X^F) denote the tuple of feasible solutions to MIP-CC-D, and Z^G denote the set of feasible solutions to MIP.

Given an instance of MIP, we construct an instance of MIP-CC-D using the following conditions,

P1. For each project j, we have $R_j(\cdot) = R(\cdot)$ s.t. $\sum_j R(x_j, \cdot) = R(\sum_j x_j, \cdot)$.

P2. Let the remaining contribution be $C_{j-i'} = c_j - r_j,\ \forall j \in \mathcal{P}_m$.

Using P1 and P2, we now show that any $(z^*, x^*) \in OPT_{MIP-CC-D}$ implies that $z^* \in OPT_{MIP}$.

Claim 6.3. *For the specific setting defined by P1-P2 and given (Z^F, X^F) and Z^G as the tuple of feasible solutions to MIP-CC-D and MIP in Eq. 6.3, respectively, we have $Z^G \subseteq Z^F$.*

Proof. For the proof, we show that given any solution $z \in Z^G$, we can also construct a solution $(z, x) \in (Z^F, X^F)$. That is, given z, we can construct x as follows. Let $x_j = r_j$ if $z_j = 1$ and $x_j = 0$ if $z_j = 0$.

We now have to show that x's construction does not break the feasibility of a solution in MIP-CC-D. Observe that, from Eq. 6.2,

- Trivially, we have $\sum_j x_j = \sum_j z_j r_j \leq \gamma$ and $x_j \leq r_j = c_j - \mathbf{x}_{j-i'}$. That is, the budget constraint and the project's funding conditions are satisfied.

- If $z_j = 0$, then both $(x_j - c_j + \mathbf{x}_{j-i'}) \cdot z_j \geq 0$ and $x_j - c_j + \mathbf{x}_{j-i'} < z_j$ hold.

- Likewise, if $z_j = 1$ then both $(x_j - c_j + \mathbf{x}_{j-i'}) \cdot z_j \geq 0$ and $x_j - c_j + \mathbf{x}_{j-i'} < z_j$ hold.

Hence for every $z \in Z^G$ we can construct (z, x) s.t., $(z, x) \in (Z^F, X^F)$. Hence, every solution that is feasible for MIP is also feasible for MIP-CC-D. □

With this, consider the following claim.

Claim 6.4. *For the specific setting defined by P1-P2 and any fixed $z \in Z^F$, let $\hat{x}^z_j = r_j$ where $z_j = 1$ and $r_j = c_j - \mathbf{x}_{j-i'}$ and $\sum_j \hat{x}^z = \gamma$. Then $\exists \hat{x}^z \in X^F$, such that $F(z, \hat{x}^z) \geq F(z, x)$, $\forall x \in X^F$.*

Proof. Given a fixed $(z, x) \in (Z^F, X^F)$, observe that,

$$
\begin{aligned}
F(z,x) &= \sum_{j \in \mathcal{P}_m} z_j \cdot (\theta_j - x_j) + \sum_{j \in \mathcal{P}_m} (1 - z_j) \cdot R(x_j, \cdot) \\
&= \sum_{j \in \mathcal{P}_m} z_j(\theta_j - x_j) + R(\sum_{j \in \mathcal{P}_m} (1 - z_j)x_j, \cdot) \qquad \left(\text{As } \sum_k R(x_k, \cdot) = R(\sum_k x_k, \cdot)\right) \\
&= \sum_{j \in \mathcal{P}_m} z_j(\theta_j - x_j) + R(\sum_{j \in \mathcal{P}_m} x_j - \sum_{j \in \mathcal{P}_m} z_j x_j, \cdot) \\
&\leq \sum_{j \in \mathcal{P}_m} z_j(\theta_j - x_j) + R(\gamma - \sum_{j \in P} z_j x_j, \cdot) \qquad \left(\text{As } \sum_j x_j \leq \gamma \text{ and } R \text{ satisfies CM}\right) \\
&\leq \sum_{j \in \mathcal{P}_m} z_j(\theta_j - r_j) + R(\gamma - \sum_j z_j x_j, \cdot) \qquad (\text{From Eq. 6.2},\ x_j < r_j \implies z_j = 0) \\
&= \sum_{j \in \mathcal{P}_m} z_j(\theta_j - r_j) + R(\gamma - \sum_{j \in \mathcal{P}_m} z_j r_j, \cdot) = F(z, \hat{x}^z).
\end{aligned}
$$

Hence, $F(z, \hat{x}^z) = \sum_{j \in \mathcal{P}_m} z_j(\theta_j - r_j) + R(\gamma - \sum_j z_j r_j, \cdot) \geq F(z, x)$ where $\hat{x}^z_j = r_j$ if $z_j = 1$ and $\sum_j \hat{x}^z_j = \gamma$.

To conclude the proof, we now show that the contribution set is also feasible. To this end, observe that from the claim statement, we have $\sum_j \hat{x}^z = \gamma$. Further, since for each $z_j = 1$, we have by construction $\hat{x}^z_j = r_j \leq c_j - \mathbf{x}_{j-i'}$ and for each $z_j = 0$ we have $\hat{x}^z_j < r_k$, the second feasibility condition is also met. □

We can trivially observe that $G(z) = F(z, \hat{x}^z)$ where $\hat{x}^z_j = r_j$ if $z_j = 1$ and $\sum_j \hat{x}^z_j = \gamma$. Let $(z^*, x^*) \in OPT_{MIP-CC-D}$, then $F(z^*, x^*) \geq F(z^*, \hat{x}^{z^*})$ and from Claim 6.4, $F(z^*, \hat{x}^{z^*}) \geq F(z^*, x^*)$, hence $x^* = \hat{x}^{z^*}$. That is,

$$G(z^*) = F(z^*, x^*) = \max_{z \in Z^F} F(z, \hat{x}^{z^*}) = \max_{z \in Z^F} G(z).$$

Given that (i) the feasibility set of MIP is a subset of the feasibility set of MIP-CC-D and (ii) z^* is also feasible to MIP since $\sum_j x^*_j \leq \gamma$ and $x^*_j \leq r_j,\ \forall j$, we have $\sum_j z^*_j r_j \leq \gamma$. Hence, z^* is also optimal for MIP.

In summary, Part B shows that MIP in Eq. 6.3 reduces to MIP-CC-D. That is, any solution to MIP-CC-D can be used to determine a solution to MIP in Eq. 6.3. Hence, MIP-CC-D is also NP-Hard. □

The following corollary follows from Theorem 6.5.

Corollary 6.2. *Given an instance of $\mathcal{M}_{CC}$ with discrete contributions and for any set of $(R_j)_{j\in\mathcal{P}_m}$ satisfying Condition 5.1, if all agents except i', $N \setminus \{i'\}$, follow a specific strategy that funds $P^\star \subset \mathcal{P}_m$, then computing the optimal deviation for agent i' is NP-Hard.*

Proof. Let $\mathcal{A}\setminus\{i'\}$ agents follow a specific strategy and contribute to the projects s.t. $P^\star$ is not funded. For any such strategy, given $\mathcal{A}\setminus\{i'\}$ contributions, agent i''s optimal deviation is again given by MIP-CC-D. Hence, NP-Hardness follows directly from Theorem 6.5. □

6.4 Experiments

Motivation. Theorem 6.3 proves that the optimal subset $P^\star$ may not be funded at equilibrium due to agents' strategic deviations. However, computing an agent's optimal deviation is also NP-Hard (Corollary 6.2). These observations highlight that computing closed-form equilibrium strategies in Budget Deficit Combinatorial CC, similar to Theorem 2.8 and Theorem 6.2, for agents is challenging. Given this challenge and the hardness of strategic deviations, agents may employ heuristics to increase utility [311, 180]. Next, we propose five heuristics for agents to employ in practice and study their impact on agent utilities and the generated welfare.

6.4.1 Heuristics and Performance Measures

Heuristics. Given the conflict between agent utilities and $P^\star$'s funding (Theorem 6.3), we propose the following heuristics for agent $i \in \mathcal{A}$, for each project $j \in \mathcal{P}_m$, to employ in practice and observe their utility vs. welfare trade-off.

1. *Symmetric*: $x_{ij} = \min(\theta_{ij}, \gamma_i/m)$
2. *Weighted*: $x_{ij} = \left(\frac{\theta_{ij}}{\sum_{k \in \mathcal{P}_m} \theta_{ik}}\right) \gamma_i$
3. *Greedy-θ*: Greedily contribute $x_{ij} = \bar{x}_{ij}$ in descending order of the projects sorted by $\theta_{ij}, \forall j$
4. *Greedy-ϑ*: Greedily contribute $x_{ij} = \bar{x}^*_{ij}$ in descending order of the projects sorted by $\frac{\vartheta_j}{c_j}, \forall j$
5. *OptWelfare*: $x_{ij} = \bar{x}_{ij}$, $\forall j \in P^\star$ and evenly distribute the remaining budget across $\mathcal{P}_m \setminus P^\star$

Agents contribute the minimum amount of what is specified by the five heuristics and the amount left to fund the project. OptWelfare is the *baseline* (preferred) heuristic since it generates optimal welfare, i.e., funds $P^\star$.

Performance Measures. To study the welfare vs. agent utility trade-off, we consider the following performance measures: (i) *Normalized Social Welfare* (SW_N) – Ratio of the welfare obtained and the welfare from $P^\star$ and (ii) *Normalized Agent Utility* (AU_N) – Ratio of the agent utility obtained w.r.t. to the utility when each agent has enough budget to play its PPR contribution $\forall\ j \in \mathcal{P}_m$ (see Theorem 2.8).

We compare the heuristics when $\alpha \in (0, 1]$ fraction of the total agents *deviate*, i.e., choose heuristic $\in$ {Symmetric, Weighted, Greedy-θ, Greedy-ϑ}. The remaining $1 - \alpha$ fraction of agents use the baseline OptWelfare.

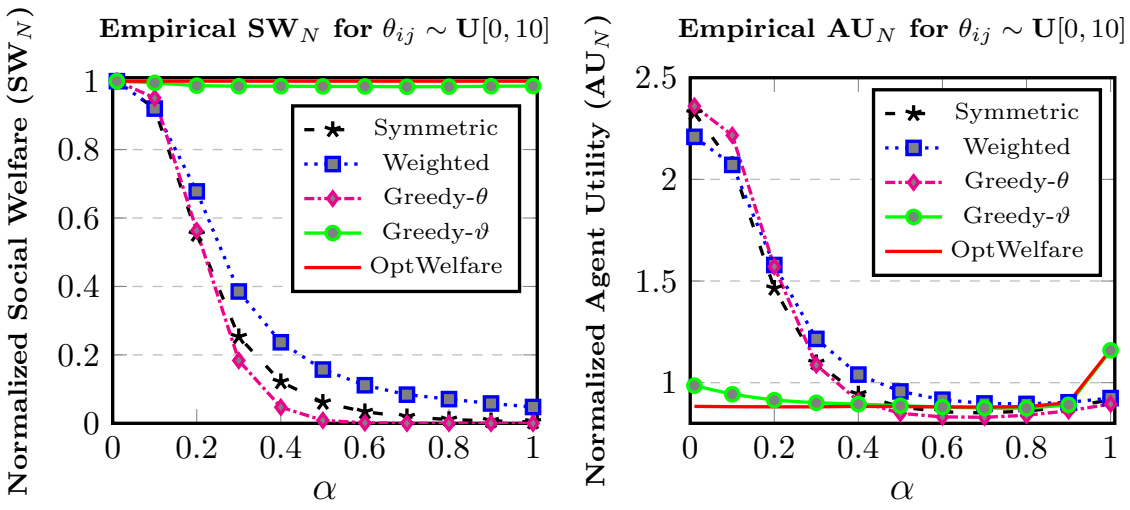


Figure 6.4: Empirical SW_N and AU_N for $\theta_{ij} \sim \mathbf{U}[0, 10]$

6.4.2 Simulation Setup and Results

Setup. We simulate the combinatorial CC game with $n = 100$, and $m = 10$ and PPR refund scheme (Eq. 2.15).[4] We sample θ_{ij}s, for each $i \in N$ and $j \in P$, using (i) Uniform Distribution, i.e., $\theta_{ij} \sim \mathbf{U}[0, 10]$, and (ii) Exponential Distribution, i.e., $\theta_{ij} \sim \mathrm{Exp}(\lambda = 1.5)$. Here, λ is the rate parameter. When $\theta \sim \mathbf{U}[0, 10]$, we get agents whose per-project valuations differ significantly. The agents have approximately similar per-project valuations for $\theta \sim \mathrm{Exp}(\lambda = 1.5)$.

We ensure $\vartheta_j > v_j$ and $B_j \in (0, \vartheta_j - v_j]$ for each project j and that the properties Budget Deficit and Subset Feasibility for $P^\star$ are satisfied. We run each simulation across 100k instances and observe the average SW_N and AU_N for each of the five heuristics. We depict our observations with Figure 6.4, Figure 6.5 and Figure 6.6 when $\theta_{ij} \sim \mathbf{U}[0, 10]$ and $\theta_{ij} \sim \mathrm{Exp}(\lambda = 1.5)$.

[4]The experimental trends presented remain same for different (n, m) pairs, such as $(50, 10)$, $(500, 10)$, and $(500, 20)$.

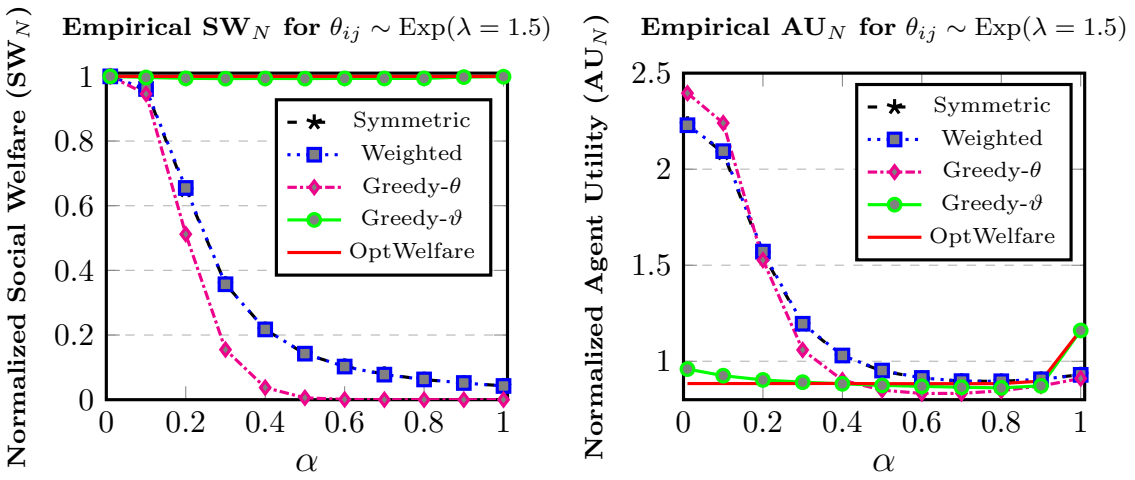


Figure 6.5: Empirical SW_N and AU_N for $\theta_{ij} \sim \text{Exp}(\lambda = 1.5)$

Average SW_N and AU_N. Figure 6.4 depicts the results when $\theta_{ij} \sim \mathbf{U}[0, 10]$. We make three main observations. First, deviating from the baseline heuristic (OptWelfare) is helpful only when a few agents deviate, i.e., for smaller values of α. Despite such a deviation, we observe that the corresponding decrease in social welfare is marginal. On the other hand, the increase in α reduces the amount of the contributions, and the projects remain unfunded, reducing the social welfare and the agent utilities. Second, deviating from OptWelfare always increases the average agent utility – at the cost of the overall welfare. Third, Greedy-ϑ almost mimics OptWelfare for both SW_N and AU_N.

AU_N for Deviating vs. Non-deviating Agents. In Figure 6.6, we compare the average utility for the agents who deviate versus those who do not. We let $\alpha = 0.2$ fraction of the agents deviate and follow the other four heuristics. From Figure 6.6, we observe that upon deviating to Symmetric, Weighted or Greedy-θ, the $\alpha = 0.2$ fraction of agents obtains higher AU_N (red grid bars) compared to the remaining non-deviating agents who do not deviate (green grid bars). In contrast, Greedy-ϑ shows non-deviation to be beneficial. Since Greedy-ϑ performs close to OptWelfare, the AU_N for deviation remains low compared to OptWelfare. Crucially, the deviation is not majorly helpful when many agents deviate.

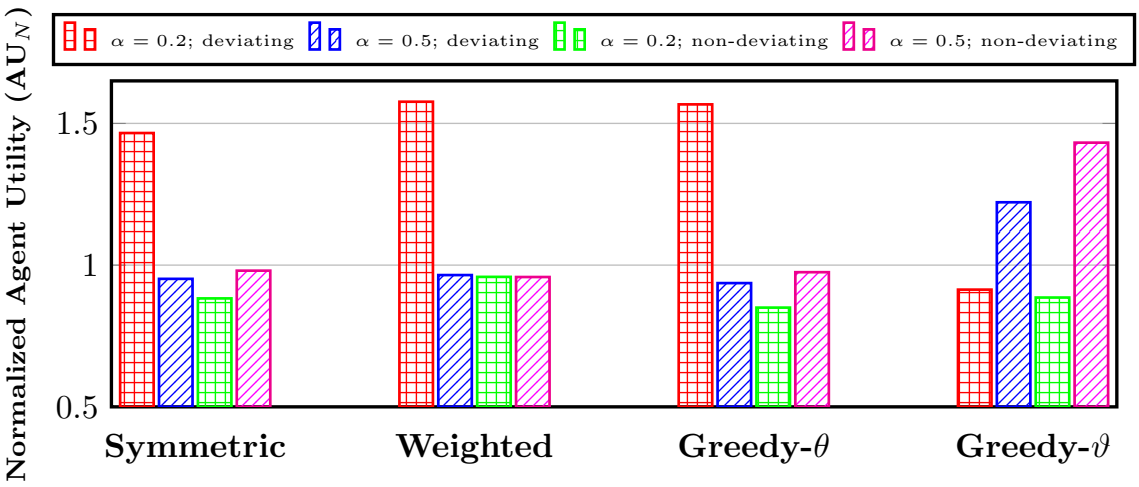


Figure 6.6: AU_N for α fraction of Agents Deviating vs. $1-\alpha$ fraction Non-deviating

When $\alpha = 0.5$, we see comparable average AU_N for agents who deviate and those who do not (blue-lined vs. magenta-lined bars, respectively). While deviating to Greedy-ϑ remains non-beneficial.

6.5 Discussion and Conclusion

Discussion and Future Work. From Figures 6.4 and 6.6, we see that Greedy-ϑ performs similar to OptWelfare (which funds $P^\star$). Thus, as the number of projects p increases, to maximize social welfare, it may be beneficial for the agents to adopt Greedy-ϑ instead of deriving sophisticated strategies based on $P^\star$ (since computing $P^\star$ is NP-Hard).

Generally, it is challenging to determine PSNE contributions for combinatorial CC with budgeted agents. We propose four heuristics and study their welfare and agent utility trade-offs. Future work can explore other heuristics that achieve better trade-offs and welfare guarantees. One can also study strategies that perform better on average, such as Bayesian Nash Equilibrium. The experiments show that deviating from OptWelfare may increase agent utility. Thus, one can explore strategies such as ϵ-Nash Equilibrium, which

approximates a worst-case ϵ increase in utility with unilateral deviation. Approximate strategies may also be desirable since finding optimal deviation is NP-Hard. However, the approximation must provide a desirable trade-off between agent utility and welfare..

Conclusion. This chapter focuses on the funding guarantees of the projects in combinatorial CC. Based on the overall budget, we categorize combinatorial CC into (i) Budget Surplus and (ii) Budget Deficit. First, we prove that Budget Surplus cannot guarantee projects' funding at equilibrium. Introducing the stronger criteria of Subset Feasibility guarantees the projects' funding at equilibrium under Budget Surplus. However, for Budget Deficit, we prove that the optimal welfare subset's funding can not be guaranteed at equilibrium despite Subset Feasibility. Next, we show that computing an agent's optimal strategy (and consequently, its optimal deviation), given the contributions of the other agents, is NP-Hard. Lastly, we propose specific heuristics and observe the empirical trade-off between agent utility and social welfare.

Chapter 7

Analyzing Crowdfunding of Public Projects Under Dynamic Beliefs

Our discussion of Civic Crowdfunding (CC) has focused on the role of refunds in mechanisms like PPR [312] or PPRx (Chapter 4). However, despite their attractive incentive properties, these mechanisms assume that the agent's beliefs about the project getting funded do not change over time, i.e., their beliefs are static. Unlike the known CC mechanisms in this chapter, we model the evolution of agents' beliefs as a random walk. We study PPRx in this dynamic belief setting and refer to it as PPRx-DB for readability. We prove that in PPRx-DB, the project is funded at equilibrium. More significantly, we prove that under certain conditions on agent's belief evolution, agents will contribute as soon as they arrive at the mechanism. Thus, we believe that by incorporating dynamic belief evolution in analysis, the social planner can mitigate the concern of race conditions in many mechanisms.

⋆ ⋆ ⋆ ⋆ ⋆

7.1 Introduction

Generally, Civic Crowdfunding (CC) is conducted in the following two modes: (i) *Offline*: in which the participating agents are not aware of the history of the contributions and the net contribution at any epoch. (ii) *Online*: where the net and the history of contributions are visible to each participating agent (e.g., [156, 120, 294]). We refer to crowdfunding over online settings as *sequential crowdfunding*.

Particularly for sequential crowdfunding, *blockchain*-based online are becoming popular, as first discussed in Chapter 5. Blockchain is an immutable, decentralized, and public ledger [203]. These ledgers allow pseudo-anonymous, transparent, and verifiable payments while eliminating the middle person. As we have already seen in Chapter 5, CC mechanisms are now also being deployed as *smart contracts* over publicly distributed ledgers such as the *Ethereum blockchain* [264, 290]. Carrying out transactions on Etheurem incurs *gas* (a form of payment), and thus, there is a need to design efficient mechanisms for sequential crowdfunding. We refer the reader to Chapter 3.2.2.3 for a brief overview of Ethereum and gas costs.

For an offline setting, PPR is an excellent choice. However, PPR induces a simultaneous game ([52] or Claim 2.4). In sequential crowdfunding, such a game result in the agents deferring their contribution until the deadline, which in turn may result in the project not getting funded [52, 47], i.e., a "race" condition (RC). Chapter 5 studies various aspects of refund schemes to avoid the race condition and for efficient deployment in blockchain-based online settings.

Information Structure (Chapter 4). In addition to the above, CC also depends on the information available to the participating agents. To capture this, we define the tuple consisting of each agent's (i) valuation and (ii) belief as its *information structure*. The existing literature majorly assumes that each agent is interested in the funding of the public project, i.e., its *valuation* towards the project's funding is non-negative. Additionally, the literature also assumes that each agent has *symmetric* belief, i.e., each agent believes that

the public project will be funded with probability 1/2 and not with 1/2. Note that in the real world, the beliefs may be asymmetric. Chapter 4 presents PPRx (which leverages PPR) for public projects when information structure allows positive valuation with asymmetric, yet *static*, beliefs.

7.1.1 Chapter Contributions

With this background, we highlight a few observations that must be addressed in crowdfunding mechanism design.

Observation 7.1 ([157])**.** *Empirically, the probability of funding a project* decreases *with an* increase *in its duration.*

Observation 7.2. *As the agents can observe the net contribution to the project in sequential crowdfunding, their beliefs toward the project's funding evolve.*

Observation 7.3 ([47])**.** *Empirically, agents prefer to contribute only once.*

Belief Evolution. The above observations indicate a change in the agent's belief regarding funding the public project. For instance, with Observation 7.1, we see that agents become reluctant to fund projects whose target deadlines are greater. With Observation 7.3, we can note that agents contribution behavior changes with different refund schemes. This behavior can be attributed to a change in their belief in the project's funding. Moreover, from Observation 7.2, it is natural to assume that the availability of critical information, such as net contribution and the remaining time, will also impact the agent's belief.

This chapter refers to evolving beliefs as *dynamic* beliefs. We model this belief evolution as a *random walk*. We argue that each agent's step size, at any epoch, will be a posterior update depending on its prior belief and other auxiliary information (e.g., net contribution, time remaining).

PPRx-DB. This work primarily incorporates dynamic beliefs to analyze incentive-based civic crowdfunding mechanisms. To the best of our knowledge, PPRx is the only mechanism that incorporates asymmetric but static beliefs. We study PPRx under dynamic beliefs, and to distinguish our setting; we refer to it as *Provision Point Mechanism for agents With Dynamic Belief* (PPRx-DB). We argue that agents' belief evolution will be a random walk. We identify conditions on the random walk to characterize the sub-game perfect equilibria of the sequential game induced by PPRx-DB. In particular, by utilizing the evolution of each agent's random walk as a martingale, super/sub-martingale, we identify conditions wherein agents are naturally incentivized to contribute as soon as they arrive (i.e., avoid the race condition). Thus, though theoretically sound, complex mechanisms such as PPS may not be warranted in practice.

7.1.2 Chapter Notations and Solution Concepts

Throughout this chapter, we use the notations introduced during the description of CC in Chapter 2.3. We focus on the single-project case, i.e., $m = 1$ in Definition 2.26. Thus, we drop the subscript "j" from this chapter. For instance, agent i contributing x_{ij} to project $j \in [m]$ is simply agent i contributing x_i as there is only a single project ($m = 1$). Table 7.1 tabulates the notations used in this chapter.

We also build on the existing PPRx mechanism (Chapter 4). Lastly, the solution concepts used are Nash equilibrium (Definition 2.3) and sub-game perfect Nash equilibrium (Definition 2.4).

7.1.2.1 Additional Preliminaries

In this chapter, we also need the following notations.

7.1.2.1.1 Funded and Unfunded Expected Utilities in PPRx To incorporate asymmetric beliefs, in Chapter 4 we introduce PPRx. PPRx consists of two phases, (i)

Belief Phase (BP) with budget B_B and deadline τ_B wherein each agent i arrives at $a_{i,1}$ to the crowdfunding platform and submits its (static) belief k_i at $t_{i,1}$ towards the funding of the project, based on which it gets a reward b_i (known as Belief Based Reward (Eq. 4.6)); followed by (ii) *Contribution Phase* (CP) with budget B_C and deadline τ_C ($\tau = \tau_C + \tau_B$) wherein each agent i arrives at $a_{i,2}$ to the crowdfunding platform and contributes x_i at $t_{i,2}$ to the project. Thus, $\sigma_i = (b_i, t_{i,1}, x_i, t_{i,2})$ is each agent i's strategy in PPRx.

For this chapter, we create the following subsets for agents with "high" and "low" belief: $\mathcal{A}_H = \{i | \forall i \in \mathcal{A} \text{ s.t. } k_i \geq 1/2\}$ and $\mathcal{A}_L = \{i | \forall i \in \mathcal{A} \text{ s.t. } k_i < 1/2\}$. Trivially, $\mathcal{A} = \mathcal{A}_H \sqcup \mathcal{A}_L$. The utilities for agents in PPRx are given in Eq. 4.7 and Eq.4.8. Based on these, we denote the utility for agent $i \in \mathcal{A}$ when $\mathbf{x} \geq c$ as u^F (i.e., funded) and when $\mathbf{x} < c$ as u^{UF} (i.e., unfunded). That is,

- $\forall i \in \mathcal{A}_H$:

$$\mathbb{E}[u_i^F(x_i, \cdot)] = k_i \cdot (\theta_i - x_i + b_i)$$
$$\mathbb{E}[u_i^{UF}(x_i, \cdot)] = (1 - k_i) \cdot \left(\frac{x_i}{\mathbf{x}} \cdot B_C\right)$$

- $\forall i \in \mathcal{A}_L$:

$$\mathbb{E}[u_i^F(x_i, \cdot)] = k_i \cdot (\theta_i - x_i)$$
$$\mathbb{E}[u_i^{UF}(x_i, \cdot)] = (1 - k_i) \cdot \left(\frac{x_i}{\mathbf{x}} \cdot B_C + b_i\right)$$

7.1.2.1.2 Martingale Theory A *martingale* is a sequence of random variables such that the next value equals the current value in expectation, conditioned over all prior values. However, for several applications, one cannot always guarantee this equality. To analyze such scenarios, we interest ourselves in *bounding* the expected values. Such a sequence corresponds to a *super-martingale* or a *sub-martingale*. Consider a discrete sequence of random variables $X_0, X_1, \ldots$ evolving over time. Such a collection of random variables is referred to as a *stochastic process*, denoted by $\{X_t\}_{t \in [\tau]}$.

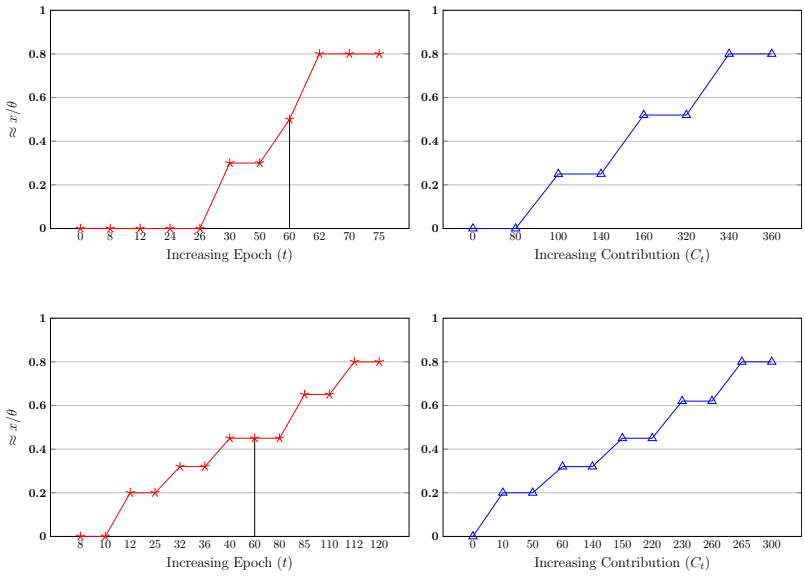


Figure 7.1: Plotting $\approx x/\theta$ for two randomly sampled agents using the dataset available with [47]. The black vertical line in the left plots represents the end of the refund period. We observe that the agents contribute even after the refund stage, possibly implying a change in their beliefs.

Definition 7.1: Martingales [295]

A stochastic process $\{X_t\}_{t\in[\tau]}$ such that $\mathbb{E}[X_t] < \infty$, is a

- Martingale if $\mathbb{E}[X_{t+1}|X_0, \ldots, X_t] = X_t$
- Sub-martingale if $\mathbb{E}[X_{t+1}|X_0, \ldots, X_t] \geq X_t$; and
- Super-martingale if $\mathbb{E}[X_{t+1}|X_0, \ldots, X_t] \leq X_t$.

In mechanism design literature, martingale theory is popularly used to model the dynamic evolution of an agent's private information. For e.g., Chawla et al. [57] model agent's dynamic valuation for a product (such as Netflix subscription) over time as a Martingale.

Balseiro et al. [22] model agent's expected utility as a Martingale to design a dynamic auction.

Notation	Definition
$\mathcal{P}_m, m = 1$	Set of Projects to be crowdfunded
$\mathcal{A} = \{1, \ldots, n\}$	Set of Agents
$\theta_i \in \mathbb{R}_{\geq 0}$	Agent i's valuation for the project
$\vartheta = \sum_{i \in \mathcal{A}} \theta_i$	Overall valuation of the project
$c \in \mathbb{R}_{\geq 0}$	Target cost of the project
$\tau \in \mathbb{R}_{\geq 0}$	Deadline of the project
$x_i \in \mathbb{R}_{\geq 0}$	Agent i's contribution to the project
$\mathbf{x} = \sum_{i \in \mathcal{A}} x_i$	Total contribution to the project
$B > 0$	Refund bonus
R	Refund Bonus Scheme
$r_i = R(x_i; \mathbf{x}, c, B, \cdot)$	Refund bonus to Agent i from the project
$u_i(\theta_i, x_i; \mathbf{x}, c, B)$	Agent i's utility from the project
k_i	Agent i's belief regarding the project's funding
$B_B > 0$	Bonus budget of the Belief Phase
$\tau_B \in \mathbb{R}_{\geq 0}$	Deadline of the Belief Phase
$B_C > 0$	Bonus budget of the Contribution Phase
$\tau_C \in \mathbb{R}_{\geq 0}$	Deadline of the Contribution Phase
u_i^F	Funded Utility of Agent i
u_i^{UF}	Unfunded Utility of Agent i

Table 7.1: Chapter Notations

7.2 PPRx-DB: CC for Agents with Dynamic Beleifs

This section analyzes PPRx with dynamic beliefs, namely PPRx-DB. We first present the agent's dynamic belief model. We then introduce PPRx-DB and provide agents' equilibrium contribution and the equilibrium time of contribution. We next present empirical evidence that agents' beliefs evolve during crowdfunding.

Observing Agent's Belief Evolution. Cason et al. [47] conduct real-world experiments to primarily test the impact of early refund bonus on a crowdfunding project's success. We use their data to provide the following insight regarding an agent's evolving belief.

Figure 7.1 plots x/θ, with varying t and $\mathbf{x}$, for two random agents from the dataset. Post $t > 60$ seconds; the agents do not get refunds for their contributions. Yet, we observe that agents contribute post $t > 60$ (Figure 7.1(left)). Agents' contribution pattern also evolves with $\mathbf{x}$ (Figure 7.1(right)).

7.2.1 Agent Dynamic Belief Model

We model the evolution of each agent's belief as a stochastic process over discrete epochs. Observe that this belief evolution may on available information at an epoch (e.g., net contribution). After each epoch, as an agent's belief can increase or decrease, we model it as a random walk.

For each agent $i \in \mathcal{A}$, let $\{k_{i,t}\}_{t\in[\tau]}$ denote the random walk with $X_{i,t}$ as the random variable for the step size at an epoch t. More formally, let each agent i's *prior* belief regarding the project's funding be $k_{i,0} \in [0, 1]$. At each epoch t, the agent's belief evolves per the available information, e.g., $\mathbf{x}_t$, i.e., the current contribution at epoch t or remaining epochs $\tau - t$.

Now, at each epoch $t \geq 1$, we denote agent i's *posterior* belief regarding the project's funding as: $k_{i,t} = k_{i,t-1} + X_{i,t}$. The sizes of the positive ($s_{i,+}()$) and negative ($s_{i,-}()$)steps

(with "$\circ$" as auxiliary information) are:

$$X_{i,t} = \begin{cases} s_{i,+}(\mathbf{x}_t, \tau - t, \circ), & \text{with probability } p \in [0,1] \\ s_{i,-}(\mathbf{x}_t, \tau - t, \circ), & \text{with probability } 1 - p \end{cases}$$

Note. $X_{i,t}$ captures the agent's belief evolution through the size of the step sizes, dependent on the available information. We now have a model for the random walk, $\{k_{i,t}\}_{t\in[\tau]}$. Our goal is to analytically derive equilibrium strategies for the agents conditioned on the behavior of $\{k_{i,t}\}_{t\in[\tau]}$.

7.2.2 PPRx-DB: Theoretical Analysis

In this subsection, we first discuss the funding of the public project at equilibrium. Second, we provide the upper bound of the agents' equilibrium contribution. Last, we present the equilibrium time of contribution for agents based on the underlying condition of the agent's belief evolution.

7.2.2.1 Project Status at Equilibrium

In PPRx-DB, the public project is funded at equilibrium. That is, at equilibrium, the total contribution *equals* the provision point, i.e., $\mathbf{x} = c$, when $\vartheta > c$. Consider the following lemma.

Lemma 7.1. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$, $B_B, B_C > 0$ and the utility structure for each agent $i \in \mathcal{A}$ given by Eq. 4.7 and Eq.4.8. At equilibrium, the public project is funded i.e., $\mathbf{x} = c$.*

Proof. From Eq. 4.7 and Eq. 4.8, at equilibrium, $\mathbf{x} < c$ cannot hold, since $\exists i \in \mathcal{A}_H$ with $x_i < \theta_i + b_i$ or $\exists i \in \mathcal{A}_L$ with $x_i < \theta_i$, at least, as $\vartheta > c$. Such an agent i can obtain a greater refund bonus by marginally increasing its contribution x_i as $B_C > 0$. Moreover, if $\mathbf{x} > c$, any contributing agent can increase its utility by marginally decreasing its contribution. Thus, $\mathbf{x} = c$ holds at equilibrium, i.e., the project is funded at equilibrium. □

7.2.2.2 Equilibrium Contribution: Upper Bound

We now analyze the equilibrium contribution of each agent $i \in \mathcal{A}$ in PPRx-DB. As each agent $i \in \mathcal{A}$ submits its belief $\hat{b}_i$ in the BP, PI can categorize each agent i to the sets $\mathcal{A}_H$ or $\mathcal{A}_L$. We next independently compute equilibrium contributions for the agents in $\mathcal{A}_H$ and $\mathcal{A}_L$, respectively.

7.2.2.2.1 For Agents with High Belief Lemma 7.2 presents the equilibrium contribution $x_i^\star$ analysis for each agent $i \in \mathcal{A}_H$. For the proof, we solve for $x_i^\star$ such that the (expected) funded utility is greater than equal to the (expected) unfunded utility. This is because, from Lemma 7.1, we know that at equilibrium, the contributions are such that $\mathbf{x} = c$.

Lemma 7.2. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$, $B_B, B_C > 0$ and the utility structure for each agent $i \in \mathcal{A}_H$ given by Eq. 4.7. For each $i \in \mathcal{A}_H$, its equilibrium contribution is*

$$x_i^\star \leq \frac{ck_{i,t_{i,2^\star}}(\theta_i + b_i)}{B_C(1 - k_{i,t_{i,2^\star}}) + ck_{i,t_{i,2^\star}}}, \tag{7.1}$$

where $t_{i,2^\star} \in [\tau_C]$ is its time of contribution at equilibrium.

Proof. Since at equilibrium $\mathbf{x} = c$, each agent i will contribute such that its funded utility is no less than the highest possible unfunded utility. That is, we solve for $x_i^\star$ such that $\mathbb{E}[u_i^F] \geq \mathbb{E}[u_i^{UF}]$, for each $i \in \mathcal{A}_H$. That is,

$$\begin{aligned} k_{i,t_{i,2^\star}} \cdot (\theta_i - x_i^\star + b_i) &\geq (1 - k_{i,t_{i,2^\star}}) \cdot \frac{x_i^\star}{c} \cdot B_C \\ \implies x_i^\star &\leq \frac{ck_{i,t_{i,2^\star}}(\theta_i + b_i)}{B_C(1 - k_{i,t_{i,2^\star}}) + ck_{i,t_{i,2^\star}}} \end{aligned}$$

This proves the lemma. □

7.2.2.2.2 For Agents with Low Belief Similar to our analysis for the set of agents in $\mathcal{A}_H$, Lemma 7.3 presents the equilibrium contribution analysis of each agent $i \in \mathcal{A}_L$.

Lemma 7.3. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$, $B_B, B_C > 0$ and the utility structure for each agent $i \in \mathcal{A}_L$ given by Eq. 4.8. For each $i \in \mathcal{A}_L$, its equilibrium contribution is*

$$x_i^\star \leq \frac{ck_{i,t_{i,2^\star}}\theta_i + cb_i(1 - k_{i,t_{i,2^\star}})}{B_C(1 - k_{i,t_{i,2^\star}}) + ck_{i,t_{i,2^\star}}}, \tag{7.2}$$

where $t_{i,2^\star} \in [\tau_C]$ is its time of contribution at equilibrium.

Proof. Similar to Lemma 7.2, we again solve for $x_i^\star$ such that $\mathbb{E}[u_i^F] \geq \mathbb{E}[u_i^{UF}]$, for each $i \in \mathcal{A}_H$. That is,

$$\begin{aligned} k_{i,t_{i,2^\star}} \cdot (\theta_i - x_i) &\geq (1 - k_{i,t_{i,2^\star}}) \cdot \left(\frac{x_i^\star}{c} \cdot B_C + b_i\right) \\ \implies x_i^\star &\leq \frac{ck_{i,t_{i,2^\star}}\theta_i + cb_i(1 - k_{i,t_{i,2^\star}})}{B_C(1 - k_{i,t_{i,2^\star}}) + ck_{i,t_{i,2^\star}}} \end{aligned}$$

This proves the lemma. □

7.2.2.3 Time of Equilibrium Contribution

Firstly, note that the refund bonus scheme in PPR (or PPRx) is independent of time. This induces a simultaneous-move game in PPR [52] or PPRx (Chapter 4). However, in PPRx-DB, the dynamic evolution of agents' belief towards the public project results in *variable* expected utility for each agent – dependent on their belief at each epoch. Thus, unlike PPR and PPRx, PPRx-DB does not induce a simultaneous-move game and can be deployed in sequential settings.

The challenge remains to identify when an agent will contribute to the public project. Recall that we denote the funded utility for agent i as u_i^F and the unfunded utility as u_i^{UF}.

Now, the complete utility structure for agent i is,

$$u_i(\cdot) = \mathbb{1}_{\mathbf{x} \geq c} \cdot u_i^F(\cdot) + \mathbb{1}_{\mathbf{x} < c} \cdot u_i^{UF}(\cdot).$$

At equilibrium, the expected funded payoff is equal to the expected not funded payoff (Lemmas 7.2 and 7.3). Thus, we have $\mathbb{E}[\pi_i] = \mathbb{E}[\pi_i^{UF}]$, $\forall i$ at equilibrium. In PPRx-DB, from Eq. 4.7 and Eq. 4.8, we also have

$$u_i^{UF}(x_i) = \frac{x_i}{\mathbf{x}} \cdot B_B + \kappa,$$

where $\kappa = 0, \forall i \in \mathcal{A}_H$ and $\kappa = b_i, \forall i \in \mathcal{A}_L$.

Now, the equilibrium time of contribution $t_{i,2^\star}$, $\forall i \in \mathcal{A}$, can be calculated as:

$$t_{i,2^\star} = \underset{t_{i,2} \in [\tau_C]}{\arg\max}\, \mathbb{E}[u_i^{UF}(x_i^\star)].$$

The subsequent results indeed derive $t_{i,2^\star}$ for the set of agents in $\mathcal{A}_H$ and $\mathcal{A}_L$. For these, we remark that when an agent i arrives at the Contribution Phase (CP), its belief at that epoch is the same as its prior belief, i.e., $k_{i,a_{i,2}} = k_{i,0}$. This stems from the fact that the agent has yet to observe the available information for any meaningful belief update.

7.2.2.3.1 For Agents with High Belief

Consider the following lemma for each agent $i \in \mathcal{A}_L$.

Lemma 7.4. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$, $B_B, B_C > 0$ and the utility structure for each agent $i \in \mathcal{A}_H$ given by Eq. 4.7. With $k^\star = \frac{\sqrt{B_C/c}}{1+\sqrt{B_C/c}}$, for each $i \in \mathcal{A}_H$, if:*

1. *$\{k_{i,t}\}_{t \in [\tau_C]}$ is a Martingale, then $t_{i,2^\star} = \tau_C$.*

2. *$\{b_{i,t}\}_{t \in [\tau_C]}$ is a Super-martingale, then*

$$t_{i,2^\star} = \begin{cases} a_{i,2} & \text{if } k_{i,0} \leq b^\star \\ t \ s.t. k_{i,t} = k^\star & \text{if } k_{i,0} > k^\star \end{cases}$$

Agent Set	$\{k_{i,t}\}_{t\in[\tau_C]}$	$x_{i,t_{i,2^\star}}$	$t_{i,2^\star}$	Race Condition
$\forall i \in \mathcal{A}_H$	Martingale	$\leq \frac{ck_{i,t_{i,2^\star}}(\theta_i+b_i)}{B_C(1-k_{i,t_{i,2^\star}})+ck_{i,t_{i,2^\star}}}$	τ_C	✓
	Super-martingale		$a_{i,2}$ if $k_{i,0} \leq k^\star$	✗
			t s.t.$k_{i,t} = k^\star$ if $k_{i,0} > k^\star$	✗
	Sub-martingale		$a_{i,2}$ if $k_{i,0} \geq k^\star$	✗
			t s.t.$k_{i,t} = k^\star$ if $k_{i,0} < k^\star$	✗
$\forall i \in \mathcal{A}_L$	Martingale	$\leq \frac{ck_{i,t_{i,2^\star}}\theta_i+cb_i(1-k_{i,t_{i,2^\star}})}{B_C(1-k_{i,t_{i,2^\star}})+ck_{i,t_{i,2^\star}}}$	τ_C	✓
	Super-martingale		$a_{i,2}$	✗
	Sub-martingale		τ_C	✓

Table 7.2: Summary of Our Results for PPRx-DB. Here, the cross mark denotes that the mechanism avoids the race condition.

3. $\{k_{i,t}\}_{t\in[\tau_C]}$ is a Sub-martingale, then

$$t_{i,2^\star} = \begin{cases} a_{i,2} & \text{if } k_{i,0} \geq k^\star \\ t \text{ s.t.} k_{i,t} = k^\star & \text{if } k_{i,0} < k^\star \end{cases}$$

Proof. We broadly divide the proof into the following two steps: (i) Firstly, we show that $\mathbb{E}[u_i^{UF}(x_i^\star)]$ is increasing in $k_{i,t}$ only if $k_{i,t} \leq b^\star$. That is, $\mathbb{E}[u_i^{UF}(x_i^\star)]$ is maximized at $k^\star$. In (ii), to decide on $t_{i,2^\star}$, we condition the underlying evolution of agent belief. Consider the following.

(i) **Deriving $k^\star$.** We have,

$$\mathbb{E}[u_i^{UF}] = (1-k_{i,t}) \cdot \frac{B_C}{c} \cdot \frac{ck_{i,t_{i,2^\star}}(\theta_i+b_i)}{B_C(1-k_{i,t_{i,2^\star}})+ck_{i,t_{i,2^\star}}}$$

Now,

$$\frac{\partial \mathbb{E}[u_i^{UF}]}{\partial k_{i,t}} > 0 \iff \frac{k_{i,t}}{1-k_{i,t}} \leq \sqrt{\frac{B_C}{c}}.$$

That is, $k_{i,t} \leq \frac{\sqrt{B_C/c}}{1+\sqrt{B_C/c}} = k^\star$.

(ii) **Deriving $t_{i,2^\star}$.** First, if $\{k_{i,t}\}_{t\in[\tau_C]}$ is a Martingale, $\mathbb{E}[k_{i,t}] = k_{i,t-1}$. Thus, on expectation, the value $\mathbb{E}[u_i^{UF}]$ does not change. As such, agent i has no incentive to contribute early, resulting in the race condition, i.e., $t_{i,2^\star} = \tau_C$.

Second, if $\{k_{i,t}\}_{t\in[\tau_C]}$ is a Super-martingale, we have $\mathbb{E}[k_{i,t}] \leq k_{i,t-1}$. Since $\mathbb{E}[u_i^{UF}]$ increases till $k^\star$, if agents initial belief is less than $k^\star$, $k_{i,t}$ will not reach $k^\star$ (in expectation) implying agent i must contribute as soon as it arrives, i.e., $t_{i,2^\star} = a_{i,2}$. However, if agent i's initial belief is greater than $k^\star$ and since $\mathbb{E}[k_{i,t}] \leq k_{i,t-1}$, the agent waits till an epoch t' s.t. $k_{i,t'} = k^\star$.

Last, if $\{k_{i,t}\}_{t\in[\tau_C]}$ is a Sub-martingale, we have $\mathbb{E}[k_{i,t}] \geq k_{i,t-1}$. Now, if the agent's initial belief is greater than $k^\star$, it is incentivized to contribute as soon as it arrives as its belief increases in expectation, resulting in lesser $\mathbb{E}[u_i^{UF}]$. Likewise, if its initial belief is less than $k^\star$, then in expectation, its belief will increase. That is, agent i waits till an epoch t' s.t. $k_{i,t'} = k^\star$.

This proves the lemma. □

7.2.2.3.2 For Agents with Low Belief Like Lemma 7.4, we now analytically present the time of equilibrium contribution for agents in $\mathcal{A}_L$.

Lemma 7.5. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$, $B_B, B_C > 0$ and the utility structure for each agent $i \in \mathcal{A}_L$ given by Eq. 4.8. With $\theta_i > b_i$ and $\theta_i < \frac{b_i \cdot c}{B_C}$ for each $i \in \mathcal{A}_L$, if*

1. *$\{k_{i,t}\}_{t\in[\tau_C]}$ is a Martingale, then $t_{i,2^\star} = \tau_C$.*
2. *$\{k_{i,t}\}_{t\in[\tau_C]}$ is a Super-martingale, then $t_{i,2^\star} = a_{i,2}$.*
3. *$\{k_{i,t}\}_{t\in[\tau_C]}$ is a Sub-martingale, then $t_{i,2^\star} = \tau_C$.*

Proof. Similar to the proof for Lemma 7.4, we broadly divide the proof into the following two steps: (i) Firstly, we show that $\mathbb{E}[u_i^{UF}(x_i^\star)]$ is increasing in $k_{i,t}$ if $\theta_i > b_i$ and $\theta_i < \frac{b_i \cdot c}{B_C}$.

In (ii), to decide on $t_{i,2^\star}$, we condition the underlying evolution of agent belief. Consider the following.

(i) $\mathbb{E}[u_i^{UF}]$ **as an Increasing Function.** We first derive the condition in which $\mathbb{E}[u_i^{UF}]$ is an increasing function. We have,

$$\mathbb{E}[u_i^{UF}] = (1 - k_{i,t}) \cdot \frac{B_C}{c} \cdot \frac{ck_{i,t_{i,2^\star}}\theta_i + cb_i(1 - k_{i,t_{i,2^\star}})}{B_C(1 - k_{i,t_{i,2^\star}}) + ck_{i,t_{i,2^\star}}}$$

Now,

$$\frac{\partial \mathbb{E}[u_i^{UF}]}{\partial k_{i,t}} > 0 \iff (c - B_C)(b_i - \theta_i)k_{i,t}^2 + 2B_C(b_i - \theta_i)k_{i,t} - (c + B_C)m_i + B_C\theta_i > 0 \tag{7.3}$$

For $\mathbb{E}[u_i^{UF}]$ to be increasing, the quadratic in Eq. 7.3 must be increasing. That is, $\Delta < 0$, and the first term must be positive. Through algebraic manipulations, we can show that these conditions will hold $\forall i \in \mathcal{A}_L$ iff $\theta_i > b_i$ and $\theta_i < \frac{b_i \cdot c}{B_C}$.

(ii) **Deriving $t_{i,2^\star}$.** As $\mathbb{E}[u_i^{UF}]$ is increasing in $k_{i,t}$ under $\theta_i > b_i$ and $\theta_i < \frac{b_i \cdot c}{B_C}$, we now derive $t_{i,2^\star}$ conditioned on the nature of the belief evolution. First, if $\{k_{i,t}\}_{t \in [\tau_C]}$ is a Martingale, then $\mathbb{E}[k_{i,t}] = k_{i,t}$. Trivially, the value $\mathbb{E}[u_i^{UF}]$ will not change in expectation for such a case. Thus, in practice, agent i will defer its contribution to the deadline, i.e., $t_{i,2^\star} = \tau_C$.

Second, if $\{k_{i,t}\}_{t \in [\tau_C]}$ is a Super-martingale, then $\mathbb{E}[k_{i,t}] \leq k_{i,t}$. In this case, a decrease in $k_{i,t}$ (in expectation) will imply a decrease in $\mathbb{E}[k_{i,t}]$. As such, agent i will contribute as soon as it arrives, i.e., $t_{i,2^\star} = a_{i,2}$.

Last, if $\{k_{i,t}\}_{t \in [\tau_C]}$ is a Sub-martingale, then $\mathbb{E}[k_{i,t}] \geq k_{i,t}$. In this case, an increase in $k_{i,t}$ (in expectation) will imply an increase in $\mathbb{E}[k_{i,t}]$. As such, agent i will defer its contribution till the deadline, i.e., $t_{i,2^\star} = \tau_C$.

This proves the lemma. □

Note. Table 7.2 summarizes the results presented in this section. We analytically provide the equilibrium contribution and time of contribution based on the underlying property of agents' dynamic belief evolution. The equilibrium time of contribution also implies whether the mechanism avoids the race condition or not. That is, when the equilibrium time of contribution equals the deadline, the race condition persists.

7.2.3 PPRx-DB: Sub-game Perfect Equilibrium Strategy

What remains to be shown is that the strategy, for each $i \in \mathcal{A}$, $\sigma_i^\star = (k_i^\star, t_{i,1^\star}, x_i^\star, t_{i,2^\star})$ where $k_i^\star = k_{i,0}, t_{i,1^\star} = a_{i,1}$, $x_i^\star$ as defined in Lemma 7.2 and Lemma 7.3 and $t_{i,2^\star}$ as defined in Lemma 7.4 and Lemma 7.5 satisfies sub-game perfect equilibrium (SPE). To this end, consider the following theorem.

Theorem 7.1. *Given $\mathcal{A}, m = 1$ and c (refer to Definition 2.26) with $0 < B < \vartheta - c$, $B_B, B_C > 0$ and the utility structure for each agent $i \in \mathcal{A}$ given by Eq. 4.7 and Eq.4.8. The public project is funded at equilibrium, i.e., $\mathbf{x} = c$. The set of strategies $\sigma_i^\star = (k_i^\star, t_{i,1^\star}, x_i^\star, t_{i,2^\star})$ where $k_i^\star = k_{i,0}, t_{i,1^\star} = a_{i,1}$ and $(x_i^\star, t_{i,2^\star})$ as defined in Lemma 7.2 and Lemma 7.4 $\forall i \in \mathcal{A}_H$ and in Lemma 7.3 and Lemma 7.5 $\forall i \in \mathcal{A}_L$.*

Proof. Firstly, from Lemma 7.1, we know that $\mathbf{x} = c$ if $\vartheta > c$. Next, $k_i^\star = k_{i,0}$ and $t_{i,1^\star} = a_{i,1}$ follows from the properties of BBR. More concretely, since BBR is incentive compatibility and decreasing with time, each agent $i \in \mathcal{A}$ reports its prior belief as soon as it arrives at the Belief Phase. Further, Lemmas 7.2,7.3 derive $x_i^\star$ and Lemmas 7.4,7.5 derive $t_{i,2^\star}, \forall i \in \mathcal{A}_H$ and $\forall i \in \mathcal{A}_L$, respectively.

The equilibrium strategy $\sigma_i^\star$, depending on the aggregate contribution and current belief, is also SPE. W.l.o.g., let agent j arrive at the Contribution Phase (CP) last. If the remaining contribution is zero, its best response is $x_{j,2^\star} = 0$. If some contribution remains to be funded, then irrespective of c and $\mathbf{x}$ its best strategy is $x_i^\star$ (defined in Lemmas 7.2,7.3) and $t_{i,2^\star}$ (defined in 7.4,7.5). Using backward induction, we argue that it is the best re-

sponse for every agent i to follow its strategy $\sigma_i^\star$, irrespective of history. That is, $\sigma_i^\star$ satisfy SPE, $\forall i \in \mathcal{A}$. □

Theorem 7.1 presents the SPE strategy for an agent. Without additional information/assumptions regarding the belief evolution or future agents' contribution, we believe these are a good starting point for mechanism design for crowdfunding public projects with dynamic beliefs.

7.3 Conclusion & Future Work

To the best of our knowledge, this chapter is the first attempt at addressing the persistent issue of static beliefs in the existing literature on crowdfunding of public projects. Toward this, we model the dynamic belief update for each agent as a random walk. Empirical evidence available justifies this argument. Next, we analyzed PPRx with dynamic beliefs as PPRx-DB. We first derived the agent's equilibrium contribution as a function of their dynamic beliefs. To derive the time of equilibrium contribution, we condition the dynamic belief as a (i) Martingale, (ii) Super-martingale, and (iii) Sub-martingale. Based on these underlying conditions, we provide the time of equilibrium contribution. Consequently, we also showed the conditions under which PPRx-DB avoids race conditions.

Discussion & Future Work. Significantly, our results highlight that simpler mechanisms may also avoid the race condition, allowing a practitioner to save on-chain deployment costs. Future work can build on these results by (i) exploring other conditions that provide an analytical characterization of the agent's equilibrium time and contribution and (ii) empirically validating the evolution of the agent's dynamic belief as a martingale. In parallel, one can even attempt to learn an ML model for an agent's belief update.

Civic Crowdfunding with Refunds: Contribution Summary

We conclude our discussion on civic crowdfunding (CC) with refunds by summarizing the different mechanisms in the literature in comparison with the mechanisms presented in this thesis (refer to Table 7.3). In Table 7.4, we also summarize the mechanisms in terms of their agent model and the properties they satisfy. Future research directions to consider will revolve around various permutations of the properties that are yet to be studied. We hope that the summary acts as a point of reference for readers interested in CC with refunds, both as a starting point and to advance the state-of-the-art.

CC Mechanisms	$\mathcal{P}$	Funding Guarantees	Equilibrium Contribution	Valuation	Refund Scheme	Utility Structure
PPR [312]	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}_+$	$r_{i1} = \frac{x_{i1}}{\mathbf{x}_1} B_1$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
PPS [52]	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}_+$	$r_{i1} = \phi(x_{i1}, q_{i1})$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
REPP-S [53]	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}_+$	$r_{i1} = \phi(x_{i1}, q_{i1}) + s(R_{M_{i1}})$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
Eqc-Learner [219]	$\mathcal{P}_m$	-	Not Defined	$\theta_{ij} \in \mathbb{R}_+$	$r_{ij} = \frac{x_{ij}}{\mathbf{x}_j} B_j$	$\mathbb{1}_{c_j \leq \mathbf{x}_j} \cdot (\theta_{ij} - x_{ij}) + \mathbb{1}_{c_j > \mathbf{x}_j} \cdot r_{ij}$
PPSN (Chapter 4)	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}$	$r_{i1} = \phi(x_{i1}, \min(q^+_{i1}, q^-_{i1}))$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
PPSx (Chapter 4)	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}_+$	$r_{i1} = \phi(x_{i1}, q_{i1})$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1} + b_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
					$r_{i1} = \phi(x_{i1}, q_{i1})$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot (r_{i1} + b_{i1})$
PPRG (Chapter 5)	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}_+$	$r_{i1} = \frac{x_{i1} + a(1/\gamma)^{\gamma-1}}{\mathbf{x}_1 + K} B_1$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
CCC (Chapter 6)	$\mathcal{P}_m$	-	May not be defined	$\theta_{ij} \in \mathbb{R}_+$	$r_{ij} = \frac{x_{ij}}{\mathbf{x}_j} B_j$	$\mathbb{1}_{c_j \leq \mathbf{x}_j} \cdot (\theta_{ij} - x_{ij}) + \mathbb{1}_{c_j > \mathbf{x}_j} \cdot r_{ij}$
PPRx-DB (Chapter 7)	$\mathcal{P}_1$	Funded	Closed-form	$\theta_{i1} \in \mathbb{R}_+$	$r_{i1} = \frac{x_{i1}}{\mathbf{x}_1} B_1$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1} + b_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot r_{i1}$
					$r_{i1} = \frac{x_{i1}}{\mathbf{x}_1} B_1$	$\mathbb{1}_{c_1 \leq \mathbf{x}_1} \cdot (\theta_{i1} - x_{i1}) + \mathbb{1}_{c_1 > \mathbf{x}_1} \cdot (r_{i1} + b_{i1})$

Notations. (i) $\mathbb{1}_X = 1$ if X is true, $\mathbb{1}_X = 0$ otherwise. (ii) $\phi(x, q) = C_0^{-1}(x + C_0(q)) - q - x$, $C_0 : \mathbb{R} \to \mathbb{R}$: The prediction market's cost function with q as the total market securities. (iii) $S(R_{M_{i1}})$: referral bonus. (iv) $b(\cdot)$: Belief-based reward. (v) $a > 0, \gamma > 1$ and $K = \frac{a\gamma}{\gamma - 1}$.

Table 7.3: Summarizing Different Mechanisms for CC with Refunds for an agent $i \in \mathcal{A}$.

CC Mechanisms	Setting	Information Structure					Avoids Race Condition	Supports Multiple Projects	Blockchain
		Valuation			Beliefs				
		Positive	Negative	Dynamic	Symmetric	Asymmetric	Dynamic		
PPR [312]	Offline	✓	✗	✗	✓	✗	✗	✗	✗
PPS [52]	Online	✓	✗	✗	✓	✗	✗	✗	Inefficient
REPP-S [53]	Online	✓	✗	✗	✓	✗	✗	✗	Inefficient
EqC-Learner [219]	Online	✓	✗	✗	✓	✗	✗	✓	✗
PPRx (Chapter 4)	Offline	✓	✗	✗	✗	✓	✗	✗	✗
PPSx (Chapter 4)	Online	✓	✗	✗	✗	✓	✗	✗	Inefficient
PPRN (Chapter 4)	Offline	✓	✓	✗	✓	✗	✗	✗	✗
PPSN (Chapter 4)	Online	✓	✓	✗	✓	✗	✗	✗	Inefficient
PPRG (Chapter 5)	Online	✓	✗	✗	✓	✗	✗	✗	Efficient
CCC (Chapter 6)	Offline	✓	✗	✗	✓	✗	✗	✓	✗
PPRx-DB (Chapter 7)	Online	✓	✗	✗	✗	✓	✓	✗	✓

Table 7.4: Summary of Works for Civic Crowdfunding of Public Projects with Refunds (CC). The green check mark implies that a mechanism satisfies the given property. Future research directions to consider will revolve around various permutations of the properties that are yet to be studied.

Chapter 8

Achieving Fairness in Transaction Fee Mechanism Design

Popular cryptocurrencies, like Bitcoin and Ethereum, process more than a million daily transactions. Given the demand, a strategic miner of the block is likely to include transactions whose fees maximize its utility. Recent research introduces a transaction fee mechanism (TFM) design to study the optimal behavior of miners and transaction creators. The current TFM literature focuses on satisfying standard incentive properties – which may not suffice for the widespread adoption of cryptocurrencies. We argue that a TFM is "fair" to the transaction creators if it satisfies specific notions, namely Zero-fee Transaction Inclusion and Monotonicity. First, we prove that, generally, one cannot ensure both these properties and prevent a miner's strategic manipulation. We then show that existing TFMs either do not satisfy these notions or do so at a high cost to the miners' utility. To this end, we introduce a novel TFM using on-chain randomness – rTFM. We prove that rTFM satisfies incentive compatibility for both miner and transaction proposer while ensuring our novel fairness constraints.

⋆ ⋆ ⋆ ⋆ ⋆

8.1 Introduction

Decentralized cryptocurrencies, e.g., Bitcoin [203] and Ethereum [42], support payments as digital *transactions*. The currencies' rise in popularity is evident as Bitcoin and Ethereum process 1M+ transactions per day [302, 303]. These currencies maintain their state through a distributed ledger, namely *blockchain*. A blockchain is a sequence of cryptographically linked blocks comprising a set of transactions [203]. In Proof-of-Work (PoW) based cryptocurrencies such as Bitcoin and (formerly) Ethereum, block proposers (aka "miners") generate a valid block by solving a cryptographic puzzle. Miners receive a block reward for their efforts.

Explosion in Transaction Fees. A miner adds transactions from the pool of outstanding transactions (aka "mempool"). Transaction creators (or *agents*) optionally send a *transaction fee* as a commission to the miners to incentivize them to add their transactions. We remark that transaction fees were envisioned as optional in Bitcoin [293]. Unfortunately, due to surplus demand [263] and the volatile nature of cryptocurrencies, in Bitcoin (i) transactions offering 0-2 Satoshi per bytes as fees have an unbounded waiting time [233] and (ii) 30% of Bitcoin fees are two orders of magnitude more than recommended [193]. Given this, recent research focuses on the *strategic* interaction between the miner and agents through mechanism design, referred to as Transaction Fee Mechanism (TFM) design and introduced formally in Chapter 3.2.2.4.

TFMs: Brief Overview. The miner-agent interaction is analogous to an auction setting (with some blockchain-induced idiosyncrasies). Indeed, Bitcoin [203] implements a "first-price" auction with a miner maximizing its revenue by greedily adding transactions to its block. An agent's transaction fee indicates its willingness to pay for its transaction to be added. Roughgardern [245] formalizes TFM design in terms of the (i) *allocation rule* (adding transactions from the mempool to a block), (ii) *payment rule* for the payment

to the miner, and (iii) *burning rule*[1]. Unlike classical auction settings, in blockchain-based cryptocurrencies, the miners have complete control over the transactions they add. Consequently, the author proposes *miner incentive compatibility* (MIC) in addition to the standard *dominant strategy incentive compatibility* (DSIC) for the agents. MIC states that the proposed TFM must incentivize miners to follow the intended allocation rule truthfully. DSIC ensures that agents offer their transaction's valuation as a transaction fee. Next, to curb miner-agent off-chain collusion, Roughgarden [245] also introduces *off-chain collusion proofness* (OCAP). The author studies popular TFMs like first-price, second-price, and Ethereum's new dynamic posted-price mechanism, namely EIP-1559 [43], in terms of the properties they satisfy. Subsequent works [103, 62] enrich the TFM literature by proposing a dynamic posted-price mechanism and providing significant foundational results, respectively.

TFM Design: Challenge with Incentives. To satisfy DSIC, MIC, and OCAP, TFMs introduce payment and burning rules based on transaction fees. However, we believe that (and as originally intended in Bitcoin [293]) TFMs must also support including transactions with zero fees. This inclusion will also be beneficial for the greater adoption of cryptocurrencies. As a concrete example, commission-based digital payment networks (e.g., VISA/MasterCard) are losing ground to commission-less networks (e.g., UPI) [1]. Commission-less payment networks admit ≈ 7.5 times *higher* transaction volume compared to their commission-based counterparts (`rbi.org.in`). That is, zero transaction fees are instrumental for greater market penetration. As such, we argue that it is necessary to design TFMs with a provision for zero-fee transaction inclusion.

[1]Burning refers to removing tokens from the cryptocurrency's supply forever. E.g., by transferring them to an address with no private keys. Such addresses can only receive tokens, thus effectively making the tokens inaccessible.

8.1.1 Chapter Contributions

8.1.1.1 Fairness Notions

We introduce (i) *Zero-fee Transaction Inclusion* (ZTi) and (ii) *Monotonicity*. A TFM satisfies ZTi if it ensures that zero-fee transactions have a non-zero probability of getting accepted in a block.[2] Reflecting on the success of commission-less digital payment networks [46] and the unbounded waiting time for transactions with marginal fees in Bitcoin, we argue that such a notion is critical for a cryptocurrency's greater adoption.

However, guaranteeing ZTi must still ensure that the probability of a transaction's inclusion increases with an increase in its fee. For example, randomly including trivial transactions ensures ZTi but may be unfair for a company that desires swift confirmation to meet the scheduled launch or if the transaction fixes a critical bug. To capture this, we introduce Monotonicity, which states that a TFM must ensure that transactions with a higher transaction fee are more likely to get included in the block. Such a notion allows for *priority-based* transaction confirmation.

Given the impossibility of satisfying DSIC, MIC, and a collusion-proofness against a single creator and the miner collusion simultaneously [62], we say a TFM is *fair* if it meets the above two criteria and DSIC. That is, fairness in TFMs w.r.t. the transaction creators (or agents). As TFM design generally focuses on maximizing the miner's utility, it fails to satisfy ZTi. Moreover, existing TFMs either do not satisfy our notion of fairness or do so at a high cost to the miners' utility. As such, we introduce (i) Softmax TFM (STFM) and (ii) Randomized TFM (RTFM), two TFMs that satisfy our fairness notions and study their incentive properties.

[2]We assume that miners/agents are myopic [245, 103, 62], i.e., they only consider their utility from the next block. Thus, ZTi deals with a transaction's probability of inclusion for the next block and *not* "eventual" confirmation. The myopic assumption is reasonable since pending transactions are typically never confirmed (e.g., in Ethereum).

8.1.1.2 Softmax TFM (STFM)

With STFM, we introduce a novel allocation rule wherein a miner samples transactions from a distribution generated by applying the *softmax* function [38] to the set of outstanding transactions. We prove that STFM satisfies our notion of fairness with bounded utility loss for the miners. Unfortunately, STFM does not satisfy MIC, as a strategic miner can always maximize its revenue by optimally selecting transactions instead of following STFM's randomized allocation.

8.1.1.3 Randomized TFM (rTFM)

We next propose rTFM: a TFM that satisfies our fairness notions while guaranteeing MIC (for an appropriate payment rule). In rTFM, we introduce another novel allocation rule that requires the miner to create two sets of transactions. In the first set, the miner optimally selects the transactions to add to its block (i.e., exactly like it currently does in Bitcoin). In the second set, the miner *uniformly* adds transactions from the mempool to its block, but crucially, does *not* receive any fees for these transactions. That is, the miner has no incentive to deviate from the uniform allocation in this set. The miner broadcasts both these sets, and we show that the blockchain network can randomly confirm one of the two sets through a *trusted* coin-flip mechanism. Intuitively, such an allocation gives a non-zero probability of acceptance for zero-fee transactions (due to the uniform sampling in the second set). As the miner has no control over the confirmed set, rTFM satisfies MIC for an appropriate payment rule (e.g., Bitcoin's first-price auction).

8.1.1.4 Contributions

In summary, (i) we introduce the notion of fair TFMs based on two fairness criteria: ZTi and Monotonicity (Chapter 8.2). (ii) We prove the impossibility of a TFM simultaneously satisfying ZTi and MIC (Chapter 8.2.2). (iii) For an improved trade-off between the miner's utility and ZTi, we first propose STFM, a novel softmax-based allocation rule, and prove

Notation	Definition
	TFM Model
$\mathcal{H}$	Blockchain history
$M := \{1, \ldots, m\}$	Mempool (set of outstanding transactions)
B_k	Current Block
C	Size of the block
$\mathbf{x}$	Block allocation rule
$\mathbf{p}$	Payment rule
$\mathbf{q}$	Burning rule
$\tau \in \{\tau_D, \tau_R\}$	TFM's type – either deterministic (τ_D) or randomized (τ_R)
$\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$	TFM Tuple
$\mathbf{b} := \{b_1, \ldots, b_m\}$	Bids present in the mempool
$\Theta := [\theta_i]$	Set of valuations with each agent i's valuation θ_i
$s_t,\ \forall s_t \in M$	Size of each transaction s_t
$\mathbf{b}_I$	Bids of users included in the block
$\mathbf{b}_{I \setminus i}$	Bids of users included in the block without agent i
$u_i(\cdot)$	Utility of agent i
F	Set of fake transactions added by the miner
$u_{\mathsf{M}}(\cdot)$	Utility of the miner

Table 8.1: Chapter Notations

that it satisfies our fairness notions and other incentive properties (Chapter 8.3). (iv) We then introduce rTFM: a novel TFM based on a trusted coin-flip mechanism and prove that it satisfies our fairness notions while simultaneously satisfying MIC (Chapter 8.4).

8.1.2 Chapter Notations and Additional Background

This chapter extends the Transaction Fee Mechanism (TFM) design literature introduced formally in Chapter 3.2.2.4. As such, we follow the notations from Chapter 3.2.2.4. The solution concepts focused on in this chapter include DSIC in the TFM context (Def-

inition 3.22) and Miner Incentive Compatibility (MIC) (Definition 3.23). For rTFM we require hash functions (Definition 3.1) and the following cryptographic primitives. Table 8.1 tabulates the notations used in this chapter.

8.1.2.1 Additional Preliminaries

Merkle Tree (MT) [192]. These are complete binary trees where every parent node is a hash of its children. In blockchains like Bitcoin, each block comprises an MT such that the parents are hashes of transactions that are included in the block. More concretely, the value of a parent node a is the hash of the concatenation of its two children nodes b, c, i.e. $a = \text{Hash}(b||c)$. The Merkle root $\text{MT}_{\mathsf{root}}$ is the hash value of the root node of MT.

Proof-of-Work (PoW) [203]. In blockchains like Bitcoin [203], PoW is a protocol to propose new blocks. Here, miners use the blockchain's history $\mathcal{H}$ (comprising previously mined blocks, say up till B_{k-1}) and $\text{MT}_{\mathsf{root}}$ of the set of transactions to be included in their block, B_k. The block header of B_k is made up of the hash of the parent block B_{k-1}, $\text{MT}_{\mathsf{root}}$, and a randomly generated nonce. The block is considered mined if the miner finds a nonce such that the hash value of the block $h = \text{Hash}(B_k)$ is lesser than *target difficulty* (TD) as decided by the system, i.e., $h < TD$.

On-chain Trusted Randomness. Micali et al. [194] introduce *verifiable random functions*, which take inputs and generate pseudorandom outputs that can be publicly verified. In the blockchain context, this often implies functions whose randomness depends on the information available to the blockchain (aka verifiable or trusted on-chain randomness). E.g., Chung et al. [62] propose a randomized second-price TFM that uses such randomness to confirm transactions added to its block by the miner.

8.1.2.2 Additional Related Work

We now place this chapter's contribution concerning the existing literature for (i) TFM design and (ii) fairness in the context of blockchain.

Transaction Fee Mechanism (TFM) Design. Roughgarden [245] presents the seminal work that describes the "inclusion of transactions in a block" in the language of mechanism design. The author shows that EIP-1559 satisfies DSIC and MIC and is OCAP (under some constraints on the base fee). Ferreira et al. [103] present a novel dynamic posted-price TFM with an equilibrium characterization of the posted-price. Most recently, Chung et al. [62] provide several foundational results for TFM design based on underlying incentives and allocation rules. While the works [245, 103, 310, 62] are complementary, they do not focus on transaction fairness in TFMs.

Parallely, works also exist that empirically analyze TFMs to optimize transaction fees [165, 278]. Tedeschi et al. [278] suggest a Deep Neural Network-based approach to predict miners' behavior in terms of including transactions in their blocks. The authors show that their approach reduces transaction fees and improves the confirmation time.

Fairness in Blockchain. Fairness is studied in various contexts, including network latency [144, 185], transaction ordering [116, 15, 262, 216, 155, 162] and price of transaction consumption [25, 258].

Fairness in transaction order focuses on the latency in transaction confirmation. E.g., miners may discriminate among specific transaction creators or only include transactions of the creators they know prior. This line of work [116, 15, 262, 216, 155, 162] does not model game-theoretic interactions and focuses on verifiable methods of ensuring "fairness" using cryptographic primitives. Moreover, there is no provision for the inclusion of zero-fee transactions. E.g., Sokolik et al. [262] present a fair approach that prioritizes transactions with significant waiting time. Orda et al. [216] provide techniques that enforce that transactions are allocated randomly to each block.

BitcoinF 's [258] allocation rule splits the block with dedicated sections for standard transactions and low-fee transactions. The authors argue that this allows miners to maximize their utility (through the standard section) while also processing low-fee transactions. With a strong assumption that transaction influx equals the cryptocurrency's throughput,

they empirically argue that BitcoinF provides a lower consumption price. Also, they do not provide any theoretical guarantees for strategyproofness or fairness.

In summary, the TFM literature does not focus on the fairness of transactions w.r.t. transaction creators. Other works on fairness do not provide theoretical guarantees for fairness, miner's utility, or the inclusion of zero-fee transactions.

8.2 Fairness in TFMs

This section (i) presents our novel fairness notions, (ii) discusses the impossibility of simultaneously maximizing the miner's utility and ZTi, (iii) studies the fairness guarantees of BitcoinF when $\delta = 0$ and (iv) introduces Softmax TFM (STFM).

8.2.1 Fairness Notions

We propose the following fairness notions for the general adoption of a blockchain-based cryptocurrency.

8.2.1.1 Zero-fee Transaction Inclusion (ZTi)

Based on the market penetration of digital payments due to non-commission-based payment platforms [46] and the unbounded waiting time for transactions with marginal fees in Bitcoin, we introduce *Zero-fee Transaction Inclusion* (ZTi) as a critical fairness notion for a TFM to satisfy. That is, our first fairness notion ensures that a transaction with zero fees must have a non-zero probability of getting included in a block.

Definition 8.1: Zero-fee Transaction Inclusion (ZTi)

We say $\mathcal{T}^{TFM}$ (refer to Definition 3.21) satisfies ZTi if the probability with which a transaction t with transaction fee $b_t = 0$ gets included in a block B_k is strictly non-zero, i.e., $\Pr(t \in B_k) > 0$.

As the agents and miners are myopic, ZTi only considers a transaction's probability of being included in the next block.

8.2.1.2 Monotonicity

This notion focuses on the probability of including a bidding agent's transaction being proportional to the transaction fee. Naturally, a bidding agent would expect a higher probability of accepting its transaction if it increases the transaction's fee. Such a scenario is also desirable in practice, e.g., startups/applications may want faster transaction acceptance to meet launch dates or deployment targets. To this end, we introduce *monotonicity* as defined next.

Definition 8.2: Monotonicity

We say $\mathcal{T}^{TFM}$ (refer to Definition 3.21) satisfies Monotonicity if the probability with which a transaction t gets accepted in a block B_k increases with an increase in its transaction fee b_t, given the remaining bids $\mathbf{b}_{-t}$ are fixed. That is, $\Pr(t \in B_k \mid \mathbf{b}_{-t}, b_t + \epsilon) > \Pr(t \in B_k \mid \mathbf{b}_{-t}, b_t)$ for any $\epsilon > 0$ and fixed $\mathbf{b}_{-t}$.

Note. We remark that most existing TFMs satisfy monotonicity. However, designing TFMs that simultaneously satisfy monotonicity and ZTi is non-trivial. Trivially, a TFM satisfying both our fairness notions ensures that each transaction has a non-zero probability of getting accepted!

8.2.2 Impossibility of Simultaneously Maximizing Miner Utility and Satisfying ZTi

Before presenting the main impossibility, we first analyze the fairness guarantees for EIP-1559 [43].

Remark 8.1. EIP-1559 satisfies (i) Monotonicity but does not satisfy (ii) ZTi. As each transaction must at least pay the base fee, no honest/strategic miner will include zero-fee transactions to preserve the validity of their blocks, i.e., if $b_t = 0 \implies \Pr(t \in B) = 0$. EIP-1559 satisfies monotonicity since increasing the payment $b_t - \lambda$ will increase the chance of the transaction being part of the optimal set in Eq. 3.10.

Theorem 8.1 adds to Remark 8.1 by showing that any TFM that allows a strategic miner complete control over which transactions to add cannot satisfy ZTi, for any non-trivial payment rule. A *trivial payment rule* is $p_t = 0,\ \forall t \in B_k$. For the proof, we provide a counterexample s.t. $\forall t \in M,\ b_t = 0 \implies \Pr(t \in B_k = 0)$.

Theorem 8.1. *No $\mathcal{T}^{TFM}$ (refer to Definition 3.21) with a non-trivial payment rule which provides a strategic miner complete control over the transactions to add to its block, satisfies Zero-fee Transaction Inclusion (ZTi).*

Proof. Consider the following example. Let the transaction bid and size pair in the mempool be denoted by $\mathcal{P} = [(b_i, s_i)] = \{(10, 10), (10, 10), (5, 10), (0, 10), (0, 10)\}$. If the block B_k can admit a total transaction size of 30, then the miner can maximize its utility from 3.10 by selecting the first three transactions in $\mathcal{P}$. That is, $\mathbf{x}^{TFM} = \{1, 1, 1, 0, 0\}$ with $u_{\mathsf{M}}^{TFM} = 25$. This implies that $\Pr(t_4 \in B_k) = \Pr(t_5 \in B_k) = 0$, thus, ZTi is not satisfied. □

8.2.3 BitcoinZF: BitcoinF with Zero Fees

We tweak the block allocation rule in BitcoinF [258] (Chapter 3.2.2.4) to introduce a provision for transactions with zero fees. We set $\delta = 0$ so that the miner *randomly* adds zero-fee transactions to fill the $1 - \alpha$ section, followed by *greedily* adding transactions with bid b to the α section. Formally, from Eq. 3.11, the change is as follows.

$$\left.\begin{array}{ll} \max_{\mathbf{x}^{BZ}} & \sum_{i\in M} x_i^{BZ}\cdot p_i^{BZ}(\mathcal{H},B_k)\cdot s_i \\ \text{s.t.} & \sum_{t\in M,b_t\neq 0} s_t\cdot x_t^{BZ}(\mathcal{H},M)\leq C_\alpha \\ & \sum_{t\in M,b_t=0} s_t\cdot x_t^{BZ}(\mathcal{H},M)=C_{1-\alpha} \text{ and} \\ & x_t^{BZ}(\mathcal{H},M)\in\{0,1\},\forall t\in M. \end{array}\right\} \tag{8.1}$$

Furthermore, with base fee λ, for each i in the α section we have $p_i^{BZ} = b_i - \lambda$ and $q_i^{BZ} = \lambda$. For each i in the $1-\alpha$ section we have $p_i^{BZ} = q_i^{BZ} = 0$. In summary, BitcoinZF is denoted by the tuple $\mathcal{T}^{BZ} = (\mathbf{x}^{BZ}, \mathbf{p}^{BZ}, \mathbf{q}^{BZ}, \tau_D)$.

8.2.3.1 Fairness Notions

Theorem 8.2 shows that BitcoinZF satisfies the two fairness notions if each zero-fee transaction's size is less than $C_{1-\alpha}$. In other words, BitcoinZF satisfies ZTi if none of the zero-fee transactions are of significant size.

Theorem 8.2. *BitcoinZF ($\mathcal{T}^{BZ}$, Eq. 8.1) satisfies (i) Zero-fee Transaction Inclusion and (ii) Monotonicity only if $\forall\ t_i \in M$ with $b_i = 0$, we have $s_i \leq C_{1-\alpha}$.*

Proof. Without considering the inclusion of fake bids from the miner, i.e., $F = \emptyset$, the optimization in BitcoinZF is given in Eq. 8.1. To show that BitcoinZF satisfies Monotonicity, we have to show that by increasing its bid b_i, agent i's transaction t_i has a higher probability of getting accepted in B_k. Indeed, this is the case in BitcoinZF, since increasing b_i to $b_i + \epsilon$ s.t. $\epsilon > 0$ can only increase the probability of t_i's inclusion in B_k. This is because of the KNAPSACK definition from Eq. 8.1.

Furthermore, since the miner receives no utility from any transaction in the "$1-\alpha$" section, it can uniformly sample zero-fee transactions in this section. There is a subtle point here: the miner does not get any utility by adding these transactions to the $1-\alpha$ section. It can, in effect, leave the section empty or add its own transactions. However,

since such deviations will not yield the miner any increase in utility, we can state that BitcoinZF satisfies ZTi. □

8.2.3.2 Cost of Fairness (CoF)

Unfortunately, there is a "cost" to the fairness guarantees in BitcoinZF. Ensuring ZTi *hurts* the miner's utility. To this end, consider the following definition.

Definition 8.3: Cost of Fairness CoF

We define the cost of fairness (CoF) of $\mathcal{T}^{TFM} = (\mathbf{x}, \mathbf{p}, \mathbf{q}, \tau)$ as $\mathsf{CoF}_{TFM} = \max_{\mathbf{b} \neq 0} \frac{OPT}{u_{\mathsf{M}}^{TFM}}$. Here, u_{M}^{TFM} is the miner's utility from the indented allocation $\mathbf{x}$ and OPT its utility from Eq. 3.10 with $p_t = b_t$ and $q_t = 0,\ \forall t \in B_k$.

Trivially, *lesser* the CoF, *greater* the miner's utility from following $\mathcal{T}^{TFM}$. Claim 8.1 presents the CoF for BitcoinZF for the specific case when for every $t_i, t_j \in M$ s.t. $i \neq j$, we have $s_i = s_j$. That is, all transactions are of the same size. The proof follows from algebraic manipulations.

Claim 8.1. *For every $t_i, t_j \in M$ s.t. $i \neq j$, if we have $s_i = s_j$, then $\mathsf{CoF}_{BZ} = \frac{OPT}{u_M^{BZ}} = 1/\alpha$ where $\alpha \in (0, 1]$.*

Proof. W.l.o.g., let the optimal set of bids (sorted in non-decreasing order) which maximizes the miner's utility in Eq. 3.10 with $p_t = b_t$ and $q_t = 0,\ \forall t$ be $\{b_1, \ldots, b_c\}$. Then with $\alpha = \frac{k}{c}$ s.t. $k \leq c$, we can write BitcoinZF's bid set as $\{b_1, \ldots, b_k\}$ (since the miner will maximize utility in the "α" section of the block). Observe that,

$$\begin{aligned}\frac{OPT}{u_{\mathsf{M}}^{BZ}} &= \frac{b_1 + \ldots + b_c}{b_1 + \ldots + b_k} = 1 + \frac{b_{c-k+1} + \ldots + b_c}{b_1 + \ldots + b_k} \\ &\leq 1 + \frac{(c-k)b_k}{k \cdot b_k} \leq 1 + \frac{c}{k} - 1 \leq \frac{c}{k} = 1/\alpha.\end{aligned}$$

This completes the claim. □

Challenges with BitcoinZF. Despite satisfying our fairness notions, BitcoinZF has the following challenges. First, Claim 8.1 only holds when each transaction's size is equal. With different transaction sizes, $\frac{OPT}{u_{\mathsf{M}}^{BZ}}$ can be arbitrarily bad. E.g., if the size of the transaction with the highest bid in M is greater than C_α, $OPT/u_{\mathsf{M}}^{BZ} \to \infty$. Second, when $1-\alpha$ is small, zero-fee transactions of sufficient size will deterministicly never get included in the block. Formally, if $\exists\ t_i \in M$ s.t. $b_i = 0$ and $s_i > C_{1-\alpha}$, we have $\Pr(t_i \in B_k) = 0$.

8.3 Softmax TFM: Fairness using Randomization

We now introduce Softmax TFM (STFM), which comprises an intuitive, randomized allocation rule that guarantees ZTi and Monotonicity. To begin with, it's important to note that a straightforward allocation rule that uniformly selects transactions from M will trivially satisfy both our fairness notions.

Remark 8.2. Consider a TFM with an allocation rule that uniformly samples transactions, i.e., $\forall t_i \in M\ \Pr(t_i \in B_k) = 1/n$. Trivially, such a TFM (i) satisfies Zero-fee Transaction Inclusion but (ii) does not satisfy Monotonicity since as b_i increases, $\Pr(t_i \in B_k)$ remains the same.

We next (i) introduce Softmax TFM (STFM) and (ii) discuss its fairness and incentive guarantees.

Algorithm 6 Softmax TFM (STFM) Allocation

Input: Block Size C, Mempool M, History $\mathcal{H}$, Temperature γ

Output: Set of allocated transactions in B_k, i.e., $\mathcal{X}_k$

procedure STFMAllocation($C, M, \mathcal{H}$)

  $S = 0, \mathcal{X}_k = \emptyset$

  $\Gamma_k = \left[\frac{\exp(b_t/\gamma)}{\sum_{t' \in M} \exp(b_{t'}/\gamma)}\right]_{\forall t \in M}$ ▷ Generate the Softmax distribution

  while $C - S > 0$ **do**

    $t \sim \Gamma_k$ ▷ Sample a transaction

    $S \leftarrow S + s_t$ ▷ Add to the current block consumption

    $\mathcal{X}_k \leftarrow \mathcal{X}_k + \{t\}$

    $\Gamma_k = \left[\frac{\exp(b_t/\gamma)}{\sum_{t' \in M \setminus \mathcal{X}_k} \exp(b_{t'}/\gamma)}\right]_{\forall t \in M \setminus \mathcal{X}_k}$ ▷ Re-generate the Softmax distribution

  end while

  return $\mathcal{X}_k$

end procedure

8.3.1 Softmax TFM

For a given on-chain history $\mathcal{H}$, the mempool M and the current block B_k, STFM can be expressed in the TFM language as $\mathcal{T}^{STFM} = (\mathbf{x}^{STFM}, \mathbf{p}^{STFM}, \mathbf{q}^{STFM}, \tau_R)$. We begin by defining the allocation rule $\mathbf{x}^{STFM}$.

8.3.1.1 STFM Allocation

Unlike deterministic TFMs like FPA and EIP-1559, STFM is a randomized allocation rule. The miner does not compute the optimal allocation set as in Eq. 3.10 but instead *samples* a feasible set of transactions. These transactions are sampled through a distribution generated by applying the softmax with temperature function [38] to the set of the

outstanding transactions in M. The softmax function with the temperature parameter $\gamma \in \mathbb{R}^+$ and for any real-valued vector $\mathbf{z} = (z_1, \ldots, z_n)$ is defined $\forall i$ as follows [38].

$$\Gamma(\mathbf{z})_i = \frac{\exp(z_i/\gamma)}{\sum_{i' \in \mathbf{z}} \exp(z_{i'}/\gamma)}. \tag{8.2}$$

Algorithm 6 presents the procedure with which the miner randomly samples a feasible set of transactions in STFM. With this, we can define $\mathbf{x}^{STFM}$ as follows.

Definition 8.4: STFM Allocation Rule

Given $\mathcal{H}$, M and B_k, let $\mathbf{x}^{STFM}$ denote a feasible allocation rule with $\Pr(t \in B_k)$, $\forall t \in M$ generated from the Softmax distribution (refer Algorithm 6). Formally, given the set of transactions sampled, $\mathcal{X}_k \leftarrow \text{STFMAllocation}(C, M, \mathcal{H})$, we have $\mathbf{x}^{STFM} = [x_t^{STFM}]$ s.t.

$$x_t^{STFM} = \begin{cases} 1 & \text{if} \quad t \in \mathcal{X}_k, \\ 0 & \text{otherwise.} \end{cases} \tag{8.3}$$

STFM Payment and Burning Rules. The allocation rule $\mathbf{x}^{STFM}$ can be coupled with any payment ($\mathbf{p}^{STFM}$) and burning ($\mathbf{q}^{STFM}$) rules to define $\mathcal{T}^{STFM}$. E.g., similar to FPA, we can create $\mathcal{T}^{STFM}$ such that each bidding agent i whose $t_i \in B_k$ pays $p_i^{STFM} = b_i$ and $p_i^{STFM} = 0$ otherwise. Furthermore, $q_i^{STFM} = 0, \ \forall i$.

8.3.1.2 STFM: Fairness Properties

The choice of the payment and burning rules impact the strategyproofness, w.r.t. both the agent and the miner, of the resulting STFM mechanism. However, Theorem 8.3 proves that the STFM allocation from Definition 8.4 is sufficient to satisfy both our fairness notions.

Theorem 8.3. *$\mathcal{T}^{STFM}$ (refer to Definition 8.4) with $\gamma \in (0, \infty)$ satisfies (i) Zero-fee Transaction Inclusion and (ii) Monotonicity.*

Proof. We first prove that STFM satisfies Monotonicity irrespective of the payment and burning rules.

For this, we must show that $\forall t_i, t_j \in M$ s.t. $t_i \neq t_j$ if $b_i > b_j$, $\Pr(t_i \in B_k) > \Pr(t_j \in B_k)$. We remark that $\mathbf{x}^{STFM}$ admits transactions with a distribution generated by applying the softmax function on the transactions in M (refer Algorithm 6).

For sampling the first transaction, the probability distribution is $\Pr(t_i \in B_k) = \frac{\exp(b_i/\gamma)}{\sum_{i' \in M} \exp(b_{i'}/\gamma)}$, $\forall i \in M$. Trivially, we have $\frac{\exp(b_i/\gamma)}{\sum_{i'} \exp(b_{i'}/\gamma)} > \frac{\exp(b_j/\gamma)}{\sum_{i'} \exp(b_{i'}/\gamma)}$ if $b_i > b_j$ and $\gamma > 0$, implying STFM satisfies Monotonicity in this case. Next, w.l.o.g., we assume a transaction t_l was sampled. The re-generated probability distribution becomes, $\Pr(t_i \in B_k) = \frac{\exp(b_i/\gamma)}{\sum_{i' \in M \setminus \{l\}} \exp(b_{i'}/\gamma)}$, $\forall i \in M \setminus \{l\}$. Still, we have $\frac{\exp(b_i/\gamma)}{\sum_{i'} \exp(b_{i' \in M \setminus \{l\}}/\gamma)} > \frac{\exp(b_j/\gamma)}{\sum_{i'} \exp(b_{i' \in M \setminus \{l\}}/\gamma)}$ if $b_i > b_j$ and $\gamma > 0$. That is, Monotonicity still holds. Along similar lines, we can show that Monotonicity holds for each sampling stage.

Trivially, we can also show that STFM satisfies Zero-fee Transaction Inclusion (ZTi). For each $t_i \in M$ with $b_i = 0$, we have $\Pr(t_i \in B_k) = \frac{\exp(b_i/\gamma)}{\sum_{i' \in M} \exp(b_{i'}/\gamma)} = \frac{1}{\sum_{i' \in M} \exp(b_{i'}/\gamma)} > 0$, irrespective of the size of M. □

8.3.1.3 Softmax TFM: Incentive Properties

As aforementioned, the incentive properties of STFM are a function of the underlying payment and burning rules. Theorem 8.4 presents the general impossibility of MIC for STFM with any payment rule, which increases monotonically with the transaction fees.

Theorem 8.4. *Given $\gamma \in (0, \infty)$ and with any non-trivial, monotonically increasing $\mathbf{p}^{STFM}$, i.e., for $b_i > b_j \implies p_i > p_j$ $\forall i, j$ s.t. $i \neq j$, $\mathcal{T}^{STFM}$ does not satisfy MIC.*

Proof. For the proof, we have to show that for any non-trivial payment rule, the intended allocation $\mathbf{x}^{STFM}$ in $\mathcal{T}^{STFM}$ is such that the miner has the incentive to deviate.

Given the mempool M, denote $Z \subset M$ as the set of all zero-fee transactions, i.e., $Z = \{t_i \mid t_i \in M \text{ and } b_i = 0\}$. For all game instances of $\mathcal{T}^{STFM}$ where the block B_k's size C is less than the size of the transactions in $M - Z$, we have $\Pr(t_i \in B_k) = 0, \forall t_i \in Z$. That is, the miner has no incentive to add transactions in Z to B_k. This is because as the payment rule increases with the transaction fees, the miner's utility from greedily adding transactions from $M - Z$ will be strictly greater than including even a single transaction from Z. □

We now discuss DSIC and MIC guarantees for STFM with FPA and EIP-1559 payment and burning rules.

Remark 8.3. From the perspective of the bidding agent, STFM's allocation rule does not change its behavior as the allocation rule satisfies Monotonicity. As such, any instance of STFM with FPA does not satisfy DSIC, as the first-price payment rule is well-known not to be DSIC. Furthermore, STFM with EIP-1559 satisfies DSIC only when the underlying EIP-1559 is DSIC (refer to Chapter 3.2.4).

8.3.1.4 STFM: Cost of Fairness

Similar to CoF guarantees for BitcoinZF, Theorem 8.5 provides an upper bound on CoF for STFM. We obtain the bound by selecting the worst-case distribution of bids which maximize $\mathbb{E}[OPT]$ and minimize $\mathbb{E}[u_{\mathsf{M}}^{STFM}]$.

Theorem 8.5. *For STFM with FPA, average* $\mathsf{CoF}_{STFM} = \frac{OPT}{\mathbb{E}_{\mathbf{x}\sim\Gamma_k}[u_M^{STFM}]} = \frac{n}{c} + 1$. *Here, n denotes the total transactions in M and c the maximum number of transactions included in B_k.*

Proof. Denote C and N as the block and mempool sizes, respectively. Let OPT denote miner's utility from Eq. 3.10 with $p_t = b_t$ and $q_t = 0$, $\forall t$ and u_m^{STFM} denote miner's utility for $\mathcal{T}^{STFM}$. Let M comprise n transactions with fees $\{b_i\}_{i=1}^n$. W.l.o.g, we consider $b_1 \geq b_2 \geq \ldots \geq b_n$. Let c denote the maximum number of transactions in a block. The block size is C, and for simplification, we assume that transactions are the same size.[3]

Miner's optimal utility from Eq. 3.10 is: $OPT = \sum_{i=1}^{c} b_i$. Let X denote the utility from sampling one transaction from M using $\mathcal{T}^{STFM}$. Then, $\mathbb{E}[X] = \sum_{i=1}^{n} \Pr(t_i \in B_k) \cdot b_i$. Further, if X_i is the utility from i^{th} sampled transaction (out of total c transactions present in a block), then the expected utility is given by

$$\mathbb{E}[u_m^{STFM}] = \mathbb{E}[\sum_{x=1}^{c} X_x] = |c| \sum_{i=1}^{n} b_i \Pr(t_i \in B_k).$$

We get the last equation using the linearity of expectations. Therefore, the ratio of utilities is,

$$\frac{OPT}{\mathbb{E}[u_{\mathsf{M}}^{STFM}]} = \frac{\sum_{i=1}^{c} b_i}{|c| \sum_{i=1}^{n} b_i \Pr(t_i \in B_k)}$$

For maximizing $\frac{OPT}{\mathbb{E}[u_{\mathsf{M}}^{STFM}]}$, we need to maximize the numerator and minimize the denominator. This is achieved by taking $b_1 = b_2 = \ldots = b_c = b$ and $b_{c+1} = b_{c+2} = \ldots = b_n = 0$. That is,

$$\frac{OPT}{\mathbb{E}[u_{\mathsf{M}}^{STFM}]} = \frac{c \cdot b}{c(\sum_{i=1}^{c} b \cdot \frac{e^{\frac{b}{\gamma}}}{n-c+c \cdot e^{\frac{b}{\gamma}}} + 0)}$$

$$\frac{OPT}{\mathbb{E}[u_{\mathsf{M}}^{STFM}]} = \frac{n - c + c \cdot e^{\frac{b}{\gamma}}}{c \cdot e^{\frac{b}{\gamma}}} = \frac{n}{c} + 1 - e^{-\frac{b}{\gamma}}$$

Upper bound on utility-loss ($\frac{OPT}{\mathbb{E}[u_{\mathsf{M}}^{STFM}]}$) is found when $b \to \infty$ and is equal to $\frac{n}{C} + 1$. □

Note. Despite STFM satisfying ZTi and Monotonicity, Theorem 8.4 states that it is not MIC under any monotone payment rule. To this end, we next leverage the blockchain's

[3] if N and C are large enough, then with very high probability, the number of transactions n (or c) in a pool (or block) of size N (or C) will deviate from n (or c) negligibly. This observation follows from the Chernoff bound.

verifiable randomness to propose rTFM a TFM that satisfies our fairness notions while simultaneously guaranteeing MIC.

8.4 rTFM: Fairness in Transaction Fees Mechanism using Randomization

We now propose a rTFM: a TFM that uses trusted on-chain randomness to guarantee both our fairness constraints, namely (i) *ZTi* (Zero-Fee Transaction inclusion) and (ii) *Monotonicity*. In addition to this, the proposed rTFM is both Miner Incentive Compatible (MIC) and Dominant Strategy Incentive Compatible (DSIC).

We next (i) introduce randomized Transaction Fees Inclusion rTFM, (ii) show that when paired with the payment rules of FPA and EIP-1559, preserves their incentive guarantees while simultaneously satisfying the fairness notions (i) Monotonicity and (ii) ZTi.

8.4.1 rTFM: Randomized TFM

We denote rTFM as the tuple $\mathcal{T}_{\phi}^{\text{rTFM}} = \left(\mathbf{x}_{\phi}^{\text{rTFM}}, \mathbf{p}, \mathbf{q}, \tau_{\mathbf{R}}\right)$. At its core, rTFM comprises a novel allocation rule, $\mathbf{x}^{\text{rTFM}}$, and can be paired with any payment and burning rule. The allocation rule uses two sub-procedures: (i) *transaction sampling* and (ii) *biased coin-toss*. We first introduce these procedures and subsequently use them to formally define $\mathbf{x}_{\phi}^{\text{rTFM}}$.

8.4.1.1 Transaction Sampling

An honest miner of a block adds transactions from the mempool M to its block using the following rules.

- Rule 1: The miner uniformly adds transactions from the mempool M to its block B_k. But, for each transaction $t \in B_k$, the miner receives zero fees. That is, $\forall t \in B_k, p_t = 0$. Denote the Merkle tree constructed using these transactions as MT_{rand} with root root_{rand}.

- Rule 2: The miner selects the transactions optimally, i.e., using Eq. 3.10. Denote the Merkle tree constructed using these transactions as MT_{opt} with root $\mathsf{root}_{\mathsf{rand}}$.

While mining a block, the miner selects transactions and constructs Merkle trees according to Rule 1 and Rule 2. Denote the transaction selection rule, given M, be represented as $\text{SAMPLE}(M) = ((\mathsf{root}_{rand}, MT_{\mathsf{rand}}), (\mathsf{root}_{\mathsf{rand}}, MT_{opt}))$.

Algorithm 7 Randomized TFM (rTFM) based Mining

Input: Block Size C, Mempool M, Zero-Fees probability ϕ, parent Block B_{k-1}, Target difficulty TD

Output: (MT_k, B_k) Merkle Tree MT_k of selected transactions and Mined block B_k

1: **procedure** MINEBLOCK(C, M, p, B_{k-1})
2: $((root_{zf}, MT_{\mathsf{rand}}), (root_{opt}, MT_{opt})) \leftarrow$ SAMPLE(M)
3: $n \leftarrow$ RANDOM() ▷ Select a random nonce
4: $B_k \leftarrow (B_{k-1}, root_{zf}, root_{opt}, n)$ ▷ Construct block B_k
5: **while** HASH(B_k) $\geq TD$ **do**
6: $n \leftarrow$ RANDOM()
7: $B_k \leftarrow (B_{k-1}, root_{zf}, root_{opt}, n)$
8: **end while**
9: **if** HASH(B_k) $\leq \phi \cdot TD$ **then** ▷ Outcome of biased coin toss
10: **return** $(MT_{\mathsf{rand}}, B_k)$
11: **else**
12: **return** (MT_{opt}, B_k)
13: **end if**
14: **end procedure**

8.4.1.2 Trusted Biased Coin Toss

rTFM's allocation rule selects one out of the two sets of transactions created from Rules 1 and 2. We now introduce an *on-chain-based* biased coin toss method to select between the two sets. Let $\phi \in [0,1]$ denote the probability of heads (or 0) and $1-\phi$ denote the probability of tails (or 1).

From Chapter 8.1.2.1, a miner mines its block B_k at height k using the hash of the parent block $\textsc{Hash}(B_{k-1})$, the random nonce `rand`, the block height k, the two Merkle roots root_{rand} and $\mathsf{root}_{\mathsf{rand}}$. If the block is mined, i.e., $\textsc{Hash}(B_k) < TD$ for target difficulty TD, then the toss' outcome is considered as follows:

$$O(\textsc{Hash}(B_k), \phi) = \begin{cases} 0 & \text{if } \textsc{Hash}(B_k) < \phi \cdot TD \\ 1 & \text{otherwise} \end{cases} \tag{8.4}$$

Remark 8.4. *Invoking $O(\textsc{Hash}(B_k), \phi)$ (Eq. 8.4) for a mined block B_k is equivalent to a biased coin toss with ϕ as the probability of heads.*

Proof. From Eq. 8.4, we get $O(\textsc{Hash}(B_k), \phi) = 1$ if $\textsc{Hash}(B_k) < \phi \cdot TD$. For any hash function $\textsc{Hash} : \{0,1\}^* \to \{0,1\}^\lambda$, the pre-image guarantee [242] implies $\textsc{Hash}(B_k) \in_R \{0,1\}^\lambda$. However, since we are considering invocation for a mined block, we have $\textsc{Hash}(B_k) \in_R \{0,1,\ldots,TD-1\}$. As such, $\Pr(\textsc{Hash}(B_k) < \phi \cdot TD) = \phi$. The outcome $O(\textsc{Hash}(B_k), \phi) = 1$ is with probability ϕ. We, therefore, get the equivalence by mapping this outcome to "Heads" in a biased coin toss. □

Given this, Algorithm 7 provides the procedural outline of $\mathbf{x}_\phi^{\text{rTFM}}$. The procedure is summarized as follows:

Step 1. Miner samples two merkle trees MT_{rand} and MT_{opt} by invoking $\textsc{sample}(M)$ and include both merkle roots $root_{zf}$ and $root_{opt}$ in block header B_k.

Step 2. Miner selects a random nonce for the block header B_k until the block is mined; i.e. $\textsc{Hash}(B_k) < TD$.

Step 3. Miner invokes biased coin toss $O(\textsc{Hash}(B_k), \phi)$ (Eq. 8.4). If the outcome is 1, then MT_{opt} (optimally selected transactions) is considered part of the blockchain. If the outcome is 0, then MT_{rand} (Merkle tree with transactions sampled uniformly from M but with $p_t = 0$) is considered part of the blockchain.

To summarize, Definition 8.5 formally defines $\mathbf{x}^{\text{rTFM}}$.

Definition 8.5: rTFM Allocation Rule

Given $\mathcal{H}, M$ and B_k, let x_ϕ^{rTFM} denote a feasible allocation rule generated using Algorithm 7. Formally, the set of allocated transactions $x^{\text{rTFM}}(\mathcal{H}, M, B_k, C, \phi) = \mathcal{X}_k$ for block B_k is obtained from $(\mathcal{X}_k, B_k) \leftarrow \textsc{MineBlock}(C, M, p, \mathcal{H})$.

rTFM Payment and Burning Rule. The allocation rule $\mathbf{x}_\phi^{\text{rTFM}}$ can be coupled with any payment ($\mathbf{p}$) and burning ($\mathbf{q}$) rules to define $\mathcal{T}_\phi^{\text{rTFM}}$. E.g., similar to FPA, we can create $\mathcal{T}_\phi^{\text{rTFM}}$ such that each bidding agent i whose $t_i \in \mathcal{X}_k$ for $(\mathcal{X}_k, B_k) \leftarrow \textsc{MineBlock}(C, M, p, \mathcal{H})$ has $p_i^{FPA} = b_i$ else $p_i^{FPA} = 0$. In both cases, $q_i^{\text{rTFM}} = 0$.

8.4.2 rTFM: Fairness Properties.

Here, we show that $\mathcal{T}_p^{\text{rTFM}} = \left(\mathbf{x}^{\text{rTFM}}, \mathbf{p}^{\mathbf{EIP-1559}}, \mathbf{q}^{\mathbf{EIP-1559}}, \tau_{\mathbf{R}}\right)$, i.e. TFM with rTFM transaction selection mechanism and EIP-1559 payment rule satisfies both Monotonicity and Zero-Fee Transaction Inclusion properties.

Theorem 8.6. rTFM *with EIP-1559 satisfies (i) Monotonicity and (ii) Zero-Fee Transaction Inclusion for any* $\phi \in (0, 1)$.

Proof. We show the proof in two steps.

- **Monotonicity.** To show that rTFM satisfies Monotonicity, we have to show that increasing the bid b_t of an arbitrary transaction t increases its probability of acceptance in rTFM (given the remaining bids are fixed). For this, first, let us write down the probability of any transaction $t \in M$ with bid b_t getting added to the block B_k. We have,

$$\Pr(t \in B_k) = \phi \cdot \Pr(t \in MT_{opt}) + (1 - \phi) \cdot \Pr(t \in MT_{\mathsf{rand}}) \tag{8.5}$$

  Now, assume that the new bid is $b_t + \epsilon$ for any $\epsilon > 0$. If $\Pr(t \in B_k)$ increases in Eq. 8.5 for $b_t + \epsilon$ (compared to the bid b_t), rTFM satisfies Monotonicity. Note that the term $\Pr(t \in MT_{\mathsf{rand}}$ remains the same for both bids b_t and $b_t + \epsilon$ as the miner does the allocation uniformly. The term $\Pr(t \in MT_{opt})$ can only increase for $b_t + \epsilon$ compared to b_t, for some ϵ. This is because, in MT_{opt}, the miner adds the transactions optimally, i.e., using Eq. 3.10. That is, rTFM satisfies Monotonicity.

- **Zero-fee Transaction Inclusion.** For any $\phi \in (0, 1)$ and Eq. 8.5, the probability of any transaction t with $b_t = 0$ being part of the block B_k is trivially non-zero. This is because $\Pr(t \in MT_{\mathsf{rand}}$ will be non-zero even if $b_t = 0$.

These two steps complete the proof of the theorem. □

8.4.3 rTFM : Incentive Properties

From Definition 3.23, TFMs comprise a transaction selection mechanism, $\mathbf{x}_{\phi}^{\text{rTFM}}$, payment $\mathbf{p}^{\text{rTFM}}$ and burning $\mathbf{q}^{\text{rTFM}}$ rules. We now discuss the incentive properties of $\mathcal{T}_{\phi}^{\text{rTFM}}$ with payment rules of (i) First Price Auction (FPA) and (ii) EIP-1559.

First, we show that rTFM satisfies MIC for both FPA and EIP-1559 payment rules. Moreover, rTFM is DSIC when the payment rule is EIP-1559. Following this, we also show that rTFM satisfies fairness properties, namely (1) *ZTi* and (2) *Monotonicity*.

rTFM with FPA. The payment rule for FPA for any selected transaction t_i with bid b_i is $p_i^{FPA} = b_i$ if $t_i \in B_k$ and $p_i^{FPA} = 0$ otherwise. In both cases, the burning rule is $q_i^{FPA} = 0$. Trivially, rTFM with FPA is not DSIC, while Theorem 8.8 proves that it satisfies MIC.

rTFM with EIP-1559. The EIP-1559 payment rule implies that for each bidding agent i whose $t_i \in B_k$ and $b_i \neq 0$ has $\mathbf{p_i^{EIP-1559}} = \mathbf{b_i} - \lambda$ and $\mathbf{q_i^{EIP-1559}} = \lambda$. Here, λ is the posted price determined by the network (refer to Chapter 3.2.4). With this, Theorem 8.7 shows that rTFM with EIP-1559 is DSIC. The mechanism is also MIC, as shown in Theorem 8.8.

Theorem 8.7. $\mathcal{T}_\phi^{\text{rTFM}} = \left(\mathbf{x}_\phi^{\text{rTFM}}, \mathbf{p}^{\mathbf{EIP-1559}}, \mathbf{q}^{\mathbf{EIP-1559}}, \tau_{\mathbf{R}}\right)$ *with EIP-1559 payment rule satisfies* Dominant Strategy Incentive Compatibility *(DSIC), if* λ *is excessively low.*

Proof. Theorem 8.7 follows trivially by observing that an agent's strategy does not depend on rTFM's allocation but only on the payment and the burning rule. The TFM also satisfies Monotonicity (Theorem 8.6). Thus, the DSIC guarantee of EIP-1559 carries over for rTFM with EIP-1559. □

Theorem 8.8. $\mathcal{T}_\phi^{\text{rTFM}} = \left(\mathbf{x}_\phi^{\text{rTFM}}, \mathbf{p}^{\text{rTFM}}, \mathbf{q}^{\text{rTFM}}, \tau_{\mathbf{R}}\right)$ *satisfies Miner Incentive Compatibility (MIC) when transaction allocation rule is* $\mathbf{x}_\phi^{\text{rTFM}}$ *and payment scheme* $\mathbf{p}^{\text{rTFM}}$ *and burning rule* $\mathbf{q}^{\text{rTFM}}$ *are either (1) First Price Auction or, (2) EIP-1559.*

Proof. To show that the TFMs satisfy MIC, we remark that the selecting between the optimal and zero-fee transactions (refer Algorithm 7) is carried out by the blockchain in a trusted manner (Eq. 8.4). As the miner has no control over the random outcome of $O(\textsc{Hash}(B_k), \phi)$ (Remark 8.4), its strategy involves (i) optimally selecting the transactions and (ii) either sampling the transactions uniformly from M, or using any other way, or keeping them empty. For (i), we know that both EIP-1559 and FPA payment rules satisfy MIC. For (ii), any strategy results in zero utility for the miner since the transaction

payments are set to zero. So, the miner has no incentive to deviate from the uniform sampling. That is, rTFM is MIC for the miner. □

Note. Theorem 8.1 does not apply to rTFM as the miner does not have control over which set of transactions are selected with $\mathbf{x}^{\text{rTFM}}$. We can trivially extend these results to show that rTFM with FPA satisfies both fairness notions and DSIC and MIC.

8.4.4 rTFM: Choosing ϕ

rTFM's allocation rule is parameterized by the probability ϕ of mining a block where each transaction t_i has bid $b_i = 0$. We now discuss the impact of ϕ on CoF and the variation in the miner's revenue.

Cost of Fairness (CoF). From Definition 8.3, CoF is the ratio of the utilities u_{OPT} (refer to Eq. 3.10) and u_{rTFM} (i.e., miner's utility when the transactions are selected according to $\mathbf{x}_{\phi}^{\text{rTFM}}$).

The miner's utility in rTFM is dependent on the output of random variable $O(\text{Hash}(B_k), \phi)$. If $O(\text{Hash}(B_k), \phi) = 0$ (occurs with probability ϕ), then each selected transaction t_i has $b_i = 0$ resulting in zero revenue for the miner. In contrast, with probability $1 - \phi$, we have $O(\text{Hash}(B_k), \phi) = 1$, such that the optimal transactions are selected. Here, the miner's revenue equals u_{OPT}. That is,

$$\mathbb{E}_{\phi}[u_{\text{rTFM}}] = \phi \cdot 0 + (1 - \phi) \cdot u_{OPT}.$$

This implies that, $CoF_{\text{rTFM}} = \frac{u_{OPT}}{\mathbb{E}_{\phi}[u_{\text{rTFM}}]} = \frac{1}{1-\phi}$.

Impact of ϕ on CoF. Trivially, an increase in ϕ increases ZTi. On the other hand, this also increases CoF, reducing the miner's revenue. However, since rTFM (with an appropriate payment rule) is MIC, we believe that the system designers must choose an appropriate ϕ which (i) incentivizes the miner to not abstain from the system and (ii) allows for a desirable percentage of zero-fee transactions that increases the cryptocurrency's market penetration.

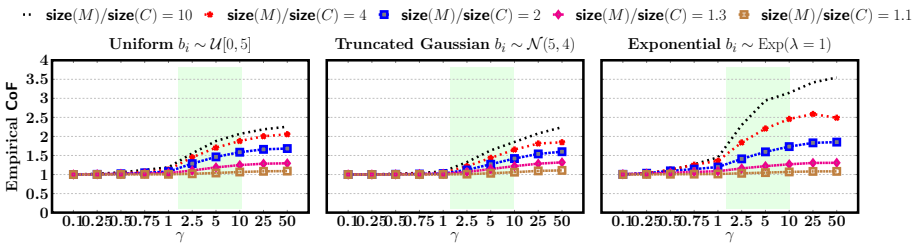


Figure 8.1: Empirical CoF for the distributions: (D1) Uniform, (D2) Truncated Gaussian and (D3) Exponential.

Coefficient of Variation (CoV). An increase in ϕ not only decreases the miner's expected revenue but will also *increase* its variance. More concretely, denote σ_{OPT} as the standard deviation and π_{OPT} as the miner's expected utility when it optimally selects the transactions. Likewise, σ_{RTFM} and π_{RTFM} are the standard deviation and expectation in the miner's utility from RTFM. We know that the Coefficient of Variation (CoV) is given by $\frac{\sigma}{\mu}$. By trivial arguments, we know the following:

$$CoV_{OPT} = \frac{\sigma_{OPT}}{u_{OPT}} = 1 \textbf{ and } CoV_{\text{RTFM}} = \frac{\sigma_{\text{RTFM}}}{\mathbb{E}_{\phi}[u_{\text{RTFM}}]} = \left(\frac{1-\phi}{\phi}\right)^{1/2}$$

Ideally, we want to choose ϕ such that $CoV^2_{OPT}/CoV^2_{\text{RTFM}}$ is maximized. Towards this, we observe that as $\phi \to 0$, the CoV ratio increases monotonically.

Choosing ϕ. The trade-off between (1) CoF, (2) CoV and (3) ZTi (probability of accepting zero-fee transactions) is such that as ϕ increases, CoF increases, CoV-ratio decreases, and ZTi increases. If we wish to increase the number of accepted zero-fee transactions, we must compromise with utility and suffer higher variance. Given these three factors and a suitable trade-off, the blockchain network must decide on an appropriate ϕ.

Discussion. Table 8.2 tabulates the results presented in this chapter. In summary, both STFM and RTFM satisfy ZTi and Monotonicity, with RTFM also being MIC (for an appropriate payment rule). However, STFM is relatively simpler to implement since it only requires miners to adapt to the new allocation rule. In contrast, RTFM will require

Table 8.2: Summary of our results. In conclusion, for appropriate payment and burning rules, rTFM simultaneously satisfies MIC and our novel fairness notions.

TFM	**DSIC**	**MIC**	**Monotonicity**	**ZTi**
EIP-1559 [43]	✓*	✓	✓ (Rem. 8.1)	✗ (Rem. 8.1)
Uniform TFM [62]	✗	✗	✗ (Rem. 8.2)	✓ (Rem. 8.2)
BitcoinZF [258]	✗	✗‡ (Clm. 8.1)	✓ (Thm. 8.2)	✓† (Thm. 8.2)
STFM + FPA	✗ (Rem. 8.3)	✗ (Thm. 8.4)	✓ (Thm. 8.3)	✓ (Thm. 8.3)
STFM + EIP-1559	✓* (Rem. 8.3)	✗ (Thm. 8.4)	✓ (Thm. 8.3)	✓ (Thm. 8.3)
rTFM + FPA	✗	✓ (Thm. 8.8)	✓	✓
rTFM + EIP-1559	✓* (Thm. 8.7)	✓ (Thm. 8.8)	✓ (Thm. 8.6)	✓ (Thm. 8.6)

†: Only if $\forall\ t_i \in M$ with $b_i = 0$ we have $s_i \leq C_{1-\alpha}$.

$\star$: Only if λ is not excessively low

a fork of the blockchain. The coin-toss mechanism introduced for rTFM also requires a PoW blockchain, which is often resource-intensive. Future work can extend rTFM's allocation rule for other blockchains (e.g., Proof-of-Stake blockchains).

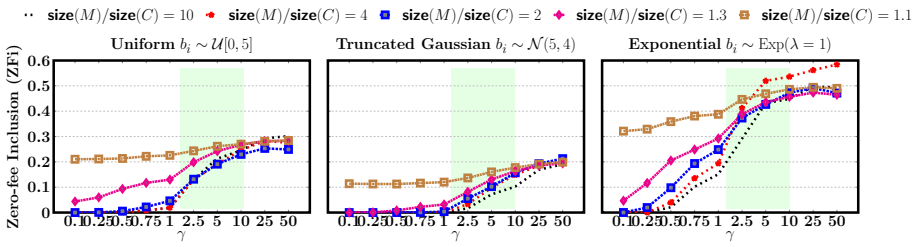


Figure 8.2: Zero-fee Inclusion (ZFi) for the distributions: (D1) Uniform, (D2) Truncated Gaussian and (D3) Exponential.

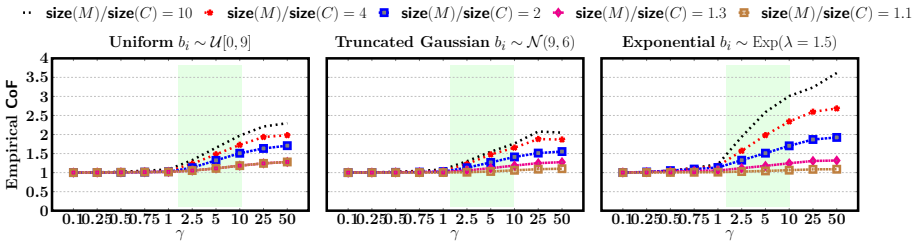


Figure 8.3: Empricial CoF: Miner's Utility Ratio for the distributions: (D1) Uniform, (D2) Truncated Gaussian and (D3) Exponential

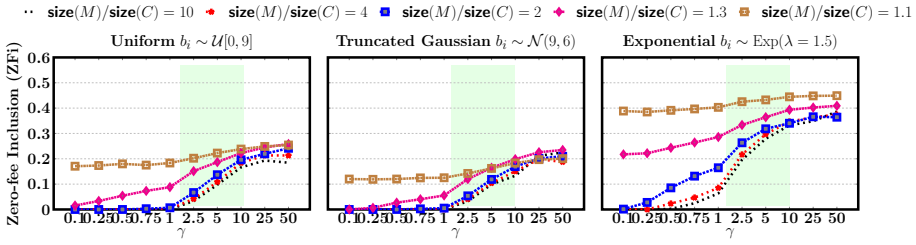


Figure 8.4: Zero-fee Inclusion (ZFi) for the distributions: (D1) Uniform, (D2) Truncated Gaussian and (D3) Exponential

8.5 Simulations

We now empirically validate STFM's performance with regard to the loss in the miner's utility and the fraction of zero-fee transactions included in the block.

8.5.1 Experimental Setup & Performance Measures

To simulate STFM, we need to configure the size of the mempool M, block size C, temperature parameter γ, and each agent's transaction fees and their sizes. In our experiments, we vary the ratio of the sizes of the mempool and block size, say $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$, in the set $\{1.1, 1.3, 2, 4, 10\}$ and $\gamma \in [0.1, 50]$. To concretely mimic all possible real-world scenarios, for each $t_i \in M$, the agent i samples its bid b_i from the following three distributions[4]: (D1) Uniform, i.e., $b_i \sim \mathcal{U}[0, 5]$, (D2) Truncated Gaussian, i.e., $b_i \sim \mathcal{N}(5, 4)$, and (D3) Exponential, i.e., $b_i \sim \text{Exp}(\lambda = 1)$.

Likewise, for each $t_i \in M$, the agent i samples the transaction's size $s_i \sim \text{Exp}(\lambda = 1)$. This choice is reasonable since smaller transactions (e.g., payer-payee token transfer) are more common than larger transactions (e.g., smart contract deployment). To measure STFM's performance, we also define the following measures.

1. Empirical CoF. This is the ratio of the miner's utility by greedily adding transactions to the block with the utility from STFM's allocation. The smaller the CoF, the better.
2. Zero-fee Inclusion (ZFi). ZFi is the ratio of the size of zero-fee transactions in the block with the total size of all the transactions in the block.

For each $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$ and γ, we sample the agent's bids based on D1, D2 and D3. We simulate the resulting game instances 100 times and report the average CoF and ZFi values.

8.5.2 Results & Discussion

Figure 8.1 and Figure 8.2 depict our results. Details follow.

[4]We observe similar trends for other distribution parameters.

Empirical CoF: Miner's Utility Ratio. We first discuss the change in CoF with varying γs and $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$ values for D1, D2 and D3. For all three distributions, we observe a consistent increase in CoF as γ increases, i.e., $\gamma \uparrow \implies u_{\mathsf{M}} \downarrow$. For $\gamma \in (0, 1)$, CoF is < 1.5 implying that miner's utility drop is > 0.67 times OPT. For $\gamma \geq 1$ CoF increases, but remains < 2.5 for D1, D2 and < 3.5 for D3. E.g., for $\gamma = 5$ and the worst-case value of $\frac{\mathsf{size}(M)}{\mathsf{size}(C)} = 10$, CoF values are 1.88 (D1), 1.63 (D2) and 2.93 (D3).

Furthermore, one way to interpret decreasing $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$ is an increase in the block size, $\mathsf{size}(C)$. As $\mathsf{size}(C) \to \mathsf{size}(M)$, the randomized allocation adopted with STFM plays a lesser role as the block gets large enough to accommodate most transactions. From Figure 8.1, we see that decreasing $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$ decreases CoF, i.e., an increase in the miner's utility.

Zero-fee Inclusion (ZFi). We empirically show that STFM admits zero-fee transactions with Figure 8.2. For varying γ, we plot the ratio of the size of zero-fee transactions included in the block with the total block size (aka ZFi). We make five major observations. First, for $\gamma \in (0, 1]$ and high $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$, ZFi values are ≈ 0. Second, ZFi consistently increases as γ decreases. Third, for $\gamma > 5$, ZFi values almost saturates at ≈ 0.3 (D1, D2) and ≈ 0.6 (D3) for all values of $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$. Fourth, as $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$ decreases, we observe significant ZFi even for $\gamma \in (0, 1)$. This is because smaller $\frac{\mathsf{size}(M)}{\mathsf{size}(C)}$ implies enough room for most of the available transactions. Lastly, since with D3, there is a greater chance of sampling lower b_is, its ZFi values are greater than D1 and D2.

The green shaded region depicts the range of γ with a practical CoF-ZFi trade-off. Specifically, for $\gamma \in (2, 10)$, we observe CoF < 2 and ZFi > 0.1, for all three distributions.

8.6 Conclusion

In this chapter, we focused on the need for fairness in TFMs regarding the transaction fees for the transaction creators. We argued that including zero-fee transactions is necessary for the greater adoption of cryptocurrencies. We introduced two novel fairness notions:

Zero-fee Transaction Inclusion (ZTi) and Monotonicity. We showed that existing TFMs do not satisfy at least one of these notions or do so for smaller transaction sizes and at a high cost to the miner's utility. To resolve these limitations, we first introduced STFM, which samples transactions through the distribution generated from the softmax with temperature (γ) function. We showed that STFM is a fair TFM but not MIC. To this end, we introduced rTFM which simultaneously satisfies MIC and our fairness notions.

Future Work. We believe these fair TFMs may further democratize cryptocurrencies by contributing to their broader accessibility and enhancing their adoption in the market. Future work can further study the role of ϕ in rTFM towards striking a desirable balance between a miner's revenue and the fraction of zero-fee transactions included. Last, as aforementioned, one can also explore extending rTFM for Proof-of-Stake blockchains.

Chapter 9

Designing Redistribution Mechanisms for Reducing Transaction Fees in Blockchains

Transaction Fee Mechanisms (TFMs) allocate agent transactions to blocks and determine their payments (i.e., transaction fees). Increasing demand and scarce block resources have led to high agent transaction fees. As these blockchains are a public resource, it may be preferable to reduce these transaction fees. To this end, we introduce Transaction Fee Redistribution Mechanisms (TFRMs) – redistributing VCG payments collected from such TFM as rebates to minimize transaction fees. Classic redistribution mechanisms (RMs) achieve this while ensuring Allocative Efficiency (AE) and Agent Incentive Compatibility (DSIC). Our first result shows the non-triviality of applying RM in TFMs. More concretely, we prove that it is impossible to reduce transaction fees when (i) transactions that are not confirmed do not receive rebates and (ii) the miner can strategically manipulate the mechanism. Driven by this, we propose Robust *TFRM (R-TFRM): a mechanism that compromises on an honest miner's individual rationality to guarantee strictly positive rebates to the agents. We then introduce* robust *and* rational *TFRM (R^2-TFRM) that uses trusted on-chain randomness that additionally guarantees miner's*

individual rationality (in expectation) and strictly positive rebates. Our results show that TFRMs provide a promising new direction for reducing transaction fees in public blockchains.

⋆ ⋆ ⋆ ⋆ ⋆

9.1 Introduction

In Chapter 3.2.2.4 and Chapter 8 we introduce Transaction Fee Mechanisms (TFMs) and present a fair TFRM, namely rTFM, respectively. In a nutshell, TFMs study the strategic interaction between the miner and the transaction creators (henceforth referred to as *agents*). Unfortunately, as previously discussed, an increasing demand, cryptocurrency's market volatility, and supply-demand economics have led to agents' over-paying [24]. E.g., Messias et al. [193] show that 30% of Bitcoin fees are two orders of magnitude more than recommended.

Considering public blockchains as a shared resource, it's desirable not to impose charges for transaction confirmation. However, given the infeasibility of confirming every transaction due to resource constraints, one may prefer only to confirm transactions with higher value (pertaining to their importance to agents). The absence of transaction fees could lead agents to misrepresent the value of their transactions to secure confirmation. Therefore, this chapter aims to design TFMs that minimize transaction fees while upholding other incentive-related properties.

Clearly, minimizing transaction fees is at odds with the miner's objective of maximizing revenue. Thus, the task of designing TFMs to minimize fees is more intricate than in the classical auction setting, primarily because, in TFMs, miners have complete control over the transactions they include in their blocks [245].

9.1.1 Chapter Contributions

9.1.1.1 Goal

We aim to design a TFM that satisfies certain game theoretic properties like (i) Allocative Efficiency (AE), confirmed transactions maximize the overall valuation, (ii) Agent Incentive Compatibility (DSIC): agents bid their true valuation, and Individual Rationality (IR): agents receive a non-negative utility. At the same time, the TFM must actively reduce transaction fees for agents, thereby enhancing the blockchain's appeal. Unfortunately, from the famous Green-Laffont Impossibility Theorem [188], we know that it is impossible to design a TFM that is both AE and DSIC and which guarantees *zero net transaction fees* – in mechanism design commonly referred to as *strong budget balance*.

Given this, our objective is to design a TFM that is both AE and DSIC while minimizing the transaction fees (or is *weakly budget balanced)*. Motivated from Maskin et al. [188], the mechanism design literature proposes the use of *Groves' Redistribution Mechanism* (RM) for this purpose [49, 132, 188]. In Groves' RM, the VCG mechanism is executed, and then the surplus money is redistributed among the agents while preserving other game-theoretic properties.

Along similar lines, this chapter introduces *Transaction Fee Redistribution Mechanisms* (TFRMs): a general class of TFMs based on RMs where the miner offers rebates from the transaction fees collected to the agents while retaining AE, DSIC, and IR. By offering agents rebates, TFRMs, in effect, reduce the transaction fees they pay. Figure 9.1 provides an overview.

9.1.1.2 TFRM: Challenges

Designing such TFRMs has the following primary challenges.

Miner IC (MIC): As miners possess complete control over the transactions included in their blocks [245], they may deviate from the intended TFM allocation rule (i.e., select-

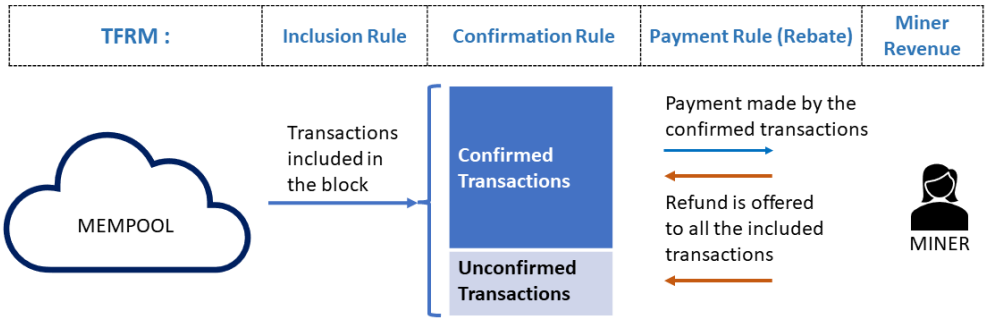


Figure 9.1: Overview of the framework for Transaction Fee Redistribution Mechanisms (TFRMs).

ing a different subset from the mempool). They may introduce "fake" transactions (i.e., transactions created strategically to increase their revenue) into their blocks [245]. This is similar to *shill bidding* [231] in traditional auctions. Thus, it is imperative that a TFRM maintains AE, DSIC, and low transaction fees even in the face of miner manipulation (or, alternately, in the presence of a *strategic miner*).

Roughgarden [245] introduces the notion of *miner IC* (MIC) to model the miner's strategic behavior. In auction theory, the Myerson-Satterthwaite impossibility theorem [202] states that it is impossible to design a mechanism that is AE, IR, weakly budget balanced, and IC for both sides of the market. Designing TFRMs is analogous to such a two-sided auction, and achieving both sides' IC (with other properties) is elusive.

Agent IC (DSIC): Typically, RMs ensure DSIC by offering rebates to everyone participating in the auction (irrespective of the allocation). In TFRMs, the transactions that are only part of the mempool (i.e., are not part of the block) are not available to the blockchain. Thus, unlike RMs, in TFRMs, we cannot offer rebates for each available transaction. As some transactions do not receive rebates, we can easily construct instances where the agents of these transactions have an incentive to overbid to get included in the block and receive rebates. Thus, ensuring DSIC in TFRM is non-trivial. As such, we propose *restricted*

DSIC (RDSIC), which ensures that bidding truthfully is a weakly dominant strategy *only* for the agents whose transactions are included in the block.

9.1.1.3 Chapter Contributions

Broadly, we (i) formally introduce TFRMs (refer to Figure 9.1 for an overview), (ii) analyze the challenges due to miner manipulation in vanilla-TFRMs, and (iii) introduce two novel TFRMs, namely R-TFRM and R^2-TFRM that are robust to miner manipulation. We discuss these in detail next.

1. `Ideal-TFRM`. As we cannot offer rebates to all transactions in the mempool, we begin our analysis with an "`Ideal-TFRM`" that offers non-zero rebates only to confirmed transactions. Unfortunately, we show that it is impossible for `Ideal-TFRM` to satisfy DSIC while offering non-zero rebates to confirmed transactions (Theorem 9.1).

2. **TFRM: Effect of A Strategic Miner.** We shift our focus to TFRMs that provide rebates to all transactions included in the block. An RM's effectiveness is measured using the Redistribution Index (RI) [132], which is the fraction of the VCG surplus redistributed. To absorb the effect of strategic miners, we introduce *Resilient Redistribution Index* (RRI). RRI measures the fraction of redistributed funds under optimal miner manipulation. We prove that it is impossible to design a TFRM that satisfies AE, RDSIC, and is IR for both agents (IR_u) and miners ($\text{IR}_\mathbf{M}$), while guaranteeing strictly positive RRI (Theorem 9.2).

3. **Robust TFRM (R-TFRM).** Given these impossibilities, we propose R-TFRM: a TFRM that guarantees strictly positive RRI *and* satisfies all agent-specific properties. However, R-TFRM is not individually rational for the miner[1].

[1]We remark that miners, or block proposers in general, often have alternate revenue streams (e.g., block rewards [203] or attestation rewards [260]). These rewards can primarily help absorb the reduction in revenue due to reduced transaction fees. They may also alleviate the lack of $\text{IR}_\mathbf{M}$ guarantee in R-TFRM.

At its core, R-TFRM builds on an RM with VCG payments as transaction fees and a linear rebate function. The rebate function maximizes the worst-case rebate while satisfying RDSIC, IR_u, and Approx-$\text{IR}_{\mathbf{M}}$. Designing such a rebate function is equivalent to solving the linear program in Figure 9.4 for its coefficients. We finally show that the payments are reduced by a fraction k/n, where k transactions are confirmed out of the n included in the block (Theorem 9.4). The fraction remains the same, even with miner manipulation, i.e., RRI is also k/n. In other words, each confirmed agent sees a reduction by $(1 - k/n)$ in its transaction fee compared to the equivalent VCG-based TFM.

4. **Robust and Rational TFRM (R^2-TFRM).** R-TFRM ensures positive RRI by compromising $\text{IR}_{\mathbf{M}}$. Another way of ensuring positive RRI is by randomly offering rebates to the agents. Such an approach guarantees $\text{IR}_{\mathbf{M}}$, in expectation. R^2-TFRM uses this approach wherein each agent receives the rebate given by R-TFRM with probability α and does not receive any rebate probability $1 - \alpha$. The randomization is carried out by the blockchain in a trusted manner [62]. Theorem 9.5 shows that for $\alpha \in (0, \overline{\alpha})$ & $\overline{\alpha} < 1$, R^2-TFRM is AE, IR_u, and RDSIC and $\text{IR}_{\mathbf{M}}$, in expectation. Further, it ensures an expected RRI of $\alpha \cdot k/n$.

9.1.2 Chapter Notations and Additional Background

This chapter extends the Transaction Fee Mechanism (TFM) design literature introduced formally in Chapter 3.2.2.4. As such, we follow the notations from Chapter 3.2.2.4. The solution concepts focused on in this chapter include DSIC in the TFM context (Definition 3.22) and Miner Incentive Compatibility (MIC) (Definition 3.23). To design Transaction Fee Redistribution Mechanisms (TFRM), we also rely on Redistribution Mechanisms (RMs), formally introduced in Chapter 2.2.5.7. The notations used in this table are summarized in Table 9.1.

9.1.2.1 TFM: Model

The TFM model considered in this chapter differs (slightly) compared to the original model from Chapter 3.2.2.4, or the 'fairer' TFMs presented in Chapter 9. Particularly, (i) we assume that all transactions are of the *same* size[2], and (ii) we extend the block space so that it contains *confirmed* transactions and *included* (or price-setting) transactions. More formally, our TFM definition is as follows:

Definition 9.1: Transaction Fee Mechanism (TFM) [62, 245]

Consider the TFM model from Definition 3.21 such that for any two transactions i, j $(i \neq j)$ in the mempool M, are of the same size, $s_i = s_j$. Given a bid profile $\mathbf{b}$, we define TFM as the tuple $\mathcal{T} := (\mathbf{x}^I, \mathbf{x}^C, \mathbf{p})$ where:

- $\mathbf{x}^I$ is a feasible block inclusion rule, i.e., $\sum_{i \in M} x_i^I(\mathbf{b}) \leq n$ where $x_i^I(\cdot) \in \{0, 1\}$. Let the set of included transactions be $I = \{i | x_i^I = 1, i \in M\}$
- $\mathbf{x}^C$ is a feasible block confirmation rule, i.e., $\sum_{i \in M} x_i^C(\mathbf{b}) \leq k$ where $x_i^C(\cdot) \in \{0, 1\}$. Let the set of confirmed transaction be $C = \{i : x_i^C = 1, i \in M\}$
- $\mathbf{p}$ is the payment rule with the payment for each included transaction, i.e., $\forall i \in \mathbf{x}^I$ the payment is denoted by $p_i(\mathbf{b}, \mathbf{x}^I, \mathbf{x}^C)$.

We use the example of a second-price TFM to explain Definition 9.1 better.

[2]From Chapter 2.2.5.7, this assumption makes our TFM setting analogous to the Redistribution Mechanism (RM) setting with homogeneous items with unit demand.

Example 9.1: Second-price TFM (SPA) [62, 245]

W.l.o.g., assume that $\mathbf{b} = (b_1, \ldots, b_m)$ are bids in decreasing order. Now, the inclusion rule is $x_i^I = 1,\ \forall i \in \{1, \ldots, n\}$ and zero otherwise, i.e., the top n transactions are included in the block. With $k = n - 1$, the confirmation rule is $x_i^C = 1,\ \forall i \in \{1, \ldots, k\}$ and zero otherwise. The top k (among n) transactions are confirmed, and the last included transaction is the price-setting transaction. Each confirmed agent $i \in [k]$ pays $p_i = b_{k+1}$ to the miner and unconfirmed agent $(i \in I \setminus C)$ pays $p_i = 0$. The miner's net revenue is $k \cdot b_{k+1}$.

9.1.2.2 TFM: Additional Incentive Properties

For this chapter's purposes, we will need an additional set of properties over the ones already defined in Chapter 3.2.2.4. We begin by defining Individual Rationality.

9.1.2.2.1 Individual Rationality (IR). To incentivize participation, mechanism designers also focus on IR (first introduced in Chapter 2).

Definition 9.2: (Ex-post) Individual Rationality (IR)

Given a TFM $\mathcal{T} = (\mathbf{x}^I, \mathbf{x}^C, \mathbf{p})$, we say that it satisfies IR for both the agent and miners if their utility post participation in the mechanism is non-negative, i.e., $u_i(\cdot) \geq 0, \forall i \in M$ and $u_{\mathbf{M}}(\cdot) \geq 0$.

Note. We denote a mechanism that is IR w.r.t. miner as $\text{IR}_{\mathbf{M}}$ and IR w.r.t. agent as IR_u.

As we show later, ensuring DSIC while minimizing transaction fees in a TFM is challenging. Hence, we focus on the incentive compatibility of a 'restricted' set of agents whose transactions are included in the block. We define restricted DSIC (RDSIC), which states that for all the included agents, reporting truthfully is IC irrespective of what the remaining included agents report.

Definition 9.3: Restricted DSIC (RDSIC)

Given a TFM $\mathcal{T} = (\mathbf{x}^I, \mathbf{x}^C, \mathbf{p})$, we say that RDSIC is satisfied if, $\forall i$ included in the block i.e., $\forall i \in I$ we have,

$$u_i(\theta_i, b_i^\star = \theta_i, \mathbf{b}_{I\setminus i}) \geq u_i(\theta_i, b_i, \mathbf{b}_{I\setminus i}), \forall \theta_i, \mathbf{b}_{I\setminus i}$$

where $\mathbf{b}_{I\setminus i}$ is the bids of agents included in the block excluding agent i.

We will also study other game-theoretic properties such as Allocative Efficiency (Definition 2.18) and (strong) budget balance (Definition 2.19).

Notation	Definition
	TFM Model
$\mathcal{H}$	Blockchain history
$M := \{1, \ldots, m\}$	Mempool (set of outstanding transactions)
$\mathbf{x}^I$	Block inclusion rule
$\mathbf{x}^C$	Block confirmation rule
$\mathbf{p}$	Payment rule
$\mathcal{T} := (\mathbf{x}^I, \mathbf{x}^C, \mathbf{p})$	TFM Tuple
$\mathbf{b} := \{b_1, \ldots, b_m\}$	Bids present in the mempool
$\Theta := [\theta_i]$	Set of valuations with each agent i's valuation θ_i
$n \in \mathbb{Z}_{\geq 1}$	Number of included transactions
$k \in \mathbb{Z} : k \in [1, n]$	Number of confirmed transactions
$\mathbf{b}_I$	Bids of users included in the block
$\mathbf{b}_{I \setminus i}$	Bids of users included in the block without agent i
$u_i(\theta_i, \mathbf{b})$	Utility of agent i
F	Set of fake transactions added by the miner
$u_{\mathsf{M}}(F, \mathbf{b})$	Utility of the miner
	TFRM Model
$g(\cdot)$	Rebate Function
r_i	Rebate to user i
$c_i \in \mathbb{R},\ \forall i \in \mathbf{b}_I$	Rebate function constants
e_{wc} and e_{avg}	Worst-case/Average Redistribution Index
$\hat{e}_{\mathsf{wc}}$	Resilient Redistribution Index (RRI)

Table 9.1: Chapter Notations

9.2 `Ideal-TFRM`: Impossibility of Achieving Strictly Positive Redistribution Index

We now present a first attempt at implementing an RM for minimizing transaction fees using the second-price TFM from Example 9.1). In a second-price TFM, the transactions/bids are sorted in decreasing order. The top k bids are confirmed with the $n = (k+1)^{th}$ transaction as the price-setting one. Each confirmed transaction pays $p = b_{k+1}$; with the net miner revenue as $k \cdot b_{k+1}$. To minimize the agent fees, we must "redistribute" the collected surplus.

It may be preferable to only provide rebates to agents whose transactions are confirmed (i.e., are among the top k bids) since each such agent pays b_{k+1}. If we also provide rebates to the remaining $n-k$ agents, the remaining transactions in the mempool may prefer to overbid just enough to get included in the block. By doing so, they can grab rebates for free[3]. Thus, to achieve DSIC, we must not provide rebates to unconfirmed transactions. With this motivation, we propose the following.

`Ideal-TFRM`. The goal is to maximize the fraction of VCG payments redistributed to the agents, denoted by f while ensuring non-zero rebates only to confirmed transactions. Further, in `Ideal-TFRM`, we would like the rebate offered to each agent to be less than the payment it makes. Eq. 9.1 captures this optimization.

$$\left.\begin{aligned} \max_{r_i, i\in\mathbf{x}} f \ \textbf{s.t.} \quad & \textstyle\sum_{i\in C} r_i \geq f \cdot \sum_{i\in C} p_i \\ & \textbf{and } p_i \geq r_i, \ \forall i \in C \ \textbf{and } r_i = 0, \ \forall i \in I \setminus C \end{aligned}\right\} \tag{9.1}$$

Here, $r_i = g(b_1, b_2, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n)$ is the rebate (Definition 2.24) for each agent $i \in M$ with p_i as the VCG payment. The goal is to find an optimal $g(\cdot)$ such that e_{wc} is maximized.

[3]Assigning future costs to transactions not confirmed, e.g., as in [62], may help overcome such manipulation. We leave the analysis for future work.

Unfortunately, we now show that for both the worst and average-case, `Ideal-TFRM` admits zero rebates for the agents with confirmed transactions, i.e., $r_i = 0,\ \forall i \in \mathbf{x}^C$ while guaranteeing DSIC.

9.2.1 Worst-case Rebate

Theorem 9.1 formally shows the impossibility of simultaneously guaranteeing DSIC and minimizing transaction fees in `Ideal-TFRM`.

Theorem 9.1 (`Ideal-TFRM` Impossibility)**.** *If $r^\star$ is an anonymous rebate function that satisfies Theorem 2.7, no `Ideal-TFRM` can guarantee a non-zero redistribution index (RI) in the worst case.*

Proof. Consider the bid vector, $(v_1, \ldots, v_n)$ and a DSIC rebate function $r_i^\star = g(v_{-i})$ as given by Theorem 2.7 satisfying Equation 9.1. W.l.o.g we assume $r_1^\star = g(v_2, \ldots, v_n) > 0$, that is agent 1 receives strictly positive rebate. Further $r_n^\star = g(v_1, \ldots, v_{n-1}) = 0$ since the last agent is not confirmed and receives strictly zero rebates as per the constraint of Eq. 9.1. We now construct another bid vector $(v_1', \ldots, v_n')$ such that $v_1' = v_n, v_2' = v_1, v_3' = v_4, \ldots, v_n' = v_{n-1}$. Under this bid profile, $r_1^\star = g(v_2', \ldots, v_n') = g(v_1, \ldots, v_{n-1}) = 0$, since for agent n, $r_n^* = 0$. Hence, the worst-case refund for the first agent will be 0, contradicting our assumption that $r_1^\star > 0$. Similarly, we can construct bid vectors to show that for every agent $i \in C$, the worst case refund $r_i^* = 0$. □

9.2.2 Average-case Rebate

Theorem 9.1 shows that $e_{\mathsf{wc}} = 0$ in `Ideal-TFRM`, for a linear rebate function. We now aim to find a *non-linear* rebate function that maximizes e_{avg} in `Ideal-TFRM`. However, it is analytically intractable to characterize similar results to show the outcome of a rebate function that maximizes e_{avg}. As such, we simulate the optimization in Eq. 9.1 as a Neural Network (NN), similar to [184, 270, 90].

9.2.2.1 Architecture & Setup

We consider a typical 3-layer feed-forward NN with bias, ReLU activation, and with AdamW optimizer. The input to our NN is the n-dimensional bid vector $\mathbf{b}_I$ sampled from a specific distribution. Each hidden layer comprises $2n$ neurons, with n as the output layer's dimension. Given $\mathbf{b}_I$, the NN computes the payments and rebates to the confirmed and included transactions.

9.2.2.2 Loss Function

For optimization, our loss function is a weighted sum of the following three quantities: (i) average rebate to the n bidders (denote as r_{avg}), (ii) feasibility, i.e., $\sum_i r_i \leq \sum_i p_i$ (denote as r_{feas}) and (iii) zero-rebate, i.e., $r_i = 0, \forall i \in I \setminus C$ (denote as r_{zero}. More concretely, for weights $\beta_1, \beta_2 \in (0, 1)$ the loss function takes the form: $\mathsf{Loss} = r_{\mathsf{avg}} + \beta_1 \cdot r_{\mathsf{feas}} + \beta_2 \cdot r_{\mathsf{zero}}$.

9.2.2.3 Training Details

We keep $n = 10$ with number of confirmed transactions as $k = 7$. For the optimizer, we choose a fixed learning rate $\eta = 5e - 4$. The batch size is 1000, and we train for 50,000 epochs. The code is available with the supplementary.

9.2.2.4 Results

We observe that $e_{\mathsf{avg}} \approx 0$ when transactions are sampled from $\mathcal{U}[0, 1]$ and $\mathcal{N}(0, 1)$. That is, the average case rebate to confirmed transactions is zero, even with non-linear rebate functions.

We conclude that it is impossible to design a TFRM with a linear rebate function that is DSIC (Theorem 2.7) and offers a non-zero rebate to any agent. Our experiments also highlight the same for non-linear rebate functions. Therefore, the next section introduces the general TFRM framework, where we focus on restricted DSIC.

1. *Inclusion Rule* ($\mathbf{x}^I$). Select highest n transactions from the mempool, M. W.l.o.g., assume that these n transactions are ordered as $b_1 \geq b_2 \geq \ldots \geq b_n \implies x_i^I = 1,\ \forall i \in \{1, \ldots, n\}$.
2. *Confirmation Rule* ($\mathbf{x}^C$). Select highest k bids from the n included, $x_i^C = 1, \forall i \in [k]$, where $k \leq n-2$.
3. *Payment Rule* ($\mathbf{p}$). Each confirmed agent i (i.e., $i \in C$) pays $p_i = b_{k+1} - r_i$. Each included but not confirmed agent j ($j \in I \setminus C$) pays $p_j = -r_j$.
4. *Miner Revenue Rule.* The miner receives the net revenue of $\sum_{i=1}^{n} p_i$.

Figure 9.2: Transaction Fees Redistribution Mechanism (TFRM): General Framework

9.3 Transaction Fee Redistribution Mechanism (TFRM)

As both $e_{\mathsf{avg}} \approx e_{\mathsf{wc}} = 0$ for `Ideal-TFRM`, we must also provide rebates to agents whose transactions are included but not confirmed. With this, we present the general TFRM framework in Figure 9.2.

In a TFRM, out of the m outstanding transactions in the mempool, we include the n highest bids in the block (denoted by the set I). Among the bids in the block, we confirm the k highest bids (denoted by the set C) where $n \geq k+2$. The remaining bids (denoted by the set P) are included but not confirmed; we refer to them as included or price-setting transactions. That is, $|I| = n, |C| = k$ and $|P| = n-k$. W.lo.g., we assume that $b_1 \geq b_2 \geq \ldots \geq b_n$. Hence, $C = \{b_1, \ldots b_k\}$ and $P = \{b_{k+1}, \ldots, b_n\}$. Each agent i's payment is computed based on the VCG payments and a rebate function, with r_i as the rebate to agent i. Since k bids are confirmed, the VCG payment for the confirmed transactions is the $(k+1)^{th}$ highest bid, i.e., b_{k+1}.

9.3.1 TFRM: RDSIC

While the rebate function satisfies Theorem 2.7, note that TFRM is not DSIC. E.g., agents not part of the block may report $b > \theta$ to grab the additional rebate, as only confirmed bids pay the transaction fee. However, TFRM satisfies RDSIC, i.e., it is DSIC for agents included in the block to bid their true valuation. All included agents, confirmed or not, are offered rebates, implying that k slots offered to n agents is equivalent to the allocation of k resources among n agents. Thus, by Theorem 2.7, TFRM is RDSIC.

We now show that in the presence of a strategic miner, the TFRM in Figure 9.2 results in net zero rebates to confirmed agents.

9.3.2 TFRM: Effect of Strategic Miners

In general, strategic miners may introduce fake transactions to increase their revenue. In the second-price TFM itself (Example 9.1), the miner may introduce a fake bid $\hat{b}_{k+1} = b_k$ to increase its revenue to $k \cdot b_k$ from the intended $k \cdot b_{k+1}$. Similarly, fake transactions can also affect the rebate offered by a TFRM. The miner's deviation may result in the following: (i) Fake bids affect the rebate of all agents, potentially reducing the rebate, and (ii) As a fake bidder, the miner pays the rebate to itself.

Thus, designing TFRMs to minimize transaction fees may only work if made resilient to such strategic manipulations. Unlike RMs, TFRMs must quantify the rebate redistributed to the genuine agents. Towards this, we define the following metric:

Definition 9.4: Resilient Redistribution Index (RRI)

Given that the miner manipulates the bids $\mathbf{b}$ to $\hat{\mathbf{b}}$, RRI is the fraction of the received payments that are redistributed in the worst case to the actual agents. Given C confirmed and P unconfirmed agents, let $S \subseteq I$ be the subset of agents that are not impersonated by the miner. Then: $\hat{e}_{\mathsf{wc}} = \inf_{\hat{\mathbf{b}}, p(\cdot,\hat{\mathbf{b}})\neq 0} \frac{\sum_{i\in S} r_i}{p(\cdot;\hat{\mathbf{b}})}$.

9.3.3 TFRM: Impossibility of Strictly Positive RRI

It is desirable to have a TFRM that is AE, RDSIC, $\text{IR}_{\mathbf{M}}$ and IR_u while ensuring strictly positive RRI in the worst-case, i.e., $\hat{e}_{\mathsf{wc}} > 0$. Unfortunately, Theorem 9.2 proves that it is impossible to design such a TFRM with strategic miners.

Theorem 9.2 (TFRM: RRI Impossibility)**.** *Given a strategic miner, it is impossible to design a TFRM with a linear rebate function that is RDSIC, AE, both IR_u and $IR_{\mathbf{M}}$, and guarantees a strictly positive RRI, i.e., $\hat{e}_{\mathsf{wc}} > 0$.*

Proof. Consider selecting k transactions from the total included n with $n - k$ as price-setting transactions. Any deterministic linear rebate function that is RDSIC must have the following form (from Theorem 2.7):

$$r_i = c_0 + c_1 b_1 + \ldots + b_i c_{i+1} + \ldots + c_{n-1} b_n$$

The above r_is satisfy both IR_u and $\text{IR}_{\mathbf{M}}$ only when the constants $c_i = 0, \forall i = 1, \ldots, k$. The proof for this can be found in [133, Claim 1]. Hence, in the rebate function, the top k agents have the following form:

$$r_i = c_{k+1} b_{k+2} + \ldots + c_{n-1} b_n, \ \forall i \leq k$$

The rebate offered for the price-setting transactions is given by,

$$\begin{aligned} r_{k+1} &= c_{k+1} b_{k+2} + \ldots + c_{n-1} b_n \\ r_{k+2} &= c_{k+1} b_{k+1} + \ldots + c_{n-1} b_n \\ &\vdots \\ r_n &= c_{k+1} b_{k+1} + \ldots + c_{n-1} b_{n-1} \end{aligned}$$

Now, consider the following miner deviation:

$$\hat{b}_{k+1} = b_k, \hat{b}_{k+2} = \ldots = \hat{b}_n = 0$$

The total VCG payments obtained is $k\hat{b}_{k+1} = kb_k$ and $r_i = 0, \forall i \leq k+1$. Hence the top $k+1$ transactions receive zero rebate. Now the miner has impersonated as $k+2, \ldots, n$ price setting agents. Hence the miner receives back the rebate of $r_{k+2} = \ldots = r_n = c_{k+1}\hat{b}_{k+1} = c_{k+1}b_k$. Therefore the effective redistribution to the agents is $\sum_{i=1}^{k} r_i = 0 \implies \hat{e}_{\mathsf{wc}} = 0$. □

From these results, we establish that preventing agent manipulation entirely is not possible in the TFRM framework. Therefore, we focus on ensuring Restricted DSIC (RDSIC), which ensures that agents of transactions included in the block will not misreport their values. Further, we know from Theorem 9.2 that even with RDSIC, any known RM that satisfies all the desirable properties can be easily manipulated by the miner. The theorem also shows that the manipulation will lead to strictly zero rebate. Thus, in the next section, we propose Robust TFRM (R-TFRM), which relaxes $\text{IR}_{\mathbf{M}}$, to ensure a positive rebate even with miner manipulation.

9.4 R-TFRM: A TFRM Robust to Miner Manipulation

To ensure strictly positive RRI, we compromise on $\text{IR}_{\mathbf{M}}$, i.e., the utility of an honest miner may be negative. However, we ensure that when the miner is strategic, it can always guarantee itself a non-zero utility. We denote such a TFRM that is resilient to miner manipulation by Robust TFRM (R-TFRM). Designing R-TFRM involves constructing an appropriate rebate function. We focus on a linear rebate function that maximizes the worst-case redistribution index RRI while ensuring IR_u. We still want RDSIC; hence, we use the rebate function as given in Theorem 2.7.

9.4.1 IR_u Constraints

Each included agent must have a non-negative utility, i.e., $u_i \geq 0, \forall i \in I$. W.l.o.g., we assume that $b_1 \geq b_2 \geq \ldots b_n$ and IR for agent n is ensured when $r_n \geq 0$ as $u_n = r_n$. In Claim 9.1, we show that $r_i \geq 0, \forall i$ if $r_n \geq 0$; hence R-TFRM will be IR_u.

Maximize: f ▷RRI

Subject to: For every $\mathbf{b} \geq 0$,

$$r_n \geq 0 \quad \triangleright \text{IR}_u$$

$$\sum_{i=1}^{n} r_i \leq k \cdot b_k \quad \triangleright \text{Approx-IR}_m$$

$$\sum_{i=1}^{k} r_i \geq f \cdot k \cdot b_{k+1} \quad \triangleright \text{Worst-case Fraction}$$

where $r_i = c_0 + c_1 b_1 + \ldots + c_{i-1} b_{i-1} + c_i b_{i+1} + \ldots + c_{n-1} b_n$

Figure 9.3: R-TFRM: Linear Program for Rebate Function with Approx-IR$_m$

Claim 9.1. R-TFRM *with n included transactions and rebates $(r_1, \ldots, r_n)$ is agent IR (IR$_u$) if $r_n \geq 0$.*

Proof. Suppose there exists a bid vector $\mathbf{b} = (b_1, \ldots, b_n)$ and some agent $i < n$ such that $r_i < 0$. r_i is computed using $\mathbf{b}_{-i}$, i.e, $r_i = c_0 + c_1 b_1 + \ldots + c_{i-1} b_{i-1} + c_i b_{i+1} + \ldots + c_{n-1} b_n$. Now, consider a new bid vector $\mathbf{b}'$ s.t., $b'_1 = b_2, b'_2 = b_3, \ldots, b'_n = 0$. Observe that $\mathbf{b}'_{-n} = \mathbf{b}_{-i}$, hence $r_i(\mathbf{b}_{-i}, \cdot) = r_n(\mathbf{b}'_{-n}, \cdot) < 0$. Thus, TFRM will be IR$_u$ if for any bid vector $r_n \geq 0$. □

9.4.2 Approx-IR$_\mathbf{M}$ Constraints

In any classic RM, to ensure IR$_\mathbf{M}$, there is an additional constraint to ensure that the total rebate is less than the VCG payments, i.e., $\sum_i r_i \leq k \cdot b_{k+1}$. We modify this constraint to ensure $\sum_i r_i \leq k \cdot b_k$. This change is because a strategic miner can manipulate the VCG auction to insert a fake bid $\hat{b}_{k+1} = b_k$. Figure 9.3 describes the linear program to solve for such a rebate while maximizing the RRI fraction f.

Maximize: f

Subject To: $\sum_{j=k}^{i} c_j \geq f,\ \forall i \in \{k, \ldots, n-1\}$

$$(n-k) \cdot c_k \leq k \textbf{ and } n \sum_{j=k}^{n-1} c_j \leq k$$

$$n \sum_{j=k}^{k+i-1} c_j + (n-k-i) \cdot c_{k+i} \leq k,\ \forall i \in [n-k-1]$$

Figure 9.4: R-TFRM: Linear Program Independent of the bid vector $\mathbf{b}$

We now aim to write the linear program in Figure 9.3 such that it is only dependent on $n, k, c_i's$ and independent of the bid vector, $\mathbf{b}$. For this purpose, we now state the following claims.

Claim 9.2. *If $c_0, \ldots, c_{n-1}$ satisfy IR$_u$ and Approx-IR$_\mathbf{M}$, then $c_i = 0$ for $i = 0, \ldots, k-1$.*

Proof. First, we show that $c_0 = 0$. Consider a bid vector $\hat{b}_i = 0$ for all i, then for IR$_u$ we must have $c_0 \geq 0$ (from Claim 9.1). To satisfy Approx-IR$_\mathbf{M}$, we must have $c_0 \leq 0$ hence $c_0 = 0$. If $c_i = 0, \forall i$, we are done hence consider $j = \min\{i | c_i \neq 0\}$. Let $j < k$, then from IR$_u$ constraint we have that $r_n = c_0 + c_1\hat{b}_1 + \ldots + c_{n-1}\hat{b}_{n-1} \geq 0$. Consider a bid vector s.t. $\hat{b}_i = 1$ for $i \leq j$ and $\hat{b}_i = 0$ for the rest. For this, $r_n = c_j$ therefore $c_j \geq 0$. While the Approx-IR$_\mathbf{M}$ constraint states that $r_1 + \ldots + r_n \leq kb_k$ for the above bid vector $\hat{b}$, we obtain $r_i = 0$ for $i \leq j$ as c_j is multiplied with a bid that is j^{th} highest hence 0. Whereas $r_i = c_j$ for $i > j$, because the j^{th} highest bid is 1. Therefore, the constraint can be rewritten as $c_j(n-j) \leq k\hat{b}_k$. Because $j < k$, $\hat{b}_k = 0$ so the right-hand side is 0. Also $n - j > 0$ because $j < k < n$, therefore $c_j \leq 0$. But we already know from IR$_u$ that $c_j \geq 0$, hence $c_j = 0$ for $j < k$. Hence by contradiction, $j \geq k$, i.e., $c_j = 0$ for $j < k$. □

Claim 9.3. *The IR_u constraint $r_n \geq 0$ and the worst-case fraction constraint (refer Figure 9.3) is equivalent to having $\sum_{j=k}^{i} c_j \geq f,\ \forall i \in \{k, \ldots, n-1\}$.*

Proof. We consider the following lemma to prove the claim.

Lemma 9.1. *[132, Lemma 1] Given $n \in \mathbb{N}$ and set of real constants $s_1, \ldots, s_n$, $(s_1 t_1 + \ldots + s_n t_n \geq 0$ for any $t_1 \geq \ldots \geq t_n \geq 0)$ iff $\sum_{i=0}^{j} s_i \geq 0$ for $j = 1, 2 \ldots, n$.*

The proof of the above lemma is given in [132]. From Claim 9.2, we know that $c_i = 0$ for $i < k$. Hence, we can re-write the worst-case constraint as follows,

$$\sum_{i=1}^{k} r_i \geq f \cdot k b_{k+1}$$

$$k \cdot (c_k b_{k+1} + \ldots + c_{n-1} b_n) \geq f \cdot k b_{k+1}$$

$$(c_k - f) b_{k+1} + \ldots + c_{n-1} b_n \geq 0$$

Given that $b_{k+1} \leq b_{k+2} + \ldots + b_n \geq 0$, we can invoke the above Lemma and thus, we obtain the following condition: $c_k - f + c_{k+1} + \ldots + c_i \geq 0,\ \forall i = k, \ldots, n-1$. Thus, the worst-case fraction constraint can be written as $\sum_{j=k}^{i} c_j \geq f, i \in \{k, \ldots, n-1\}$. □

Claim 9.4. *The Approx-$IR_{\mathbf{M}}$ constraint can be replaced by:*

$$(n-k)c_k \leq k \ \textbf{\textit{and}}\ n \cdot \sum_{j=k}^{n-1} c_j \leq k \ \textbf{\textit{and}}$$

$$n \sum_{j=k}^{k+i-1} c_j + (n-k-i)c_{k+i} \leq k,\ \ i \in \{1, \ldots, n-k-1\}$$

Proof. We prove this using [132, Lemma 1] on Approx-$IR_{\mathbf{M}}$ constraints. The Approx-$IR_{\mathbf{M}}$ constraint requires that $r_1 + \ldots + r_n \leq k b_k$ where $r_i = c_0 + c_1 b_1 + \ldots + c_i b_{i+1} + \ldots + c_{n-1} b_n$. Since $c_i = 0, i < k$, the constraint is further simplified as follows,

$$q_k b_k + q_{k+1} b_{k+1} + \ldots + q_n b_n \geq 0$$

where, $q_k = k - (n-k)c_k$ and $q_i = -(i-1)c_{i-1} - (n-i)c_i$, for $i = \{k+1, \ldots, n-1\}$ and $q_n = -(n-1)c_{n-1}$. By [131, Lemma 1], the Approx-IR$_{\mathbf{M}}$ constraint is equivalent to having $\sum_{i=k}^{j} q_i \geq 0$ for all $j = \{k, \ldots, n\}$. This can be simplified to obtain the following:

$$
\begin{aligned}
q_k \geq 0 &\iff (n-k)c_k \leq k \\
q_k + \ldots + q_{m+i} \geq 0 &\iff n \sum_{j=k}^{k+i-1} c_j + (n-k-i)c_{k+i} \leq k, \\
& \qquad i \in \{1, \ldots, n-k-1\} \\
q_k + \ldots + q_n \geq 0 &\iff n \sum_{j=k}^{n-1} c_j \leq k
\end{aligned}
$$

This proves the claim. □

Using Claims 9.3 and 9.4, we reformulate the linear program in Figure 9.3 so that it is independent of the bid vectors. Figure 9.4 presents this reformulated LP.

9.4.3 Optimal worst-case Redistribution Fraction

We next provide the analytical solution to the linear program in Figure 9.4 and thereby also state the optimal worst-case fraction redistributed.

Theorem 9.3. *For any n and k such that $n \geq k+2$, the* R-TFRM *mechanism is unique. The fraction redistributed to the top-k agents in the worst case is given by: $f^* = \frac{k}{n}$. In* R-TFRM*, the rebate function is characterized by the following: $c_k^* = \frac{k}{n}$ and $c_i^* = 0, \ \forall i \neq k$.*

Proof. We first show that $c_k^* = \frac{k}{n}$, $c_i^* = 0, \forall i \neq k$ is a feasible solution to the LP given in Figure 9.4.

Note that, $\sum_{j=k}^{i} c_k^* = \frac{k}{n} \geq f^* = \frac{k}{n}, i \in \{k, \ldots, n-1\}$. Further $(n-k)c_k^* = (n-k)\frac{k}{n} < k$ as $n - k < n$. Observe that the third constraint $n \sum_{j=k}^{k+i-1} c_j^* + (n-k-i)c_{k+i}^* = nc_k^* =$

$n\frac{k}{n} \leq k, i \in \{1, \ldots, n-k-1\}$ is also satisfied. Finally, $nc_k^* \leq k$ hence, c^* and f^* is a feasible solution to the LP.

Now we show that if there exists any solution $\hat{c}, \hat{f}$ that satisfies the constraints of the LP, then $\hat{c} = c^*$ and $\hat{f} = f^*$. First, let $x_j = \sum_{i=k}^{j} c_i$ for $j = k, \ldots, n-1$. The LP constraints can be re-written as the following,

$$\begin{aligned} x_j &\geq f,\ j = \{k, \ldots, n-1\} \\ (n-k)x_k &\leq k \\ (k+i)x_{k+i-1} + (n-k-i)x_{k+i} &\leq k\ i = \{1, \ldots, n-k-1\} \\ nx_{n-1} &\leq k \end{aligned}$$

We know that $x_k^* = f^*$, from first constraint we have $\hat{x}_k \geq \hat{f} \geq f^* = x_k^*$, i.e., $\hat{x}_j \geq x_j^*$ for $j = \{k, \ldots, n-1\}$. From second constraint we have that, $x_k^* \leq$ From the third constraint, we obtain, for i = 1, $(k+1)x_k^* + (n-k-1)x_{k+1}^* = k$ and for any $\hat{x}$, $(n-k-1)\hat{x}_{k+1} \leq k - (k+1)\hat{x}_k \leq k - (k+1)x_k^* = (n-k-1)x_{k+1}^*$, i.e., $\hat{x}_{k+1} \leq x_{k+1}^*$. Similarly we can obtain $\hat{x}_k \leq x_k^*$ further using $i = 2, \ldots, n-k-1$ and the fourth constraint we obtain $\hat{x_j} \leq x_j^*$, for $j = k+2, \ldots, n-1$. From the first and fourth constraint, we can conclude that $\hat{x}_j = x_j^*$ for $j = k, \ldots, n-1$. This completes the proof of the theorem. □

Observe that the total redistribution to the agents, when the miner is honest for R-TFRM, is given by

$$\sum_{i=1}^{n} r_i = \frac{k}{n}\left[(k \cdot b_{k+1}) + (n-k)b_k\right] \tag{9.2}$$

This value may exceed $k \cdot b_{k+1}$, thus violating $\text{IR}_{\mathbf{M}}$. However, it satisfies Approx-$\text{IR}_{\mathbf{M}}$ as $\sum_{i=1}^{n} r_i \leq k \cdot b_k$ (refer Figure 9.3). R-TFRM is similar to the Bailey-Cavallo mechanism [49]. The primary difference is due to the Approx-$\text{IR}_{\mathbf{M}}$ constraint, which makes $c_k = k/n$ instead of c_{k+1} as in Bailey-Cavallo.

9.4.4 R-TFRM: Analyzing Impact of Miner Manipulation on Rebate and Miner Revenue

With an honest miner, R-TFRM maximizes the worst-case redistribution index such that it is AE, RDSIC, IR_u, and Approx-$\text{IR}_{\mathbf{M}}$. We now analyze the effect of miner manipulation on R-TFRM. Previously, we saw that it is impossible to ensure non-zero RRI (Theorem 9.2), but with R-TFRM we show that RRI is strictly positive even with miner manipulation.

9.4.4.1 Reduction in Transaction Fees

The rebate function for R-TFRM is characterized by the constants given in Theorem 9.3. With these, We now calculate RRI (Definition 9.4), i.e, $\hat{e}_{\mathsf{wc}}$, for R-TFRM. Theorem 9.4 shows that irrespective of miner manipulation, $c_k^* = \frac{k}{n}$ fraction of payments will be returned.

Theorem 9.4. *Consider n included transactions with the set C as confirmed transactions such that $|C| = k$ with the remaining $n - k$ as price-setting transactions. Irrespective of any miner manipulation,* R-TFRM *ensures strictly positive RRI or $\hat{e}_{\mathsf{wc}} = c_k^* = \frac{k}{n}$.*

Proof. The VCG payments to the miner and rebates both depend on the unconfirmed transaction b_{k+1}. Manipulating b_{k+1} would change both the payments and refund in a way

that the fraction of redistribution remains constant. From Definition 9.4, we know that,

$$\begin{aligned}
\hat{e}_{\mathsf{wc}} &= inf_{\hat{\mathbf{b}}} \frac{\sum_{i \in S} r_i}{p(\hat{\mathbf{b}})} \\
&= inf_{\hat{\mathbf{b}}} \frac{k(c_k \hat{b}_{k+1})}{k\hat{b}_{k+1}} \\
&\quad \Big(\text{As } S = \{1, \ldots, k\} \text{ and } p(\hat{\mathbf{b}}) \text{ is VCG}\Big) \\
&\quad (r_i \text{ given by Theorem 9.3}) \\
&= c_k = \frac{k}{n}
\end{aligned}$$

This proves the theorem. □

From Theorem 9.3 and Theorem 9.4, we see that in R-TFRM, the fraction of payments redistributed to the top-k agents, i.e., k/n, is the same for honest and strategic miner. This implies that R-TFRM is resilient to miner manipulation while being worst-case optimal.

9.4.4.2 Utility of Strategic Miner

From Theorem 9.4, we know that if a miner is strategic and impersonates the price-setting transactions, the miner will receive positive utility. The miner will preferably set the fake bid $\hat{b}_{k+1}$ close to b_k. Hence, the maximum utility to a miner that deviates by impersonating the price-setting bids is: $u_{\mathbf{M}} = (1 - k/n) \cdot k \cdot b_k$. As we assume $n \geq k + 2$, the miner's maximum utility is minimized for $k = n - 2$.

The fraction redistributed to the genuine agents is still $\frac{k}{n}$ of the payments received even when the miner impersonates the confirmed transactions. We illustrate this with an example next.

Example 9.2: R-TFRM: Utility of Strategic Miner

It is possible for the miner to insert fake transactions with high enough bids such that they are confirmed. Consider $n = 5$ and $k = 3$ where $b_1 = b_2 = 100$, $b_3 = 10$ and $b_4 = b_5 = 4$. If the miner is only impersonating the price setting transactions, then it puts $\hat{b}_4 = b_3$ and arbitrary $\hat{b_5} < b_3$, then its overall utility is $\left(1 - \frac{k}{n}\right) kb_k = 12$. Whereas if the miner is given more flexibility to insert a fake transaction within the confirmed and unconfirmed bids, it receives more payments. For e.g., let $\hat{b}_1 = 200$ and $\hat{b}_3 = 100$ hence the ordered bids are $\hat{b}_1 \geq b_1 \geq b_2 \geq \hat{b}_3$. Therefore, effectively, the first two transactions are confirmed and pay 100 each. Further, due to R-TFRM, it returns a rebate of $\frac{k}{n}100$ to each of the two agents, thus obtaining an overall utility of $(1 - 3/5)\, 200 = 80$.

9.5 R^2-TFRM: Robust and Rational TFRM

R-TFRM compromises Miner IR to ensure positive RRI. We now introduce randomness in R-TFRM to obtain a mechanism that ensures positive utility to an honest miner, i.e., satisfies IR_m. Towards this, we propose Robust and Rational TFRM, R^2-TFRM (Figure 9.5). In R^2-TFRM, the rebate is not guaranteed for every included transaction. Instead, an included transaction gets a rebate with probability α, $\alpha \in [0, 1]$ where the rebate value is calculated using R-TFRM. Hence R^2-TFRM reduces to R-TFRM when $\alpha = 1$. On the other extreme, when $\alpha = 0$, R^2-TFRM reduces to a second price auction.

9.5.1 R^2-TFRM: On-chain Randomness

As stated, each transaction receives a rebate with a probability α. Similar to other TFMs [62], we employ trusted on-chain randomness for this randomization. Researchers have proposed such trusted randomized protocols using various cryptographic primitives [31, 81]. Significantly, the miner of the block cannot exert any influence on this randomization.

1. *Inclusion Rule* ($\mathbf{x}^I$). Select highest n transactions from the mempool, M. W.l.o.g., assume that these n transactions are ordered as $b_1 \geq b_2 \geq \ldots \geq b_n \implies x_i^I = 1,\ \forall i \in \{1, \ldots, n\}$.

2. *Confirmation Rule* ($\mathbf{x}^C$). Select highest k bids from the n included, $x_i^C = 1, \forall i \in [k]$, where $k \leq n - 2$.

3. *Payment Rule* ($\mathbf{p}$). Each confirmed agent i (i.e., $i \in C$) pays

$$p_i = \begin{cases} b_{k+1} - r_i & \text{w.p. } \alpha \\ b_{k+1} & \text{w.p. } (1-\alpha) \end{cases}$$

   Each included but not confirmed agent j (i.e., $j \in I \setminus C$) pays

$$p_j = \begin{cases} -r_j & \text{w.p. } \alpha \\ 0 & \text{w.p. } (1-\alpha) \end{cases}$$

   The rebate r_i is given by Theorem 9.3.

4. *Miner Revenue Rule.* The miner receives the net revenue of $\sum_{i=1}^{n} p_i$.

Figure 9.5: R^2-TFRM: Robust and Rational Transaction Fees Redistribution Mechanism.

9.5.2 R^2-TFRM: Incentive and RRI Guarantees

Theorem 9.5 proves that R^2-TFRM mimics the incentive guarantees of R-TFRM. Moreover, for an appropriate α, R^2-TFRM is also $\text{IR}_{\mathbf{M}}$ in expectation.

Theorem 9.5. *For any n and k such that $n \geq k + 2$ and any bid profile $\mathbf{b} = (b_1, \ldots, b_n)$, and probability $\alpha \in (0, 1)$* R^2-TFRM *has an expected redistribution fraction (expectation over α) $f^* = \alpha \cdot \frac{k}{n}$. Further it satisfies AE, RDSIC, IR_u, and is IR_m when $\alpha \leq \overline{\alpha} = \frac{n}{k+(n-k)b_k/b_{k+1}}$.*

Proof. R^2-TFRM is randomized R-TFRM, where randomness is introduced due to the α parameter.

R^2-TFRM is AE as the k highest bids are allotted the slots according to the confirmation rule (Figure 9.5).

R^2-TFRM is IR_u as utility is non-negative for each included transaction. For the confirmed transaction $i \in \{1, \ldots, k\}$, utility is

$$u_i = \begin{cases} \theta_i - \left(\frac{n-k}{n}\right) b_{k+1} \geq b_i - \left(\frac{n-k}{n}\right) b_{k+1} \geq 0 \quad \text{w.p.} \;\; \alpha \\ \theta_i - b_{k+1} \geq b_i - b_{k+1} \geq 0 \quad \text{w.p.} \;\; (1-\alpha) \end{cases}$$

as $\theta_i \geq b_i$ and $b_i \geq b_{k+1}$. Similary for the included but non-confirmed transaction, $j \in \{k+1, \ldots, n\}$, the utility is $u_j = \frac{k}{n} b_k \geq 0$ w.p. α or $u_j = 0$.

R^2-TFRM is $RDSIC$-in-expectation. For any confirmed transaction $i \in \{1, \ldots, k\}$ expected utility is $u_i = \theta_i - \left(\frac{n-\alpha k}{n}\right) b_{k+1}$. The agent may submit a dishonest bid b_i*' s.t., i) $b'_i > \theta_i$ or ii) $b'_i < \theta_i$.

i) When $b'_i > \theta_i$, the bid is still confirmed and the expected utility is unchanged.

ii) When $b'_i < \theta_i$, s.t. $b'_i > b_{k+1}$, the transaction remains confirmed and the expected utility is unchanged. Thus, consider $b'_i < b_{k+1}$, then the transaction is not confirmed and now $b_k = b_{k+1}$ therefore the expected utility to i is $\alpha \frac{k}{n} b_{k+1} \leq u_i$. Hence no confirmed agent gains in expected upon underbidding.

For any transaction i.e., included but not confirmed, $j \in \{k+1, \ldots, n\}$, the expected utility is $u_i = \alpha \frac{k}{n} b_k$ and we have the same two cases as above,

i) When $b'_j > \theta_j$, s.t., $b'_j < b_k$ the transaction remains unconfirmed, and the expected utility is unchanged. Thus, consider $b'_j \geq b_k$, the transaction gets confirmed and now, $b_{k+1} = b_k \geq \theta_j$. Thus the expected utility of j is given by $\theta_j - b_k + \alpha \frac{k}{n} b_k \leq u_j$. Hence the agent does not gain in expectation upon overbidding.

ii) When $b'_j < \theta$, the agent remains non-confirmed, and the maximum expected utility remains unchanged.

R^2-TFRM satisfies AE, IR_u and $RDSIC$-in-expectation.

R^2-TFRM is $IR_{\mathbf{M}}$-in-expectation. The payment obtained by an honest miner is given by kb_{k+1}. The expected rebate paid by the honest miner is given by, $\alpha\left[k\frac{k}{n}b_{k+1} + (n-k)\frac{k}{n}b_k\right]$. If $\alpha = 1$, the refund is equal to the refund in R-TFRM given by Equation 9.2. In order to ensure $IR_{\mathbf{M}}$ for the honest miner, the following must be true,

$$
\begin{aligned}
kb_{k+1} &\geq \alpha\left[k\frac{k}{n}b_{k+1} + (n-k)\frac{k}{n}b_k\right] \\
b_{k+1} &\geq \alpha\left[\frac{k}{n}b_{k+1} + \frac{(n-k)}{n}b_k\right] \\
\alpha &\leq \frac{nb_{k+1}}{kb_{k+1} + (n-k)b_k}
\end{aligned}
$$

Therefore R^2-TFRM satisfies $IR_{\mathbf{M}}$ when $\alpha(b) = \frac{nb_{k+1}}{kb_{k+1}+(n-k)b_k}$ and total rebate is kb_{k+1}. □

We make the following observations based on Theorem 9.5.

$b_{k+1} \to b_k \implies$ **R-TFRM** $\iff$ **R^2-TFRM.** From Theorem 9.5, an honest miner obtains non-negative utility when,

$$\alpha \leq \frac{n}{k + (n-k)b_f} \tag{9.3}$$

where, $b_f = \frac{b_k}{b_{k+1}}$ is the bid ratio. W.l.o.g as the bids are ordered (Figure 9.5) $b_f \geq 1$. When $b_f = 1$, i.e., $b_k = b_{k+1}$ we have $\alpha = 1$ and R-TFRM $\iff$ R^2-TFRM. This implies that every agent receives a rebate, and the miner IR is not violated.

This may seem to contradict R-TFRM not being IR$_{\mathbf{M}}$; however, we can think of $b_{k+1} \to b_k$ as one of the deviations of the strategic miner. And R-TFRM is IR$_{\mathbf{M}}$ when the miner is strategic. Furthermore, as b_f increases, the upper bound on α becomes smaller. To still guarantee IR$_{\mathbf{M}}$, the overall rebate (i.e., $\alpha \cdot \frac{k}{n}$) decreases.

9.5.2.1 R^2-TFRM: Analyzing Miner Manipulation

Like R-TFRM, R^2-TFRM also ensures strictly positive RRI even with miner manipulation as formally stated in Theorem 9.6.

Theorem 9.6. *Consider n included transactions with the set C as confirmed transactions such that $|C| = k$ with the remaining $n-k$ as price-setting transactions. Irrespective of any miner manipulation* R^2-TFRM *ensures: expected RRI or $\mathbf{E}_\alpha[\hat{e}_{\mathsf{wc}}] = \alpha \cdot \frac{k}{n}$.*

Proof. Similar to Theorem 9.4, the fraction of redistribution remains constant. For every true agent (not fake), the $\alpha k/n$ fraction of the payment is returned back as the rebate in expectation. □

We observe that irrespective of the miner manipulation; the agents receive back the fraction $\alpha \cdot \frac{k}{n}$ of the payment made on expectation. Based on what fake transactions the miner inserts, the payments change, but the refund fraction remains the same as for the case when the miner is honest.

9.6 Conclusion

In this chapter, we argued the importance of minimizing agent costs in a TFM. Our key idea is to employ a redistribution mechanism-based approach for determining the transaction fees, which we call the Transaction Fee Redistribution Mechanism (TFRM). Due to strategic miner manipulation, we first show that guaranteeing a strictly positive rebate in a TFRM and other desirable properties is impossible. Hence, we propose R-TFRM, which ensures strictly positive rebates even in the worst case but compromises the miner's IR. However, we show that in R-TFRM, a strategic miner will never incur negative utility while still guaranteeing strictly positive rebates to the agents. We also propose R^2-TFRM which uses blockchain's inherent randomness to guarantee the agents a positive rebate while respecting the miner's IR.

Future Work. Future directions can explore TFRMs with randomized rebate functions, which may likely satisfy stronger notions of IC and IR. Another approach may be to explore non-linear rebate functions, which may provide a better redistribution index on average. In addition, unlike this work, the heterogeneous RM setting, i.e., transactions with varying sizes, can also be explored.

PART B: Security and Privacy

PART B *is the privacy-focused section of this thesis. This part explores the privacy of the agents in three distinct yet interconnected multi-agent systems (MAS): (i) privacy-preserving voting application over blockchain, (ii) privacy-preserving combinatorial auction over blockchain, and (iii) differentially private distributed constraint optimization (DCOP). First, we have FASTEN [73], where we look at the design and implementation of a voting application leveraging blockchain technology while prioritizing voter's (and their vote's) privacy. We utilize the distributed trust provided by a blockchain with smart contract support to present a voting mechanism that is scalable even for general elections. Second, we introduce STOUP [70], which focuses on privacy concerns within the realm of combinatorial auctions conducted over a blockchain infrastructure. We present cryptographic protocols and smart contract mechanisms to enable secure and private bidding. Emphasis is placed on protecting bid details, participant identities, and auction outcomes. Last, we propose P-Gibbs [80], a differentially private Distributed Constraint Optimization (DCOP) algorithm that preserves the privacy of the agent's constraints. This involves incorporating differential privacy principles into the DCOP framework, ensuring that the optimization process does not compromise the privacy of individual agents. In summary, the overarching theme of* PART B *is to contribute to the evolving landscape*

of privacy-preserving MAS. By examining voting applications, combinatorial auctions, and DCOPs, the goal is to advance the understanding and implementation of privacy-enhancing mechanisms in diverse scenarios, fostering a more secure and confidential digital environment.

Chapter 10

FASTEN: Fair and Private Voting

Electing democratic representatives via voting has been common since the 17th century. However, these mechanisms raise concerns about fairness, privacy, vote concealment, fair calculations of tally, and proxies voting on behalf of the voters. Ballot voting, and in recent times, electronic voting via electronic voting machines (EVMs), improves fairness by relying on centralized trust. Homomorphic encryption-based voting protocols also assure fairness but cannot scale to large-scale elections such as presidential elections. In this chapter, we leverage the blockchain technology of distributing trust to propose a smart contract-based protocol, namely, FASTEN*. There are many existing protocols for voting using smart contracts. We observe that these either are not scalable or leak the vote tally during the voting stage, i.e., do not provide vote concealment. In contrast, we show that* FASTEN *preserves voters' privacy, ensures vote concealment and immutability, and avoids double voting. We prove that the probability of privacy breaches is negligibly small. Further, our cost analysis of executing* FASTEN *over Ethereum is comparable to most of the existing costs of elections.*

⋆ ⋆ ⋆ ⋆ ⋆

10.1 Introduction

As mentioned earlier in Chapter 1.5.3.1, elections are fundamental to democratic governance. This chapter focuses on *fair elections* that allow every eligible individual to participate in the decision-making process by registering their vote. We argue that a fair election is possible only when the voter can freely vote for its desired preference.

One must also ensure an agent's participation in the voting process is hidden. This can be achieved by eliminating the *link* between the voter and its vote, i.e., *anonymous* voting. We first define the following essential properties to design a fair election with anonymous voting, i.e., fair and secure election (FSE).

1. **Voter Anonymity (VA).** A vote cannot be traced back to the voter during or after the election.

2. **Vote Concealment (VC).** The vote's value should remain hidden from the system (voters, candidate, election commission). This, in turn, ensures that the vote tally remains a mystery to the system until the voting window has expired.

3. **Vote Immutable (VI).** Once a voter casts its vote, altering it to anyone else's vote should be impossible.

4. **Double Voting Inhibition (DVI).** A voter should be allowed to vote only once in a specific election.

Fair and Secure Election (FSE) Overview. Towards FSE, the most traditional voting method is *paper ballots*. It partially ensures anonymity, vote concealment, and vote immutability. The major drawback of ballot-based voting is that it involves tiresome manual work in counting the votes. Along with the risk of unintentional and intentional human error, the non-durability of paper and the lack of a robust mechanism to avoid double voting are some of the other challenges involved with this system.

Election through *electronic voting machines* (EVM)s is a technological upgrade over the paper ballot system. EVMs provide voter anonymity and do not take voter ID as a parameter. But, they fail to guarantee vote immutability. This is because the voter needs to trust the EVM software company for vote concealment and immutability. They also entrust the company with shipping the EVMs with the correct version of the firmware, and thus, the EVM remains a black box to the voter. Besides, the double voting inhibition problem is still there.

Micali et al. [195] propose a protocol for secure auctions that applies to voting. However, the body conducting the elections, *election commission* (EC), will know all the votes after the voting stage. The authors propose an expensive zero-knowledge proof of the result to overcome the "centralized trust" placed on EC. Consequently, the protocol is also not scalable. Adida [5] proposes a most popular scheme *Helios*, which relies on the security of one server and is not viable for nationwide elections. Thus, there is a need to look for a completely different approach to conduct an FSE.

Fair and Secure Election (FSE) over Blockchain. With Ethereum[1] [42], we observe that blockchain can not only be used to solve the problem of designing a cryptocurrency but can also be used to implement *smart contracts* [42]. A smart contract allows blockchain to establish an interactive platform for n parties. Such a contract enforces the outcome of any event through a set of rules. These rules correspond to a programming language that is understandable to the execution system. The key concept here is distributing trust rather than relying on a single party. Thus, we explore ways to leverage such a distributed trust to conduct an FSE.

The important steps in the voting procedure for FSE are: (i) *voter authentication*, i.e., a person claiming to be a voter should be an eligible voter; (ii) *vote registration*, which preserves the privacy of the voter as well as its vote; (iii) *outcome verification* that counts the tally of votes in a verifiable manner. Zhao et al. [309] propose a voting protocol based

[1]While we focus on Ethereum, we remark that our solution works for any blockchain with smart contract support.

on *Commit-Publish* mechanisms that also leverages smart contract. As the authors' main goal is boardroom voting, it does not address step (i). It solves steps (ii) and (iii). To the best of our knowledge, in a plethora of voting schemes over blockchain, except [307], no protocol satisfies all the four desirable properties of a fair election. The challenge with [307] is scalability as it is Zcash-based, which is *considerably* slower than Bitcoin.

10.1.1 Chapter Contributions

We propose a novel protocol for FSE, namely, FASTEN: *FAir and Secure disTributEd votiNg*. We partially rely on EC for authentication, which issues a random but *unique* token for each voter after authenticating the voter. The token is unique to the voter and the particular election. If the voter tries to obtain multiple tokens for the same election – it will receive the same token. Therefore, FASTEN resists *Sybil* attacks. This authentication is similar to several secure applications over blockchain (e.g., [179, 87]). In this work, we assume that EC does not store the link between a voter and its token[2]. Next, the smart contract considers tokens as eligibility to vote, which the EC grants. After this, each voter registers its encrypted vote[3] and token to the smart contract. The smart contract[4] holds the hashes of all the tokens that EC issues. It computes the hash of the token registered by the voter and checks if it is in the database. Once the entry is confirmed, the encrypted votes are registered in the voter database. After the vote-casting window expires, the smart contract decrypts all the votes and computes the tally.

Since our protocol deploys smart contracts based on blockchain, one may implement it as a *Decentralized Application* (DApp). DApps comprise a friendly UI for any smart

[2]Such trusted third-party authentications also have a close parallel with the ZCash Parameter Generation [223]. These links thus correspond to "toxic waste" – to be destroyed.

[3]The encryption/decryption keys are generated by semi-trusted third parties, namely, wardens, as defined later in Chapter 10.2.1.

[4]In this chapter, we present FASTEN as an Ethereum-based smart contract. The protocol can also be easily conducted through computational logic over other distributed platforms like *Hyperledger Fabric* [10] or *Quorum* [20].

contract, thereby allowing laypersons to interact with said contracts. Thus, FASTEN helps improve fairness in elections, i.e., designing FSE and improving voter participation.

With FASTEN, we show that a straightforward protocol will achieve an FSE. We believe that the simple design of our protocol is desirable and will benefit the end-user. This simplicity is in contrast with sophisticated protocols that deploy heavy cryptographic and security primitives to achieve FSE. These protocols, especially how they reach the desired privacy, are challenging to explain to a layperson. These limit their use in general elections where participating voters are in millions. We outline some of them next.

Contributions. We propose a protocol, namely FASTEN to conduct a fair election. We leverage blockchain technology as the foundation for our protocol. While our protocol is not the first to leverage blockchains to develop a voting protocol, it is the first protocol using blockchains that proposes a feasible model for a large-scale election such as nationwide elections. The protocol enables all the voters and candidates to verify the voting process and simultaneously preserves the votes' privacy. In short, FASTEN satisfies all the four desirable characteristics required for an election to be fair. We prove that the probability of breach of any of the four characteristics is negligibly small (Claims 10.1, 10.2, 10.3,10.4). Our cost analysis shows that the cost of running an election through FASTEN is very competitive w.r.t. existing costs of conducting elections.

10.1.2 Additional Background

Here, we provide the related literature for FSE and chapter notations.

10.1.2.1 Related Work

There have been few attempts before to design a voting protocol based on blockchains, but they all use *Commit-Publish* mechanisms to conceal the vote tally. A *Commit-Publish* mechanism requires voters to submit a deposit before committing to a vote and are refunded the deposit when they publish their vote. Both the Open Vote Network Protocol

Paper	VA	VC	VI	DVI	Scalable
[195] (No Blockchain)	✓	×	✓	✓	✗
[5] (No Blockchain)	✓	✓	✓	✓	✗
[189, 271, 309]	✓	✓	✓	✓	✗
[27, 110, 136, 304]	✓	×	✓	✓	✓
[307]	✓	✓	✓	✓	✗
FASTEN (Our Protocol)	✓	✓	✓	✓	✓

Table 10.1: Comparison of Different Secure Voting Protocols

implemented by McCorry et al. [189] and the Bitcoin-based protocol by Zhao et al. [309] use this mechanism. While this mechanism could be used for boardroom voting, it cannot be used for mass elections like the general presidential election of nations because most voters would be reluctant to submit the deposit. This is because, in a population of millions, voters will not consider their vote to be of much importance to begin with. In addition, once the voters commit, they have to participate in the protocol again to reveal their commitment. Note that several nations are struggling to bring voters to voting booths to cast their vote when they just have to come once. Thus, we believe that it is unrealistic to believe that a voter will choose to participate twice in the same election. With FASTEN, we require the voters to come on-chain, i.e., online on the blockchain, only once to register their votes and without any deposit.

Other works, e.g., FollowMyVote [110], use trusted authorities to ensure voter anonymity. The voters cast their votes in plain text, therefore risking the concealment of the vote tally.

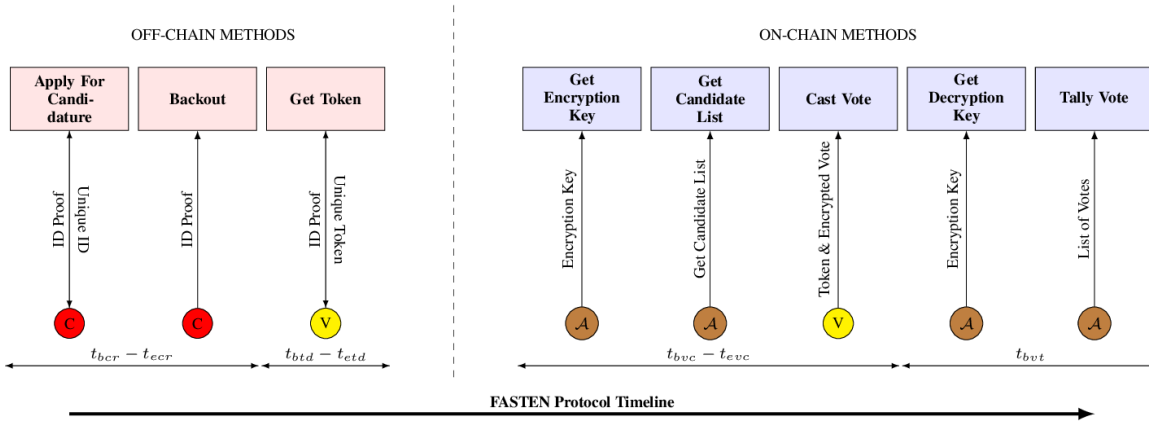


Figure 10.1: Illustration of the protocol timeline in FASTEN. Here, C: Candidate, V: Voter, and $\mathcal{A}$: Other agent

In contrast, voters in FASTEN must submit encrypted votes, "cipher-texts", thereby concealing their votes. Our encryption-decryption mechanism ensures the concealment of the vote tally until the vote-casting window expires. Due to space constraints, we present several other works through Table 10.1 to place FASTEN with respect to the existing literature in FSEs. We also refer the interested reader to [285, Table 1] for a more comprehensive comparison of secure voting protocols over the blockchain. We next present the preliminaries required for the design and analysis of FASTEN.

10.1.2.2 Model and Chapter Notations

The main stakeholders of any election are the voters, the candidates, and the Election Commission (EC). EC represents the governing body of the election. Once an election is announced, interested candidates must register their candidature with EC before a certain deadline, which we denote as t_{ecr}. It is the EC's responsibility to ensure fair candidature registration. For the time period between the beginning and end of token distribution, i.e., t_{btd} and t_{etd}, EC issues tokens to the voters after authenticating them. Actual vote casting begins at time t_{bvc} and ends at t_{evc}. In FASTEN, the voters submit their encrypted votes

over the Ethereum-based smart contract. We show how EC (or any other interested party) can get the vote tally using the smart contract. We refer to the procedural methods of the overall protocol, which are on the smart contract as *on-chain methods* and the remaining procedures as *off-chain methods*. Figure 10.1 illustrates the temporal aspect of FASTEN.

We use an Ethereum-based smart contract for on-chain methods, which relies on blockchain technology. We refer the reader to Chapter 3.2 for an overview of blockchains and the Ethereum network.

10.1.2.3 Threat Model

In this chapter, we assume that all parties are *honest-but-curious* and *passive*. They follow the protocol's steps honestly but try to infer private information from the available data. E.g., looking at the (encrypted) votes available on the blockchain, an adversary can try to extract total votes for a candidate. The adversary can also use the on-chain data to link a voter to a vote.

10.2 FASTEN: Our Approach

As a smart contract resides on the blockchain, we inherit the decentralized feature of the blockchain with it in our protocol. In designing a FSE, verifying the eligibility of a voter before allowing it to vote is mandatory. This verification process requires examining the voter's real-world identity and examining the same on-chain risks to the privacy/anonymity of the voter. This is because it links the real-world voter identity with the encrypted vote. Therefore, there is a need for separating voter ID from encrypted vote. For the same purpose, we take help from the respective EC to distribute tokens that separate the voter's real ID from the vote. The tokens are randomly distributed off-chain (off the blockchain, which is not part of the contract) to voters after verifying their identity and eligibility. Since time is important in an election, the voting protocol strictly follows a timeline of

events to avoid liabilities. Every contract method has a time bound out of which the contract *reverts* the call.

Role of Election Commission (EC). To ensure that only eligible candidate gets to vote, we take help from the EC to verify voter eligibility. Every data on the smart contract is on blockchain and thus is publicly available. This implies that the voter database cannot be made public to preserve voter anonymity. Also, verifying voter ID on blockchain and registering votes through that ID violates voter anonymity. To resolve these, we let EC verify voter ID off-chain and distribute random tokens to the voters as a grant to vote. Note that every voter must be given only one token. The smart contract assumes that EC distributes the token truthfully, securely, and privately without keeping any trace of voter ID with the token. The smart contract holds a pre-stored database of hashes of the token issued by the EC. Since the token is distributed by verifying the voter eligibility, the smart contract needs only check if the token is genuine and not fake. It does so by checking if the hash of the token exists in the token hash database.

Also, the voters in FASTEN are not bothered with any commitments through deposits as in previously proposed voting mechanisms based on blockchain. In contrast, they are simply required to get the encryption key and ID, encrypt the vote, and send it through the smart contract along with their token. The overhead of submitting the decryption keys relies on a different set of agents. We refer to such a third party as a *Warden*. These agents store the decryption key safely instead of the voters, the process of which is explained later. Each warden holds a key that corresponds to a batch of votes. These wardens represent a distributed trust system and are incentivized to act honestly in the system. We also assume the candidate registration to be handled by EC and the candidate list to be provided by them in advance.

Role of Smart Contracts. EVM(s), which are currently used for conducting election systems, are black boxes to the voter because the software used inside EVM cannot be inspected by them. This raises concerns regarding EVM code tempering. Unlike the code

in EVM, a smart contract's code resides on a public distributed ledger, i.e., blockchain. This enables anyone to inspect the methods deployed by a smart contract. Also, since smart contract resides on the blockchain, every data/transaction that is part of the contract also becomes a part of the blockchain, which ensures the permanence of the history of the vote done by the voter. Thus, it results in the *verifiability* of the vote counting.

After time t_{evc}, when the decryption key is released to the public, it can be used to decrypt the encrypted votes stored safely on the blockchain and compare with the result achieved by the smart contract. Since the encryption and corresponding decryption keys have to be different by the protocol's design, we use asymmetric cryptography. Any asymmetric cryptographic system will work for our protocol, such as ElGamal cryptography system [95]. With these as a backdrop, we formally present our FSE protocol, FASTEN.

10.2.1 FASTEN: Protocol Design

In FASTEN, we use Ethereum-based smart contracts to enforce the protocol. As stated in Chapter 3.2.2.3, smart contract transactions in Ethereum consume gas, a form of commission paid to the Ethereum network. As such, we require the EC to reimburse the cost associated with Ethereum transactions to all parties involved in FASTEN. Later in Chapter 10.2.3, we present a rough cost analysis of FASTEN

The contract maintains strict adherence to the time constraints. Therefore, each method in the contract is bound by a time window outside of which the contract reverts the call made to it. Table 10.2 defines and explains these time constraints. FASTEN also uses predefined variables and databases fed into the smart contract beforehand. We describe them as follows.

FASTEN: Variables. Table 10.3 provides the variables in FASTEN[5]. From Table 10.3, *hashDatabase* is a mapping from hash string to boolean such that $True$ value means the hash is present in the database. It is pre-populated beforehand. Further, *voteBatch* is a

[5]For older versions of Solidity, i.e., $< 0.7.0$, *block.timestamp* corresponds to *msg.now*.

Notation	Definition
t_{bcr}	time-stamp for beginning the candidature registration (Candidates can begin registration after this time)
t_{ecr}	time-stamp for ending the candidature registration (All candidates must register before this time)
t_{btd}	time-stamp for beginning the token distribution (Voters can collect their tokens after this time)
t_{etd}	time-stamp for ending the token distribution (All voters must collect their token before this time)
t_{bvc}	time-stamp for beginning the vote casting (Voters can begin casting their vote after this time)
t_{evc}	time-stamp for ending the vote casting (All voters must cast their vote before this time)
t_{bvt}	time-stamp for beginning the vote tally (Anyone can ask for a vote tally only after this time)

Table 10.2: Time Constraints in FASTEN

double array where $voteBatch[i]$ represents the array of encryption votes with encryption id i. The list of encrypted votes gets aggregated in the course of the election. These variables are part of the underlying methods deployed in FASTEN. We now describe these methods next.

Variables	Definition
block.timestamp	Current time-stamp
candList	List of the registered candidates
numKeys	Total number of the encryption keys available
idCounter	To allocate the encryption-decryption key pair
enKeys	List of all encryption keys provided
deKeys	List of decryption keys aggregated by the wardens
hashDatabase	Database of the hashes of all the tokens
voteBatch	Database to store votes
securityAmt	Security amount wardens deposit before registration
refundAmt	Array to store amount to be refunded to the wardens
reward	Additional reward to the wardens
wardens	Mapping from warden address to key id
sampleText	Sample file to authenticate key pair(s)
tallyDone	Flag to ensure votes are only counted once

Table 10.3: Variables in FASTEN

10.2.1.1 FASTEN: Underlying Methods

In FASTEN, we consider a *hybrid* model for improved scalability and efficiency, i.e., some of the underlying methods are on-chain while others are off-chain. We describe our protocol by first presenting the off-chain and on-chain methods. Later, we describe the control flow of the protocol using these methods.

10.2.1.1.1 Off-Chain Methods These methods are not part of the smart contract, i.e., are performed locally by the concerned party. These methods are aimed at setting up the election process by letting the candidates register (ApplyForCandidature), allowing a candidate to withdraw from the election (Backout), and allowing *all* parties (e.g., voters and candidates) to receive a token (GetToken) used by the smart contract for identification purposes.

1. ApplyForCandidature. This method allows eligible candidates to register themselves as the candidate for election. For the implementation, we suggest using a biometric scanner for ID authentication. The method generates a unique voting ID for the candidate. It can only be used during (t_{bcr}, t_{ecr}). For an offline, non-blockchain election, one can think of this process as akin to candidates filing their nominations with the Election Commission (EC).

2. Backout. Registered candidates can use this to *backout* from the election. The candidates must first authenticate their identity using the biometric scanner. The method can also only be used during (t_{bcr}, t_{ecr}).

3. GetToken. This method is not part of the smart contract but a facility provided outside the contract to get the token privately. This method will distribute a unique token that the voter can use to cast a vote. The token will only be distributed after a successful ID authentication through biometrics. It can be used only in the duration (t_{btd}, t_{etd}). For an offline, non-blockchain election, one can think of this process as akin to getting a voter ID card.

10.2.1.1.2 On-Chain Methods While the off-chain methods essentially help set up the election/voting process, the on-chain methods comprise those that occur during the vote-casting and vote-tallying phase and are conducted using the smart contract. We classify them under two subsets: "General Public" methods and "Warder-specific" methods.

This categorization is based on who can invoke these methods. We now formally present each of these methods using pseudo-codes.

General Public methods. These methods are concerned with allowing the general public access to the candidate list, querying the public key encryption keys to encrypt their vote and casting their vote. After the vote-casting phase is over, the public can also get the decryption keys so that anyone can compute the election result.

- GetCandidateList. This method returns the list of all the candidates participating in the election. This is simply a *getter* method, i.e., it has no cost. E.g., a function that is set to "view" in Solidity. The method requires no authentication and can be invoked during (t_{ecr}, t_{bvc}).

- GetEncryptionKey: This method shares the encryption key with the caller who invokes it. It also requires no authentication. The caller is provided with an encryption ID and the corresponding encryption key. It can be invoked only in the duration (t_{bvc}, t_{evc}). We present the pseudo-code in Method 1.

Method 1 GetEncryptionKey

1: **require**$(block.timestamp \in (t_{bvc}, t_{evc}))$
2: Int $i \leftarrow idCounter + 1$
3: $idCounter = (idCounter + 1)\%numKeys$
4: bytes32 $ek \leftarrow enKeys[i]$
5: **return** $[i, ek]$

- CastVote: Presented in Method 2, it allows the eligible voters to register their vote. Required parameters are token[6], encryption ID, and encrypted vote. The token is

[6]This is the only time the voter is required to send a token to the smart contract.

validated, after which the encrypted vote is stored corresponding to its encrypted ID. This method can be used only in the duration (t_{bvc}, t_{evc}).

Method 2 CastVote

1: **Input:** Token t, *Int* i, *bytes32* ev
2: **require**$(block.timestamp \in (t_{bvc}, t_{evc}))$
3: *bytes32* $h \leftarrow$ sha3(t)
4: **require**$(hashDatabase[h] == true)$
5: $hashDatabase[h] = false$
6: $voteBatch[i].push(ev)$

- GetDecryptionKeys. Similar to Method 1, this method shares the list of decryption keys with the caller who invokes it. The method requires no authentication *but* can only be invoked after t_{bvt}.

- TallyVote: This decrypts the votes, counts them, and returns the result (Method 3). It requires no authentication. Once the votes have been decrypted and counted, the pre-computed result is returned in the subsequent calls. This method can only invoke t_{bvt}.

Method 3 VoteTally

1: **require**$(block.timestamp > t_{bvt})$
2: **if** $tallyDone == false$ **then**
3: **for** $Int\ i = 0; i < numKeys; i++$ **do**
4: **for** $bytes32\ ev : voteBatch[i]$ **do**
5: $bytes32\ dk = deKeys[i]$
6: $bytes32\ dv \leftarrow decrypt(ev)$
7: $bytes32\ candId = extract(dv)\ candTally[candId] += 1$
8: **end for**
9: **end for**$tallyDone = true$
10: **end if**
11: **return** $candTally$

Warden-specific Methods. We remark that the General-public methods essentially mimic a classic, offline election process. These methods allow voters to vote (post authentication), see the candidates nominated, and query encryption/decryption keys for FASTEN to guarantee certain privacy and security. The Wardens perform the warden-specific methods. The methods' aim is to disincentivize warden no-shows (DepositSecurity), allow them to submit their encryption and decryption keys (SubmitEncryptionKey, SubmitDecryptionKey), and also collect rewards for their participation (WithdrawReward). We present these methods formally next.

- DepositSecurity. The wardens invoke this method to deposit monetary value as security against their honest behavior and to confirm their registration. The method is presented in Method 4.

Method 4 DepositSecurity

1: **require**$(wardens[msg.sender] > 0)$

2: **require**$(block.timestamp < t_{bvc})$

3: **require**$(msg.value > securityAmt)$

4: $refundAmt[msg.sender] = msg.value - securityAmt$

- SubmitEncryptionKey. As described in Method 5, this method allows the wardens to submit the encryption key in the required duration.

Method 5 SubmitEncryptionKey

1: **Input:** *bytes32 ek*

2: $uint\ id = wardens[msg.sender]$

3: **require**$(id > 0)$

4: **require**$(block.timestamp < t_{bvc})$

5: **require**$(refundAmt[msg.sender] > 0)$

6: $enKeys[id] = ek$

- SubmitDecryptionKey. This method is used by the wardens to submit the correct decryption key in a timely. We present the method in Method 6.

Method 6 SubmitDecryptionKey

1: **Input:** *bytes32 dk*
2: $uint\ id = wardens[msg.sender]$
3: **require**$(wardens[msg.sender] > 0)$
4: **require**$(block.timestamp \in (t_{evc}, t_{bvt}))$
5: **require**$(refundAmt[id] > 0)$
6: $bytes32\ ek = enKeys[id]$
7: **require**$(sampleText == decrypt(encrypt(sampleText, ek), dk))$
$deKeys[id] = dk$
8: refundAmt[msg.sender] += securityAmt + reward

- WithdrawReward: From Method 7, wardens can use this method to collect their reward after the successful submission of the decryption key.

Method 7 WithdrawReward

1: **require**$(wardens[msg.sender] > 0)$
2: **require**$(block.timestamp > t_{bvt})$
3: $uint\ amt = refundAmt[msg.sender]$
4: $refundAmt[msg.sender] = 0$
5: **if** amt >0 **then**
6: $msg.sender.transfer(amt)$
7: **end if**

As aforementioned, in FASTEN, we use tokens distributed by EC to register a voter's vote. We now explain how we use those tokens to guarantee secure voting.

10.2.1.2 FASTEN: Token Validation Process

The tokens are distributed through an off-chain system. Tokens get pre-generated and stored securely and privately with EC. We assume that tokens will be distributed privately and securely by the respective EC. The voter provides the token only once while sending its encrypted vote. The smart contract keeps the hash[7] of all the tokens in its on-chain database. It calculates the hash of the token provided by the voter and checks if the hash of the token exists in the database. If found, it removes the entry of the token from the database and registers the voter's vote.

Algorithm 15 Voting Procedure()

1: **procedure** VOTE CASTING WINDOW OPENS
2: token $t \leftarrow getToken()$
3: Candidate List $candList \leftarrow$ GetCandidateList()
4: Let $canList$ be the preferred candidate of Gretel from the list $candList$
5: Encryption ID, Encryption Key $i, ek_i \leftarrow$ GetEncryptionKey()
6: Encrypted Vote $v \leftarrow Decrypt()$, encryption done off-chain by the voter CastVote(t, v, i)
7: **end procedure**
8: **procedure** VOTE TALLYING WINDOW OPENS
9: Vote Count[] $vc \leftarrow$ TallyVote()
10: vc_i represents vote count of i^{th} candidate in the $canList$
11: Decryption Key List $dk_i \leftarrow$ GetDecryptionKeys(), where dk_i represents decryption key associated with the encryption id i
12: The list of decryption keys can be used to get the vote count manually and check its correctness with the tally calculated by the contract.
13: **end procedure**

[7] We suggest hash functions which are efficient over a smart contract such as *sha3* [42].

10.2.1.3 FASTEN: Voting Procedure

Now that we have provided the core protocol design, we give a sample walk-through of the protocol. A voter who wishes to use FASTEN can use it by following Algorithm 15. We will now describe how the decryption key is kept secret in FASTEN.

10.2.1.4 FASTEN: Warden Assistance

As a smart contract cannot store data privately, we take assistance from wardens outside the contract to submit the decryption keys on time. In FASTEN, wardens are appointed to keep the decryption keys off the chain. They provide the keys to the smart contract through a special transaction when the time comes, i.e., during (t_{evc}, t_{bvt}).

Consider a set of wardens W. Each warden $w_i \in W$ has a decryption key dk_i and is assigned a batch of votes which can be decrypted through that key. Trivially, $|W|$ is the total number of batches. Let B be the dataset of batches, i.e., where $b_i \in B$ is the batch corresponding to w_i. Observe that every time GetEncryptionKey() (Method 1) is called, the contract provides the voter with the encryption id i corresponding to w_i and an encryption key ek_i. The voter encrypts the vote using the encryption key and then sends the encrypted vote and "i" as parameters to CastVote() (Method 2). The contract then assigns the encrypted vote to b_i. As a result, after t_{evc}, if we have n votes in total, every batch b_i will hold roughly $\frac{n}{|W|}$ number of encrypted votes.

In FASTEN, we do not take for granted that the wardens will be honest. Towards this, we design our protocol to minimize the loss in case any warden turns out to be dishonest. Observe that any dishonest warden can cause trouble in two ways: aborting its duty and/or leaking the decryption key. We now provide our solution to these two problems.

10.2.1.4.1 Abortion of Duty Each warden w_i submits a deposit dk_i prior to its participation in FASTEN. The warden's address will be stored in the smart contract database. As a result, it will get its deposit refunded only if it submits the correct decryption key

on time. The smart contract will verify the decryption key by trying the encryption-decryption key pair on some r random strings. Once the decryption key is verified to be correct, the warden will get the deposit refunded. We also suggest providing an additional bonus amount as a reward for honest behavior.

10.2.1.4.2 Leaking Decryption Key We remark that it is practically impossible to prevent any warden w_i from leaking its decryption key dk_i. If a warden w_i leaks the decryption key, on average, it will leak the tally of only $\frac{n}{|W|}$ votes. Thus, we suggest choosing the number of wardens $|W|$ to minimize this loss to a very minute percentage. For example, having less than $\frac{1}{1000}^{\text{th}}$ fraction of voters with each warden will ensure that a single dishonest warden cannot leak a tally of more than 0.1% of votes. The number of wardens is a protocol parameter and can be increased depending on the budget and risk tolerance.

10.2.2 FASTEN: Proof of Fairness and Security

As aforementioned, an election is said to be fair if it satisfies the four properties: vote anonymity, vote concealment, vote immutability, and double voting inhibition. That is, every voter can vote discretely and anonymously, and the vote tally remains hidden until the end of the election. Further, nobody can cast a vote without authentication nor cast the vote twice. We now (i) provide the formal definitions and (ii) prove that FASTEN satisfies all these properties.

Definition 10.1: Voter Anonymity

An election protocol is said to satisfy the *Voter Anonymity* property if the probability of finding the vote v of any voter i is negligibly small as compared to the size of the population, i.e., $\Pr[v|i] \leq \mathsf{negl}(\lambda)$, with λ as the security parameter.

Claim 10.1. FASTEN *satisfies voter anonymity with the probability of guessing any voter's token being at most* $\frac{n}{2^l}$ *where* n *is the number of voters and tokens are* l*-bit long.*

Proof. The smart contract takes the token as an attribute from the voter to register their encrypted votes. We demand that voters use a new Ethereum address to cast their encrypted vote since the old Ethereum address might have been compromised by the voter. Since the token is randomly generated and privately distributed off the blockchain, it cannot be linked back to the voter through the blockchain because Ethereum addresses are randomly generated and have no relation with the user's identity. The public ledger will only show that a certain encrypted vote has been registered for a certain token. This implies that even after the vote is encrypted, it is only related to the token and has no links with the previous token holder. Thus, the vote can never be traced back to the voter except possibly random guessing. If the token is l-bit long, and the size of the voter population is n, the random guessing will not succeed by probability more than $\frac{n}{2^l}$. We can ensure voter anonymity by choosing $l >> \log_2 n$. □

Definition 10.2: Vote Concealment

An election protocol is said to satisfy the *Vote Concealment Property* if no vote's value is revealed before the end of the election vote casting period with probability more than $\frac{k}{|W|}$ if no more than $k << |W|$ wardens are dishonest.

Claim 10.2. FASTEN *satisfies vote concealment property.*

Proof. As we have highlighted earlier, the voters will register their encrypted vote to the contract. The contract will provide the voters with the encryption key and the corresponding encryption ID. The voter encrypts the vote through the encryption key of the blockchain on her device and sends it to the smart contract along with the encryption ID

and the respective token. Since the vote is encrypted off the chain, the real value of the vote remains hidden until the decryption key is out. The decryption is made public only after the end of the casting window, thereby concealing the vote until the voting period ends. Hence, the property of Vote Concealment is preserved.

More formally, no probabilistic polynomial-time (PPT) adversary will be able to know a vote's value by looking at the encryption unless it can solve the Discrete-log Problem (DLP) (Definition 3.5), which is computationally infeasible. That is, a PPT adversary can only solve the DLP with probability $\mathsf{negl}(\lambda)$, given λ as the security parameter. Another way for a vote to be revealed is directly by the warden (the warden reveals its secret key). If a warden $w_i \in W$ is dishonest, a voter's vote is leaked with probability $\frac{1}{|W|}$. Similarly, if k such wardens are dishonest, the probability of the vote leaking is $\frac{k}{|W|}$. By keeping $|W|$ large enough, i.e., $k << |W|$, this probability is $\mathsf{negl}(|W|)$ implying Vote Concealment is preserved. □

Definition 10.3: Vote Immutability

An election protocol is said to satisfy the *Vote Immutability* property if no vote's value can be altered once it has been cast.

Claim 10.3. FASTEN *satisfies vote immutability under the assumption that the majority of the nodes in the network are honest.*

Proof. Ethereum blockchain stores all the transactions permanently by default. Once a transaction is part of the public ledger, it cannot be removed or changed unless 51% or more nodes are corrupt. Also, once the hash of a token is matched to that of the database on the blockchain, it is immediately removed; thereby, no overwriting of the vote is possible. □

Definition 10.4: Double Voting Inhibition

An election protocol is said to satisfy the *Double Voting Inhibition* property if the probability that any voter can cast more than one vote in the election is negligible.

Claim 10.4. FASTEN *satisfies the double voting inhibition property.*

Proof. The smart contract has an on-chain database containing the pre-calculated hashes of the tokens distributed to the voters. The voters are required to send their tokens to register their vote. The smart contract calculates the hash of the received token and compares it with the ones stored in the database. If a match is found, the token hash is removed from the database, and the vote is registered. Since the token is immediately removed from the database before registering the encrypted vote, we can safely claim that every token is used only once to register the vote, thus inhibiting double voting. The only possibility for a voter to double vote would be to cast a vote with its valid token and guess a random token whose hash matches that one in the database. If the hash is of length h-bits, even by birthday-paradox, the voter needs to try $2^{\frac{h}{2}}$ trials to guess a first valid token apart from its own. Typically, hash sizes (h) are 256 bits. Hence, the probability of successful double-voting is negligible. □

Thus, we have proved theoretically that FASTEN achieves fairness in the election, i.e., FASTEN is a solution for FSE. Now we perform a cost analysis of FASTEN.

10.2.3 FASTEN: Protocol Analysis

10.2.3.1 Cost Analysis

We analyze the cost in terms of Ethereum gas, which is the constant cost of network resources/utilization. We use the gas estimation given in Ethereum docs, and the Ethereum

rate card also given in the docs to estimate cost per vote [42]. We estimate the cost per vote in 2 stages: (i) Voter side cost and (ii) Warder side cost.

10.2.3.1.1 Voter Side Cost In FASTEN, each voter has to follow a particular sequence of methods to cast a vote. From Procedure 15, the voter follows the given sequence: (i) GetCandidateList, which has its gas requirement of 26 units; (ii) GetEncryptionKey, which consumes 667 gas units; and (iii) CastVote, which consumes 739. In total, the vote casting consumes 1432 Gas units.

Further, TallyVote computes the vote count for each candidate when called for the first time and returns the result. After this, it returns the pre-computed result for subsequent calls. This optimization ensures that there are no redundant calculations. Therefore, every vote is decrypted and counted once. This ensures that TallyVote is equivalent to counting a single vote, which we estimate as 300000 gas units. The estimation is high as we encrypt votes using 160-bit ElGamal encryption. Thus, the decryption cost is itself high.

Fortunately, this can be easily optimized as well. We suggest decrypting the vote on DApp instead of the smart contract. As smart contracts are mainly used for maintaining persistent states throughout the network, and the decryption won't alter the state, the decryption can be done on the DApp. By avoiding the decryption cost, we get an upper bound of 1500 gas units per vote for the voter side.

10.2.3.1.2 Warden Side Cost In FASTEN, each warden uses the following methods in the order given: (i) DepositSecurity, which requires 23 gas units; (ii) SubmitEncryptionKey, which requires 629 gas units; (iii) SubmitDecryptionKey, with 600755 gas units; and (iv) WithdrawReward, which requires 21629. That is, each warden consumes 623036 gas units in total. The reason for such high cost is the encryption and decryption operations done in *SubmitDecryptionKey* method to check the authenticity of the decryption key. However, as mentioned during the analysis of the voter side's cost, one can avoid the cost of encryption and decryption by computing it on the DApp instead of a smart contract. This

also reduces the warden side cost to an upper bound of 23036 gas units. Suppose there exists a total of n voters; then every warden holds the decryption key for $\frac{n}{|W|}$ voters, on average. Therefore, the cost per vote from the warden's side is $\frac{23036 \cdot |W|}{n}$ gas units.

For a reasonable choice of $\frac{n}{|W|}$, such as 1000, this cost comes to be ≈ 23 gas units. Adding this to the voter side cost calculated above, we can set the upper bound for the total cost per vote of in FASTEN in terms of Ethereum gas units as 1600 (when $\frac{n}{|W|} = 1000$). Further note that, 1600 gas units corresponds to 0.000064 ETH as 1 gas unit (typically) equals $4 \cdot 10^8$ ETH [97]. As at the time of writing of this thesis, we have 1 ETH ≈ 2396 USD, the overall cost comes out to be ≈ 1.53 USD per vote.

10.2.4 Time Analysis

As shown, the total gas consumption in FASTEN is 1600 per vote (when $\frac{n}{|W|} = 1000$). When writing the thesis, Ethereum's block gas limit is, on average, 8×10^6 [98]. Thus, each Ethereum block can hold 5000 votes. Further, when writing this thesis, each block takes $\approx 15s$ to get on the Ethereum Network [16]. This corresponds to a processing capacity of roughly 20000 votes per hour[8].

Inference from Analysis. A rough estimate of the cost per vote is 1.53 USD (when $\frac{n}{|W|} = 1000$). This is lower than the amount of money spent in countries worldwide for mass elections. For Example, the per-vote election cost for the UK European Parliament Election 2014 is 5.54 USD [280].

10.3 Conclusion and Discussion

We showed that FASTEN is a solution for FSE through blockchain and smart contracts. It also provides a transparent, decentralized system through which the stakeholders can

[8]We remark that this estimate can be significantly improved by deploying a more scalable underlying blockchain consensus protocol such as [85, 281, 12].

verify the result. We also argued that our protocol is cost-efficient compared to the existing election methods. In summary,

1. FASTEN does not reveal voter identity at any stage of the process.
2. In FASTEN, votes cannot be revealed at any time before the vote-casting window expires. Even the vote count is concealed from everyone until the window expires. Thus, FASTEN ensures that voting is not influenced at any stage.
3. FASTEN allows anyone to check the end-to-end voting procedure independently. This is because all the relevant information is on a public ledger. As proved, this is achieved without compromising voters' privacy, as every vote is cryptographically secure and cannot be traced back to the voter.
4. As we do not assume nor use any constraints on voting population, FASTEN can be used for a large population.
5. Our cost analysis shows that FASTEN is cost-efficient as compared to existing protocols.
6. Unlike prior works, in FASTEN voters do not have to commit and return to the protocol to reveal their votes.

10.3.1 Discussion

Oracles: An alternative to Warden Assistance. As smart contracts cannot access and fetch data outside the blockchain, in FASTEN, we rely on wardens to provide the decryption keys. Another way to achieve this is through *oracles*. An oracle is a third-party service designed for smart contracts and can feed data from outside the blockchain to it as and when required. However, relying on such third-party oracles can compromise the distributed trust model of the underlying blockchain. As a result, in FASTEN, we leverage wardens.

To use oracles, we need a way to ensure the data provided is genuine and not tempered. Oraclize [214] is one such service provider that claims to be *provably honest*, i.e., which provides unaltered data to smart contracts. They do so by accompanying the returned

data together with a document - referred as an "authenticity proof" - which can be requested using the *oraclize_setProof* function provided by the service. The authenticity proofs build upon different technologies such as auditable virtual machines and trusted execution environments (refer [215]).

Liquid Democracy. The use of transferring voting rights to an informed voter has been referred to as *liquid democracy* [149]. Kahng et al. [149] showed the existence of certain delegation mechanisms that can outperform traditional voting in terms of selecting better candidates. We believe such mechanisms can be easily implemented through FASTEN. It must be transacted between the voter (willing to transfer voting rights) and the delegate to transfer its token. We leave a security analysis of this for future work.

Chapter 11

STOUP: Secure and Trustworthy Combinatorial Auction

In multi-agent systems (MAS), i.e., when multiple agents interact with the system, we must aim to preserve the privacy of participants' information in such applications. Towards this, Yao's Millionaires' problem (YMP), i.e., to determine the richer among two millionaires privately, finds relevance. This chapter presents a novel, practical, and verifiable solution to YMP, namely, Secure Comparison Protocol (SCP). We show that SCP achieves this comparison in a constant number of rounds without using encryption and without requiring the participants' continuous involvement. SCP uses semi-trusted third parties- privacy accountants- for the comparison, who do not learn any information about the values. That is, the probability of information leak is negligible in the problem size. We also leverage the Ethereum network in SCP for pseudo-anonymous communication, unlike computationally expensive secure channels such as Tor. We present a Secure, Truthful Combinatorial Auction Protocol (STOUP) for single-minded bidders to demonstrate SCP's significance. We show that STOUP, unlike previous works, preserves the privacies relevant to an auction, even from the

auctioneer. We demonstrate the practicality of STOUP through simulations.

⋆ ⋆ ⋆ ⋆ ⋆

11.1 Introduction

Multi-agent systems (MAS) such as *distributed constraint optimization*, *e-commerce*, and *e-voting* mechanisms continue to grow in popularity. Consequently, the need for privacy of the information exchange within these platforms has become imperative and is an area of active research [126, 212, 274, 275, 276]. The participants (e.g., bidders), being *strategic agents*, prefer the preservation of their *private* information (e.g., bids) as well as (often) their public identities from other competitive agents.

With *blockchain* gaining momentum, MAS is now being conducted through computational logic over distributed platforms such as the *Ethereum* blockchain network [42]. Specifically, Ethereum allows for *smart contracts*, which are computer protocols intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract (refer to Chapter 3.2.2.3 for details on Ethereum and smart contracts). Since these are on a publicly distributed ledger, they are open to any interested agent while making sensitive information (e.g., bids) and executing payments publicly verifiable, transparent, and pseudo-anonymous. Consequently, an agent's private information is publicly available for anyone to see and use. This further necessitates securing (privacy-preserving) MAS over a blockchain.

11.1.1 Chapter Contributions

Yao's Millionaires' Problem (YMP). At the heart of several MAS mentioned above is comparing two numeric values. Therefore, to build a protocol that preserves each agent's private information, we require a method for comparing these values while preserving their

privacy. In the literature, this challenge is referred to as *Yao's Millionaires' Problem* (YMP) [300] of securely determining the richer between two different agents and has been extensively studied.

YMP is as follows: Two agents (millionaires), *Alice* and *Bob*, are interested in determining the richer among them – without revealing their actual wealth. Motivated by this, in this chapter, we introduce a novel method for comparing two integers $x, y \in \mathbb{Z}$ securely, i.e., a practical solution to YMP designed for secure multi-agent applications. We refer to our method as *Secure Comparison Protocol* (PPC). In PPC, we assume that there are approved *cryptographic accountants* in the system, which assist the *central server* (CS) in determining whether $x \geq y$ or not. We show that the probability that CS (or any other party) learns any information regarding x or y, in PPC, is negligible. Further, we show that the integer comparison in PPC is *verifiable* by leveraging *zero-knowledge proof* (ZKP) techniques.

To securely deploy these AI applications over blockchain, we present PPC over the Ethereum network[1]. To the best of our knowledge, we are the first to introduce a dedicated and verifiable solution, in constant rounds, for YMP over the blockchain. We demonstrate the significance of PPC by presenting a Secure, Truthful Combinatorial Auction Protocol l (STOUP) for single-minded bidders.

STOUP. More concretely, in STOUP, we create a privacy-preserving auction atop the greedy single-minded combinatorial auction presented in Chapter 2.2.5.6.3, Algorithm 1. With STOUP, we show how to sort and compare the bidding information of agents without revealing them, with the help of accountants. The accountants do not learn of any *bidding information*, i.e., bid values and the items that are bid for. Towards this, we assume that each agent's bundle size is ≥ 2. Otherwise, the items in a bidder's bundle may be revealed to the auctioneer in our protocol. Note that in STOUP, the accountant's role is only to

[1]PPC may also be deployed using computationally expensive secure channels like Tor [84]. We ensure efficient pseudo-anonymous communication by coupling blockchain with an asymmetric encryption scheme.

assist the auctioneer in determining winners and their payments when the bid values and items are hidden.

Threat Model. This chapter assumes that CS is *semi-honest* or *honest-but-curious*. This implies that while CS can observe and cipher any information, it *will not* deviate from the defined protocol. However, unlike much of the previous works, we assume that all *other* agents, i.e., Alice, Bob, accountants, bidders, etc., are *strategic-but-curious*. These agents do not deviate from the protocol but may potentially submit/send incorrect information to gain an advantage or increase their utility. Additionally, like [126, 274], we also assume that agents do not collude.

11.1.2 Chapter Notations and Additional Background

As mentioned above, we build our privacy-preserving auction over Algorithm 1. As such, this chapter follows the notations introduced in Chapter 2.2.5.6.3. We also employ encryptions and commitment schemes (Chapter 3.1.1.2) and zero-knowledge proofs (ZKPs) (Chapter 3.1.2) for the verification of the auction outcome. Table 11.1 tabulates the notations used in this chapter.

11.1.2.1 Additional Preliminaries

We leverage the following known cryptographic techniques to construct our solution for YMP (Chapter 3.1.1.2). In particular, let p and q denote large primes such that q divides $p-1$, with G_q as the unique subgroup of $\mathbb{Z}_p^*$ of order q, and g as a generator of G_q.

Pedersen Commitment [229]. Let g and $h = g^a (\text{mod } p)$ be elements of G_q such that $log_g h$ is intractable, where $a \in \mathbb{Z}_q$ is the secret key. Then, a *Pedersen commitment scheme* is the commitment of a message $x \in \mathbb{Z}_q$, with a random help value $r \in \mathbb{Z}_q$, as, $C(x,r) = g^x h^r \ (\text{mod } p)$. We denote a_i as a party i's secret key, with $h_i = g^{a_i} (\text{mod } p)$.

Random Number Representation [195]. A *random number representation* of a number x, $R(x)$, is a representation of x as the pair (u,v) where $u, v \in \mathbb{Z}_q$ and $x = (u+v) \text{mod} q$,

Notations	Definitions
	Combinatorial Auction
AU	Auctioneer
$\mathcal{A} := \{1, \ldots, n\}$	Set of Agents
$\mathcal{M} := \{1, \ldots, m\}$	Set of Items Auctioned
$\mathbf{p}$	Payment Rule
K	Allocation Rule
$S_i \subseteq \mathcal{M}$	Subset of items bid (item bundle) by Agent i
$\vartheta_i(S)$	Valuation of Agent i for bundle S
W	Set of Winning Bidders
	Cryptographic Model
p, q	Large primes s.t. q divides $p-1$
G_q	Unique subgroup of $\mathbb{Z}_p^\star$ of order q
$g \in G_q$	Generator of G_q
$a \in \mathbb{Z}_p$	Secret Key
$h := g^a \pmod p$	Public Key
$C(x, r)$	Pedersen Commitment for $x \in \mathbb{Z}_p$ and randomness $r \in \mathbb{Z}_p$
$R(x)$	Random number representation of x
$C(R(x))$	Commitment pair for $R(x)$
n^1, n^2	Pair of Accountants assigned
$H(\cdot)$	Collision-resistant Hash function

Table 11.1: Chapter Notations

Note that, to find $R(x)$ of a number x, any party can randomly choose u and then pick $v = (x - u) \bmod q$.

Value Comparison [195]. For two integers $x, y < q/2$,

$$x - y \leq q/2 \iff x \geq y \text{ and } x - y > q/2 \iff x < y \tag{11.1}$$

Therefore, to compare x and y, we only need to check whether $x - y \leq q/2$.

Other Notations. We utilize the following notations for PPC as well as throughout the chapter.

- $C(R(x))$ represents the Pedersen commitment of x as $R(x) = (u, v)$, i.e., $C(R(x))$ denotes the pair of commitments $\big(C(u, r), C(v, r')\big)$.
- $A \xrightarrow{x} B$ denotes a party A submitting a value x to a smart contract, such that x is encrypted using B's public key.
- $H(\cdot)$ denotes a *collision-resistant* hash function.
- $E_A(x)$ represents the ElGamal encryption [95] of x using party A's private key.

11.1.2.2 Related Work

Yao's Millionaires' Problem (YMP). Yao [300] introduces YMP and its first solution. However, the presented solution is exponential in time and space. Several protocols improve over the seminal solution [26, 56, 124]. Ioannidis et al. [143] present a two-round protocol which is polynomial while the authors in [32, 174] provide a single-round solution which is linear in the order of the length of the integers to be compared. For their solutions, Ioannidis et al. [143] uses complex *bitwise operators* while [32, 174] use *Paillier homomorphic encryption* and zero-knowledge proof. The computational cost per comparison in Blake et al. [32] is $(4b + 1)(\log p) + 6b$ and in Lin et al. [174] is $5b(\log p) + 4b - 6$, where b is the bit number and p modulus of the Paillier scheme. Recently, Liu et al. [175] proposes

Paper	Homomorphic Encryption	Garbled Circuit	Continuous User Involvement	Dependent on Input size
[67, 108, 121, 147, 175]	✓	✗	✓	✓
[4]	✗	✗	✗	✗
[199]	✗	✓	✗	✓
[26, 56]	✗	✓	✓	✓
[44]	✓	✗	✗	✓
[301]	✓	✗	✗	✓
PPC	✗	✗	✗	✗

Table 11.2: Comparing existing YMP protocols with PPC. The "green cross-mark" is desirable.

a single-round solution using Paillier encryption and *vectorization* method. However, the solution is of the order $2(s+2)\log p$, where s is the vector dimension.

We place PPC with some of the plethoras of protocols available for YMP using Table 11.2. To the best of our knowledge, the existing protocols comprise one or more of (i) garbled circuits, (ii) partial-homomorphic encryption, and (iii) continuous involvement of the parties during execution. Note that [4] is limited as it only applies to integers in the $\approx 2^{60}$, whereas PPC works for any range. Consequently, these protocols *can not* be adapted towards designing lightweight and secure AI applications. Aggravating this limitation is that the number of comparisons needed for such AI applications is significant – in the order of polynomials or more. One can not assign trusted third parties for the operations as it may reveal the owners' private information.

Combinatorial Auctions. A combinatorial auction, where the parties can bid for a combination(s) of items, yields a higher revenue (lesser costs) than selling (buying) the items individually. E.g., wireless spectrum auctions [190] or allocating airport landing take-off slots [237]. Combinatorial auctions have an exponential number of possible valuations

for each party and are NP-Complete [244]. Hence, we focus on a *single-minded* case. The parties are interested in a specific bundle of items and obtain a particular value if they get the whole bundle (or any super-set) and zero otherwise. Unfortunately, even single-minded combinatorial auctions, being NP-Hard [248], are solved approximately. In particular, [169] proposes a polynomial-time algorithm for single-minded combinatorial auction, which gives $\sqrt{m}$-approximate winner and payment determination payment rule, which we refer to as ICA-SM (Incentive Compatible Approximate auctions for Single-minded bidders). Here, m denotes the number of items being auctioned.

Secure Auctions. Micali and Rabin [195] solve single-item and multi-unit auctions while preserving the privacy of the bids using Pedersen commitment but reveal the bid information to the auctioneer after the end of the bidding phase. Similarly, [204, 225] present single and multi-unit auctions that reveal the bid-topology to third parties. The authors in [36] give a practical, multi-unit auction that does not reveal any private information to a third party, even after the auction closes. Parkes et al. [226] use *clock-proxy* auction to solve a privacy-preserving combinatorial auction, revealing private information to the auctioneer after the end of the clock phase. The protocol is linear in the size of the original computational time, from exponential. Suzuki and Yokoo [267] propose a privacy-preserving, secure combinatorial auction without revealing any bid information to a third party. The authors use dynamic programming, and [164] extends the work to add verifiability. The protocol, however, is exponential in the size of the number of bids. The protocol is thus impractical even for a small number of bids.

We leverage PPC to present a secure combinatorial auction protocol, namely STOUP. To the best of our knowledge, this chapter is the first work to present an efficient, secure combinatorial auction protocol that preserves the bidding information's privacy, even from the auctioneer.

11.2 Secure Comparison Protocol (PPC)

We now describe PPC which securely compares two integers x and y owned by two agents, Alice and Bob. Towards this, let $||x||$ denote the number of bits required to represent the integer x. Now, in PPC, we assume that a *central server* (CS) coordinates the comparison. Note that, as shown later, the CS only aids the comparison and only learns additional information about the values of x and y with negligible probability, $\mathsf{negl}(\lambda)$, where λ is the security parameter.

We assume that $x, y < \frac{q}{2 \cdot d_{\max}}$, where $d_{max} \in (1, 2^{(||q||-1)})$. In PPC, we require Alice and Bob to privately select an *integer* $d_{Alice}, d_{Bob} \in (1, d_{\max}]$, respectively. Let, $D = (d_{\mathsf{Alice}} \oplus d_{\mathsf{Bob}})$. Observe that, we have $||D|| < ||\frac{q}{2}||$. For readability, we also denote $||d_{\max}||$ as $\mathbf{d}$.

Before describing PPC, we present the following claim. For this, let $R(x) = (u_1, v_1)$, $R(y) = (u_2, v_2)$, $val_1 = (u_1 - u_2) \bmod q$, and $val_2 = (v_1 - v_2) \bmod q$ with $x, y < \frac{q}{2 \cdot d_{max}}$.

Claim 11.1. *(i)* $D \cdot (val_1 + val_2) \bmod q \leq q/2 \iff x \geq y$; *and (ii)* $D \cdot (val_1 + val_2) \bmod q > q/2 \iff x < y$.

Proof. The claim is a simple rearrangement of the result presented in Eq. 11.1. □

11.2.1 PPC: Protocol

For PPC, we consider a smart contract $\mathtt{SC}$, which allows agents to post and get relevant information. Figure 11.1 presents the procedure for PPC, while Figure 11.2 presents the procedure for ZKP of PPC. Note that, for the ZKP, CS acts as $\mathcal{P}$. Trivially, PPC is independent of the length of the binary representation of x or y and hence is of constant order ($O(1)$) of computation rounds. We illustrate the protocol timeline of PPC with Figure 11.3.

PPC Procedure.

Let, $(n^1_{Alice}, n^2_{Alice})$ and (n^1_{Bob}, n^2_{Bob}) be Alice and Bob's pair of *distinct* assigned accountants, respectively.

1. Alice generates $R(x) = (u_1, v_1)$ and broadcasts $C(R(x))$ while Bob generates $R(y) = (u_2, v_2)$ and broadcasts $C(R(y))$; through SC.

2. $$\text{Alice} \xrightarrow{u_1, r_1, d_{Alice}} n^1_{Alice} \text{ and Alice} \xrightarrow{v_1, r'_1, d_{Alice}} n^2_{Alice}$$
   $$\text{Bob} \xrightarrow{u_2, r_2, d_{Bob}} n^1_{Bob} \text{ and Bob} \xrightarrow{v_2, r'_2, d_{Bob}} n^2_{Bob}$$

3. $$n^1_{Alice} \xrightarrow{u_1} n^1_{Bob} \text{ and } n^2_{Alice} \xrightarrow{v_1} n^2_{Bob}$$
   $$n^1_{Bob} \xrightarrow{d_{Bob}} n^1_{Alice} \text{ and } n^2_{Bob} \xrightarrow{d_{Bob}} n^2_{Alice}$$

4. $$n^1_{Alice} \xrightarrow{E_{CS}(d_{Alice} \oplus d_{Bob})} n^1_{Bob}$$
   $$n^2_{Alice} \xrightarrow{E_{CS}(d_{Alice} \oplus d_{Bob})} n^2_{Bob}$$

5. $$n^1_{Bob} \xrightarrow{X = E_{CS}(d_{Alice} \oplus d_{Bob} \cdot (u_1 - u_2) \bmod q)} CS$$
   $$n^2_{Bob} \xrightarrow{Y = E_{CS}(d_{Alice} \oplus d_{Bob} \cdot (v_1 - v_2) \bmod q)} CS$$

6. CS then checks the following,
   if $(X + Y)$ *mod* $q = 0$ *return* "**equal**"
   if $(X + Y)$ *mod* $q < q/2$ *return* " $>$ "
   else return " $<$ "

Figure 11.1: PPC Procedure

ZKP for PPC Procedure.

1. CS acts as $\mathcal{P}$. $\mathcal{V}$ submits $\beta,\ g^{\beta} \pmod p$ through `SC`, with $\beta \in \mathbb{Z}_q$ as the random challenge.
2. The accountants, n^1_{Alice} and n^2_{Alice}, send the values $C(u_1, r_1)^D,\ C(u_2, r_2)^{-D},\ C(v_1, r'_1)^D$, and $C(v_2, r'_2)^{-D}$, using `SC`. Note that the exponentiation is modular and over p. Here, $D = d_{Alice} \oplus d_{Bob}$.
3. $\mathcal{P}$ computes, for some *private* $\alpha \in \mathbb{Z}_q$:

$$\begin{aligned} c_1 &= C(u_1, \cdot)^D C(u_2, \cdot)^{-D} C(v_1, \cdot)^D C(v_2, \cdot)^{-D} \pmod p \\ c_2 &= g^{(\alpha+\beta) \bmod q} \pmod p \\ c_3 &= c_1 \cdot c_2 \pmod p \end{aligned}$$

4. $\mathcal{P}$ submits the following values using `SC`.

$$\begin{aligned} c_4 &= H(c_1, c_2, c_3) \\ c_5 &= (\alpha + \beta + X + Y) \bmod q. \end{aligned}$$

5. $\mathcal{P}$, similar to X and Y, i.e., Figure 11.1, gets the values,

$$\begin{aligned} H_1 \bmod q = &\quad (D \cdot (r_1 + r'_1)) \bmod q \\ H_2 \bmod q = &\quad -(D \cdot (r_2 + r'_2)) \bmod q. \end{aligned}$$

6. $\mathcal{V}$ verifies:

$$\left.\begin{aligned} c_3 &\stackrel{?}{=} c_1 \cdot c_2 \pmod p \\ c_4 &\stackrel{?}{=} H(c_1, c_2, c_3) \\ c_3 &\stackrel{?}{=} g^{c_5} \cdot h_{Alice}^{H_1 \bmod q} \cdot h_{Bob}^{H_2 \bmod q} \pmod p \end{aligned}\right\} \tag{11.2}$$

7. $\mathcal{V}$ accepts that the comparison was correctly computed only if Eq. 11.2 holds.

Figure 11.2: ZKP for PPC Verification.

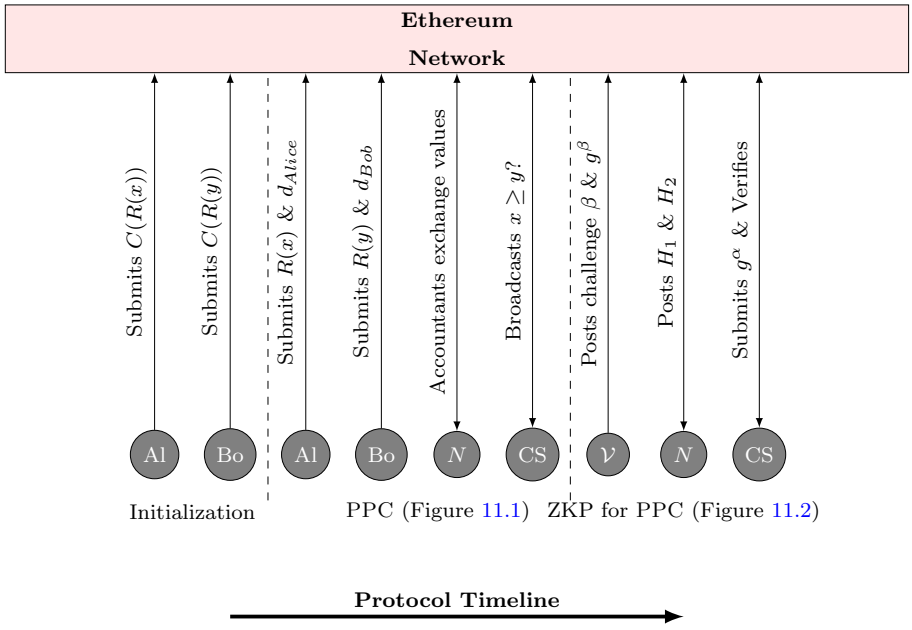


Figure 11.3: Illustration of the timeline in PPC. Here, Al: Alice, Bo: Bob, and N: set of assigned accountants.

11.2.2 PPC: Security and Privacy Analysis

PPC (Figure 11.1) preserves privacy of the values x and y from CS since CS only knows the values[2] $(D \cdot val_1) \bmod q$ and $(D \cdot val_2) \bmod q$. It is trivial to see that CS cannot find anything about the values of (u_1, v_1) and (u_2, v_2). In addition, every accountant only has one component of the other party's (Alice or Bob) value, which implies that it can not either find out anything about the other party's value. However, as $\mathbf{d}$ is publicly known, the value $X + Y$ leaks an *upper bound* of the value $x - y$. We now show that the following results show that the probability of finding the actual value of $x-y$ from $X+Y$ is *negligible*.

[2]This follows as the values $(D \cdot val_i), \forall i \in \{1, 2\}$ are reduced over mod q. For the modular multiplication of $a \cdot b \pmod q$, where q is a prime, and no information of a is known, all possible values of b are equally likely.

Theorem 11.1. *In* PPC *(Figure 11.1), the probability of guessing the actual value of* $x - y$ *is* $\frac{1}{2^{\mathbf{d}}}$*, i.e.,* $\mathit{negl}(\mathbf{d})$ *in* $\mathbf{d}$.

Proof. Observe that, as $d_{\mathsf{Alice}}, d_{\mathsf{Bob}} \in (1, d_{\max}]$, we have $D \in [1, d_{\max}]$. Further, as the value of D is never revealed, we have,

$$\begin{aligned} & X + Y = D \cdot (x - y) \\ \implies & x - y = \frac{X + Y}{D} \\ \implies & x - y \in \left[\frac{X + Y}{d_{\max}}, X + Y\right] \end{aligned}$$

Thus, the value of $X + Y$ bounds the value of $x - y$. However, no other information about the value of $x - y$ is leaked during or after the execution of PPC. This implies that any value in the range $\left[\frac{X+Y}{d_{\max}}, X + Y\right]$ is *equally likely* for $x - y$. Note that, the range $\left[\frac{X+Y}{d_{\max}}, X + Y\right]$ corresponds to $2^{\mathbf{d}}$ guesses. As a result, the probability of finding the actual value of $x - y$ from $X + Y$ is $1/2^{(\mathbf{d})}$, i.e., $\mathsf{negl}(\mathbf{d})$ in $\mathbf{d}$. □

The following corollary also follows directly from Theorem 11.1.

Corollary 11.1. *In* PPC *(Figure 11.1), the probability of finding the actual value of* x *(*y*) from* $X + Y$*, with the knowledge of the other value* y *(*x*) is* $\frac{1}{2^{\mathbf{d}}}$*, i.e.,* $\mathit{negl}(\mathbf{d})$ *in* $\mathbf{d}$.

Probability Threshold. Observe that the agents Alice and Bob know one of the terms in $D = d_{\mathsf{Alice}} \oplus d_{\mathsf{Bob}}$, but not the value itself. These agents also know one of the values, i.e., x or y. This implies, from Corollary 11.1, that the probability with which these agents can guess the value of x or y from $X + Y$ is $\frac{1}{2^{\mathbf{d}}}$. Thus, in PPC, the threshold of the probability with which a party can guess the value of x (y) is $\frac{1}{2^{\mathbf{d}}}$, which is negligible in $\mathbf{d}$.

11.2.2.1 ZKP in PPC

We now show that the ZKP described in Figure 11.2 satisfies the three properties required for a ZKP outlined in Chapter 3.1.2. That is,

- *Completeness.* It is trivial to see that if Eq. 11.2 holds, then the comparison was correct. That is, an honest $\mathcal{P}$ will be able to convince $\mathcal{V}$ that the comparison was correct.

- *Soundness.* If Eq. 11.2 does not hold, i.e., Alice and/or Bob misreported their values, then there can not be a case where $\mathcal{P}$ can find other values except for $(X+Y) \mod q, (H_1) \mod q$ and $(H_2) \mod q$ for which Eq. 11.2 holds, with high probability. This is because Pedersen commitments are computationally binding[3].

- *Zero-knowledge.* Observe that, similar to the argument given for PPC $\mathcal{V}$ does not gain any knowledge of the committed values or the help values through the values $(X+Y) \mod q$, $H_1 \mod q$, and $H_2 \mod q$. Moreover, the value $C(\cdot)^z \mod p$ does not reveal any information about the value of z at any stage of the procedure because of the hardness of the discrete-log problem. □

We now use PPC introduced for secure comparison of two integers to present a novel, secure combinatorial auction for the single-minded case that preserves the privacy of each agent's bidding information even after the bidding phase is over, namely, STOUP.

11.3 STOUP Auction Protocol

Auction Background. As mentioned above, we build our privacy-preserving auction over Algorithm 1. As such, this chapter follows the notations introduced in Chapter 2.2.5.6.3. To

[3]This property also compares *robust* to any misreporting done by the accountants. As we assume the accountants to be strategic-but-curious, they may strategically misreport information. However, Figure 11.2 will allow any $\mathcal{V}$ to detect the misreporting. Thus, PPC (Figures 11.1 and 11.2) is *robust* to any misreporting done by the accountants.

recall, we have an auctioneer AU, the seller itself, interested in selling $\mathcal{M} = [m]$ indivisible items to $\mathcal{A} = [n]$ agents (we assume that $|\mathcal{A}| = n \geq 2$). In this chapter, we also assume a set of privacy accountants, N, can assist AU in determining the winners and their payments. As already seen, Algorithm 1 is DSIC, IR, with an approximation factor $\sqrt{m}$. These make it a desirable candidate to be used for combinatorial auctions where the bidders are single-minded. Note that, as the agents are single-mined, we use $\vartheta_i(s)$ to denote agent i's valuation for the bundle s.

Security & Privacy Properties in Auctions. We now describe the required cryptographic properties of a privacy-preserving auction protocol.

1. *Non-repudiation.* This deals with the inability of an auctioneer or an agent to retract from their actions. These may include exclusion of correctly submitted bid(s) in case of an auctioneer or altering of the bid(s) in case of an agent. Auction protocols must be able to commit an agent to its bid and prove the exclusion of any bid by the auctioneer.

2. *Verifiability.* The public, including the agents, must be shown conclusive proof of the correctness of the auction protocol. The protocol must enforce correctness; an auctioneer should not present valid proofs for invalid winners or incorrect payments.

3. *Privacy.* An auction protocol should hide the bidding information of an agent from the other participating agents. Only after the auction should the information revealed by the winning agents be known. The types of privacies relevant to an auction are defined below. For this, let W be the set of winning agents.

Definition 11.1: Agent Privacy

No agent should be able to discover each other's identity, i.e., for an agent $a \in \mathcal{A}$ **during** the auction and for an agent $a \in \mathcal{A} \setminus W$ **after** the auction, no other agent $b \in \mathcal{A}$ should know about a's participation in the auction[a].

[a]Note that, this definition allows the auctioneer AU to know the agent identities.

Definition 11.2: Bid Privacy

No agent should be able to know any agent's bid valuation, i.e., the probability with which an agent $a \in \mathcal{A} \setminus \{i\}$ can guess agent i's bid valuation ϑ_i is $\ll 1/\vartheta_i$.

Definition 11.3: Bid-Topology Privacy

No agent should be able to know any other agent's bundle of items, i.e., the probability with which an agent $a \in \mathcal{A} \setminus \{i\}$ can guess the item bundle S_i of an agent $i \in \mathcal{A} \setminus \{a\}$ **during** the auction and of an agent $i \in \mathcal{A} \setminus \{\{a\} \cup W\}$ **after** the auction is negligible in the number of items being auctioned.

With this backdrop, consider the following definition.

Definition 11.4: Trustworthy Implementation

An auction protocol that provides non-repudiation and verifiability while preserving agent, bid, and bid-topology privacy and being dominant strategy incentive compatible (DSIC) and (ex-post) individually rational (IR) is a trustworthy implementation of an auction.

11.3.1 STOUP: Protocol

In STOUP, $\mathcal{A}$ is the set of (bidding) agents, and AU is the seller itself. All arithmetic operations (except the payments) are modulo p for the commitments and modulo q for the

values to be committed and the help[4] values. Further, AU acts as the CS (refer to PPC protocol). As aforementioned, we assume that AU is honest-but-curious while the bidders and the set of accountants strategic-but-curious.

Item Bundle. In STOUP, an agent i submits its *item bundle* $\mathcal{S}_i$, consisting of commitments of its preferred items *at least* once as well as *different* commitments of some (or all) of their preferred items randomly such that $|\mathcal{S}_i| = m,\ \forall i \in \mathcal{A}$.

Definition 11.5: Item Bundle

An agent i's item bundle is defined as $\mathcal{S}_i = \{C \cup D\}$ where $C = \{C(R(j)) \,|\, \forall j \in S_i\}$ and $D = \{C(R(k)) \,|\, \forall k \in S_i'\}$,

where S_i' is the set of non-distinct items randomly chosen from S_i such that $|C| + |D| = m$. The idea behind the "item bundle" is to hide the size of an agent's actual bundle from an adversary. The size may (along with other public information) be used to compromise the agent's bid items or bid values[5].

Bid Tuple. With Definition 11.5, each bidder $i \in \mathcal{A}$ participates in STOUP, by submitting the following bid tuple,

$$\mathbf{BT}_i = \Big\langle C(\vartheta_i), C(|S_i|), C\big(R(\vartheta_i/\sqrt{|S_i|}), \mathcal{S}_i \Big\rangle$$

where $w_i = \vartheta_i/\sqrt{|S_i|}$.

[4]The cryptography literature sometimes refers to the randomness in encryption/commitments as help values.

[5]This also does not hurt us computationally as Algorithm 1 only cares about *one* common item among any two (distinct) agent bundles.

Algorithm 16 STOUP: Protocol

procedure AUTHENTICATION PHASE
    Each agent $a \in \{\mathcal{A} \cup N\}$ gives its public id's to AU
    AU assigns each agent a a secret identifier id_a[6]
    AU generates a random id for each item
    AU randomly assigns $\left(n^1_{id_i}, n^2_{id_i}\right) \in N$ to each $id_i \in \mathcal{A}$
end procedure
procedure BIDDING PHASE
    Each bidder $id_i \in \mathcal{A}$ submits $\mathbf{BT}_{id_i}$ to SC
    Each bidder $id_i \in \mathcal{A}$ sends (u, r, d_{id_i}) to $n^1_{id_i}$ and (v, r', d_{id_i}) to $n^2_{id_i}$ for w_{id_i} and $\mathcal{S}_{id_i}$ with SC as described in Figure 11.1
end procedure
procedure WINNER DETERMINATION PHASE
    AU determines – in coordination with the assigned accountants – the set of the winning bidders W consisting of each winner's identifier and calculates payments as defined in Algorithm 1
    AU submits W and the payments with SC
end procedure

11.3.1.1 STOUP Protocol

For STOUP, similar to PPC, we consider a smart contract SC which allows agents to post and get relevant information. Algorithm 16 illustrates STOUP with Figure 11.4 illustrating the protocol flow. As we use PPC for winner(s) and payment(s) determination, we require $w_i < \frac{q}{2 \cdot d_{\max}}$, $\forall i$. We next describe how AU solves steps (1), (2), and (3) of Algorithm 1 in STOUP.

[6] After this assignment, we may use a and id_a interchangeably.

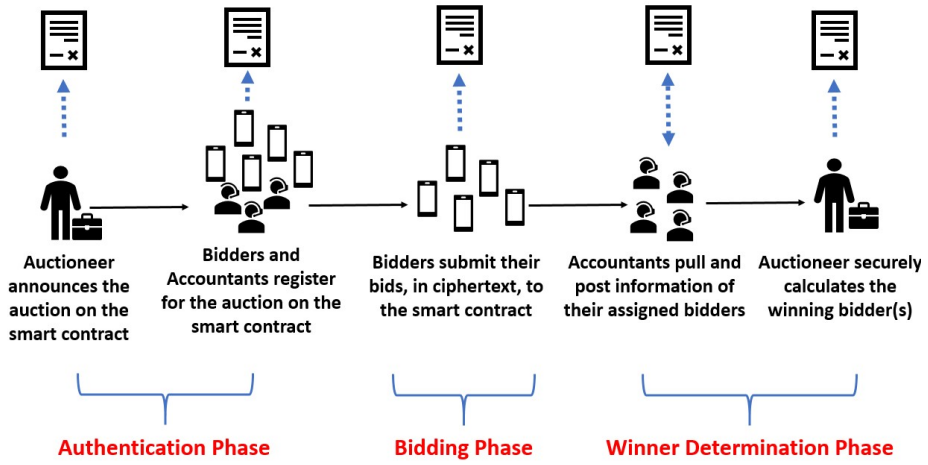


Figure 11.4: Overview of STOUP

11.3.1.2 Bid Initialization

AU sorts the bids based on the values w_{id_i} $\forall id_i$, using any comparison based sorting with the comparison done through PPC (Figure 11.1). For the winner and payment determination phase, w.l.o.g., let the highest agent's identifier be denoted as id_1, the second highest agent as id_2, and so on. Let I consist of the set of identifiers $\{id_1, \ldots, id_n\}$, $\mathcal{S}$ as the set of preferred item bundles of every agent $\{\mathcal{S}_{id_1}, \ldots, \mathcal{S}_{id_n}\}$ and W as the set of winners initialized to $\emptyset$.

11.3.1.3 Winner Determination

AU carries out winner determination (Algorithm 1), in co-ordination with accountants. In this, the highest agent is automatically selected, and its *identifier* is added to W. To determine the other winners, *AU* compares every pair of item, $\forall\, id_j \in I \setminus \{id_1\}$ with every id_k currently in W, using Figure 11.1.

If AU does not find any identical pair of items for an agent id_j for every id_k currently in W i.e., $\mathcal{S}_{id_j} \cap (\cup_{k \in W} \mathcal{S}_{id_k}) = \emptyset$, it adds id_j to W. Otherwise, it discards that agent and continues with the next highest agent.

Note. As the set of items $\mathcal{M}$ is finite, i.e., there are only $\binom{m}{2}$ distinct combinations possible, AU can deterministically get the items, x and y, being compared from the value $x - y$. By using PPC however, the AU will get the value $X + Y$. With this, if $x \neq y$, i.e., $X + Y \neq 0$, all possible $\binom{m}{2}$ combinations will be equally likely.

11.3.1.4 Payment Determination

The payments for every winner $id_i \in W$ are as described in Algorithm 1. AU can find out an agent id_j, $\forall\, id_i \in W$, where j is the smallest index such that $\mathcal{S}_{id_i} \cap \mathcal{S}_{id_j} \neq \emptyset$, and an agent id_k for $k < j$, $id_k \neq id_i$ such that $\mathcal{S}_{id_k} \cap \mathcal{S}_{id_j} = \emptyset$, similar to the procedure to the winner determination described in Chapter 11.3.1.3. If such id_j and id_k exists, then AU asks the assigned accountant $n^1_{id_j}$ of id_j to calculate the payment $p_{id_i} = \vartheta_{id_j} / \sqrt{|S_{id_j}|/|S_{id_i}|}$. The agent id_j opens its commitment $C(R(w_{id_j}))$ for $n^1_{id_j}$, securely. AU asks id_i to open its commitment for $C(|S_{id_i}|)$, and sends the value to $n^1_{id_j}$, which calculates p_{id_i} and sends it to AU. If no such id_j or id_k exist, then $p_{id_{b_i}} = 0$.

11.4 STOUP: Security and Privacy Analysis

STOUP preserves non-repudiation since all the relevant information is submitted on the blockchain, an append-only ledger. We now look at verifiability and the nature of the privacy guarantees as provided by STOUP. Here, we denote the identifier $id_i \in \mathcal{A}$ as i for simplicity of notation.

Verifiability. A prover $\mathcal{P}$ (i.e., the AU in STOUP) proves to a verifier $\mathcal{V}$ the correctness of the order $w_1 \geq \cdots \geq w_n$ and the correctness of the comparisons for $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, for each

$i, j \in \mathcal{A}$. As all values *as well as* item comparisons in STOUPare done using PPC, the ZKP for the comparisons follows the same as described in Figure 11.2[7].

11.4.1 Privacy Analysis

STOUP provides the following privacy guarantees.

Proposition 11.1. STOUP *preserves agent privacy (Definition 11.1).*

Proof. In STOUP, all agents first communicate with *AU* over a blockchain with smart contract support (e.g., Ethereum). We know that communication over the blockchain is pseudo-anonymous, i.e., these addresses can not be linked to the real-world identity of the agents with high probability. *AU* assigns a random *id* to all the agents that are also used throughout the execution of STOUP for communication. Thus, there is no step in the protocol from which the agent identities will be leaked unless *AU* leaks the identities.

More concretely, assuming the *AU* does not leak the agent identities, an adversary must break the cryptographic primitives used in STOUP and (say) the Ethereum blockchain. STOUP relies on the Pedersen commitment and a public-key encryption scheme (e.g., ElGamal encryption (Definition 3.10). Breaking these will require the adversary to solve the Discrete Log Problem (Definition 3.5), which is computationally infeasible for any probabilistic polynomial-time (PPT) adversary.

The adversary can instead try attacking the public address used by an agent for communication over Ethereum in STOUP. However, the safety of the public address relies on the security guarantees of the ECDSA digital signature scheme and the computational infeasibility of inverting the Keccak-256 (or SHA3-256) hash function for any PPT adversary. □

[7] As Pedersen commitments are computationally binding, $\mathcal{V}$ does not require multiple proofs for different commitments of the same values. This significantly reduces the computational time, unlike [195, 225].

Proposition 11.2. STOUP *preserves each agent's bid privacy (Definition 11.2).*

Proof. Each agent is required to submit only the Pedersen commitments of their bids, which are perfectly hiding. That is the probability with which an adversary will win the hiding game – knowing the message $x \in \mathbb{Z}_q$ given only the commitment $C(x, r)$ – is negligible in the security parameter λ. More formally, let U_λ denote the uniform distribution over all possible values to open the commitment. Now, the Pedersen commitments satisfy the fact that the $\forall x \neq x'$, no probabilistic polynomial-time (PPT) adversary can computationally distinguish between the probability ensembles $\{C(x, U_\lambda)\}$ and $\{C(x', U_\lambda)\}$, i.e., $|\sum_{r \in U_\lambda} \Pr[x \mid C(x, r)] - \Pr[x' \mid C(x', r)]| \leq \mathsf{negl}(\lambda)$.

Hence, the adversary can, at best, try a brute force other than through a brute force attack. Typically, 64 bits are enough to represent each agent's bid valuation, while $|q|$ is chosen to be 1024 bits. Thus, we have $1/2^{|\frac{q}{2 \cdot D_{\max}}|} \ll 1/\vartheta_i \; \forall i \in \mathcal{A}$. □

Proposition 11.3. STOUP *preserves bid and bid-topology privacy from the accountants.*

Proof. Since each individual notary will only know one value (either u or v) of its assigned agent i's w_i or $\mathcal{S}_i$ and assuming they do not collude among each other or with AU, STOUP leaks no information regarding the bids or the items in the bid tuple the bid and bid-topology privacy from the notaries. The formal proof follows from Corollary 11.1. □

Lemma 11.1. *In* STOUP*, the probability with which AU can know at least one item in agent j's bid-topology (Definition 11.3) is $1/s_i$. The probability with which AU can know the complete bid-topology of an agent j is,*

$$P_j(s_i) = \frac{1}{2^m - 2^{m - s_i}} \tag{11.3}$$

$\forall j \in \mathcal{A} \backslash W$, *such that* $i \in W$ *is that agent for which* $S_j \cap S_i \neq \emptyset$ *in Step 2 of Algorithm 1,* $s_i = |S_i|$ *and* m *are the number of items.*

Proof. AU, through the bidding topology of the winners and its knowledge about which agents have at least one item in common, can infer some information about the bid-topology of an agent $j \in \mathcal{A} \setminus W$. With this, AU can know that j's bid-topology consists of at least one item belonging to i's preferred bundle of items, S_i. The probability with which AU can figure that item is $1/s_i$.

Further, with this information, AU can eliminate all possible subsets of $\mathcal{A}$ which do not consist of any item in S_i. Then the probability with which STOUP leaks each agent j's bid-topology to AU is given by Eq. 11.3, as all the remaining subsets i.e., $2^m - 2^{m-s_i}$ are equally likely. □

Interpreting Lemma 11.1. The above lemma follows by observing that AU through the bidding topology of the winners and its knowledge about which agents have at least one item in common, can infer some information about the bid-topology of an agent $j \in \mathcal{A} \setminus W$. We then get the probability defined in Eq. 11.3 by eliminating the subsets that do not comprise the shared item.

From Eq. 11.3, STOUP preserves bid-topology privacy with high probability when $s_i \geq 2, \forall i \in W$. For the analysis of the result, observe that Eq. 11.3 can be written as,

$$P_j(s_i) = \frac{1}{2^m - 2^{m-s_i}} = \frac{2^{s_i}}{2^{s_i} - 1} \left(\frac{1}{2^m} \right).$$

Thus, the increase in the probability with which AU can determine the complete bid-topology of an agent with respect to randomly guessing the complete bid-topology is by a *constant factor*, i.e., by $\frac{2^{s_i}}{2^{s_i}-1}$. Assuming that each agent's bundle size is ≥ 2, the worst case follows when $s_i = 2$. The probability that AU can know the complete bid-topology of an agent b_j in this case is, $P_j(s_i) = \frac{4}{3}\left(\frac{1}{2^m}\right)$, which is an increase by a factor $\frac{4}{3}$ or an increase by 33.33% of $O(\frac{1}{2^m})$ which is negligible in m, i.e., $\mathsf{negl}(m)$.

The probability result follows from the fact that at no point during the auction or post-auction and $\forall j \in \mathcal{A} \setminus W$, the cardinality of the preferred bundle of items of an agent j, i.e., s_j, is revealed to AU in STOUP. Note that Eq. 11.3 does not hold for an auction protocol that leaks the cardinality of S_j of an agent j. For instance, if AU knew that for an agent j, $s_j = m$, the probability with which agent j's bid-topology is leaked to AU would be 1. Lastly, combining these privacy guarantees implies the following theorem.

Theorem 11.2. STOUP *is a trustworthy implementation of Algorithm 1.*

Proof. Propositions 11.1, 11.2, and 11.3 and Lemma 11.1 show that STOUP preserves agent, bid and bid-topology privacy with high probability (the probability of guessing improves only by $O\left(\frac{1}{2^m}\right)$) and is non-repudiate and verifiable. Since the protocol also solves the winner and payment determination problem through Algorithm 1, it is DSIC and ex-post IR. Thus, the theorem follows from Definition 11.4. □

11.5 STOUP: Implementation

To avoid any floating-point number, AU can announce at the start of the auction that w_i for every agent i, will have x-*precision* i.e., each value w_i will be significant up to x decimal places.

Simulation Setup. We generate all auction instances as a CATS file using the SATS command-line tool [291]. We calculate the optimal social welfare by solving the winner determination problem for the general single-minded case through FRODO 2.0 [168]. For this calculation, the generated CATS file is parsed through the inbuilt FRODO 2.0 parser to convert it to XCSP. The XCSP file is then solved using optimal algorithms (such as DPOP, P-DPOP, etc.) provided in FRODO 2.0 (through GUI or command line). Further, the primes p and q are of size 1024 bits. We use a quad-core Intel i5-4210U CPU with a 1.70GHz processor and 8GB RAM for the simulations. We also assume no latency in inter-party communication. Consequently, the computational bottleneck of STOUP

n	m	**Upper Bound** ($\sqrt{m}$)	$\frac{\textbf{Optimal Welfare}}{\textbf{Approximate Welfare}}$	**Time Taken (mins)**
25	9	3	1.11905993576	2.1826
25	12	3.4641	1.1313692063	5.21355
25	15	3.8729	1.05711039103	11.103467
100	9	3	-	11.59642
100	12	3.4641	-	19.72178
100	15	3.8729	-	54.084380

Table 11.3: STOUP bound for 25 random auction instances

corresponds to the verification of *every* value and item comparison, i.e., Figure 11.2. Table 11.3 presents the results. Note that, for large n, it is difficult to calculate the optimal welfare[8], as the problem is NP-Hard.

Simulation Analysis. As stated, the mean time taken for STOUP in Table 11.3 includes verifying every value and item comparison done throughout the execution of STOUP. However, the time consumed to verify the value and item comparisons is significantly less than other secure auction protocols such as [225]. For comparison, a 100 bid *single-item* auction (i.e., $n = 100$ and $m = 1$) takes approximately 2.51 hours in [225] (see [225, Table 2]), while a 100 bid *single minded* combinatorial auction (i.e., $n = 100$) even with $m = 15$, only takes approximately 0.91 hours in STOUP. This decrease in the run-time shows the practicality of PPC.

Gas Consumption. A smart contract is compiled as bytecode and executed on the Ethereum Virtual Machine (EVM). EVM charges a fee per computational step executed in a contract or transaction to prevent deliberate attacks and abuse on the Ethereum network. This fee is measured in terms of *gas* units (refer to Chapter 3.2.2.3).

[8] As discussed in Chapter 2.1, for an auction, *social welfare* is the summation of all the winning bidders' valuations.

The estimate depends on the post operations described in STOUP. For example, each bidder submits its bid tuple and other required information. Then, AU and the accountants further exchange information on-chain. The EVM uses 256-bit as default, so changing p does not affect the estimate. Smaller p's will result in greater gas consumption as the EVM "downscales" the values. Typically, the gas associated with a post-operation for the *uint256* variable in *Solidity* is ≈ 62664. With this in STOUP, AU will consume 313320 per comparison while each accountant consumes 501312. Each participating bidder will at-worst consume $(3 + m) \cdot 62664$ gas units.

11.6 Conclusion

This chapter observed that Yao's Millionaires' Problem (YMP) is fundamental to designing secure AI applications. Towards this, we presented a practical and verifiable solution to YMP, namely, PPC (Figures 11.1 and 11.2). PPC uses third-party agents to securely compare two integers that do not learn any information (Theorem 11.1). Significantly, PPC achieves the comparison in constant time and one execution of Figure 11.1.

To demonstrate the effectiveness of PPC, we use it to design a Secure, Truthful cOmbinatorial aUction Protocol (STOUP) for single-minded bidders (Algorithm 16). STOUP preserves an agent's bid valuation and topology at any time during the auction and post-auction, even to the auctioneer, unlike prior works. The bid-topology is preserved with high probability when every agent's bundle size is ≥ 2, which is a fair assumption in practice for combinatorial auctions (Lemma 11.1). We further believe that PPC will find an application for other secure multi-agent systems (MAS), including different auctions, voting, and distributed optimization.

Chapter 12

Differentially Private Multi-agent Constraint Optimization

Distributed Constraint Optimization (DCOP) is a framework in which multiple agents with private constraints (or preferences) cooperate to achieve a common goal optimally. DCOPs are applicable in several multi-agent coordination/allocation problems, such as vehicle routing, radio frequency assignments, and distributed scheduling of meetings. However, optimization scenarios may involve multiple agents wanting to protect their preferences' privacy. Researchers propose privacy-preserving algorithms for DCOPs that provide improved privacy protection through cryptographic primitives such as partial homomorphic encryption, secret-sharing, and secure multiparty computation. These privacy benefits come at the expense of high computational complexity. Moreover, such an approach does not constitute a rigorous privacy guarantee for optimization outcomes, as the result of the computation may compromise agents' preferences. In this work, we show how to achieve privacy, specifically Differential Privacy, by randomizing the solving process. In particular, we present P-Gibbs, which adapts the current state-of-the-art algorithm for DCOPs, namely SD-Gibbs, to obtain differential privacy guarantees with much higher computational efficiency. Experiments

on benchmark problems such as Ising, graph-coloring, and meeting-scheduling show P-Gibbs' privacy and performance trade-off for varying privacy budgets and the SD-Gibbs algorithm. More concretely, we empirically show that P-Gibbs provides fair solutions for competitive privacy budgets.

⋆ ⋆ ⋆ ⋆ ⋆

12.1 Introduction

The idea of *distributed computation* has been a trending topic among computer scientists for decades. Distributing the computation has several well-known advantages over centralized computing. These include no single-point failure, incremental growth, reliability, open system, parallel computing, and easier management of resources. In this chapter, we focus on the distributed analog of constrained optimization [243], namely *distributed constraint optimization problem* (DCOP), first introduced in [305].

As mentioned earlier in Chapter 12.1, in DCOPs, agents compute their value assignments to maximize (or minimize) the sum of resulting constraint rewards. These constraints quantify an agent's preference for each possible assignment. E.g., distributed scheduling of meetings and graph-coloring-related applications such as mobile radio frequency assignments are modeled using DCOPs. As stated in Chapter 12.1, using the meeting-scheduling for several Chief Executive Officers (CEOs) scenario, DCOP algorithms may leak sensitive information regarding the CEOs' schedule during the computation process or through the outcome [100].

12.1.1 Privacy in DCOPs

In general, the need to preserve the privacy of an agent's sensitive information in AI/ML solutions is paramount [3, 33, 138]. Despite its distributed nature, 'solving' a DCOP in-

stance transfers information across agents, which may leak sensitive information to the other participants, such as the agent's preferences. In the above example, an information leak may involve a CEO inferring critical information about the other participating CEOs during the information exchange. Thus, privacy-preserving solutions to DCOPs are necessary and form the basis of this work. Before discussing the existing privacy-preserving DCOP literature, we first summarize the existing DCOP algorithms.

12.1.1.1 DCOP Algorithms

Solving a DCOP instance is NP-hard [197]. Nevertheless, the field has grown steadily over the years, with several algorithms being introduced to solve DCOP instances, each providing some improvement over the previous. These algorithms are either: (1) search-based algorithms like SynchBB [135], ADOPT [197] and its variants, AFB [115] and MGM [183], where the agents enumerate through sequences of assignments in a decentralized manner; (2) inference-based algorithms like DPOP [230] and Max-Sum [102], where the agents use dynamic programming to propagate aggregated information to other agents; (3) sampling-based algorithms like DUCT [217, 218], where the agents iteratively sample promising assignments. We refer the reader to [105] for a comprehensive survey on DCOP algorithms.

This chapter focuses on SD-Gibbs (and its parallel analog PD-Gibbs) [209], the current state-of-the-art algorithms for approximately solving DCOPs. SD-Gibbs is known to run faster (e.g., compared to DUCT [209]), find a better quality of solutions (e.g., compared to MGM and DUCT [218]), and be applicable for larger problems (e.g., compared to DPOP [218] and DUCT [209]).

12.1.1.2 Privacy-preserving DCOP Algorithms

In literature, several algorithms exist to preserve privacy in DCOPs. Unfortunately, we identify that such existing privacy-preserving algorithms have two significant drawbacks.

Firstly, these algorithms lack scalability with respect to the number of agents and constraints. Secondly, privacy-preserving complete algorithms for DCOPs converge at the optimal solution. As such, the said solution may be used to infer potentially critical information regarding the DCOP instance. We next discuss these drawbacks in detail.

12.1.1.2.1 Non-scalability of Private DCOPs As DCOPs are NP-hard, complete DCOP algorithms such as DPOP do not scale as it is. The added complexity of the underlying cryptographic primitives further hits the scalability of its privacy variants: P-DPOP [100], $P^{3/2}$-DPOP [167], and P^2-DPOP [167]. Of these, P-DPOP scales the best, primarily due to its weaker privacy guarantees. Even for P-DPOP, the algorithm is known to only scale up to 12 agents for graph-coloring, and 6 agents for meeting-scheduling – two popular benchmark problems in DCOP literature.

On the other hand, more recent algorithms like P-Max-Sum [274] and P-SyncBB [127] scale better, either in part to the underlying approximate algorithm (P-Max-Sum) or efficient secure multi-party computation protocols (P-SyncBB). However, the algorithms are still computationally intensive. For instance, P-Max-Sum requires a computational overhead ranging from minutes to an hour. Also, the algorithm's run-time increases by a factor of 1000s over its non-private variant [274].

12.1.1.2.2 Solution Privacy In addition to their lack of scalability, privacy-preserving DCOP algorithms built atop complete algorithms output the optimal assignment (or solution). However, these final assignments cannot be private and, in turn, may leak critical information about agents' preferences [100]. We refer to this information leak as *solution privacy*. For complete DCOP algorithms such as DPOP, their privacy variants built through cryptographic primitives such as P-DPOP [100], P $^{3/2}$-DPOP [167], and P ^{2}-DPOP [167] trivially do not satisfy solution privacy.

12.1.2 Chapter Contributions

In summary, we note that while algorithms exist that realize constraint privacy, their non-scalability hinders their practical use. The cryptographic primitives used to achieve privacy further exaggerate this lack of scalability. Moreover, information leaked through the algorithms' output can be used to extract significant private information, especially when the problem is solved repeatedly. Motivated by these, we aim to construct a scalable DCOP algorithm while providing rigorous and provable privacy guarantees for agents' constraints and one that satisfies solution privacy.

Note that the non-guarantee of solution privacy is an inevitable outcome of a cryptographically secure algorithm. However, it is possible to make the final assignment of a DCOP algorithm *differentially private* [92]. Consequently, to achieve such a private and scalable algorithm, we focus on the strong notion of *differential privacy* (DP) [92, 93]. In particular, we focus on achieving privacy in SD-Gibbs using DP techniques. Furthermore, we consider a *stronger* local model of privacy [93], which ensures the indistinguishability of any two agents.

Our Approach and Contributions. Differential privacy (DP) is usually achieved through randomization. This makes it natural to consider randomized algorithms, such as SD-Gibbs [209], which also, at the same time, are much more computationally efficient. However, these algorithms by themselves do not protect privacy, and we develop additional mechanisms to ensure DP of the entire process. More concretely, we consider the following approach to design a scalable DCOP that preserves constraint privacy.

Identifying Privacy Leaks in SD-Gibbs. We first show that SD-Gibbs may leak information about agent constraints during execution. More concretely, during the algorithm's execution, agents send and receive information that directly depends on their utility functions, i.e., functions that quantify the preferences for each constraint. What is more, SD-Gibbs' iterative nature may further lead to a high privacy loss over the iterations.

As such, we must construct an algorithm that not only preserves constraint privacy but incurs minor privacy leaks across iterations.

P-Gibbs. Towards this, we develop a new differentially private variant of SD-Gibbs. We present a novel algorithm *P-Gibbs*, which crucially differs from SD-Gibbs in three key aspects with respect to preserving constraint privacy.

1. *Sampling through Soft-max with Temperature.* Sampling through Gibbs distribution [172] in SD-Gibbs leaks information about the underlying utilities. This leak is because a value with greater utility is more likely to be sampled. We use *soft-max with temperature* over the Gibbs distribution to overcome this. This process smooths out sampling distributions in SD-Gibbs.

2. *Adding Gaussian Noise to Relative Utilities.* Each agent in SD-Gibbs sends its relative utility to its immediate parent. These relative utilities are the difference between its previous and current assignments. As such, these values leak vital information about the utilities. E.g., if a particular assignment has a high utility for agent j, but low for others (and it is known), an intermediate agent will learn about agent j even from the aggregated utility. To this end, we add *Gaussian* noise to the relative utility in our algorithm. The added noise helps to perturb each agent's relative utilities such that information regarding the utilities is protected.

3. *Subsampling.* As aforementioned, the iterative nature of SD-Gibbs implies that the privacy loss is accumulated over the iterations. We propose that each agent must sample a new assignment with a subsampling probability q to limit this loss. This limits the information leaked at each iteration, resulting in bounded privacy loss.

In addition, we provide a refined analysis of privacy within the framework of (ϵ, δ)-DP for P-Gibbs. We simulate P-Gibbs on three benchmark problems in DCOP literature, namely Ising [50], graph-coloring, and meeting-scheduling [181]. Our experiments demonstrate

our novel algorithm's practicality and robust performance for a reasonable *privacy budget*, i.e., ϵ, with SD-Gibbs as the baseline. Specifically, we show that P-Gibbs provides only a marginal drop in solution qualities compared to SD-Gibbs for a desirable privacy budget.

12.1.3 Chapter Overview

The chapter's structure is as follows. In Chapter 12.1.4, we place P-Gibbs concerning the privacy-preserving DCOP literature. We formally introduce DCOP, describe SD-Gibbs, and define differential privacy (DP) in our context in Chapter 12.1.5. We illustrate the nature of privacy leaks in SD-Gibbs with Chapter 12.2. Chapter 12.3 introduces our novel privacy variant P-Gibbs, including a refined analysis of (ϵ, δ)-DP. Next, in Chapter 12.4, we empirically validate P-Gibbs over several problem instances of benchmark problems in DCOP literature. Our experiments highlight our privacy variant's efficiency. Chapter 12.5 concludes the chapter and discusses future research directions.

12.1.4 Chapter Background: Related Work

This section places our work concerning the general DCOP literature, focusing on privacy-preserving DCOPs. Table 12.1 compares the works described in this section with our novel privacy variant, P-Gibbs, regarding their privacy guarantees and scalability.

12.1.4.1 Distributed Constraint Optimization Problem (DCOP)

As mentioned in Chapter 12.1, despite the computationally hard nature of DCOPs, researchers have proposed various algorithms that aim to solve them completely or approximately. Outside of the popular algorithms like DPOP [230], ADOPT [197], Synch-BB [135], and Max-Sum [102], the field has also seen some recent sampling-based algorithms. Details follow.

Ottens et al. [217] propose Distributed Upper Confidence Tree (DUCT), an extension of UCB [107] and UCT [114]. While DUCT outperforms the algorithms above, its per-agent

memory requirement is exponential in the number of agents. It prohibits it from scaling up to larger problems.

Nguyen et al. [209] improve upon DUCT through their sampling-based DCOP algorithms: *Sequential Distributed Gibbs* (SD-Gibbs) and *Parallel Distributed Gibbs* (PD-Gibbs). These are distributed extensions of the Gibbs algorithm [172]. SD-Gibbs and PD-Gibbs have a linear-space memory requirement, i.e., the memory requirement per agent is linear in the number of agents. The authors empirically show that SD-Gibbs and PD-Gibbs find better solutions than DUCT, run faster, and solve large problems that DUCT fails to solve due to memory limitations. Therefore, in this work, we focus on SD-Gibbs. Our results can be trivially extended for PD-Gibbs.

SD-Gibbs [209] is a *sampling-based* algorithm in which the authors use the Gibbs distribution [172] to solve DCOPs. The algorithm can be broadly categorized into the following four phases. *(i)* Initialization: Each agent initializes its algorithm-specific variables. *(ii)* Sampling: Agents sample an assignment to their variable based on the Gibbs distribution and depending on the assignments of their neighboring agents. *(iii)* Backtracking: After each agent has sampled its assignment, they calculate their *relative utilities*. That is the difference between their previous assignment with their current assignment. The agents then send the utilities to their immediate parents. The parents add their utilities to the ones received, and the process continues till the root agent. This concludes one *iteration*. *(iv)* Deriving Solution: The backtracking process results in the root agent holding the global relative utility. Based on the solution observed thus far, the root throws away or keeps the solution.

This chapter focuses on SD-Gibbs due to its improved solution quality and computational efficiency.

12.1.4.2 Privacy in DCOPs

The existing literature focuses on the following techniques to ensure privacy in DCOPs.

Algorithm	Complete	Agent Privacy	Topology Privacy	Constraint Privacy	Decision Privacy	Solution Privacy	Collusion Resistance	No Privacy Overhead
P-DPOP [100, 167]	✓	✓	○	○	○	✗	✗	✗
P $^{3/2}$*-DPOP* [167]	✓	✓	○	○	✓	✗	✗	✗
P ^{2}*-DPOP* [167]	✓	✓	○	✓	✓	✗	✗	✗
P-SyncBB [127]	✓	✗	✓	✓	✓	✗	✗	✗
P-Max-Sum [274]	✗	✗	✓	✓	✓	✗	✗	✗
P-RODA [128]	✗	✗	✓	✓	✓	✗	✗	✗
PC-SyncBB [273]	✓	✗	✗	✓	○	✗	✓	✗
MD-Max-Sum [160]	✗	✗	✓	✓	✓	✗	✓	✗
P-Gibbs	✗	✓†	✓†	✓‡	✗	✓	✓	✓

†: P-Gibbs can support agent and topology privacy through anonymous communication (e.g., using codenames [100])

‡: Differentially-private guarantee

Table 12.1: Comparing existing literature in privacy-preserving DCOPs with our novel privacy variant, P-Gibbs. Here, the "green check" mark denotes the realization of the property, "○" that the property is realized partially, and the "red cross" mark if the property is not realized. Note that the rest of the algorithms provide a cryptographic guarantee outside of P-Gibbs.

12.1.4.2.1 Achieving Privacy through Cryptosystems Privacy in DCOPs has focused on using cryptographic primitives, such as *partial homomorphic encryption* and *secret-sharing methods*. To quantify the privacy guarantees, researchers propose the following four notions.

1. *Agent privacy* [100, 167]. No agent must learn the existence of its non-neighboring agents, i.e., agents it does not share constraints with.

2. *Topology privacy* [100, 167]. No agent must discover the existence of topological constructs in the constraint graph, such as nodes (i.e., variables), edges (i.e., constraints), or cycles, unless it owns a variable involved in the construct.

3. *Constraint privacy* [100, 167]. No agent should be able to discover the nature of a constraint that does not involve a variable it owns.

4. *Decision privacy* [100, 167]. No agent should be able to discover the value that another agent's variable takes in the solution chosen for the problem (modulo semiprivate information).

Several privacy-preserving algorithms exist, using *secure multi-party computation* [300] atop existing DCOP algorithms to provide cryptographic privacy guarantees. These include P-DPOP [100], $P^{3/2}$-DPOP, P^2-DPOP [167], which builds on the DPOP algorithm; P-SyncBB [127] and PC-SyncBB [273] over SynchBB; P-Max-Sum [274] over Max-Sum; and P-RODA [128] which is privacy variant for algorithms which fit in Region Optimal DCOP Algorithm (RODA) [128] framework. In Table 12.1, we provide the known private algorithms and the privacy notions they satisfy; details follow.

To guarantee agent and (partial) topology privacy, the algorithms P-DPOP, P $^{3/2}$-DPOP, and P ^{2}-DPOP use "codenames" (randomly generated numbers) in place of the actual variable names and domains. These codenames are used for information exchange between agents. The agent selected as the root then "decrypts" these values to arrive at the solution. In contrast, P-Max-Sum and PC-SyncBB support topology privacy. These algorithms also use information-hiding public-key encryption and random shifts and permutations.

P^2-DPOP completely preserves constraint privacy. The algorithm uses the partial homomorphic property of ElGamal encryption for the same. This technique is unlike P-DPOP and $P^{3/2}$-DPOP, which merely adds the random numbers communicated by agents, inadvertently leaking privacy. P-Max-Sum also preserves constraint privacy by communicating information through encryptions or random shares.

DCOPs are also solved using region-optimal algorithms such as KOPT [153] or DALO [158]. Grinshpoun *et al.* present an umbrella setup, namely RODA, that generalizes these region-optimal algorithms. The authors present P-RODA [128], which implements the privacy-

preserving implementations of these region-optimal algorithms. P-RODA uses cryptographic primitives such as secret sharing and homomorphic encryptions. As such, P-RODA perfectly simulates RODA but with a significant computational overhead.

The algorithms mentioned above, except PC-SyncBB, assume that agents do *not* collude. Note that any two or more colluding agents can leak sensitive information about the other agents. Using secure multi-party computation, Tassa *et al.* [273] show that PC-SyncBB is *collusion resistant* as long as the majority of the agents do not collude. Most recently, Kogan *et al.* [160] introduced MD-Max-Sum, a privacy-preserving, collusion-resistant DCOP algorithm built atop the Max-Sum algorithm. Crucially, MD-Max-Sum uses third parties, namely *mediators*, to guarantee collusion resistance and has a reduced run time compared to PC-SyncBB. The algorithm satisfies constraint, topology, and decision privacy.

12.1.4.2.2 Solution Privacy Research on privacy-preserving algorithms for DCOP typically focuses on complete algorithms guaranteed to compute the optimal assignment (solution) [100, 167]. Obviously, one cannot keep the solution secret, so the information leaked by knowledge of the solution has generally been considered an inevitable privacy loss. Moreover, as the optimization outcome cannot be preserved, the computation may compromise agents' preferences, thereby violating constraint privacy.

However, it is possible to make the solution, and therefore, any information that can be inferred from it differentially private. We call this property *solution privacy* and add it as an additional objective for privacy-preserving DCOP. We show that our differentially private variant, P-Gibbs, satisfies solution privacy through randomization of the computation process (Table 12.1).

Solution Privacy and Decision Privacy. In this chapter, we follow the classic security principle – "no security through obscurity" – meaning that we cannot assume privacy would be kept by simply hiding decisions from some agents (the server or other agents might reveal

them; agents who get your decision can be malicious; decisions may be observed through agents actions by outsiders; and so on). Thus, in our context, the privacy notions of solution privacy and decision privacy are not equivalent.

12.1.4.2.3 Non-scalability of Existing Private DCOP Algorithm Unlike our privacy variant P-Gibbs (Table 12.1), cryptographic primitives and the computationally expensive nature of DCOPs results in the algorithms mentioned above not being *scalable* in terms of the number of agents and constraints. More concretely, we say that a private DCOP algorithm admits a *privacy overhead* if it is significantly more computationally expensive compared to its non-private variant.

Our definition implies that private algorithms based on cryptographic primitives incur a significant privacy overhead. E.g., P-DPOP [100] scales up to 12 agents for graph-coloring and 6 agents for meeting-scheduling. Such a lack of scalability is also present in privacy-preserving algorithms built atop approximate algorithms. E.g., P-Max-Sum's run-time increases three-fold in magnitude over its non-private variant [274].

12.1.4.2.4 Other Privacy Notions Information Entropy. In a parallel line of work, the authors in [182] use *information entropy* to quantify the privacy loss incurred by an algorithm in solving a distributed constraint problem. The result is later furthered by [39, 125]. Grinshpoun et al. [125] present private local-search algorithms based on the algorithms above. The authors use this quantification to show that their algorithms provide high-quality solutions while preserving privacy. While the privacy loss metric defined in [182] is interesting, it does not offer a worst-case guarantee. Practically, even a minor leak may result in information being revealed completely.

Utilitarian DCOPs. Savaux *et al.* [252, 251] propose *Utilitarian* DCOPs (UDCOPs) where privacy leakage is correlated to the quality of the final assignment. They assume that each agent also maintains a privacy cost for each assignment's utility, which captures

the desire of the agent to preserve that utility's privacy. With this modeling, they can derive the overall privacy cost and the final solution.

The authors introduce private DCOP algorithms based on this idea (e.g., DBOU and DSAU, which are extensions of DBO and DSA, respectively). The key privacy idea in these algorithms is that agents randomly sample new assignments and only broadcast the information if it positively changes their overall utility.

While such a utility-based privacy cost is another interesting way of quantifying privacy leaks in DCOPs, we believe a (ϵ, δ)-DP approach is a more robust measure of the same. First, the privacy budget used in UDCOPs appears to be agent-specific (i.e., agents may define it arbitrarily). As such, it may not be applicable in practice as the agents may find it difficult to quantify the privacy cost of revealing information about a certain resource. Furthermore, privacy implications can change with time, even if an agent can estimate its cost at one point. That is, the quality of the obtained solution may not be useful in the future. In contrast, there is a clear consensus on appropriate values of ϵ and δ in DP, implying quantifiable privacy guarantees.

Second, to the best of our knowledge, while we provide worst-case privacy guarantees for P-Gibbs, similar to the information entropy-based privacy measure, there are no worst-case guarantees for the algorithms in [252, 251]. So, even if we disregard each agent's arbitrary privacy cost assignments, it is impossible to say if the solution reveals something about the true utilities. And if there is a correlation between privacy costs and utilities, it might even reveal more.

12.1.5 Chapter Background: Preliminaries

We now formalize DCOPs, summarize SD-Gibbs, and define privacy definitions relevant to our work.

12.1.5.1 Distributed Constraint Optimization Problem (DCOP)

Distributed Constraint Optimization Problem (DCOP) is a class of problems comprising a set of variables, a set of agents owning them, and a set of constraints defined over the set of variables. These constraints reflect each agent's *preferences*.

Definition 12.1: Distributed Constraint Optimization Problem DCOP

A Distributed Constraint Optimization Problem (DCOP) is a tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{D}, \mathcal{F}, \alpha \rangle$ wherein,

- $\mathcal{X} = \{x_1, \ldots, x_p\}$ is a set of variables;
- $\mathcal{A} = \{1, \ldots, m\}$ is a set of agents;
- $\mathcal{D} = D_1 \times \ldots \times D_p$ is a set of finite domains such that D_i is the domain of x_i;
- $\mathcal{F}$ is a set of utility functions $F_{ij} : D_i \times D_j \to \mathbb{R}$. F_{ij} gives the utility of each combination of values of variables in its scope. Let $var(F_{ij})$ denote the variables in the scope of F_{ij}.
- $\alpha : \mathcal{X} \to \mathcal{A}$ maps each variable to one agent.

In this work, w.l.o.g [306], we assume that $p = m$, i.e., the number of agents and variables, are equal. We also assume $D = D_i = D_j, \ \forall i, j$, i.e., all variables have the same domain. Total utility in DCOP, for a complete assignment $\mathbf{X} = (x_1, \ldots, x_p)$ is:

$$F(\mathbf{X}) \triangleq \sum_{i=1}^{m} \left(\sum_{j} F_{ij}(\mathbf{X}_{||D}) \right), \tag{12.1}$$

where $\mathbf{X}_{||D}$ is the projection of $\mathbf{X}$ to the subspace on which F_{ij} is defined. The *objective* of a DCOP is to find an assignment $\mathbf{X}^*$ that maximizes[1] the total utility, i.e., $F(\mathbf{X}^*) = \max_{\mathbf{X} \in \mathcal{D}} F(\mathbf{X})$.

[1] We can also define a DCOP that minimizes the total utility, i.e., $F(\mathbf{X}^*) = \max_{\mathbf{X} \in \mathcal{D}} - F(\mathbf{X})$.

In DCOPs, a combination of variables (or alternately, agents) is referred to as a *constraint*. The utility functions over these constraints quantify how much each agent *prefers* a particular constraint. This constraint structure is captured through a *constraint graph*.

Definition 12.2: Constraint Graph (CG)

Given a DCOP defined by $\langle \mathcal{X}, \mathcal{A}, \mathcal{D}, \mathcal{F}, \alpha \rangle$, its constraint graph $\mathcal{G} = \langle \mathcal{X}, \mathcal{E} \rangle$ is such that $(x_i, x_j) \in \mathcal{E},\ \forall j \in var(F_{ij})$.

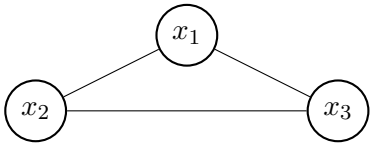
A *pseudo-tree* arrangement has the same nodes and edges as the constraint graph. The tree satisfies (i) there is a subset of edges, called *tree edges*, that form a rooted tree; and (ii) two variables in a utility function appear in the same branch of that tree. The other edges are referred to as *back edges*. Nodes connected via a tree edge are referred to as parent-child nodes. Likewise, back edges connect the pseudo-parent and its pseudo-children. Such an arrangement can be constructed using a distributed-DFS [134].

For the algorithms presented in this chapter, let N_i refer to the set of neighbors of x_i in CG. Also, let $\mathcal{C}_i$ denote the set of children x_i in the pseudo-tree, P_i as the parent of variable x_i, and PP_i as the set of pseudo-parents of x_i. Furthermore, we specifically focus on constraint privacy, which is formally defined in our context next.

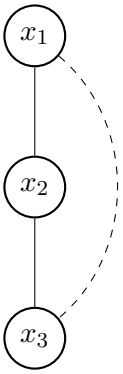
Definition 12.3: Constraint Privacy

Given a DCOP defined by $\langle \mathcal{X}, \mathcal{A}, \mathcal{D}, \mathcal{F}, \alpha \rangle$, constraint privacy implies that an agent i learns no information regarding the utility function $\{F_{jk}\}_{k \in N_j}$ of any agent $j \in \mathcal{A} \setminus N_i$. That is, for any agent, it does not share a constraint with.

12.1.5.1.1 Example Consider the maximization problem depicted in Figure 12.1. Here, $|\mathcal{X}| = |\mathcal{A}|$ s.t. $\mathcal{X} = \{x_1, x_2, x_3\}$ and $D_1 = D_2 = D_3 = \{0, 1\}$. The constraint graph and one possible pseudo tree configuration are presented in Figures 12.1(a) and 12.1(b). In Figure 12.1(b), the solid edges represent the tree edges, while the dashed edges represent the



(a) Constraint Graph



(b) Pseudo Tree

x_i	x_j	F_{ij}
0	0	0
0	1	-1
1	0	-1
1	1	2

(c) Utilities (where $i < j$)

Figure 12.1: DCOP Example

back edges. Figure 12.1(c) shows the utility function that is identical for $F_{12} = F_{23} = F_{13}$. For this example, a solution is as follows. The optimal assignment is $x_1 = 1, x_2 = 1$, and $x_3 = 1$, with the overall utility 6.

12.1.5.2 Sequential Distributed Gibbs (SD-Gibbs)

We now describe Sequential Distributed Gibbs (SD-Gibbs) as first introduced in [209]. In this, the authors map DCOP to a *maximum a posteriori* (MAP) estimation problem. Consider MAP on a *Markov Random Field* (MRF). MRF consists of a set of random variables represented by *nodes*, and a set of *potential functions*. Each potential function, represented by $\theta_{ij}(x_i; x_j)$, is associated with an edge. Let the graph constituting MRF, with nodes and edges, be denoted by $\langle V, E \rangle$.

We now describe Sequential Distributed Gibbs (SD-Gibbs) as first introduced in [209]. In this, the authors map DCOP to a *maximum a posteriori* (MAP) estimation problem. Consider MAP on a *Markov Random Field* (MRF). MRF consists of a set of random variables represented by *nodes*, and a set of *potential functions*. Each potential function,

Variables	Definition
d_i and $\hat{d}_i$	Values in current and previous iteration
d_i^*	Value in the best complete solution so far
$\bar{d}_i$	Best response value
C_i and $\bar{C}_i$	Context and best-response context
$t_i, t_i^*, \bar{t}_i^*$	Time index, best-response and non-best response index
Δ_i	Difference in current and previous local solution of agent i
$\bar{\Delta}_i$	Difference in current best-response solution with previous
Ω	Shifted utility of the current complete solution
$\bar{\Omega}$	Shifted utility of the best-response solution
Ω^*	Shifted utility of the best complete solution

Table 12.2: Variables maintained by each agent x_i in SD-Gibbs

represented by $\theta_{ij}(x_i; x_j)$, is associated with an edge. Let the graph constituting MRF, with nodes and edges, be denoted by $\langle V, E \rangle$.

Let $Pr(x_i = d_i; x_j = d_j)$ be defined as $\exp(\theta_{ij}(x_i = d_i; x_j = d_j))$. Then, the most probable assignment is:

$$\Pr(\mathbf{X}) = \frac{1}{Z} \prod_{i,j \in E} e^{\theta_{ij}(x_i, x_j)} = \frac{1}{Z} \exp \left[\sum_{i,j \in E} \theta_{ij}(x_i, x_j) \right].$$

Here, Z is the normalization factor. This corresponds to the maximum solution of DCOP if,

$$F(\mathbf{X}) = \sum_{i,j \in E} \theta_{ij}(x_i, x_j).$$

12.1.5.2.1 Sampling We now describe *sampling* in SD-Gibbs. Let C_i denote agent i's *context*, defined as the set consisting of its neighbors and the value assigned to them. In

each iteration, each agent i samples a value d_i with the following equation,

$$\Pr(x_i|x_j \in N_i) = \frac{1}{Z} \exp\left[\sum_{\langle x_j, d_j\rangle \in C_i} F_{ij}(d_i, d_j)\right] \tag{12.2}$$

Let, $\mathbb{P}_i(\mathbf{x}_i) = \{\Pr(x_i|x_j \in \mathcal{X} \setminus \{x_i\})|x_i = d_i \ \forall d_i \in D_i\}$. That is, $\mathbb{P}_i$ represents SD-Gibbs' probability distribution of each agent i. The relevant notations required for the SD-Gibbs algorithm are presented in Table 12.2.

Algorithm 17 Sequential Distributed Gibbs [209]

1: Create pseudo-tree
2: Each agent x_i calls INITIALIZE()

Procedure 1 INITIALIZE() [209]

1: $d_i \leftarrow \hat{d}_i \leftarrow d_i^* \leftarrow \bar{d}_i \leftarrow \text{ValInit}(x_i)$
2: $C_i \leftarrow \bar{C}_i \leftarrow \{(x_j, \text{ValInit}(x_j))|x_j \in N_i\}$
3: $t_i \leftarrow t_i^* \leftarrow \bar{t}_i^* \leftarrow 0$
4: $\Delta_i \leftarrow \bar{\Delta}_i \leftarrow 0$
5: **if** x_i is root **then**
6: $\quad t_i \leftarrow t_i^* \leftarrow \bar{t}_i^* \leftarrow 0$
7: $\quad$ SAMPLE()
8: **end if**

Procedure 2 SAMPLE() [209]

$t_i \leftarrow t_i + 1$; $\hat{d}_i \leftarrow d_i$

$d_i \leftarrow$ Sample based on (2)

$\bar{d}_i \leftarrow \text{argmax}_{d'_i \in D_i} \sum_{\langle x_j, \bar{d}_j \rangle \in \bar{C}_i} F_{ij}(d'_i, \bar{d}_j)$

$\Delta_i \leftarrow \sum_{\langle x_j, d_j \rangle \in C_i} \Big[F_{ij}(d_i, d_j) - F_{ij}(\hat{d}_i, d_j) \Big]$

$\bar{\Delta}_i \leftarrow \sum_{\langle x_j, \bar{d}_j \rangle \in C_i} \Big[F_{ij}(\bar{d}_i, \bar{d}_j) - F_{ij}(\hat{d}_i, \bar{d}_j) \Big]$

Send VALUE$(x_i, d_i, \bar{d}_i, t_i^*, \bar{t}_i^*)$ to each $x_j \in N_i$

Procedure 3 VALUE$(x_s, d_s, \bar{d}_s, t_s^*, \bar{t}_s^*)$ [209]

Update $\langle x_s, d'_s \in C_i \rangle$ with (x_s, d_s)

if $x_s \in PP_i \cup \{P_i\}$ **then**

    Update $\langle x_s, d'_s \in \bar{C}_i \rangle$ with $(x_s, \bar{d}_s)$

else

    Update $\langle x_s, d'_s \in C_i \rangle$ with $(x_s, \bar{d}_s)$

end if

if $x_s = P_i$ **then**

    if $\bar{t}_s^* \geq t_s^*$ **and** $\bar{t}_s^* > \max\{t_i^*, \bar{t}_i^*\}$ **then**

        $d_i^* \leftarrow \bar{d}_i; \bar{t}_i^* \leftarrow \bar{t}_s^*$

    else if $t_s^* \geq \bar{t}_s^*$ **and** $\bar{t}_s^* > \max\{t_i^*, \bar{t}_i^*\}$ **then** $d_i^* \leftarrow \bar{d}_i; t_i^* \leftarrow t_s^*$

    end if

    SAMPLE()

    if x_i is a leaf **then**

        Send BACKTRACK$(x_i, \Delta_i, \bar{\Delta}_i)$ to P_i

    end if

end if

Procedure 4 BACKTRACK($x_s, \Delta_s, \bar{\Delta}_s$) [209]

1: $\Delta_i \leftarrow \Delta_i + \Delta_s; \bar{\Delta}_i \leftarrow \bar{\Delta}_i + \bar{\Delta}_s$
2: **if** Received BACKTRACK from all children in this iteration **then**
3: Send BACKTRACK($x_i, \Delta_i, \bar{\Delta}_i$) to P_i
4: **if** x_i is root **then**
5: $\bar{\Omega} \leftarrow \Omega + \bar{\Delta}_i; \Omega \leftarrow \Omega + \Delta_i$
6: **if** $\Omega \geq \bar{\Omega}$ **and** $\Omega > \Omega^*$ **then**
7: $\Omega^* \leftarrow \Omega; d_i^* \leftarrow d_i; t_i^* \leftarrow t_i$
8: **else if** $\bar{\Omega} \geq \Omega$ **and** $\bar{\Omega} > \Omega^*$ **then**
9: $\Omega^* \leftarrow \bar{\Omega}; d_i^* \leftarrow \bar{d}_i; \bar{t}_i^* \leftarrow t_i$
10: **end if**
11: SAMPLE()
12: **end if**
13: **end if**

12.1.5.2.2 Algorithm Table 12.2 presents the values each agent i maintains in SD-Gibbs. Procedure 2 describes the complete sampling function. For completeness, we present the SD-Gibbs algorithm in Algorithm 17. The algorithm can be summarized as:

1. **Initializing.** The algorithm starts with constructing the pseudo-tree and each agent initializing each of their variables, from Table 12.2, to their default values. The root then starts the sampling, as described in Procedure 2, and sends the VALUE message (line 6) to each neighbor.

2. **Sampling.** Upon receiving a VALUE message, each agent invokes Procedure 3. In it, an agent i first updates its current contexts, C_i and $\bar{C}_i$, with the sender's values. If the message is from agent i's parents, then the agent itself samples, i.e., executes

Procedure 2. This *sampling stage* continues until all the leaf agents have executed Procedure 2.

3. **Backtracking.** Each leaf agent j then sends a BACKTRACK message to its parent comprising $x_j, \Delta_j,$ and $\bar{\Delta}_j$. As described in Procedure 4, when a parent receives such a message, it sends a BACKTRACK message to its parent. The process continues until the root receives the message, which concludes one iteration.

4. **Selecting the Solution.** Each agent i uses its current (Δ_i) and current best-response ($\bar{\Delta}_i$) local utility differences to reach a solution. We refer to these differences as *relative* utilities. Upon receiving a BACKTRACK message, agent i adds the delta variables of its children to its own. Consequently, these variables for the root agent quantify the global relative utility. Based on this, at the end of an iteration, the root decides to keep or throw away the current solution (Procedure 4, line 4).

As aforementioned, in this work, we focus on *constraint privacy* to ensure the privacy of agent preferences. From Faltings et al. [100], constraint privacy states that no agent must be able to discover the nature of constraint (i.e., the utilities) that does not involve a variable it owns. Since absolute privacy is not an achievable goal [91], we formalise constraint privacy in terms of (ϵ, δ)-DP [93].

12.1.5.3 Differential Privacy (DP)

As introduced in Chapter 3.1.3, Differential Privacy (DP) [92, 93] is a popular privacy notion that aims to provide a statistical guarantee against a database that the inclusion or exclusion of any single entry will not significantly impact the results of the statistical analysis. DP is normally defined for adjacent databases. However, in this instance, we want to protect privacy against external adversaries and curious fellow agents, i.e., agents looking to decipher sensitive information. One may note that when the set of variables and agents involved is *globally* known, there are more efficient techniques for distributed optimization

using a central coordinator and stochastic gradient descent. Researchers have developed DP techniques for this context as well [140]. While such algorithms are well-suited for contexts such as federated learning, where the model parameters are common knowledge, in meeting-scheduling, they would leak the information of who is meeting with whom, which is usually the most sensitive information. Therefore, we focus on algorithms where each participant has *local* information, i.e., only knows about agents it shares constraints with and nothing about the rest of the problem. Formally, we want our algorithm for any two utility functions (vectors in $\mathbb{R}^p$) to satisfy the following definition from [93],

Definition 12.4: Local Differential Privacy

A randomized mechanism $\mathcal{M} : \mathcal{F} \to \mathcal{R}$ with domain $\mathcal{F}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-DP if for any two inputs $F, F' \in \mathcal{F}$ and for any subset of outputs $O \subseteq \mathcal{R}$ we have,

$$\Pr[\mathcal{M}(F) \in O] \leq e^{\epsilon} \Pr[\mathcal{M}(F') \in O] + \delta \tag{12.3}$$

We now present another way of interpreting the local-DP definition defined in Eq. 12.3. For this, we define the privacy loss random variable (PRVL) L as follows:

$$L^{o}_{\mathcal{M}(F)||\mathcal{M}(F')} = \ln\left(\frac{\Pr[\mathcal{M}(F) = o]}{\Pr[\mathcal{M}(F') = o]}\right) \tag{12.4}$$

12.2 Privacy Leakage in SD-Gibbs

In SD-Gibbs, constraint privacy is compromised in the following two ways:

By sampling. Each variable value in SD-Gibbs is sampled according to agent i's utility F_{ij}. As values with *more utility are more likely to be drawn*, SD-Gibbs leaks sensitive information about these utility functions. Fortunately, this stage can be secured by simply making distributions more similar across agents (Section 12.3.2).

By relative utility Δ. Every leaf agent j in the pseudo-tree sends its Δ_j and $\bar{\Delta}_j$ to its parent i. The parent agent adds the values to its Δ_i and $\bar{\Delta}_i$, respectively, and passes them on up

the tree. The process continues until the values reach the root. Thus, any intermediate agents or an adversary observing Δ can learn something about j's utility even if sampling is private. E.g., suppose a particular assignment has a high utility for agent j but is low for others (and it is known). In that case, an intermediate agent will learn about agent j even from the aggregated utility.

These privacy leaks follow by observing what critical information gets transferred by each agent i in Algorithm 17. We ignore t^* and $\bar{t}^*$ because these are simply functions of utility, i.e., will be private by post-processing property once the utility is private. We next illustrate the same.

12.2.0.0.1 Illustrating Privacy Leak in SD-Gibbs on Figure 12.1 Consider the execution of SD-Gibbs (Algorithm 17) on the example provided in Figure 12.1. Recall that we aim to preserve constraint privacy in DCOP. The agent x_1 must not learn anything about the nature of the constraint between the agents x_2 and x_3.

Upon execution of Algorithm 17 with each variable initialized with 0, the initial Gibbs distribution for x_2 and x_3 (from Eq. (12.2)) takes the form $[0.88, 0.12]$ and $[0.88, 0.45]$, respectively. As x_1 is the neighboring agent for both x_2 and x_3, it will be aware of their current and best assignments. Moreover, as the number of iterations increases, x_1 observes that its Gibbs distribution converges to $[0.002, 0.998]$. Further, one can also see that x_2's and x_3's Gibbs distribution changes to $[0.002, 0.998]$. That is, x_1 can observe that x_2 and x_3 prefer assignment 1 (with high probability) given its context. Based on the assignment sampling, x_1 already has a qualitative idea of the nature of the constraint x_2—x_3. Since it knows it prefers assignment 1, it can estimate the constraint x_2—x_3 will be such that it is less or equal in value – for any assignment other than their current $x_2 = x_3 = 1$. If not, x_2 and x_3 would have changed their assignments to grab the additional utility.

To get a quantitative estimate, x_1 can observe the relative utilities. In our example, as there are only three constraints, x_1 can use the information on the probable assignments of x_2 and x_3, 1 and 1, and the final utility 6 to derive the value $F_{23}(x_2 = 1, x_3 =$

Symbol	Definition
τ	Sensitivity
c	Clipping constant
$\mathbb{P}_i$	SD-Gibbs probability distribution for Agent i
p_i	Soft-max function on $\mathbb{P}_i$ for Agent i
γ	Soft-max temperature parameter
q	Sub-sampling probability
T	Number of iterations in P-Gibbs
$\mathcal{N}(0, \sigma^2)$	Gaussian distribution with mean zero and variance σ^2
(ϵ_s, δ)	Privacy parameters of the Sampling stage
(ϵ_n, δ)	Privacy parameters for the Relative utilities

Table 12.3: Notations

$1) = 6 - F_{12}(x_1 = 1, x_2 = 1) - F_{13}(x_1 = 1, x_3 = 1) = 6 - 2 - 2 = 2$. Therefore, x_1 can learn information regarding the constraint F_{23} violating constraint privacy. Applying similar qualitative knowledge of the assignment on each iteration's Δs can potentially leak information about the entire utility function.

With these as a backdrop, we now build upon SD-Gibbs to formally present our novel, scalable algorithm for DCOPs that preserve constraint privacy, namely P-Gibbs.

12.3 P-Gibbs: Preserving Constraint Privacy in DCOP with SD-Gibbs

In general, for DP, we need to ensure full support of the outcome distribution. If $\Pr[\mathcal{M}(D') = o] = 0$ for some output o, the privacy loss L incurred is infinite, and one cannot bound ϵ. For the specific context of ensuring constraint privacy in SD-Gibbs, this

implies that all agents must have the same domain for their variables and non-zero utility for each value within the domain.[2] In other words, $D_1 = D_2 = \ldots = D_p$ and $|F_{ij}(\cdot,\cdot)| > 0, \forall i$. Without these, the probability distributions defined in Eq. (12.2) may not be bounded as any pair of agents i and j $(i \neq j)$ may have $D_i \neq D_j$. As a result, the ϵ with respect to constraint privacy (Definition 12.3) will not be defined. Formally, consider the following claim.

Claim 12.1. *With respect to constraint privacy, SD-Gibbs (Algorithm 17) is non-private, i.e., the privacy loss variable L is not defined for SD-Gibbs.*

Proof. Consider any two agents $i, j \in \mathcal{A}$ s.t. $(i \neq j)$ and $D_i \neq D_j$. W.l.o.g., let $D_i = D_j + \{d\}$. From Definition 12.3, the privacy loss variable L (Eq. (12.4)) can be written as,

$$L^d_{\mathbb{P}_i||\mathbb{P}_j} = \ln\left(\frac{\mathbb{P}_i(x_i = d)}{\mathbb{P}_j(x_j = d)}\right) = \ln\left(\frac{v}{0}\right) \text{ where } v > 0,$$

as $d \in D_i$ while $d \notin D_j$. Thus, the privacy loss variable, L, is not defined for SD-Gibbs. □

Claim 12.1 implies that the privacy budget, ϵ in $(\epsilon, 0)$-DP, is also not-defined for SD-Gibbs. Consequently, to provide meaningful privacy guarantees for constraint privacy in DCOPs, we present P-Gibbs (Chapter 12.3). In it, we first use soft-max with temperature to bound the SD-Gibbs distributions (Chapter 12.3.2). The resulting bound only depends on the temperature parameter and does not leak any agent's sensitive information. Then, we "clip" the relative utilities to further bound the sensitivity (Chapter 12.3.3). Lastly, to reduce the growth of ϵ, we randomly select a subset of agents to sample new values at each iteration. We then provide a refined privacy analysis for the resulting (ϵ, δ)-DP (Theorem 12.1). Table 12.2 and Table 12.3 provide a reference point for the notations used in this section.

[2] If any agent has a zero utility for some value, then all agents must have zero utility, and w.l.o.g., we can simply exclude such values from all domains.

Procedure 5 P-Gibbs SAMPLE()

$t_i \leftarrow t_i + 1$; $\hat{d}_i \leftarrow d_i$

$\beta \sim \text{Uniform}(0, 1)$

// subsampling

if $\beta \in (0, q]$ **then**

    $\mathbb{P}_i(\mathbf{x}_i) \leftarrow$ from $Eq.$ (12.2)

    // Bounding SD-Gibbs distribution with Soft-max

    $p_i(\mathbf{x}_i, \gamma) \leftarrow$ from $Eq.$ (12.5)

    $d_i \leftarrow$ Sample based on $p_i(\mathbf{x}_i, \gamma)$

else

    $d_i \leftarrow d_i$

end if

$\bar{d}_i \leftarrow \text{argmax}_{d'_i \in D_i} \sum_{\langle x_j, \bar{d}_j \rangle \in \bar{C}_i} F_{ij}(d'_i, \bar{d}_j)$

$\Delta_i \leftarrow \sum_{\langle x_j, d_j \rangle \in C_i} \Big[F_{ij}(d_i, d_j) - F_{ij}(\hat{d}_i, d_j) \Big]$

$\bar{\Delta}_i \leftarrow \sum_{\langle x_j, \bar{d}_j \rangle \in \bar{C}_i} \Big[F_{ij}(\bar{d}_i, \bar{d}_j) - F_{ij}(\hat{d}_i, \bar{d}_j) \Big]$

// Clipping

if $|\Delta_i| > c$ **then** $\Delta_i = (\Delta_i \geq 0)\ ?\ c : -c$

if $\left|\bar{\Delta}_i\right| > c$ **then** $\bar{\Delta}_i = (\bar{\Delta}_i \geq 0)\ ?\ c : -c$

// Perturbing utilities with Gaussian noise

$\Delta_i \leftarrow \Delta_i + \mathcal{N}(0, \tau^2\sigma^2)$

$\bar{\Delta}_i \leftarrow \bar{\Delta}_i + \mathcal{N}(0, \tau^2\sigma^2)$

Send VALUE$(x_i, d_i, \bar{d}_i, t^*_i, \bar{t}^*_i)$ to each $x_j \in N_i$

12.3.1 P-Gibbs: Algorithm

Recall that privacy leak in SD-Gibbs is due to the qualitative and quantitative information loss due to communicating the sampled value d and the relative utilities Δ, respectively

(Section 12.2). Our privacy variant, P-Gibbs, preserves this information loss through its novel sampling procedure. We formally provide the sampling in P-Gibbs with Procedure 5. The differences, compared to SD-Gibbs' sampling procedure, are summarized as follows:

1. To preserve constraint privacy loss due to sampling (Procedure 5, Lines 3-9):
   - P-Gibbs uses soft-max function over SD-Gibbs distributions for sampling d_i's, $\forall i$. As shown later in Proposition 12.1, this bounds any two agent distributions in SD-Gibbs, resulting in finite privacy loss.
   - P-Gibbs randomly chooses subsets of agents to sample new values in each iteration. More specifically, in every iteration, each agent i samples a new value d_i with probability q, or uses previous values with probability $1 - q$.
2. To preserve constraint privacy loss due to relative utilities (Procedure 5, Lines 13-16):
   - In P-Gibbs, we sanitize the relative utilities with calibrated *Gaussian Noise*.
   - To bound the sensitivity (see Section 12.3.3), we "clip" the relative utilities by $\pm c$, where c is the *clipping constant* (Procedure 5, Lines 16 and 17).

We next formally show that soft-max bounds the SD-Gibbs probability distributions. We then provide a formal analysis for privacy loss due to sampling.

12.3.2 P-Gibbs: Bounding Sampling Divergence with Soft-max

Towards achieving bounded sampling divergence without compromising on constraint privacy itself, we propose to apply *soft-max* to sampling distributions. Let p_i be the soft-max distribution with *temperature* parameter as γ, i.e.,

$$p_i(\mathbf{x}_i, \gamma) = \left\{ \frac{\exp(\mathbb{P}_i(x_i = d_k)/\gamma)}{\sum_{d_l \in D} \exp(\mathbb{P}_i(x_i = d_l)/\gamma)}; \forall d_k \in D \right\} \tag{12.5}$$

Firstly, observe that $p_i(\cdot, \gamma)$, for a finite γ, has full support of the outcome determination. That is, $p_i(x_i, \gamma) > 0$ s.t. $x_i = d_k, \forall d_k \in D$. This observation ensures that the scenario of

an unbounded privacy loss due to $p_i(x_i, \gamma) = 0$, described earlier with Claim 12.1, will not occur for P-Gibbs.

Secondly, to also ensure that ϵ is finite, we require that the bound $\frac{p_i(\cdot)}{p_j(\cdot)}$ for any distinct pair i and j is bounded. To this end, the following claim shows that the ratio of the resulting soft-max probabilities, $p_i(\cdot)$ and $p_j(\cdot)$ for any two agents i and j, is *bounded* by $2/\gamma$. The proof uses the fact that $D = D_i = D_j$ and $1/e \leq \exp(p_i(x) - p_j(x)) \leq e$.

Proposition 12.1. *For two probability distributions using soft-max, p_i and p_j defined by Eq. (12.5), we have, $\forall i, j$, $\forall d \in D$ and $\forall D$, s.t. $|D| > 1, \gamma \geq 1$*

$$\ln \left[\frac{p_i(x_i = d, \gamma)}{p_j(x_j = d, \gamma)} \right] \leq \frac{2}{\gamma}$$

Proof. Because p_i and p_j are soft-max distributions, we have,

$$p_i(\mathbf{x}_i, \gamma) = \left\{ \frac{\exp(\mathbb{P}_i(x_i = d_k)/\gamma)}{\sum_{d_l \in D} \exp(\mathbb{P}_i(x_i = d_l)/\gamma)} \; ; \; \forall d_k \in D \right\},$$

$$p_j(\mathbf{x}_j, \gamma) = \left\{ \frac{\exp(\mathbb{P}_j(x_j = d_k)/\gamma)}{\sum_{d_l \in D} \exp(\mathbb{P}_j(x_j = d_l)/\gamma)} \; ; \; \forall d_k \in D \right\}.$$

Now, recall that $\mathbb{P}_i$ and $\mathbb{P}_j$ are the probability distributions through SD-Gibbs sampling. With this, observe the following:

$$\ln \left[\frac{p_i(x_i = d, \cdot)}{p_j(x_j = d, \cdot)} \right] \leq \ln \left[\frac{\frac{\exp(\mathbb{P}_i(x_i = d)/\gamma)}{\sum_{d_l \in D} \exp(\mathbb{P}_i(x_i = d_l)/\gamma)}}{\frac{\exp(\mathbb{P}_j(x_j = d)/\gamma)}{\sum_{d_l \in D} \exp(\mathbb{P}_j(x_j = d_l)/\gamma)}} \right] \leq \ln \left[\frac{\exp(1/\gamma(\mathbb{P}_i - \mathbb{P}_j)))}{N_1/N_2} \right] \tag{12.6}$$

Here, $N_1 = \sum_{d_l \in D} \exp(\mathbb{P}_i(x_i = d_l)/\gamma)$ and $N_2 = \sum_{d_l \in D} \exp(\mathbb{P}_j(x_j = d_l)/\gamma)$. Now, in Eq. (12.6) observe that both the numerator and denominator in r.h.s of Eq. (12.6) are positive. Further, as $\ln(x)$ is an increasing function x, this implies that r.h.s of Eq. (12.6) is maximum when the numerator is maximum, and the denominator is minimum. Thus

the difference, $\mathbb{P}_i(x_i = d) - \mathbb{P}_j(x_j = d)$, can be at-most 1. Therefore, numerator in r.h.s of Eq. (12.6) is at-most $\exp(1/\gamma) = e^{1/\gamma}$.

The denominator in r.h.s of Eq. (12.6) is minimum when N_1 is minimum and N_2 is maximum. Note that, N_1 is minimum when $\mathbb{P}_i(x_i = d) = 0, \forall d$, i.e., minimum $N_1 = |D|$. But, N_2 is maximum when $\mathbb{P}_j(x_j = d) = 1, \forall d$, i.e., maximum $N_2 = |D| \cdot e^{1/\gamma}$. Using these values in Eq. (12.6) completes the claim. □

12.3.2.1 Effect of Soft-max

We illustrate the effect of soft-max on the SD-Gibbs sampling distribution with the following example. Let $D_j = \{d_1, d_2, d_3\}, \forall j$ such that $\mathbb{P}_i = [0.8, 0.15, 0.05]$. Observe that the distribution is such that the probability of sampling d_1 is significantly more than others. Now, the corresponding soft-max distributions, from Eq. (12.5), will be: $p(\cdot, \gamma = 1) = [0.50, 0.26, 0.24]$, $p(\cdot, \gamma = 2) = [0.41, 0.30, 0.29]$, and $p(\cdot, \gamma = 10) = [0.35, 0.33, 0.32]$. That is, the soft-max distribution is more *uniform* than the original distribution. This implies that the maximum ratio of the probabilities will be smaller. That is, an adversary will be more indifferent towards the domain values while sampling. For e.g., d_1 and d_2 in $p(\cdot, \gamma = 10)$ compared to in $p(\cdot, \gamma = 1)$.

Observe that the bound provided in Proposition 12.1 *does not* depend on an agent's sensitive information. This implies that the bound does not encode (and reveal) any sensitive information. Thus, we conclude that the bound provided in Proposition 12.1 is desirable and hence use it to construct the sampling distribution in P-Gibbs.

12.3.2.2 Privacy Guarantees for Sampling in P-Gibbs

We first calculate the privacy parameters of the sampling stage, denoted by ϵ_s and δ, in P-Gibbs. We use an extension of the moments accountant method [3] for non-Gaussian

mechanisms. Following derivations by [283],

$$\Pr[L \geq \epsilon_s] \leq \max_{p_i, p_j} \; e^{\lambda \mathcal{D}_{\lambda+1}[p_i||p_j] - \lambda \epsilon_s} \tag{12.7}$$

Here, L is the privacy loss between any two agents and $\mathcal{D}_\lambda(p_i||p_j) = \frac{1}{\lambda-1} \log \mathbb{E}_{d \sim p_j} \left(\frac{p_i(d)}{p_j(d)} \right)^\lambda$ is Rényi divergence [239] of order $\lambda \in \mathbb{N}^+, \lambda > 1$. From [283], the choice of the hyperparameter λ is arbitrary since the bound holds for any feasible value of λ. Note that the value of λ determines how tight the bound is.

Also from [283], we borrow the notion of *privacy cost* $c_t(\lambda)$. By trivial manipulation, for each iteration t,

$$c_t(\lambda) = \max_{i,j} \; \lambda \mathcal{D}_{\lambda+1} \left[p_i(d) || p_j(d) \right] \leq \lambda \cdot 2/\gamma, \tag{12.8}$$

where Eq. (12.8) is due to monotonicity $\mathcal{D}_\lambda(P||Q) \leq \mathcal{D}_{\lambda+1}(P||Q) \leq \mathcal{D}_\infty(P||Q), \; \forall \lambda \geq 0$.

12.3.2.2.1 Subsampling The privacy cost c_t in Eq. (12.8) can be further reduced by subsampling agents with probability $q << 1$. Balle *et al.* [21] show that the privacy guarantees of a DP mechanism can be amplified by applying the mechanism to a small random subsample of records of any database.

We formulate the following result by reproducing the steps of the sampled Gaussian mechanism analysis by [283] for our mechanism and classical DP.

Theorem 12.1. *Privacy cost $c_t(\lambda)$ at iteration t of a sampling stage of P-Gibbs, with agent subsampling probability q, is*

$$c_t^{(s)}(\lambda) = \ln \mathbb{E}_{k \sim B(\lambda+1, q)} \left[e^{k \cdot 2/\gamma} \right], \tag{12.9}$$

where $B(\lambda, q)$ is the binomial distribution with λ experiments and probability of success as q, $\lambda \in \mathbb{N}$.

Proof. The result follows by substituting $2/\gamma$ in place of the ratio of normality distributions in [283, Theorem 3]. □

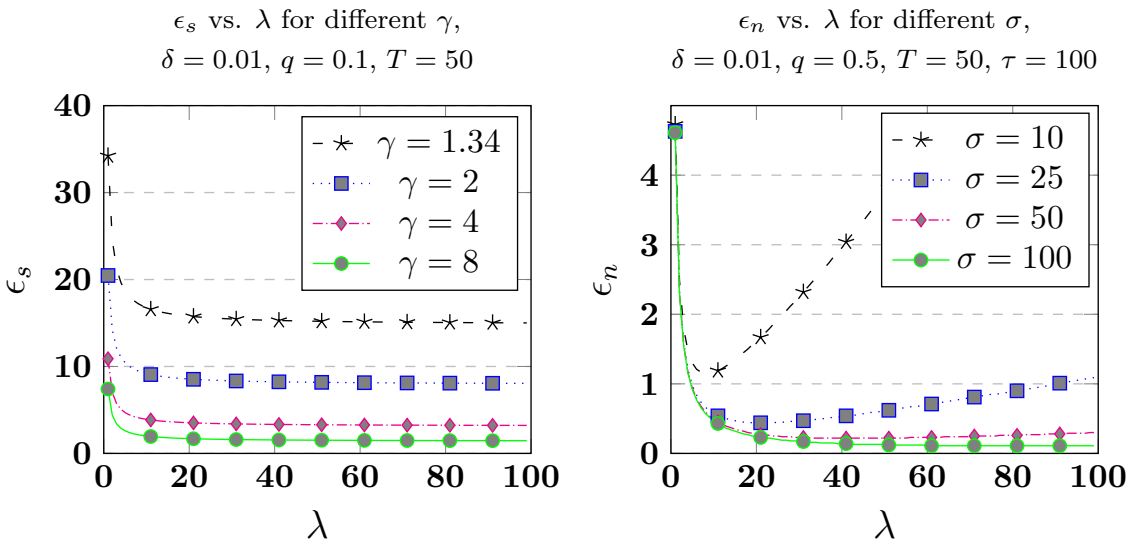


Figure 12.2: Variation of ϵ_s and ϵ_n vs. λ

Unlike the analysis in [283, Theorem 3], we do not have $c_t^L(\lambda)$ and $c_t^R(\lambda)$, as well as expectation over the data. This is because we compute the conventional differential privacy bounds instead of Bayesian DP and, thus, directly use the worst-case ratio, i.e., $2/\gamma$. Finally, merging the results, we can compute ϵ_s, δ across multiple iterations as

$$\left.\begin{aligned} \ln \delta &\leq \textstyle\sum_{t=1}^{T} c_t^{(s)}(\lambda) - \lambda \epsilon_s \\ \epsilon_s &\leq \tfrac{1}{\lambda}\left(\textstyle\sum_{t=1}^{T} c_t^{(s)}(\lambda) - \ln \delta\right) \end{aligned}\right\} \tag{12.10}$$

Figure 12.2 shows the variation of ϵ_s for different values of λ and γ, with the sampling probability $q = 0.1$. We observe that λ clearly affects the final ϵ_s value, and one should ideally minimize the bound over λ.

12.3.2.3 P-Gibbs$_\infty$: An Extreme Case

We presented P-Gibbs, which uses a soft-max with temperature function to bound the sampling divergence, thereby bounding the privacy loss incurred by sampling. We smooth the distribution using soft-max's temperature parameter to reduce further the information

encoded in SD-Gibbs sampling. We then use Theorem 12.1 to quantify privacy parameters ϵ_s and δ.

From Proposition 12.1, observe that the temperature parameter in P-Gibbs may be tuned to decrease the overall privacy budget for sampling, i.e., ϵ_s. An "extreme" case occurs when $\gamma \to \infty$. For this, we have $p_i = p_j$, which implies that $\epsilon_s \to 0$. Thus, increasing γ leads to P-Gibbs sampling distribution mimicking a *uniform* distribution, as more information on SD-Gibbs sampling distribution is lost. To distinguish this extreme case, we refer to P-Gibbs with $\gamma \to \infty$ as P-Gibbs$_\infty$.

12.3.3 P-Gibbs: Privacy of Relative Utilities (Δ)

In the previous subsection, we deal with the privacy loss occurring due to sampling in P-Gibbs. As aforementioned, the values Δ and $\bar{\Delta}$ also leak information about agents' constraints. In order to achieve DP for these Δ's, we need to bound its *sensitivity*. Sensitivity is defined as the maximum possible change in the output of a function we seek to make privacy-preserving. Formally,

Definition 12.5: Sensitivity (τ)

It is the maximum absolute difference between any two relative utility values Δ and Δ', i.e.,

$$\tau = \max_{\Delta, \Delta'} \left| \Delta - \Delta' \right| \tag{12.11}$$

As we clip the relative utilities with a constant c (see Procedure 5, Line 16-17), trivially from Eq. 12.11, $\tau = 2 \cdot c$.

Next, we must sanitize the relative utilities to preserve privacy fully. We achieve this through the *Gaussian noise mechanism* [93] defined as

$$\mathcal{M}_G(\Delta) \triangleq \Delta + Y_i,$$

where $Y_i \sim \mathcal{N}(0, \tau^2\sigma^2)$, τ is the sensitivity and σ is the noise parameter.

Algorithm	(ϵ_s, δ)	(ϵ_n, δ)	$(\epsilon = \epsilon_s + \epsilon_n, \delta)$ **for** T **iterations**
P-Gibbs	$(2/\gamma, 0)$	$(\frac{\tau}{\sigma}\sqrt{2\ln\frac{1.25}{\delta}}, \delta)$	$\left(\frac{T}{\lambda}c_t^{(s)}(\lambda) + \frac{T}{\lambda}c_t^{(n)}(\lambda) - \frac{1}{\lambda}\ln\delta,\ \delta\right)$
P-Gibbs$_\infty$	$(0, 0)$	$(\frac{\tau}{\sigma}\sqrt{2\ln\frac{1.25}{\delta}}, \delta)$	$\left(\frac{T}{\lambda}c_t^{(n)}(\lambda) - \frac{1}{\lambda}\ln\delta,\ \delta\right)$

Table 12.4: Per-iteration and final (ϵ, δ) bounds.

Privacy parameters for the relative utility Δ, denoted by ϵ_n and δ, can be computed either using the basic composition along with [93, Theorem A.1] or the moments accountant [3]. The latter can be unified with the accounting for the sampling stage by using:

$$c_t^{(n)}(\lambda) = \ln \mathbb{E}_{k\sim B(\lambda+1,q)}\left[e^{k\mathcal{D}_{\lambda+1}[\mathcal{N}(0,\tau^2\sigma^2)||\mathcal{N}(\tau,\tau^2\sigma^2)]}\right]. \tag{12.12}$$

Figure 12.2 shows the variation of ϵ_n for different values of λ and τ, with the sampling probability $q = 0.1$ and $\sigma = 1$. We observe that λ has a clear effect on the final ϵ_n value as well, although the change is virtually the same for $\tau = 10, 25$ and 50. The trend is similar to the one observed in Figure 12.2, i.e., ϵ_n decreases as λ increases. However, the decrease is not smooth when $\tau = 5$, which sees a sharp change in ϵ_n as λ increases. This change is similar to what is observed in [283, Figure 5], suggesting that one should be careful while deciding on the value of λ.

NOTE. We provide the formal sampling procedure comprising the privacy techniques discussed above with Procedure 5. The rest of the procedures are the same as provided with Algorithm 17. Table 12.4 summarises expressions for per-iteration and total ϵ values for P-Gibbs and P-Gibbs$_\infty$.

12.3.3.0.1 Collusion Resistance Recall that in the private DCOP literature, an algorithm is collusion resistant if *no* subset of agents can collude to gain additional information about the remaining agents. We remark that P-Gibbs trivially satisfies collusion-resistance. This is because each agent in P-Gibbs locally adds noise or randomness to its utility and

assignment sampling. Due to the post-processing property of DP, no subset of the agent can infer any additional information outside of the (ϵ, δ)-DP guarantee.

12.4 Experiments

We now empirically evaluate the performance of our novel algorithms, P-Gibbs w.r.t. to SD-Gibbs. This section first describes our experimental setup and benchmark problems (Section 12.4.1). Next, in Section 12.4.2, we present the results for P-Gibbs' performance in terms of solution w.r.t. SD-Gibbs. Section 12.4.3 presents criteria to empirically explain the privacy protection in P-Gibbs with regard to changes in the privacy budget. Section 12.4.4 provides a general discussion of the results presented and summarizes the advantages of our DP-based approach compared to the existing cryptographic approach for privacy-preserving DCOP algorithms.

12.4.1 Benchmark Problems and Experimental Setup

We now describe the DCOP benchmark problems and illustrate our experimental setup.

12.4.1.1 Benchmark Problems

We construct the following problem instances to test our novel differentially private variant, P-Gibbs. These are standard benchmarks in the DCOP literature.

Ising [50]. We generate 20 sample Ising problems. For this, the constrained graph is a rectangular grid with each agent/variable connected to its four nearest neighbors. The problems are such that the number of agents/variables lie between $[10, 20)$. Each agent's domain is binary, i.e., $D_i = \{0, 1\}, \forall i$. The constraints are of two types: (i) binary constraints whose strength is sampled from $\mathcal{U}[\beta, \beta]$ where $\beta \in [1, 10)$ and (ii) unary constraints whose strength is sampled from $\mathcal{U}[-\rho, \rho]$ where $\rho \in [0.05, 0.9)$. Ising is a *minimization* problem.

Graph-Coloring (GC). We generate 20 sample graph-coloring problems. The problems are such that the number of agents/variables lies between $[30, 100)$ and agents' domain size between $[10, 20)$. Each constraint is a random integer taken from $(0, 10)$. Graph-coloring is a *minimization* problem.

Meeting-Scheduling (MS) [181]. We generate 20 sample meeting-scheduling problems. The problems are such that the number of agents and variables lies between $[1, 75)$ with the number of slots, i.e., the domain for each agent randomly chosen from $[30, 100)$. Each constraint is a random integer taken from $(0, 100)$, while each meeting may randomly occupy $[1, 5]$ slots. Meeting-scheduling is a *maximization* problem.

While meeting-scheduling is a concrete problem [181], even abstract problems like graph-coloring can model real-world scenarios. E.g., Radio Frequency or Wireless Network Assignment can be modeled as a graph-coloring problem [2]. The Ising model is also a widely used benchmark in statistical physics [50].

NOTE. Importantly, we perform our experiments on much larger problems than earlier complete algorithms (e.g., [100]) can handle[3]. Concerning the infeasibility of a DCOP solution, note that incomplete (or random) algorithms like MGM, DUCT, and SD-Gibbs do not aim to solve problems with hard constraints. A hard constraint will leak vital information about the constraints, and a differentially private solution will not work in such a setting. Like [209], we focus on soft constraints; thus, infeasible solutions will not occur.

12.4.1.2 Experimental Setup

Our experimental setup is as follows.

Implementation. *pyDCOP* [247] is a *Python* module that provides implementations of many DCOP algorithms (DSA, MGM, MaxSum, DPOP, etc.). It also allows easy

[3]For e.g., DPOP, a non-private, complete algorithm timed-out after 24 hours of computing (i) an Ising instance with 10 variables, (ii) a graph-coloring instance with 12 variables and $|D| = 8$, (iii) a meeting-scheduling instance with 25 variables and $|D| = 20$.

implementation of one's DCOP algorithm by providing all the required infrastructure: agents, messaging system, and metrics collection, among others. We use pyDCOP's public implementation of the SD-Gibbs algorithm to run our experiments. In addition, we also implement P-Gibbs.

Generating Test-cases. pyDCOP allows for generating random test-cases for various problems through its command line's *generate* option. With this, we generate instances for our benchmark problems, i.e., Ising, graph-coloring, and meeting-scheduling. We test the performance of our algorithms across 20 such randomly generated problems.

Method. We consider the utility given by SD-Gibbs' solution as our baseline. Further, these algorithms, i.e., SD-Gibbs and P-Gibbs, are random algorithms. Hence, we run each benchmark problem instance 10 times for a fair comparison and use the subsequent average utility for our results.

The complete codebase is available at: `github.com/magnetar-iiith/PGiBBS`.

12.4.1.2.1 Performance Measure

We measure P-Gibbs' performance w.r.t. SD-Gibbs using the following performance measure.

Definition 12.6: Solution Quality (SQ)

Solution quality $\text{SQ}_{\mathcal{A}}$ of an algorithm $\mathcal{A}$ is defined as

$$\text{SQ}_{\mathcal{A}} = \begin{cases} \frac{U_S}{U_{\mathcal{A}}} & \text{for minimization} \\ \frac{U_{\mathcal{A}}}{U_S} & \text{for maximization} \end{cases}$$

for utility of $\mathcal{A}$ as $U_{\mathcal{A}}$ and SD-Gibbs as U_S.

With SQ, we normalize P-Gibbs' utility in the context of SD-Gibbs. SQ ≈ 1 indicates that utility does not deteriorate than SD-Gibbs. On the other hand, SQ ≈ 0 means little

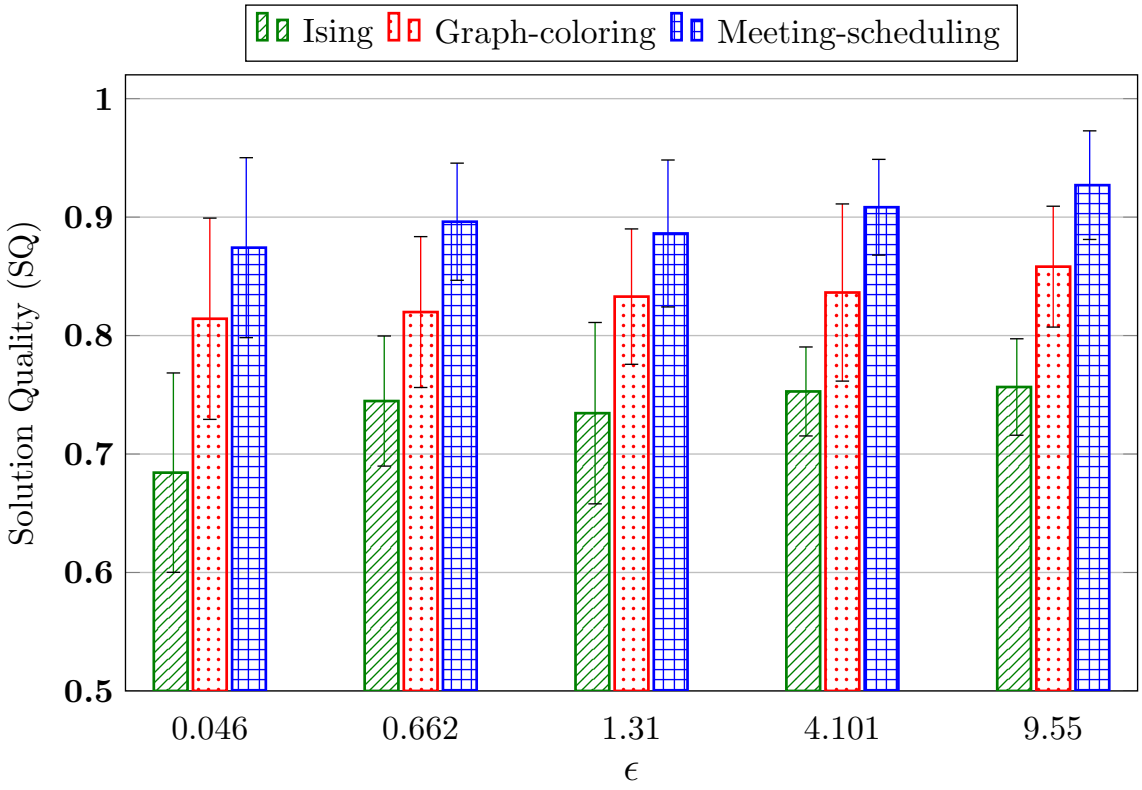


Figure 12.3: The average and standard deviation of P-Gibbs' Solution Quality (SQ) for different privacy budgets. Note that, the case with $\epsilon = 0.046$ corresponds to P-Gibbs$_\infty$ as $\gamma = \infty$.

utility as compared to the SD-Gibbs solution. It is possible that SQ > 1 due to randomness and privacy noise acting as simulated annealing[4] [159].

[4]We remark that this behavior is different from the Distributed Simulated Annealing (DSAN) algorithm for DCOPs [14, 55]. DSAN is an iterative optimization algorithm with a *temperature* parameter that aims to control the likelihood of accepting worse solutions. DSAN consists of an annealing schedule that determines the change in the temperature parameter over time. As the parameter decreases, DSAN becomes more selective and explores the solution space more effectively. Instead of selecting the next assignment through a specific, utility-based distribution like in SD-Gibbs (Eq. 12.2), in DSAN, an agent randomly chooses its next assignment. E.g., by uniform sampling or by swapping values with neighboring agents. DSAN is neither complete nor private.

12.4.2 Results

In this subsection, we present (i) the overall trend for change in P-Gibbs' SQ vs. ϵ, and (ii) the effect of hyperparameters (σ, γ, q) and the problem size on P-Gibbs' SQ.

12.4.2.1 General Trends for Solution Quality

We now provide general trends w.r.t. ϵs and SQs. More specifically, we focus on the change in the SQ with the change in ϵ (aka privacy budget).

(ϵ, δ)-bounds. Throughout these experiments, we choose $\delta = 10^{-2}$, $T = 50, \tau = 50$ and λs as 100. As standard, our choice of δ is such that $\delta < 1/m$ [93]. We calculate ϵ using different permutations of $\gamma \in \{4, 8, 20, \infty\}, q \in \{0.1, 0.2\},$ and $\sigma \in \{7, 10, 25, 1000\}$. With these, we obtain the following ϵ values: (i) $\epsilon = 0.046$ where $\sigma = 1000, \gamma = \infty$ and $q = 0.1$, (ii) $\epsilon = 0.662$ where $\sigma = 25, \gamma = 20$ and $q = 0.1$, (iii) $\epsilon = 1.31$ where $\sigma = 25, \gamma = 20$ and $q = 0.2$, (iv) $\epsilon = 4.101$ where $\sigma = 10, \gamma = 8$ and $q = 0.2$, (v) $\epsilon = 9.55$ where $\sigma = 7, \gamma = 4$ and $q = 0.2$. Note that the case with $\epsilon = 0.046$ corresponds to P-Gibbs$_\infty$ as $\gamma = \infty$.

12.4.2.1.1 Results Figure 12.3 presents the overall change in the SQ concerning an increase in ϵ. We plot SQ scores averaged across all problems. For all three benchmarks, the average SQ improves between $\epsilon \in [0.046, 9.55]$. This behavior is expected as greater ϵs imply an increase in the subsampling probability and a decrease in the noise added (σ). The increase in the probability of subsampling allows an agent to explore more values in its domain. That is an increase in the chance of encountering better assignments for itself.

We observe that the average solution quality is least for Ising and the highest for MS. The quality for GC is slightly lower than that of MS. For all three benchmarks, the quality increases sufficiently with increasing privacy budget, i.e., ϵ. P-Gibbs' performance for meeting-schedule is strong, especially for higher ϵs. Note that $\epsilon < 1$ is typically desirable. We consider $\epsilon \geq 1$ for illustrative purposes. P-Gibbs also provides good solution qualities

for $\epsilon < 1$. Specifically, for Ising, the average quality crosses 0.75. For GC, the average quality remains above 0.8 and crosses 0.92 for MS.

Coefficient of Variation. The Coefficient of Variation (CoV) is a statistical measure equal to the ratio between the standard deviation and the average. Note that lower CoVs imply a lower extent of variability with the average solution quality. We compute the CoV values for each ϵ based on the reported average and standard deviations (refer to Figure 12.3).

For Ising, for varying ϵ, we observe a maximum CoV of 0.123 and a minimum of 0.050. Likewise, for GC, we observe a minimum CoV of 0.059 and a maximum of 0.104. For meeting-scheduling, we have 0.044 (minimum) and 0.086 (maximum). Notably, for all three benchmarks, the maximum CoV corresponds to $\epsilon = 0.046$. For graph-coloring, the minimum CoV corresponds to $\epsilon = 9.55$ and for Ising and meeting-scheduling to $\epsilon = 4.101$. As expected, the maximum CoV corresponds to the lowest ϵ since the amount of noise (or the loss in SD-Gibbs' distribution) is highest. In contrast, higher ϵ infuse lesser noise, and consequently, the CoVs are lower.

The above observation supports the improvement in solution quality with increasing ϵ in Figure 12.3. As ϵ increases, the decrease in CoV values denotes a lower extent of variability concerning the average solution quality. That is, as ϵ increases, P-Gibbs is likelier to output a solution quality closer to the average quality reported.

12.4.2.2 Effect of Specific Parameters on Solution Quality

We now study the specific effect of parameters σ, γ, and q on the quality of P-Gibbs' solution. First, we vary σ while fixing the other parameters and observing SQ changes. Then, we likewise vary γ followed by q and observe the change in SQ for these. We conduct these experiments on the same 20 benchmark problem instances as earlier and report the average values across 20 runs.

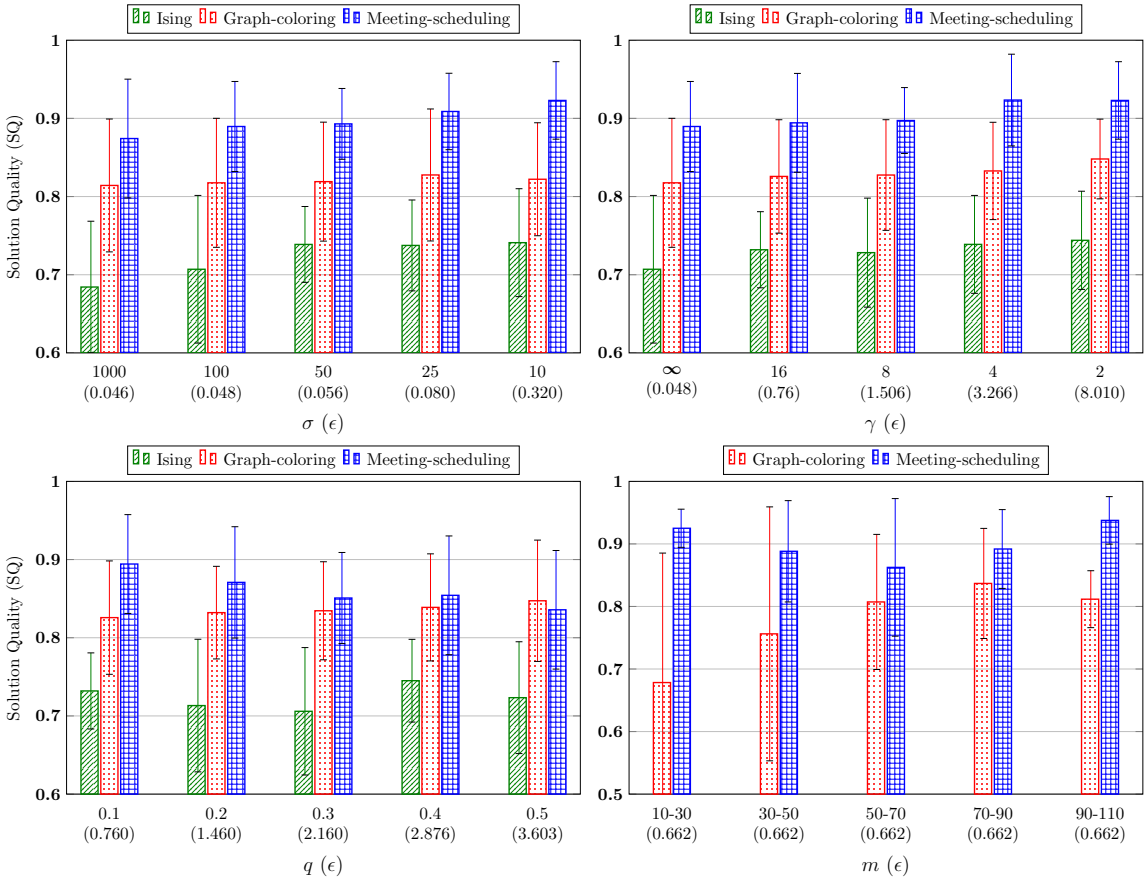


Figure 12.4: The average and standard deviation of P-Gibbs' Solution Quality (SQ) for different hyperparameters (σ, γ and q) and the problem size, m.

12.4.2.2.1 Effect of the Noise Parameter (σ) Similar to our previous subsection experiments, we let λs be 100 and $\delta = 10^{-2}$. Further, we fix $\gamma = \infty$, $\tau = 50$, and $q = 0.1$. We vary σ from the set $\{1000, 100, 50, 25, 10\}$. As σ decreases, the privacy budget ϵ increases. Intuitively, we expect the SQ to improve with a decrease in noise added.

Figure 12.4 presents the change in SQ w.r.t to the change in σ. We derive the ϵ values using Table 12.4. As expected, we observe an overall increase in the solution quality of P-Gibbs as σ decreases. However, the increase is marginal for graph-coloring, while the quality significantly improves for Ising and meeting-scheduling.

Interestingly, the solution quality for meeting-scheduling for $\sigma = 10$ ($\epsilon = 0.32$) is similar to the earlier reported quality for $\epsilon = 9.55$ (Figure 12.3). In contrast, from Figure 12.3, the SQ for graph-coloring is comfortably better for $\epsilon = 9.55$.

12.4.2.2.2 Effect of the Temperature Parameter (γ) We now turn our attention to the effect of γ on the solution qualities of P-Gibbs. For this, we fix $\tau = 50, \sigma = 100$ and $q = 0.1$ while varying $\gamma = \{\infty, 16, 8, 4, 2\}$. Similar to the case of σ, as the temperature parameter γ decreases, the privacy budget ϵ increases. As γ decreases, greater information of the original SD-Gibbs distribution is retained.

Figure 12.4 presents the results. We observe an increase in SQs as γ decreases. This increase is because an increase in γ implies that P-Gibbs' sampling distribution tends to the original SD-Gibbs' distribution. As such, the resulting solution also tends towards that of SD-Gibbs.

12.4.2.2.3 Effect of the subsampling Probability (q) To study the effect of subsampling probability q, we vary the value from the set $q \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. We fix the other parameters, i.e., $\gamma = 16$, $\sigma = 100$, and $\tau = 50$. As the probability q increases, the privacy budget ϵ increases.

Figure 12.4 presents our results. Similar to our previous results, we see an increase in solution quality for graph-coloring as the probability of sampling increases. This increase is because an increase in the subsampling probability implies an increase in each agent's probability of sampling a better assignment. However, for meeting-scheduling, we do not observe any such trend.

NOTE. These results show that the loss in sampling information deteriorates the solution quality in graph-coloring, while meeting-scheduling's solution quality largely depends on the amount of noise added. This may be due to differences between graph-coloring and meeting-scheduling [181]. In particular, we believe that abstract problems like graph-coloring better satisfy the SD-Gibbs assumption of statistical independence of variables,

while concrete problems like meeting-scheduling do not. Thus, solution quality for graph-coloring depends more on the SD-Gibbs probability distribution than meeting-scheduling.

12.4.2.2.4 Effect of Problem Size (m) We now measure the effect of the change in the number of agents (m) on P-Gibbs' SQ. To this end, we generate test cases for the two benchmarks[5] Graph-coloring (GC) and Meeting-scheduling (MS), by varying $m \in \{10\text{-}30, 30\text{-}50, 50\text{-}70, 70\text{-}90, 90\text{-}110\}$. We fix the hyperparameters in P-Gibbs to derive $\epsilon = 0.662$. We generate 10 problem instances for each m and report the average and standard deviation for P-Gibbs' SQ. Figure 12.4 depicts the results.

For GC, the average SQ generally increases as the number of agents increases from 10-30 to 90-110. Further, the standard deviation is more significant when the number of agents is small. When m is 50-70 or more, we observe greater SQ and the standard deviation is lesser than 10-30. Contrarily, for MS, the average SQ is almost similar across different problem sizes. This behavior may be due to the SQs being significantly large for each problem size.

12.4.2.2.5 Privacy Leak due to Hyperparameter Tuning Researchers have shown that hyperparameter tuning of ML models may compromise their privacy [222]. Fortunately, in our case, the tuning only corresponds to DP parameters. These parameters can be tuned via simulating privacy computation in advance, without running the actual problem-solving algorithm, and thus without revealing any information.

12.4.3 Explaining P-Gibbs' Privacy Protection for Varying ϵ

In this chapter, we provide a rigorous (ϵ, δ)-DP guarantee for P-Gibbs. Moreover, in the previous subsection, we provide the empirical quality of P-Gibbs' solution w.r.t. SD-Gibbs

[5]We omit Ising from this set of experiments, as Ising instances with > 20 agents ran out of memory during execution.

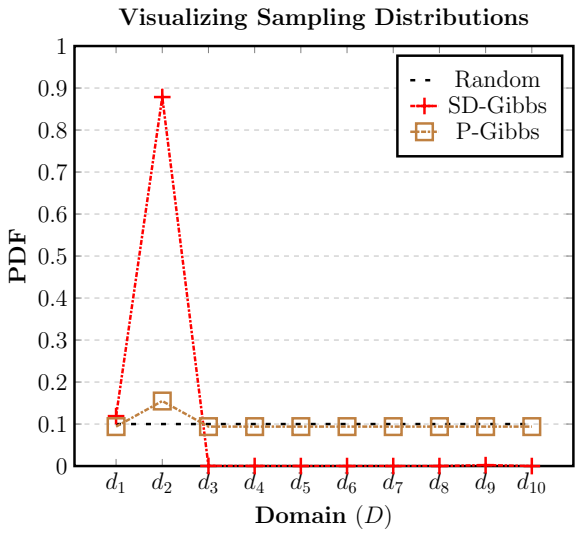


Figure 12.5: Visualizing Sampling Distributions for SD-Gibbs and P-Gibbs with a Random Assignment.

for a given privacy budget. We now demonstrate the privacy guarantee of P-Gibbs by comparing the final assignments of P-Gibbs and SD-Gibbs concerning a *random* assignment.

Note that a random final assignment will be *perfectly* privacy-preserving since no information about an agent's utility function will get encoded in the assignment. As such, the *closer* a DCOP algorithm's final assignment is to the random assignment, the *greater* the privacy protection. Figure 12.5 depicts this observation for a graph-coloring benchmark test instance with the domain $D = \{d_1, \ldots, d_{10}\}$ for any variable/agent i.

We can see that SD-Gibbs' distribution prefers the assignment d_2 with ≈ 0.9 probability. Sampling a value with the SD-Gibbs' distribution will imply that agent i greatly prefers d_2 (i.e., agent i's utility function has a sufficiently greater value for d_2 compared to $D \setminus \{d_2\}$). P-Gibbs's distribution is *closer* to random, thus plugging the information leak. To measure the distance of the assignments, we introduce the metric: *Assignment Distance*.

12.4.3.1 Assignment Distance

As depicted in Figure 12.5, we can explain the increased privacy guarantees in P-Gibbs by measuring the distance between the final assignment from P-Gibbs with a random assignment. To measure the distance, we employ the Jensen–Shannon divergence (JSD) [173][6]. Now, consider a DCOP algorithm A and domain $D = \{d_1, \ldots, d_p\}$ such that for each variable/agent i, the assignment distribution after T iterations is given by the vector $\mathbf{p}_A^i = \{p_{d_1}^i, \ldots, p_{d_p}^i\}$. Now, consider the following definition.

Definition 12.7: Assignment Distance (AD_A)

We define Assignment Distance (AD_A) of a DCOP algorithm A as the average Jensen–Shannon divergence (JSD) [173] between the vector of the assignment distribution for variable/agent i from algorithm A, i.e., $\mathbf{p}_A^i$, with the vector $\mathbf{r}$ from the random assignment, i.e., $\mathbf{r} = \frac{1}{|D|} \cdot \mathbb{1}_p$. Formally,

$$\text{AD}_A = \frac{\sum_{i \in [p]} \text{JSD}(\mathbf{p}_A^i || \mathbf{r})}{p}. \tag{12.13}$$

From Eq. (12.13), $\text{AD}_A \in [0, 1]$. When $\text{AD}_A \to 0$, it shows that algorithm A's final assignment is closer to the random assignment, i.e., A is as private as a random assignment. For $\text{AD}_A \to 1$, the assignment is farthest, implying that A encodes the maximum information possible. By comparing the assignment distance, i.e., $\text{AD}_{\text{SD-Gibbs}}$ and $\text{AD}_{\text{P-Gibbs}}$, we can explain the increased privacy of P-Gibbs.

12.4.3.2 Experimental Evaluation

To better study assignment distance, we empirically derive its values for two DCOP benchmarks, graph coloring and meeting scheduling. We omit Ising from this set of experiments as the domain there is binary.

[6] JSD [173] is a statistical method to measure the similarity of two probability distributions. It is based on KL-divergence but does not require the same support for the distributions.

Benchmark	Assignment Distance (AD)			
	SD-Gibbs	**P-Gibbs** ($\epsilon = 0.046$)	**P-Gibbs** ($\epsilon = 0.662$)	**P-Gibbs** ($\epsilon = 9.55$)
Graph-coloring	0.553	0.253	0.275	0.288
Meeting-scheduling	0.71	0.623	0.624	0.64

Table 12.5: Empirically evaluating Assignment Distance for SD-Gibbs and P-Gibbs. Note that, AD $\rightarrow$ 0 implies greater privacy protection, while AD $\rightarrow$ 1 implies maximum information leak.

12.4.3.2.1 Instance Setup For graph coloring, we have 30 agents/variables with domain size 10 and each constraint between $(0, 10]$. For meeting scheduling, we have 30 agents/variables with domain size 10 and each constraint between $[-1000, 10] \setminus \{0\}$.

12.4.3.2.2 Results We run both the benchmark instances 40 times and report the corresponding assignment distance (AD) values in Table 12.5. For graph-coloring, AD values for P-Gibbs are $\approx 50\%$ less than that for SD-Gibbs. Whereas for meeting-scheduling, it is $\approx 10\%$. This shows that P-Gibbs' final assignment encodes less information compared to SD-Gibbs', in turn, better preserving constraint privacy. Moreover, as ϵ increases, the decrease in the noise added and increase in subsampling probability results in an increase in AP values for P-Gibbs as the algorithm behaves more like SD-Gibbs.

12.4.4 Discussion

Overall, P-Gibbs provides strong solution quality for a competitive privacy budget. Concerning specific hyperparameters, we observe that the amount of noise (σ) added has the most impact on the quality of the solution – especially for meeting-scheduling. In practice, one needs to properly tune the parameters based on the problem at hand [181]. Since ours is the first method of its kind, to the best of our knowledge, we believe the results presented are strong, and future work may further improve the performance.

Advantages of our DP-based Approach We present the first differentially private DCOP algorithm with provable guarantees for constraint privacy with P-Gibbs. Here we emphasize the utility of a DP-based solution compared to the existing cryptographic solutions. Notably, as also mentioned in [100], in cryptography-based solutions, the final assignment may leak critical information about the constraints, i.e., existing algorithms do not satisfy solution privacy. Our DP-based approach overcomes this prevalent issue by randomizing the computation and perturbing agents' utility values. In turn, the final assignment may not always be optimal, i.e., with P-Gibbs, there is a drop in SQ.

Another crucial advantage of DP is the scalability of P-Gibbs. Since our privacy guarantees do not depend on computationally heavy cryptographic primitives, we conduct experiments on much larger problems than existing algorithms (e.g., [100, 167]). P-Gibbs' run time remains the same as SD-Gibbs in contrast to private DCOP algorithms based on cryptographic primitives (e.g., [274, 127, 273]).

12.5 Conclusion

In this chapter, we addressed the problem of privacy-preserving distributed constraint optimization. With our novel algorithm – P-Gibbs, we are the first to show a DP guarantee for the same. Using the local DP model, our algorithm preserves the privacy of unrelated agents' preferences. This guarantee also extends to the solution. We also achieve high-quality solutions with reasonably strong privacy guarantees and efficient computation, especially in meeting-scheduling problems.

Future Work As a first attempt at providing a differential privacy guarantee for DCOPs, we focused on the classical DP notion in this chapter. Using the notion of Bayesian DP [283] may further improve the (ϵ, δ) guarantees. More concretely, with Bayesian DP, we may be able to estimate the privacy cost $c_t(\lambda)$ with the agent's actual distribution (instead of the worst-case for all agents). We remark that such an estimation is non-trivial as it may

require certain assumptions of other agent's distribution. We leave the analysis for future work. Alternatively, one can further enrich our DCOP model by relaxing the assumption of domains being the same for each agent while ensuring meaningful privacy guarantees. Concerning P-Gibbs, we can also perform experiments on real-world datasets to further fine-tune the algorithm's hyperparameters.

Chapter 13

Conclusion and Future Work

This thesis has delved into two distinct yet interconnected realms, each contributing to the evolving landscape of multi-agent systems (MAS) with a focus on addressing challenges faced by the participating agents.

In PART A, we primarily focused on game theory and mechanism design, with dedicated applications, namely, Civic Crowdfunding (CC) and Transaction Fee Mechanisms (TFMs). In the domain of CC, our contributions have been multifaceted. We extended the CC literature by introducing mechanisms that relax traditional assumptions about agents' information, resulting in more inclusive CC mechanisms (Chapter 4). Moreover, we investigated the efficiency of CC mechanisms over a blockchain, offering the promise of transparent and fair platforms by eliminating intermediaries (Chapter 5). The exploration extended to combinatorial CC, shedding light on the simultaneous crowdfunding of multiple public projects under budget-constrained agents (Chapter 6). We concluded our look at CC mechanisms by focusing on the online setting where agents' belief of the project's funding is dynamic (Chapter 7). In the realm of TFMs, we asserted the importance of fairness, advocating for the inclusion of transactions with zero fees while maintaining bid inclusion monotonicity (Chapter 8). Additionally, we introduced Transaction Fee Redistribution Mechanisms (TFRMs), a novel class of TFMs that reduce transaction fees through calibrated rebates (Chapter 9).

PART B pivoted towards privacy concerns within multi-agent systems (MAS). We looked at three interconnected systems: privacy-preserving voting applications over blockchain, privacy-preserving combinatorial auctions over blockchain, and differentially private distributed constraint optimization (DCOP). FASTEN (Chapter 10) provided a scalable and privacy-focused voting mechanism leveraging blockchain technology. STOUP (Chapter 11) introduced cryptographic protocols and smart contract mechanisms for secure and private bidding in combinatorial auctions. Lastly, P-Gibbs (Chapter 12) presented a differentially private DCOP algorithm, ensuring the optimization process does not compromise the privacy of individual agents.

This thesis has made contributions to the fields of game theory and mechanism design, and privacy-preserving mechanisms within MAS. By addressing challenges in Civic Crowdfunding, Transaction Fee Mechanisms, privacy-preserving voting applications, combinatorial auctions, and DCOPs, the goal has been to design more inclusive, fair, strategyproof multi-agent systems that preserve the privacy of an agent's sensitive information.

Future Work

We hope that the research presented in this thesis serves as a good resource for future explorations and advancements in these evolving domains. We remark that each (contribution) chapter provides concrete (and immediate) future directions to explore. Here, we conclude the thesis by listing a few (broader) directions for future work.

- **Combinatorial Civic Crowdfunding.** Our exploration of CC mechanisms significantly extends the CC literature. However, several questions remain unexplored: (i) designing refunds for welfare-optimal combinatorial CC under budget-constrained agents, (ii) exact characterization of an agent's dynamic belief, and (iii) designing strategyproof CC mechanisms for agents with both negative valuation and asymmetric belief (which currently appears to be at odds with one another). Aside from this

theoretical analysis, it will also be important to see how well the theory aligns with real-world observations, perhaps similar to Cason et al. [47].

- **Transactions Fee Mechanism Design.** The TFM literature is relatively recent, with several directions to study. In particular, towards fairer TFMs for the transaction creators, it will be important to study the role of the fair TFMs over time. In fact, the rebates offered to creators with TFRMs may also result in complex strategies when we consider non-myopic creators or miners. Making TFMs fair while being resilient to such malicious attacks over time may be an important research direction to explore in the future.

- **Trusted Third-parties.** Our privacy-preserving applications, FASTEN (Chapter 10) and STOUP (Chapter 11), utilize (semi) trusted third parties for private computations. This is in line with other privacy-preserving solutions [179, 178]. Future work can look at designing privacy-preserving applications – with similar security and privacy guarantees – without the need for such parties (e.g., using bulletproofs [41] and zk-STARKs [28]).

Bibliography

[1] "Visa and Mastercard". *Visa and Mastercard are LOSING fast to Indian alternatives*. https://d3.harvard.edu/platform-digit/submission/visa-and-mastercard-are-losing-fast-to-indian-alternatives/. 2020.

[2] Karen I Aardal, Stan PM Van Hoesel, Arie MCA Koster, Carlo Mannino, and Antonio Sassano. "Models and solution techniques for frequency assignment problems". In: *Annals of Operations Research* 153.1 (2007), pp. 79–129.

[3] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[4] Mark Abspoel, Niek J Bouman, Berry Schoenmakers, and Niels de Vreede. "Fast secure comparison for medium-sized integers and its application in binarized neural networks". In: *Cryptographers' Track at the RSA Conference*. Springer. 2019, pp. 453–472.

[5] Ben Adida. "Helios: Web-based Open-Audit Voting." In: *USENIX security symposium*. Vol. 17. 2008, pp. 335–348.

[6] Gagan Aggarwal and Jason D. Hartline. "Knapsack Auctions". In: *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2006, pp. 1083–1092.

[7] Saeed Alaei, Azarakhsh Malekian, and Mohamed Mostagir. "A Dynamic Model of Crowdfunding". In: *Proceedings of the 2016 ACM Conference on Economics and Computation*. EC '16. Maastricht, The Netherlands: ACM, 2016, pp. 363–363. ISBN: 978-1-4503-3936-0. DOI: 10.1145/2940716.2940777. URL: http://doi.acm.org/10.1145/2940716.2940777.

[8] Porto Alegre. *Case study 2-Porto Alegre, Brazil: participatory approaches in budgeting and public expenditure management*. 2020.

[9] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. "Fair Division of Indivisible Goods: A Survey". In: *CoRR* abs/2208.08782 (2022). URL: https://doi.org/10.48550/arXiv.2208.08782.

[10] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains". In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[11] Russell Archey. *Minecraft*. 2016. URL: https://www.gamingnexus.com/Article/5012/Minecraft/.

[12] Sanidhay Arora, Anurag Jain, Sankarshan Damle, and Sujit Gujar. "ASHWAChain: A Fast, Scalable and Strategy-proof Committee-based Blockchain Protocol". In: *Workshop on Game Theory in Blockchain at WINE 2020 (GTiB@WINE 2020)*. Dec. 11, 2020. published.

[13] Kenneth J Arrow. "A difficulty in the concept of social welfare". In: *Journal of political economy* 58.4 (1950), pp. 328–346.

[14] Muhammad Arshad and Marius C Silaghi. "Distributed simulated annealing". In: *Distributed constraint problem solving and reasoning in multi-agent systems* 112 (2004).

[15] Avi Asayag, Gad Cohen, Ido Grayevsky, Maya Leshkowitz, Ori Rottenstreich, Ronen Tamari, and David Yakira. "A fair consensus protocol for transaction ordering". In: *IEEE 26th International Conference on Network Protocols (ICNP)*. 2018, pp. 55–65.

[16] *Average block time of the Ethereum Network*. URL: https://www.etherchain.org/charts/blockTime.

[17] Haris Aziz, Bo Li, Hervé Moulin, and Xiaowei Wu. "Algorithmic fair allocation of indivisible items: a survey and new questions". In: *SIGecom Exch.* 20.1 (2022), pp. 24–40.

[18] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. "Algorithms for max-min share fair allocation of indivisible chores". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 2017.

[19] Mark Bagnoli and Barton L Lipman. "Provision of public goods: Fully implementing the core through private contributions". In: *The Review of Economic Studies* 56.4 (1989), pp. 583–601.

[20] Arati Baliga, I Subhod, Pandurang Kamat, and Siddhartha Chatterjee. "Performance evaluation of the quorum blockchain platform". In: *arXiv preprint arXiv:1809.03421* (2018).

[21] Borja Balle, Gilles Barthe, and Marco Gaboardi. "Privacy amplification by subsampling: Tight analyses via couplings and divergences". In: *Advances in Neural Information Processing Systems* 31 (2018).

[22] Santiago R Balseiro, Vahab S Mirrokni, and Renato Paes Leme. "Dynamic mechanisms with martingale utilities". In: *Management Science* 64.11 (2018), pp. 5062–5082.

[23] Rui Pedro Barbosa and Orlando Belo. "Multi-agent forex trading system". In: *Agent and multi-agent technology for internet and enterprise systems*. Springer, 2010, pp. 91–118.

[24] Soumya Basu, David Easley, Maureen O'Hara, and Emin Sirer. "StableFees: A Predictable Fee Market for Cryptocurrencie". In: *Available at SSRN 3318327* (2019).

[25] Soumya Basu, David Easley, Maureen O'Hara, and Emin Gün Sirer. "Towards a functional fee market for cryptocurrencies". In: *arXiv preprint arXiv:1901.06830* (2019).

[26] Donald Beaver and Shaft Goldwasser. "Multiparty computation with faulty majority". In: *Conference on the Theory and Application of Cryptology*. 1989, pp. 589–590.

[27] Emanuele Bellini, Paolo Ceravolo, and Ernesto Damiani. "Blockchain-based e-Vote-as-a-Service". In: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE. 2019, pp. 484–486.

[28] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable, transparent, and post-quantum secure computational integrity". In: *Cryptology ePrint Archive* (2018).

[29] Piotr Berman, Juan A Garay, Kenneth J Perry, et al. "Towards optimal distributed consensus". In: *FOCS*. Vol. 89. 1989, pp. 410–415.

[30] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. "High-Speed High-Security Signatures". In: *CHES*. 2011, pp. 124–142.

[31] Adithya Bhat, Nibesh Shrestha, Zhongtang Luo, Aniket Kate, and Kartik Nayak. "Randpiper–reconfiguration-friendly random beacons with quadratic communication". In: *ACM CCS*. 2021, pp. 3502–3524.

[32] Ian F Blake and Vladimir Kolesnikov. "Strong conditional oblivious transfer and computing on intervals". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2004, pp. 515–529.

[33] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. "Practical secure aggregation for privacy-preserving machine learning". In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191.

[34] Tilman Börgers and Daniel Krahmer. *An introduction to the theory of mechanism design*. Oxford University Press, USA, 2015.

[35] Stefan Bosse, Florian Pantke, and Frank Kirchner. "Distributed computing in sensor networks using multi-agent systems and code morphing". In: *Artificial Intelligence and Soft Computing: 11th International Conference, ICAISC 2012, Zakopane, Poland, April 29-May 3, 2012, Proceedings, Part II 11*. Springer. 2012, pp. 415–423.

[36] Felix Brandt and Tuomas Sandholm. "Efficient privacy-preserving protocols for multi-unit auctions". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2005, pp. 298–312.

[37] Romaric Breil, Daniel Delahaye, Laurent Lapasset, and Eric Féron. "Multi-agent systems to help managing air traffic structure". In: *Journal of Aerospace Operations* 5.1-2 (2017), pp. 119–148.

[38] John S Bridle. "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: *Neurocomputing*. 1990, pp. 227–236.

[39] Ismel Brito, Amnon Meisels, Pedro Meseguer, and Roie Zivan. "Distributed constraint satisfaction with partially known constraints". In: *Constraints* 14.2 (2009), pp. 199–234.

[40] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short proofs for confidential transactions and more". In: *2018 IEEE S&P*. 2018, pp. 315–334.

[41] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short proofs for confidential transactions and more". In: *2018 IEEE symposium on security and privacy (S&P)*. IEEE. 2018, pp. 315–334.

[42] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform". In: *White Paper* 3.37 (2014), pp. 2–1.

[43] Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. *EIP-1559: Fee market change for ETH 1.0 chain*. eips.ethereum.org/EIPS/eip-1559. 2019.

[44] Christian Cachin. "Efficient private bidding and auctions with an oblivious third party". In: *Proceedings of the 6th ACM conference on Computer and communications security*. 1999, pp. 120–127.

[45] Robert D Carmichael. "On the numerical factors of the arithmetic forms α n$\pm\beta$ n". In: *The Annals of Mathematics* 15.1/4 (1913), pp. 49–70.

[46] Yan Carriere-Swallow, Manasa Patnam, and Vikram Haksar. *STACKING UP FINANCIAL INCLUSION GAINS IN INDIA*. imf.org/external/pubs/ft/fandd/2021/07/india-stack-financial-access-and-digital-inclusion.htm. 2021.

[47] Timothy N Cason, Alex Tabarrok, and Robertas Zubrickas. "Early refund bonuses increase successful crowdfunding". In: *Games and Economic Behavior* 129 (2021), pp. 78–95.

[48] Timothy N Cason and Robertas Zubrickas. “Enhancing fundraising with refund bonuses”. In: *Games and Economic Behavior* 101 (2017), pp. 218–233.

[49] Ruggiero Cavallo. “Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments”. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. 2006, pp. 882–889.

[50] Jesús Cerquides, Juan Antonio Rodrıguez-Aguilar, Rémi Emonet, and Gauthier Picard. “Solving highly cyclic distributed optimization problems without busting the bank: a decimation-based approach”. In: *Logic Journal of the IGPL* 29.1 (2021), pp. 72–95.

[51] Deeparnab Chakrabarty and Chaitanya Swamy. “Welfare maximization and truthfulness in mechanism design with ordinal preferences”. In: *Proceedings of the 5th conference on Innovations in Theoretical Computer Science*. 2014, pp. 105–120.

[52] Praphul Chandra, Sujit Gujar, and Y Narahari. “Crowdfunding public projects with provision point: A prediction market approach”. In: *Proceedings of the Twenty-second European Conference on Artificial Intelligence*. 2016, pp. 778–786.

[53] Praphul Chandra, Sujit Gujar, and Yadati Narahari. “Referral-Embedded Provision Point Mechanisms for Crowdfunding of Public Projects”. In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2017*. International Foundation for Autonomous Agents and Multiagent Systems. 2017, pp. 642–650.

[54] Praphul Chandra, Sujit Gujar, and Yadati Narahari. “Referral-Embedded Provision Point Mechanisms for Crowdfunding of Public Projects”. In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2018*. International Foundation for Autonomous Agents and Multiagent Systems. 2017, pp. 642–650.

[55] Archie C Chapman, Alex Rogers, and Nicholas R Jennings. "Benchmarking hybrid algorithms for distributed constraint optimisation games". In: *Autonomous Agents and Multi-Agent Systems* 22 (2011), pp. 385–414.

[56] David Chaum, Ivan B Damgård, and Jeroen Van de Graaf. "Multiparty computations ensuring privacy of each party's input and correctness of the result". In: *Conference on the Theory and Application of Cryptographic Techniques*. 1987, pp. 87–119.

[57] Shuchi Chawla, Nikhil R. Devanur, Anna R. Karlin, and Balasubranianian Sivan. "Simple Pricing Schemes for Consumers with Evolving Values". In: *ACM-SIAM SODA*. 2016, pp. 1476–1490.

[58] Xin Chen, Tao Zhang, Sheng Shen, Tianqing Zhu, and Ping Xiong. "An optimized differential privacy scheme with reinforcement learning in vanet". In: *Computers & Security* 110 (2021), p. 102446.

[59] Yiling Chen and David M. Pennock. "A Utility Framework for Bounded-Loss Market Makers". In: *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. UAI'07. Vancouver, BC, Canada: AUAI Press, 2007, pp. 49–56. ISBN: 0974903930.

[60] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. "Secureboost: A lossless federated learning framework". In: *IEEE Intelligent Systems* 36.6 (2021), pp. 87–98.

[61] Alexandra Chouldechova. "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments". In: *IEEE Big data* 5 2 (2017), pp. 153–163.

[62] Hao Chung and Elaine Shi. "Foundations of transaction fee mechanism design". In: *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2023.

[63] Edward H Clarke. "Multipart pricing of public goods". In: *Public choice* 11.1 (1971), pp. 17–33.

[64] Wikipedia Contributors. *Contract Net Protocol*. 2024. URL: https://en.wikipedia.org/wiki/Contract_Net_Protocol.

[65] ACM US Public Policy Council. "Statement on algorithmic transparency and accountability". In: *Commun. ACM* (2017).

[66] Antoine Augustin Cournot. *Researches into the Mathematical Principles of the Theory of Wealth*. New York: Macmillan Company, 1927 [c1897], 1927.

[67] Geoffroy Couteau. "Efficient Secure Comparison Protocols." In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 544.

[68] Peter Cramton, Yoav Shoham, and Richard Steinberg. *Combinatorial Auctions*. The MIT Press, Dec. 2005. ISBN: 9780262033428. DOI: 10.7551/mitpress/9780262033428.001.0001. URL: https://doi.org/10.7551/mitpress/9780262033428.001.0001.

[69] Ivan Bjerre Damgård. "A design principle for hash functions". In: *Conference on the Theory and Application of Cryptology (CRYPTO)*. Springer. 1989, pp. 416–427.

[70] Sankarshan Damle, Boi Faltings, and Sujit Gujar. "A Truthful, Privacy-Preserving, Approximately Efficient Combinatorial Auction For Single-minded Bidders." In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2019*. International Foundation for Autonomous Agents and Multiagent Systems. 2019, pp. 1916–1918.

[71] Sankarshan Damle, Boi Faltings, and Sujit Gujar. "Blockchain-based Practical Multi-agent Secure Comparison and its Application in Auctions". In: *WI-IAT '21: IEEE/WIC/ACM International Conference on Web Intelligence*. ACM, 2021, pp. 430–437.

[72] Sankarshan Damle and Sujit Gujar. "Analyzing Crowdfunding of Public Projects Under Dynamic Beliefs". In: *Proceedings of the 23rd Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2024*. International Foundation for Autonomous Agents and Multiagent Systems. 2024, pp. 2225–2227.

[73] Sankarshan Damle, Sujit Gujar, and Moin Hussain Moti. "FASTEN: Fair and Secure Distributed Voting Using Smart Contracts". In: *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2021, Sydney, Australia, May 3-6, 2021*. IEEE, 2021, pp. 1–3.

[74] Sankarshan Damle, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Aggregating Citizen Preferences for Public Projects Through Civic Crowdfunding". In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2019*. International Foundation for Autonomous Agents and Multiagent Systems. 2019, pp. 1919–1921.

[75] Sankarshan Damle, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Civic Crowdfunding for Agents with Negative Valuations and Agents with Asymmetric Beliefs". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*. 2019, pp. 208–214.

[76] Sankarshan Damle, Moin Hussain Moti, Praphul Chandra, and Sujit Gujar. "Designing Refund Bonus Schemes for Provision Point Mechanism in Civic Crowdfunding". In: *PRICAI 2021: Trends in Artificial Intelligence*. 2021, pp. 18–32.

[77] Sankarshan Damle, Manisha Padala, and Sujit Gujar. "Combinatorial Civic Crowdfunding with Budgeted Agents: Welfare Optimality at Equilibrium and Optimal Deviation". In: *AAAI Conference on Artificial Intelligence*. 2023.

[78] Sankarshan Damle, Manisha Padala, and Sujit Gujar. "Designing Redistribution Mechanisms for Reducing Transaction Fees in Blockchains". In: *Proceedings of the 23rd Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2024*. International Foundation for Autonomous Agents and Multiagent Systems. 2024, pp. 416–424.

[79] Sankarshan Damle, Varul Srivastava, and Sujit Gujar. "No Transaction Fees? No Problem! Achieving Fairness in Transaction Fee Mechanism Design". In: *Proceedings*

of the 23rd Conference on Autonomous Agents and MultiAgent Systems (AAMAS) 2024. International Foundation for Autonomous Agents and Multiagent Systems. 2024, pp. 2228–2230.

[80] Sankarshan Damle, Aleksei Triastcyn, Boi Faltings, and Sujit Gujar. "Differentially Private Multi-Agent Constraint Optimization". In: *WI-IAT '21: IEEE/WIC/ACM International Conference on Web Intelligence*. ACM, 2021, pp. 422–429.

[81] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. "Spurt: Scalable distributed randomness beacon with transparent setup". In: *IEEE S&P*. 2022.

[82] Anirban Dasgupta and Arpita Ghosh. "Crowdsourced judgement elicitation with endogenous proficiency". In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 319–330.

[83] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. "The complexity of computing a Nash equilibrium". In: *Communications of the ACM* 52.2 (2009), pp. 89–97.

[84] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004.

[85] *Documentation/TechnicalWhitepaper.md*. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. 2020.

[86] Wei Du, Depeng Xu, Xintao Wu, and Hanghang Tong. "Fairness-aware Agnostic Federated Learning". In: *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM. 2021, pp. 181–189.

[87] Huayi Duan, Yifeng Zheng, Yuefeng Du, Anxin Zhou, Cong Wang, and Man Ho Au. "Aggregating crowd wisdom via blockchain: A private, correct, and robust realization". In: *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE. 2019, pp. 1–10.

[88] Edmund H Durfee. *Coordination of distributed problem solvers*. Vol. 55. Springer Science & Business Media, 2012.

[89] Edmund H. Durfee. "Planning in Distributed Artificial Intelligence". In: *Foundations of Distributed Artificial Intelligence*. USA: John Wiley & Sons, Inc., 1996, pp. 231–245. ISBN: 0471006750.

[90] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David Parkes, and Sai Srivatsa Ravindranath. "Optimal auctions through deep learning". In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 1706–1715.

[91] Cynthia Dwork. "Differential Privacy". In: *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*. Vol. 4052. Lecture Notes in Computer Science. 2006, pp. 1–12.

[92] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer. 2006, pp. 265–284.

[93] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy." In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2014), pp. 211–407.

[94] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. "Boosting and differential privacy". In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE. 2010, pp. 51–60.

[95] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472.

[96] BrightHub Engineering. *How do Traffic Control Systems Work: Air, Road, Applications of Traffic Control System*. 2024. URL: https://www.brighthubengineering.

com/consumer-appliances-electronics/67234-know-more-about-traffic-control-systems/.

[97] *Ethereum Average Gas Price Chart*. URL: https://etherscan.io/chart/gasprice.

[98] *Evolution of the Average Gas Limit*. URL: https://www.etherchain.org/charts/blockGasLimit.

[99] Boi Faltings, Thomas Léauté, and Adrian Petcu. "Privacy Guarantees through Distributed Constraint Satisfaction". In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 2. 2008, pp. 350–358. DOI: 10.1109/WIIAT.2008.177.

[100] Boi Faltings, Thomas Léauté, and Adrian Petcu. "Privacy guarantees through distributed constraint satisfaction". In: *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 2. IEEE. 2008, pp. 350–358.

[101] Haokun Fang and Quan Qian. "Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning". In: *Future Internet* 13.4 (2021), p. 94.

[102] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. "Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm". In: *AAMAS*. 2008, pp. 639–646.

[103] Matheus V. X. Ferreira, Daniel J. Moroz, David C. Parkes, and Mitchell Stern. "Dynamic posted-price mechanisms for the blockchain transaction-fee market". In: *ACM Conference on Advances in Financial Technologies (AFT)*. 2021, pp. 86–99.

[104] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194.

[105] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. "Distributed constraint optimization problems and applications: A survey". In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 623–698.

[106] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". In: *J. ACM* 32.2 (Apr. 1985), pp. 374–382.

[107] Wilfried Fischer and Bernd W Muller. *Method and apparatus for the manufacture of a product having a substance embedded in a carrier*. US Patent 5,043,280. Aug. 1991.

[108] Marc Fischlin. "A cost-effective pay-per-multiplication comparison method for millionaires". In: *Cryptographers' Track at the RSA Conference*. 2001, pp. 457–471.

[109] Duncan Karl Foley. *Resource allocation and the public sector*. Yale University, 1966.

[110] *FollowMyVote*. URL: https://followmyvote.com/.

[111] Wikimedia Foundation. *Paillier Cryptosystem*. https://en.wikipedia.org/wiki/Paillier_cryptosystem. 2023.

[112] Evrard Garcelon, Vianney Perchet, Ciara Pike-Burke, and Matteo Pirotta. "Local differential privacy for regret minimization in reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 10561–10573.

[113] Leonardo Garrido and Katia Sycara. "Multi-agent meeting scheduling: Preliminary experimental results". In: *Proceedings of the Second International Conference on Multiagent Systems*. 1996, pp. 95–102.

[114] Sylvain Gelly and David Silver. "Combining online and offline knowledge in UCT". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 273–280.

[115] Amir Gershman, Amnon Meisels, and Roie Zivan. “Asynchronous forward bounding for distributed COPs”. In: *Journal of Artificial Intelligence Research* 34 (2009), pp. 61–88.

[116] Arthur Gervais, Srdjan Capkun, Ghassan O Karame, and Damian Gruber. “On the privacy provisions of bloom filters in lightweight bitcoin clients”. In: *Annual Computer Security Applications Conference (ACSAC)*. 2014, pp. 326–335.

[117] Allan Gibbard. “Manipulation of voting schemes: a general result”. In: *Econometrica: journal of the Econometric Society* (1973), pp. 587–601.

[118] Naman Goel and Boi Faltings. “Crowdsourcing with Fairness, Diversity and Budget Constraints”. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (2019).

[119] Naman Goel and Boi Faltings. “Deep Bayesian Trust: A Dominant and Fair Incentive Mechanism for Crowd”. In: *The Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 1996–2003.

[120] GoFundMe. *GoFundMe — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=GoFundMe. 2020.

[121] L. Gong, S. Li, C. Wu, and D. Wang. “Secure “Ratio” Computation and Efficient Protocol for General Secure Two-Party Comparison”. In: *IEEE Access* 6 (2018), pp. 25532–25542.

[122] McClure Grant, Virwaney Sandeep, and Lin Fuhua. “Integrating multiagent systems into virtual worlds”. In: *3rd International Conference on Multimedia Technology (ICMT-13)*. Atlantis Press. 2013, pp. 574–581.

[123] J.R. Green and J.J. Laffont. *Incentives in Public Decision-making*. Current Topics in Radiation Research. North-Holland Publishing Company, 1979. ISBN: 9780444851444. URL: https://books.google.co.in/books?id=0q62AAAAIAAJ.

[124] Dima Grigoriev and Vladimir Shpilrain. "YAO'S MILLIONAIRES'PROBLEM AND DECOY-BASED PUBLIC KEY ENCRYPTION BY CLASSICAL PHYSICS". In: *International Journal of Foundations of Computer Science* 25.04 (2014), pp. 409–417.

[125] Tal Grinshpoun, Alon Grubshtein, Roie Zivan, Arnon Netzer, and Amnon Meisels. "Asymmetric distributed constraint optimization problems". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 613–647.

[126] Tal Grinshpoun and Tamir Tassa. "A privacy-preserving algorithm for distributed constraint optimization". In: *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems. 2014, pp. 909–916.

[127] Tal Grinshpoun and Tamir Tassa. "P-SyncBB: A privacy preserving branch and bound DCOP algorithm". In: *Journal of Artificial Intelligence Research* 57 (2016), pp. 621–660.

[128] Tal Grinshpoun, Tamir Tassa, Vadim Levit, and Roie Zivan. "Privacy preserving region optimal algorithms for symmetric and asymmetric DCOPs". In: *Artificial Intelligence* 266 (2019), pp. 27–50.

[129] Theodore Groves. "Incentives in Teams". In: *Econometrica* 41.4 (1973), pp. 617–31.

[130] Sujit Gujar and Y. Narahari. "Redistribution Mechanisms for Assignment of Heterogeneous Objects". In: *J. Artif. Int. Res.* 41.2 (May 2011), pp. 131–154. ISSN: 1076-9757.

[131] Mingyu Guo and Vincent Conitzer. "Better Redistribution with Inefficient Allocation in Multi-unit Auctions with Unit Demand". In: *ACM EC*. 2008, pp. 210–219.

[132] Mingyu Guo and Vincent Conitzer. "Worst-case Optimal Redistribution of VCG Payments". In: *ACM EC*. 2007, pp. 30–39.

[133] Mingyu Guo and Vincent Conitzer. “Worst-case optimal redistribution of VCG payments in multi-unit auctions”. In: *Games and Economic Behavior* 67.1 (2009), pp. 69–98.

[134] Youssef Hamadi, Christian Bessiere, and Joël Quinqueton. “Distributed Intelligent Backtracking.” In: *ECAI*. 1998, pp. 219–223.

[135] Katsutoshi Hirayama and Makoto Yokoo. “Distributed partial constraint satisfaction problem”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 1997, pp. 222–236.

[136] Friðrik Þ Hjálmarsson, Gunnlaugur K Hreiðarsson, Mohammad Hamdaqa, and Gısli Hjálmtỳsson. “Blockchain-based e-voting system”. In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE. 2018, pp. 983–986.

[137] J. Hou, S. Luo, W. Xu, and L. Wang. “Fairness-based multi-task reward allocation in mobile crowdsourcing system”. In: *IET Communications* 13.16 (2019), pp. 2506–2511.

[138] Mengdi Huai, Di Wang 0015, Chenglin Miao, Jinhui Xu, and Aidong Zhang. “Privacy-aware Synthesizing for Crowdsourced Data.” In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*. ijcai.org, 2019, pp. 2542–2548.

[139] Mengdi Huai, Di Wang 0015, Chenglin Miao, Jinhui Xu, and Aidong Zhang. “Privacy-aware Synthesizing for Crowdsourced Data.” In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*. 2019, pp. 2542–2548.

[140] Zhenqi Huang, Sayan Mitra, and Nitin Vaidya. “Differentially private distributed optimization”. In: *Proceedings of the 2015 International Conference on Distributed Computing and Networking*. 2015, pp. 1–10.

[141] HudExchange. *Participatory Budgeting*. https://www.hudexchange.info/programs/participatory-budgeting/. 2022.

[142] Leonid Hurwicz. "The design of mechanisms for resource allocation". In: *The American Economic Review* 63.2 (1973), pp. 1–30.

[143] Ioannis Ioannidis and Ananth Grama. "An efficient protocol for Yao's millionaires' problem". In: *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. IEEE. 2003, 6–pp.

[144] Anurag Jain, Shoeb Siddiqui, and Sujit Gujar. "We Might Walk Together, but I Run Faster: Network Fairness and Scalability in Blockchains". In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2021, pp. 1539–1541.

[145] Alberto Jesu, Victor-Alexandru Darvariu, Alessandro Staffolani, Rebecca Montanari, and Mirco Musolesi. "Reinforcement Learning on Encrypted Data". In: *arXiv preprint arXiv:2109.08236* (2021).

[146] Radu Jurca and Boi Faltings. "Robust incentive-compatible feedback payments". In: *Agent-Mediated Electronic Commerce. Automated Negotiation and Strategy Design for Electronic Markets*. Springer, 2007, pp. 204–218.

[147] P. Kaghazgaran and B. Sadeghyan. "Secure two party comparison over encrypted data". In: *2011 World Congress on Information and Communication Technologies*. 2011, pp. 1123–1126.

[148] Daniel Kahneman, Jack L Knetsch, and Richard Thaler. "Fairness as a constraint on profit seeking: Entitlements in the market". In: *The American economic review* (1986), pp. 728–741.

[149] Anson Kahng, Simon Mackenzie, and Ariel D Procaccia. "Liquid Democracy: An Algorithmic Perspective". In: *32nd AAAI Conference on Artificial Intelligence*. 2018.

[150] Samhita Kanaparthy, Sankarshan Damle, and Sujit Gujar. "REFORM: Reputation Based Fair and Temporal Reward Framework for Crowdsourcing". In: *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022, pp. 1648–1650.

[151] Samhita Kanaparthy, Sankarshan Damle, and Sujit Gujar. "REFORM: Reputation Based Fair and Temporal Reward Framework for Crowdsourcing". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 2022, pp. 1648–1650.

[152] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. "What can we learn privately?" In: *SIAM Journal on Computing* 40.3 (2011), pp. 793–826.

[153] Hideaki Katagishi and Jonathan P Pearce. "KOPT: Distributed DCOP algorithm for arbitrary k-optima with monotonically increasing utility". In: *DCR-07* (2007).

[154] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.

[155] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. "Order-fairness for byzantine consensus". In: *Annual International Cryptology Conference (CRYPTO)*. 2020, pp. 451–480.

[156] Kickstarter. *Kickstarter — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Kickstarter. 2020.

[157] Kickstarter. *Shortening the Maximum Project Length*. 2011. URL: %7Bkickstarter.com/blog/shortening-the-maximum-project-length%7D.

[158] Christopher Kiekintveld, Zhengyu Yin, Atul Kumar, and Milind Tambe. "Asynchronous algorithms for approximate distributed constraint optimization with quality bounds." In: *AAMAS*. Vol. 10. 2010, pp. 133–140.

[159] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[160] Pablo Kogan, Tamir Tassa, and Tal Grinshpoun. "Privacy Preserving DCOP Solving by Mediation". In: *Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML*. Vol. 13301. Lecture Notes in Computer Science. 2022, pp. 487–498.

[161] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts". In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 839–858.

[162] Klaus Kursawe. "Wendy, the good little fairness widget: Achieving order fairness for blockchains". In: *ACM Conference on Advances in Financial Technologies (AFT)*. 2020, pp. 25–36.

[163] Nicolas Lambert and Yoav Shoham. "Truthful surveys". In: *International Workshop on Internet and Network Economics*. Springer. 2008, pp. 154–165.

[164] Maya Larson, Chunqiang Hu, Ruinian Li, Wei Li, and Xiuzhen Cheng. "Secure auctions without an auctioneer via verifiable secret sharing". In: *Proceedings of the 2015 Workshop on Privacy-Aware Mobile Computing*. ACM. 2015, pp. 1–6.

[165] Arnaud Laurent, Luce Brotcorne, and Bernard Fortz. "Transactions fees optimization in the Ethereum blockchain". In: *Blockchain: Research and Applications* (2022), p. 100074.

[166] Eugene L Lawler. "Fast approximation algorithms for knapsack problems". In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE. 1977, pp. 206–213.

[167] Thomas Léauté and Boi Faltings. "Protecting privacy through distributed computation in multi-agent decision making". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 649–695.

[168] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek. "FRODO 2.0: An open-source framework for distributed constraint optimization". In: *Proceedings of the IJCAI" 09 Distributed Constraint Reasoning Workshop (DCR" 09)*. LIA-CONF-2010-002. 2009, pp. 160–164.

[169] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. "Truth revelation in approximately efficient combinatorial auctions". In: *Journal of the ACM (JACM)* 49.5 (2002), pp. 577–602.

[170] Victor Lesser, Charles L Ortiz Jr, and Milind Tambe. *Distributed sensor networks: A multiagent perspective*. Vol. 9. Springer Science & Business Media, 2003.

[171] Yang Li, Yangyang Yu, Haohang Li, Zhi Chen, and Khaldoun Khashanah. "TradingGPT: Multi-Agent System with Layered Memory and Distinct Characters for Enhanced Financial Trading Performance". In: *arXiv preprint arXiv:2309.03736* (2023).

[172] JG Liao. "Variance reduction in Gibbs sampler using quasi random numbers". In: *Journal of Computational and Graphical Statistics* 7.3 (1998), pp. 253–266.

[173] Friedrich Liese and Igor Vajda. "Convex Statistical Distances". In: *Statistical Inference for Engineers and Data Scientists* (2018).

[174] Hsiao-Ying Lin and Wen-Guey Tzeng. "An efficient solution to the millionaires' problem based on homomorphic encryption". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2005, pp. 456–466.

[175] Xin Liu, Shundong Li, XiuBo Chen, Gang Xu, Xiaolin Zhang, and Yong Zhou. "Efficient solutions to two-party and multiparty millionaires' problem". In: *Security and Communication Networks* 2017 (2017).

[176] Yanchao Liu. “A multi-agent semi-cooperative unmanned air traffic management model with separation assurance”. In: *EURO Journal on Transportation and Logistics* 10 (2021), p. 100058.

[177] Magnus Ljungberg and Andrew Lucas. “The OASIS air-tra c management system”. In: *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence, PRICAI*. Vol. 92. 1992, pp. 185–190.

[178] Yuan Lu, Qiang Tang, and Guiling Wang. “Dragoon: private decentralized hits made practical”. In: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2020, pp. 910–920.

[179] Yuan Lu, Qiang Tang, and Guiling Wang. “Zebralancer: Private and anonymous crowdsourcing system atop open blockchain”. In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, pp. 853–865.

[180] Benjamin Lubin and David C Parkes. “Approximate strategyproofness”. In: *Current Science* (2012), pp. 1021–1032.

[181] Rajiv Maheswaran, Milind Tambe, Emma Bowring, Jonathan Pearce, and Pradeep Varakantham. “Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling”. In: (2004).

[182] Rajiv T Maheswaran, Jonathan P Pearce, Emma Bowring, Pradeep Varakantham, and Milind Tambe. “Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications”. In: *Autonomous Agents and Multi-Agent Systems* 13.1 (2006), pp. 27–60.

[183] Rajiv T. Maheswaran, Jonathan P. Pearce, and Milind Tambe. “A Family of Graphical-Game-Based Algorithms for Distributed Constraint Optimization Problems”. In: *Coordination of Large-Scale Multiagent Systems*. Ed. by Paul Scerri, Régis Vincent, and Roger Mailler. Boston, MA: Springer US, 2006, pp. 127–146.

[184] Padala Manisha, CV Jawahar, and Sujit Gujar. "Learning Optimal Redistribution Mechanisms Through Neural Networks". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2018, pp. 345–353.

[185] Yifan Mao and Shaileshh Bojja Venkatakrishnan. "Less is More: Fairness in Wide-Area Proof-of-Work Blockchain Networks". In: *arXiv preprint arXiv:2204.02461* (2022).

[186] Carlos Marın-Lora, Miguel Chover, José M Sotoca, and Luis A Garcıa. "A game engine to make games as multi-agent systems". In: *Advances in Engineering Software* 140 (2020), p. 102732.

[187] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[188] Eric Maskin and JJ Laffont. "A differential approach to expected utility maximizing mechanisms". In: *Aggregation and revelation of preferences* (1979).

[189] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. "A Smart Contract for Boardroom Voting with Maximum Voter Privacy." In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 110.

[190] John McMillan. "Selling spectrum rights". In: *Journal of Economic Perspectives* 8.3 (1994), pp. 145–162.

[191] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A survey on bias and fairness in machine learning". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–35.

[192] Ralph C Merkle. "A digital signature based on a conventional encryption function". In: *Conference on the theory and application of cryptographic techniques*. Springer. 1987, pp. 369–378.

[193] Johnnatan Messias, Mohamed Alzayat, Balakrishnan Chandrasekaran, and Krishna P Gummadi. "On Blockchain Commit Times: An analysis of how miners choose Bitcoin transactions". In: *The Second International Workshop on Smart Data for Blockchain and Distributed Ledger (SDBD2020)*. 2020.

[194] Silvio Micali, Michael Rabin, and Salil Vadhan. "Verifiable random functions". In: *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE. 1999, pp. 120–130.

[195] Silvio Micali and Michael O Rabin. "Cryptography miracles, secure auctions, matching problem verification". In: *Communications of the ACM* 57.2 (2014), pp. 85–93.

[196] Nolan Miller, Paul Resnick, and Richard Zeckhauser. "Eliciting informative feedback: The peer-prediction method". In: *Management Science* 51.9 (2005), pp. 1359–1373.

[197] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. "ADOPT: Asynchronous distributed constraint optimization with quality guarantees". In: *Artificial Intelligence* 161.1-2 (2005), pp. 149–180.

[198] Payman Mohassel and Yupeng Zhang. "Secureml: A system for scalable privacy-preserving machine learning". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 19–38.

[199] Hiraku Morita, Nuttapong Attrapadung, Tadanori Teruya, Satsuya Ohata, Koji Nuida, and Goichiro Hanaoka. "Constant-round client-aided secure comparison protocol". In: *European Symposium on Research in Computer Security*. Springer. 2018, pp. 395–415.

[200] Moin Hussain Moti, Dimitris Chatzopoulos, Pan Hui, and Sujit Gujar. "FaRM: Fair Reward Mechanism for Information Aggregation in Spontaneous Localized Settings". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*. ijcai.org, 2019, pp. 506–512.

[201] Roger B Myerson. "Optimal auction design". In: *Mathematics of operations research* 6.1 (1981), pp. 58–73.

[202] Roger B Myerson and Mark A Satterthwaite. "Efficient mechanisms for bilateral trading". In: *Journal of economic theory* 29.2 (1983), pp. 265–281.

[203] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[204] Moni Naor, Benny Pinkas, and Reuban Sumner. "Privacy preserving auctions and mechanism design". In: *Proceedings of the 1st ACM conference on Electronic commerce*. ACM. 1999, pp. 129–139.

[205] Yadati Narahari. *Game theory and mechanism design*. Vol. 4. World Scientific, 2014.

[206] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. "Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy". In: *arXiv preprint arXiv:2009.03561* (2020).

[207] John Nash. "Non-cooperative games". In: *Annals of mathematics* (1951), pp. 286–295.

[208] John F Nash Jr. "Equilibrium points in n-person games". In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[209] Duc Thien Nguyen, William Yeoh, Hoong Chuin Lau, and Roie Zivan. "Distributed gibbs: A linear-space sampling-based DCOP algorithm". In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 705–748.

[210] Minh Nguyen-Duc, Zahia Guessoum, Olivier Marin, Jean-François Perrot, and Jean-Pierre Briot. *A multi-agent approach to reliable air traffic control*. na, 2008.

[211] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521872820.

[212] Raz Nissim and Ronen Brafman. "Distributed heuristic forward search for multi-agent planning". In: *Journal of Artificial Intelligence Research* 51 (2014), pp. 293–332.

[213] Pablo Noriega and Carles Sierra. "Auctions and multi-agent systems". In: *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*. Springer, 1999, pp. 153–175.

[214] *Oraclize*. URL: http://www.oraclize.it/.

[215] *Oraclize Official Documentation*. URL: http://docs.oraclize.it.

[216] Ariel Orda and Ori Rottenstreich. "Enforcing fairness in blockchain transaction ordering". In: *Peer-to-peer Networking and Applications* 14.6 (2021), pp. 3660–3673.

[217] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. "DUCT: An upper confidence bound approach to distributed constraint optimization problems". In: *AAAI*. 2012, pp. 528–534.

[218] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. "DUCT: An upper confidence bound approach to distributed constraint optimization problems". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.5 (2017), pp. 1–27.

[219] Manisha Padala, Sankarshan Damle, and Sujit Gujar. "Learning Equilibrium Contributions in Multi-Project Civic Crowdfunding". In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 368–375. ISBN: 9781450391153.

[220] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 223–238.

[221] Nicolas Papernot, Martın Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. "Semi-supervised knowledge transfer for deep learning from private training data". In: *arXiv preprint arXiv:1610.05755* (2016).

[222] Nicolas Papernot and Thomas Steinke. “Hyperparameter tuning with renyi differential privacy”. In: *arXiv preprint arXiv:2110.03620* (2021).

[223] *Parameter Generation - Zcash*. URL: https://z.cash/ko_KR/technology/paramgen/.

[224] Jaehyoung Park, Dong Seong Kim, and Hyuk Lim. “Privacy-Preserving Reinforcement Learning Using Homomorphic Encryption in Cloud Computing Infrastructures”. In: *IEEE Access* 8 (2020), pp. 203564–203579. DOI: 10.1109/ACCESS.2020.3036899.

[225] David C Parkes, Michael O Rabin, Stuart M Shieber, and Christopher Thorpe. “Practical secrecy-preserving, verifiably correct and trustworthy auctions”. In: *Electronic Commerce Research and Applications* 7.3 (2008), pp. 294–312.

[226] David C Parkes, Michael O Rabin, and Christopher Thorpe. “Cryptographic combinatorial clock-proxy auctions”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2009, pp. 305–324.

[227] H Van Dyke Parunak. “Manufacturing experience with the contract net”. In: *Distributed artificial intelligence* 1 (1987), pp. 285–310.

[228] Marshall Pease, Robert Shostak, and Leslie Lamport. “Reaching agreement in the presence of faults”. In: *Journal of the ACM (JACM)* 27.2 (1980), pp. 228–234.

[229] Torben Pryds Pedersen. “Non-interactive and information-theoretic secure verifiable secret sharing”. In: *Annual International Cryptology Conference*. Springer. 1991, pp. 129–140.

[230] Adrian Petcu and Boi Faltings. “DPOP: A scalable method for multiagent constraint optimization”. In: *IJCAI 05*. CONF. 2005, pp. 266–271.

[231] Ryan Porter and Yoav Shoham. “On cheating in sealed-bid auctions”. In: *Proceedings of the 4th ACM conference on Electronic commerce*. 2003, pp. 76–84.

[232] Gujar Sujit Prakash. "Novel Mechanisms for Allocation of Heterogeneous Items in Strategic Settings". PhD thesis. Indian Institute of Science Bangalore, India, 2012.

[233] *PREDICTING BITCOIN FEES FOR TRANSACTIONS*. https://bitcoinfees.earn.com/. 2022.

[234] ProPublica. *Bias in ML*. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. 2020.

[235] Goran Radanovic and Boi Faltings. "A robust bayesian truth serum for non-binary signals". In: *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI" 13)*. 2013, pp. 833–839.

[236] Abirami Raja Santhi and Padmakumar Muthuswamy. "Influence of blockchain technology in manufacturing supply chain and logistics". In: *Logistics* 6.1 (2022), p. 15.

[237] Stephen J Rassenti, Vernon L Smith, and Robert L Bulfin. "A combinatorial auction mechanism for airport time slot allocation". In: *The Bell Journal of Economics* (1982), pp. 402–417.

[238] Éverton Rodrigues Reis. "PROFTS: a multi-agent automated trading system." PhD thesis. Universidade de São Paulo, 2019.

[239] Alfréd Rényi. "On measures of entropy and information". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Vol. 4. University of California Press. 1961, pp. 547–562.

[240] Ronald L Rivest and Michael O Rabin. "Practical Provably Correct Voter Privacy Protecting End-to-End Voting Employing Multiparty Computations and Split Value Representations of Votes". In: (2014).

[241] Ronald L Rivest, Adi Shamir, and Leonard M Adleman. *Cryptographic communications system and method*. US Patent 4,405,829. Sept. 1983.

[242] Matthew J. B. Robshaw. "One-Way Function". In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 887–888. ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_467.

[243] Mike Rosser. *Basic mathematics for economists*. London: Routledge, 2003.

[244] Michael H Rothkopf, Aleksandar Pekeč, and Ronald M Harstad. "Computationally manageable combinational auctions". In: *Management science* 44.8 (1998), pp. 1131–1147.

[245] Tim Roughgarden. "Transaction Fee Mechanism Design". In: *ACM Conference on Economics and Computation (ACM EC)*. 2021, p. 792.

[246] Tim Roughgarden. "Transaction Fee Mechanism Design". In: *CoRR* abs/2106.01340 (2021).

[247] Pierre Rust, Gauthier Picard, and Fano Ramparany. "pyDCOP: a DCOP library for Dynamic IoT Systems". In: *International Workshop on Optimisation in Multi-Agent Systems*. 2019.

[248] Tuomas Sandholm. "An algorithm for optimal winner determination in combinatorial auctions". In: (1999).

[249] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized anonymous payments from bitcoin". In: *2014 IEEE S&P*. 2014, pp. 459–474.

[250] Mark Allen Satterthwaite. "Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions". In: *Journal of economic theory* 10.2 (1975), pp. 187–217.

[251] Julien Savaux, Julien Vion, Sylvain Piechowiak, René Mandiau, Toshihiro Matsui, Katsutoshi Hirayama, Makoto Yokoo, Shakre Elmane, and Marius Silaghi. "Privacy

stochastic games in distributed constraint reasoning". In: *Annals of Mathematics and Artificial Intelligence* 88 (2020), pp. 691–715.

[252] Julien Savaux, Julien Vion, Sylvain Piechowiak, René Mandiau, Toshihiro Matsui, Katsutoshi Hirayama, Makoto Yokoo, Shakre Elmane, and Marius Silaghi. "Utilitarian Approach to Privacy in Distributed Constraint Optimization Problems." In: *FLAIRS Conference*. 2017, pp. 454–459.

[253] Florian Alexander Schmidt. "The good, the bad and the ugly: Why crowdsourcing needs ethics". In: *2013 International Conference on Cloud and Green Computing*. IEEE. 2013, pp. 531–535.

[254] Claus-Peter Schnorr. "Efficient identification and signatures for smart cards". In: *Advances in Cryptology—CRYPTO'89 Proceedings 9*. Springer. 1990, pp. 239–252.

[255] Reinhard Selten. "Axiomatic characterization of the quadratic scoring rule". In: *Experimental Economics* 1.1 (1998), pp. 43–61.

[256] Reza Shokri and Vitaly Shmatikov. "Privacy-preserving deep learning". In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (CCS)*. 2015, pp. 1310–1321.

[257] Shoeb Siddiqui, Ganesh Vanahalli, and Sujit Gujar. "BitcoinF: Achieving Fairness For Bitcoin In Transaction Fee Only Model". In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 2008–2010.

[258] Shoeb Siddiqui, Ganesh Vanahalli, and Sujit Gujar. "BitcoinF: Achieving Fairness For Bitcoin In Transaction Fee Only Model". In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2020, pp. 2008–2010.

[259] Herbert A Simon. "Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in Society Setting." In: (1957).

[260] Corwin Smith. *PROOF-OF-STAKE REWARDS AND PENALTIES*. ethereum.org/en/developers/docs/consensus-mechanisms/pos/rewards-and-penalties/. [Online]. 2023.

[261] Reid G Smith. "The contract net protocol: High-level communication and control in a distributed problem solver". In: *IEEE Transactions on computers* 29.12 (1980), pp. 1104–1113.

[262] Yaakov Sokolik and Ori Rottenstreich. "Age-aware fairness in blockchain transaction ordering". In: *IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE. 2020, pp. 1–9.

[263] Michael Spain, Sean Foley, and Vincent Gramoli. "The impact of ethereum throughput and fees on transaction latency during icos". In: *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*. 2020.

[264] Starbase. *Starbase*. Available at https://starbase.co/. 2016. URL: https://starbase.co/.

[265] Hugo Steihaus. "The problem of fair division". In: *Econometrica* 16 (1948), pp. 101–104.

[266] Jihoon Suh and Takashi Tanaka. "SARSA (0) reinforcement learning over fully homomorphic encryption". In: *2021 SICE International Symposium on Control Systems (SICE ISCS)*. IEEE. 2021, pp. 1–7.

[267] Koutarou Suzuki and Makoto Yokoo. "Secure combinatorial auctions by dynamic programming with polynomial secret sharing". In: *International Conference on Financial Cryptography*. Springer. 2002, pp. 44–56.

[268] Katia Sycara, Anandeep Pannu, M Willamson, Dajun Zeng, and Keith Decker. "Distributed intelligent agents". In: *IEEE expert* 11.6 (1996), pp. 36–46.

[269] Katia P Sycara. "Multiagent systems". In: *AI magazine* 19.2 (1998), pp. 79–79.

[270] Andrea Tacchetti, DJ Strouse, Marta Garnelo, Thore Graepel, and Yoram Bachrach. "A neural architecture for designing truthful and efficient auctions". In: *arXiv preprint arXiv:1907.05181* (2019).

[271] Pavel Tarasov and Hitesh Tewari. "Internet Voting Using Zcash." In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 585.

[272] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. "A Privacy Preserving Collusion Secure DCOP Algorithm". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 4774–4780.

[273] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. "PC-SyncBB: A privacy preserving collusion secure DCOP algorithm". In: *Artificial Intelligence* 297 (2021), p. 103501.

[274] Tamir Tassa, Tal Grinshpoun, and Roie Zivan. "Privacy preserving implementation of the Max-Sum algorithm and its variants". In: *Journal of Artificial Intelligence Research* 59 (2017), pp. 311–349.

[275] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. "Max-Sum Goes Private." In: *IJCAI*. Vol. 1360. 2015, pp. 425–431.

[276] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. "Preserving Privacy in Region Optimal DCOP Algorithms." In: *IJCAI*. 2016, pp. 496–502.

[277] Technologyify. *Wireless Sensor Networks – Definition, Applications, Design Issue, and More*. 2023. URL: https://www.technologyify.com/wireless-sensor-networks/.

[278] Enrico Tedeschi, Tor-Arne S. Nordmo, Dag Johansen, and Håvard D. Johansen. "On Optimizing Transaction Fees in Bitcoin Using AI: Investigation on Miners Inclusion Pattern". In: *ACM Trans. Internet Technol.* 22.3 (2022).

[279] Chen-Khong Tham and J-C Renaud. "Multi-agent systems on sensor networks: A distributed reinforcement learning approach". In: *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE. 2005, pp. 423–429.

[280] *The Costs of the 2014 European Parliamentary Elections*. 2016. URL: https://www.gov.uk/government/uploads/system/uploads/attachment%5C_data/file/573723/costs%5C_of%5C_the%5C_2014%5C_european%5C_parliamentary%5C_elections.pdf.

[281] Mahnush Movahedi Timo Hanke and Dominic Williams. "DFINITY Technology Overview Series Consensus System". In: CoRR abs/1805.04548 (2018), 2018. URL: https://arxiv.org/pdf/1805.04548.pdf.

[282] Tradewise. *Automated Trading Systems: Everything You Need To Know*. 2024. URL: https://tradewise.community/automated-trading-systems/.

[283] Aleksei Triastcyn and Boi Faltings. "Bayesian Differential Privacy for Machine Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020.

[284] William Vickrey. "Counterspeculation, auctions, and competitive sealed tenders". In: *The Journal of finance* 16.1 (1961), pp. 8–37.

[285] S. K. Vivek, R. S. Yashank, Y. Prashanth, N. Yashas, and M. Namratha. "E-Voting Systems using Blockchain: An Exploratory Literature Survey". In: *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2020, pp. 890–895. DOI: 10.1109/ICIRCA48905.2020.9183185.

[286] Baoxiang Wang and Nidhi Hegde. "Privacy-preserving q-learning with functional noise in continuous spaces". In: *Advances in Neural Information Processing Systems* 32 (2019).

[287] Roger Wattenhofer. *The science of the blockchain*. CreateSpace Independent Publishing Platform, 2016.

[288] Jianhao Wei, Yaping Lin, Xin Yao, and Jin Zhang. "Differential privacy-based location protection in spatial crowdsourcing". In: *IEEE Transactions on Services Computing* (2019).

[289] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3454–3469. DOI: 10.1109/TIFS.2020.2988575.

[290] WeiFund. *WeiFund - Decentralised Fundraising*. 2015. URL: http://weifund.io/.

[291] Michael Weiss, Benjamin Lubin, and Sven Seuken. "Sats: A universal spectrum auction test suite". In: *AAMAS*. 2017, pp. 51–59.

[292] Derek Wenning. "Comparative Analysis of Simultaneous Ascending Auctions and Combinatorial Auctions". In: *Indiana University* (2016).

[293] Bitcoin Wikipedia. *Historic rules for free transactions*. https://en.bitcoin.it/wiki/Miner_fees. 2022.

[294] Wikipedia contributors. *Spacehive — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Spacehive&oldid=1081450857. 2022.

[295] David Williams. *Probability with martingales*. Cambridge university press, 1991.

[296] Jens Witkowski and David C Parkes. "A Robust Bayesian Truth Serum for Small Populations." In: *AAAI*. Vol. 12. 2012, pp. 1492–1498.

[297] Gavin Wood. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151 (2014), pp. 1–32.

[298] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[299] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[300] Andrew C Yao. "Protocols for secure computations". In: *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE. 1982, pp. 160–164.

[301] Y. Yao, J. Wei, J. Liu, and R. Zhang. "Efficiently secure multiparty computation based on homomorphic encryption". In: *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*. 2016, pp. 343–349.

[302] YCHARTS. *Bitcoin Transactions Per Day*. ycharts.com/indicators/bitcoin_transactions_per_day. 2022.

[303] YCHARTS. *Ethereum Transactions Per Day*. ycharts.com/indicators/ethereum_transactions_per_day. 2022.

[304] Haibo Yi. "Securing e-voting based on blockchain in P2P network". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (2019), pp. 1–9.

[305] Makoto Yokoo, Edmund H Durfee, Toru Ishida, and Kazuhiro Kuwabara. "The distributed constraint satisfaction problem: Formalization and algorithms". In: *IEEE Transactions on knowledge and data engineering* 10.5 (1998), pp. 673–685.

[306] Makoto Yokoo, O. Etzioni, T. Ishida, and N. Jennings. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-Agent Systems*. Berlin, Heidelberg: Springer-Verlag, 2001. ISBN: 3540675965.

[307] Bin Yu, Joseph K Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. "Platform-independent secure blockchain-based voting system". In: *International Conference on Information Security*. Springer. 2018, pp. 369–386.

[308] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning". In: *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 2020, pp. 493–506.

[309] Zhichao Zhao and T-H Hubert Chan. "How to vote privately using bitcoin". In: *International Conference on Information and Communications Security*. Springer. 2015, pp. 82–96.

[310] Zishuo Zhao, Xi Chen, and Yuan Zhou. "Bayesian-Nash-Incentive-Compatible Mechanism for Blockchain Transaction Fee Allocation". In: *Crypto Economics Security Conference (CESC)*. 2022.

[311] James Zou, Sujit Gujar, and David Parkes. "Tolerable manipulability in dynamic assignment without money". In: *AAAI*. 2010, pp. 947–952.

[312] Robertas Zubrickas. "The provision point mechanism with refund bonuses". In: *Journal of Public Economics* 120 (2014), pp. 231–234.