Gamu Blue: A Practical Tool for Game Theory Security Equilibria *

Ameer Taweel
Koç University

Burcu Yıldız
EPFL

Alptekin Küpçü
Koç University

March 29, 2024

Abstract

The application of game theory in cybersecurity enables strategic analysis, adversarial modeling, and optimal decision-making to address security threats' complex and dynamic nature. Previous studies by Abraham et al. and Biçer et al. presented various definitions of equilibria to examine the security aspects of games involving multiple parties. Nonetheless, these definitions lack practical and easy-to-use implementations. Our primary contribution is addressing this gap by developing Gamu Blue, an easy-to-use tool with implementations for computing the equilibria definitions including k-resiliency, ℓ-repellence, t-immunity, (ℓ, t)-resistance, and m-stability.

1 Introduction

Game theory provides a powerful toolkit for understanding, analyzing, and addressing the complex challenges in cybersecurity. Cybersecurity researchers and professionals use game theory for strategic analysis, adversarial modeling, optimal decision-making, risk/vulnerability analysis, collaborative security, and protocol/mechanism design [1–5]. By applying game-theoretic concepts, one can analyze the interactions between attackers and defenders, identify optimal strategies, and understand the implications of different decisions in a dynamic and adversarial environment. It also facilitates collaborative approaches where multiple parties can coordinate their actions to enhance security. Moreover, game-theoretic principles guide the design of secure protocols and mechanisms that incentivize desired behaviors and discourage malicious activities [2–4, 6–8].

Security solutions based on game theory rely on the rational behavior of the parties involved. In scenarios involving multiple parties, it is often safe to assume that individuals will not engage in malicious actions if those actions are detrimental to themselves. As a result, one can focus on addressing threats that

*This paper has been accepted for publication by IEEE under copyright © 2024 IEEE.

are advantageous to the attacker rather than trying to countermeasure against all possible threats.

Previously, [9] and [6] presented equilibria definitions for analyzing games' security in the multi-party setting, but there are not any existing implementations to compute these equilibria. Our main contribution is developing Gamu Blue, an easy-to-use tool with implementations for computing these equilibria. Our tool allows game theory and cybersecurity researchers to analyze their games using the security-related game-theoretic equilibria definitions. Moreover, it provides a baseline to compare future algorithmic improvements against previous ones.

2 Preliminaries

2.1 Basics

Definition 1 (n-Player Normal-Form Game). An n-player perfect information normal-form game can be defined by a set of players $N = \{p_1, \ldots, p_n\}$, a set of actions $A_i = \{a_i^1, \ldots, a_i^{m_i}\}$ for each player i, and a utility function U_i for each player i that gives the utility of player i, given the actions chosen by all players. All players have complete and perfect knowledge of the game's state, including the other players' utility functions and available actions.

Strategy s_i of player i is a probability distribution over the possible actions of the player i. s_i is a pure strategy if it always chooses the same action. Otherwise, s_i is a mixed strategy. We will only consider pure strategies in this paper because our algorithms rely on exhaustive search, and there are infinitely many mixed strategies. $s_i(a)$ is the probability the strategy s_i assigns to the action a. $s_D = \{s_{p_a}, \ldots, s_{p_b}\}$ is a strategy profile (an ordered set) containing the strategies of all players in the set $D = \{p_a, \ldots, p_b\}$. s_{-i} is the strategy profile of all the players excluding player i. s_{-D} the strategy profile of all the players that are not in D. Given a strategy profile s_N , we denote by $U_i(s_N)$ the expected utility of the player i, if the players in the game play the strategy profile s_N.

Definition 2 (Nash Equilibrium). For any game with the player set N, a strategy profile $s_N = \{s_{p_1}, \ldots s_{p_n}\}$ is a Nash equilibrium if $\forall i \in N \ \forall s_i' \neq s_i \ U_i(s_N) \geq U_i(s_i' \cup s_{-i})$.

Nash equilibrium is a fundamental concept in game theory that describes a state in which no player has an incentive to unilaterally deviate from their chosen strategy, given the strategies of other players. John Nash proved in [10] that every finite game (games with a finite number of players and actions per player) has at least one Nash equilibrium.

Definition 3 (Weakly Dominant Strategy of a Player). A strategy s_i of a player i is its weakly dominant strategy if for all strategies $s_i' \neq s_i$ of i and for all strategy profiles s_{-i}' of players other than i, $U_i(s_i \cup s_{-i}') \geq U_i(s_i' \cup s_{-i}')$. A weakly dominant strategy is not guaranteed to exist.

Definition 4 (Coalition Utility). Let s_N be a strategy profile for all players in a game. Then, the utility $U_C(s_N)$ of a coalition C is defined as $U_C(s_N) = \sum_{i \in C} U_i(s_N)$.

This coalition utility definition simplifies the analysis by abstracting out how participants share utility internally.

Definition 5 (Weakly Dominant Strategy of a Coalition). A strategy profile s_C of a coalition C is its weakly dominant strategy if for all strategy profiles $s'_C \neq s_C$ of C and for all strategy profiles s'_{-C} of the players outside C: $U_C(s_C \cup s'_{-C}) \geq U_C(s'_C \cup s'_{-C})$.

2.2 Multi-Party Secure Equilibria Definitions Introduced In [9]

Definition 6 (k-resiliency). Given a player set $C \subseteq N$, s_C is a group best response for C to s_{-C}, if for all strategies s'_C played by C and $\forall i \in C$, we have $U_i(s_C \cup s_{-C}) \geq U_i(s'_C \cup s_{-C})$. A joint strategy s_N is a k-resilient equilibrium, if $\forall C \subseteq N$ with $|C| \leq k$, s_C is a group best response for C to s_{-C}, where $s_N = s_C \cup s_{-C}$.

The authors of [9] proposed k-resiliency to protect the mechanism outcome against a single coalition. If s_N is k-resilient, it is secure against any single coalition of size less than or equal to k because no such coalition can increase the utility of any of its members by deviating from s_N.

Definition 7 (t-immunity). A joint strategy s_N is a t-immune equilibrium, if $\forall T \subseteq N$ with $|T| \leq t$, for all strategies s'_T played by the players in T, and $\forall i \notin T$, we have $U_i(s'_T \cup s_{-T}) \geq U_i(s_N)$, where $s_N = s_T \cup s_{-T}$.

[9] proposed t-immunity to protect the mechanism outcome against Byzantine players. If s_N is t-immune, it is secure against up to t Byzantine players because such players cannot decrease the utility of any other player by deviating from s_N.

Definition 8 ((k,t)-robustness). A joint strategy s_N is a (k,t)-robust equilibrium, if $\forall C \subseteq N$, $\forall T \subseteq N$ such that $C \cap T = \emptyset$, $|C| \leq k$, and $|T| \leq t$, for all strategies s'_T played by the players in T, and for all strategies s'_C played by the players in C, and $\forall i \in C$, we have $U_i(s_{-T} \cup s'_T) \geq U_i(s_{-(C \cup T)} \cup s'_C \cup s'_T)$, where $s_N = s_C \cup s_T \cup s_{-(C \cup T)}$.

The authors of [9] proposed (k,t)-robustness to protect the mechanism outcome against a single coalition and Byzantine players. If s_N is (k,t)-robust, it is secure against up to t Byzantine players and any single rational coalition of size up to k: the Byzantine players cannot harm the other players by deviating from s_N, and no single coalition can increase the utility of any of its members by deviating from s_N.

2.3 Improved Multi-Party Secure Equilibria Definitions Introduced In [6]

Definition 9 (ℓ-repellence). Given a player set $C \subseteq N$, s_C is a best collective response for C to s_{-C}, if for all strategies s'_C played by C, we have $U_C(s_C \cup s_{-C}) \geq U_C(s'_C \cup s_{-C})$. A joint strategy s_N is an ℓ-repellent equilibrium, if $\forall C \subseteq N$ with $|C| \leq \ell$, s_C is a best collective response for C to s_{-C}, where $s_N = s_C \cup s_{-C}$.

The authors of [6] proposed ℓ-repellence to replace k-resiliency. ℓ-repellence analyzes the security of a mechanism against the presence of a single coalition. If s_N is ℓ-repellent, it is secure against any single coalition of size up to ℓ because no such coalition can increase its *total utility* by deviating from s_N. Note that ℓ-repellence improves k-resiliency because it employs the transferable utility assumption [11] and abstracts out how the utility is shared among the participants internally. Thus, rather than considering the individual members of a coalition, it considers the total benefit of the coalition.

Definition 10 ((ℓ, t)-resistance). A joint strategy s_N is an (ℓ, t)-resistant equilibrium, if $\forall C, T \subseteq N$ s.t. $C \cap T = \emptyset$, $|C| \leq \ell$, and $|T| \leq t$, for all strategies s'_T played by the players in T, and for all strategies s'_C played by C, we have $U_C(s_{-T} \cup s'_T) \geq U_C(s_{-(C \cup T)} \cup s'_T \cup s'_T)$, where $s_N = s_C \cup s_T \cup s_{-(C \cup T)}$.

[6] proposed (ℓ, t)-resistance to replace (k, t)-robustness. (ℓ, t)-resistance analyzes the security of a mechanism against the presence of a single coalition and Byzantine players. If s_N is (ℓ, t)-resistant, it is secure against up to t Byzantine players and any single coalition of size up to ℓ: the Byzantine players cannot harm the other players by deviating from s_N, and no single coalition can increase its total utility by deviating from s_N. (ℓ, t)-resistance improves over (k, t)-robustness by employing the transferable utility assumption.

Definition 11 (m-stability). A joint strategy s_N is an m-stable equilibrium, if for all natural numbers $p \leq |N|$ and for all coalitions $C_1, \ldots, C_p$ satisfying the following conditions:

1. $1 \leq |C_1|, \ldots, |C_p| \leq m$
2. $C_1 \cup \cdots \cup C_p = N$
3. $\forall i, j \in \{1, \ldots, p\}$ such that $i \neq j$, $C_i \cap C_j = \emptyset$

we have that for each $i = 1, \ldots, p$, the strategy s_{C_i} is weakly dominant for the coalition C_i and that $s_{C_1} \cup \cdots \cup s_{C_p} = s_N$.

The authors of [6] proposed m-stability to analyze the security of a mechanism against the presence of multiple coalitions. If s_N is m-stable, it is secure against any number of coalitions of size up to m because no coalition has incentive to deviate from s_N, even if it expects others to deviate from s_N. Like other definitions introduced in [6], m-stability employs the transferable utility assumption. [6] proved that $(n-1)$-stability implies $(n-1)$-repellence and $(\ell, n-\ell)$-resistance for all $\ell \leq n-1$.

3 Algorithms

We mainly focus on the optimization problem with one strategy profile. Consider, for example, ℓ-repellence: Given a game G and a strategy profile s_N, find the maximum c such that s_N is c-repellent. We can define similar sub-problems for all security equilibria definitions.

Computing Nash Equilibrium is PPAD-Complete. [12] showed that the problem of computing a Nash equilibrium with at least three players is PPAD-complete. [13] showed that the problem is also PPAD-complete for two players.

The class of PPAD problems contains computational problems for which polynomial-time algorithms are not known to exist. Therefore, PPAD problems are considered intractable.

Some of our security equilibria, like k-resiliency and ℓ-repellence, are also Nash Equilibria, so computing them is also PPAD-complete. Therefore, one should not expect our implementations to be efficient.

k-resiliency and ℓ-repellence. Algorithm 1 finds the maximal k (or ℓ), in $O(2^n \times S \times n)$ time, where n is the number of players, and S is the total number of strategy profiles. The two outer for loops go over all possible coalitions (combinations) of N, which is $O(2^n)$. Then we loop over all possible strategy profiles for each coalition, which is $O(S)$. For k-resiliency, we loop over the members of the current coalition, which is $O(n)$. For ℓ-repellence, we compute the coalition utility of each coalitions, which internally calls the utility functions of all the coalition members, which is also $O(n)$. Note that $S = \prod_i |A_i|$, so S is exponential in the number of players and number of actions.

Algorithm 1 Given a game G and a strategy profile s_N, find the maximum c such that s_N is c-resilient/repellent.

```
1: INPUT: G, s_N, type
2: N ← G.players
3: n ← |N|
4: for c ← 1, n do
5:    for C ∈ N.combinationsOfSize(c) do
6:       s_{-C} ← G.remainingStrategies(C)
7:       for s'_C ∈ G.possibleStrategies(C) do
8:          if type = resiliency then
9:             for i ∈ C do
10:               if U_i(s_N) < U_i(s'_C ∪ s_{-C}) then
11:                  return c − 1
12:         if type = repellence then
13:            if U_C(s_N) < U_C(s'_C ∪ s_{-C}) then
14:               return c − 1
15: return n
```

t-immunity. Algorithm 2 finds the maximal t in $O(2^n \times S \times n)$ time. The two outer for loops go over all possible groups of Byzantine players (combinations)

of N, which is $O(2^n)$. Then we loop over all possible strategy profiles for each group, which is $O(S)$. Finally, we loop over the players outside of the current Byzantine group, which is $O(n)$.

Algorithm 2 Given a game G and a strategy profile s_N, find the maximum c such that s_N is c-immune.

```
1: INPUT: G, s_N
2: N ← G.players
3: n ← |N|
4: for c ← 1, n do
5:    for T ∈ N.combinationsOfSize(c) do
6:       s_{-T} ← G.remainingStrategies(T)
7:       for s'_T ∈ G.possibleStrategies(T) do
8:          for i ∉ C do
9:             if U_i(s'_T ∪ s_{-T}) < U_i(s_N) then
10:               return c − 1
11: return n
```

(k,t)-robustness and (ℓ,t)-resistance. Algorithm 3 finds maximal k, t, ℓ in $O(2^{2n} \times S^2 \times n)$ time. The two outer for loops go over all possible coalitions (combinations) of N, then we loop over all possible groups of Byzantine players (combinations) of $N - C$, which is $O(2^{2n})$. Then we loop over all possible strategy profiles for each coalition and each Byzantine group, which is $O(S^2)$. For (k,t)-robustness, we loop over the members of the current coalition, which is $O(n)$. For (ℓ,t)-resistance, we compute the coalition utility of each coalitions, which is also $O(n)$.

m-stability. Algorithm 4 finds the maximal m in $O(2^n \times S^2 \times n)$ time. The two outer for loops go over all possible coalitions (combinations) of N, which is $O(2^n)$. Then we loop over all possible strategy profiles for players outside of the coalition, which is $O(S)$. We then loop over all possible strategy profile for players of the coalition, which is $O(S)$ as well. Finally, we compute the coalition utility of each coalitions, which is $O(n)$.

4 Experimental Results

4.1 Multi-Party Games

In this section, we briefly describe two games that we use in our experiments.

Incentivized Outsourced Computation (IOC) [2]: The authors of [2] defined an Incentivized Outsourced Computation (IOC) game. The game involves a boss and n rational contractors as players. The boss wants to outsource the execution of a costly algorithm to the contractors. Each contractor has the option to choose between the diligent strategy and the lazy strategy. The diligent strategy involves running the correct algorithm, which costs $c(1)$, while the lazy strategy uses a less costly algorithm known as the "q algorithm", which

Algorithm 3 Given a game G and a strategy profile s_N, find the set of maximal c_1 and c_2 such that s_N is (c_1, c_2)-robust/resistant.

INPUT: G, s_N, *type*
$S \leftarrow \{\}$
$N \leftarrow G.players$
$n \leftarrow |N|$
LABEL: Outer Loop
for $c_1 \leftarrow 1, n-1$ **do**
  for $C \in N.combinationsOfSize(c_1)$ **do**
    for $c_2 \leftarrow 1, n - c_1$ **do**
      for $T \in (N - C).combinationsOfSize(c_2)$ **do**
        $s_{-(C \cup T)} \leftarrow G.remainingStrategies(C \cup T)$
        for $s'_T \in G.possibleStrategies(T)$ **do**
          for $s'_C \in G.possibleStrategies(C)$ **do**
            if *type* $=$ *robustness* **then**
              for $i \in C$ **do**
                if $U_i(s'_T \cup s_{-T}) < U_i(s'_T \cup s'_C \cup s_{-(C \cup T)})$ **then**
                  if $t = 1$ **then**
                    return S
                  $S.append((c_1, c_2 - 1))$
                  continue Outer Loop
            if *type* $=$ *resistance* **then**
              if $U_C(s'_T \cup s_{-T}) < U_C(s'_T \cup s'_C \cup s_{-(C \cup T)})$ **then**
                if $t = 1$ **then**
                  return S
                $S.append((c_1, c_2 - 1))$
                continue Outer Loop
  $S.append((c_1, n - c_1))$
return S

Algorithm 4 Given a game G and a strategy profile s_N, find the maximum c such that s_N is m-stable.

INPUT: G, s_N
$N \leftarrow G.players$
$n \leftarrow |N|$
for $c \leftarrow 1, n$ **do**
  for $C \in N.combinationsOfSize(c)$ **do**
    $s_C \leftarrow G.remainingStrategies(N - C)$
    for $s'_{-C} \in G.possibleStrategies(N - C)$ **do**
      for $s'_C \in G.possibleStrategies(C)$ **do**
        if $U_C(s_C \cup s'_{-C}) < U_C(s'_C \cup s'_{-C})$ **then**
          return $c - 1$
return n

costs $c(q)$. The q algorithm has a probability q of producing the correct output and is assumed to be the same for all lazy players. If all contractors provide the same output, the boss accepts it as correct and rewards each contractor with reward r. However, if there is a difference in the outputs, the diligent contractors collaborate with the boss to identify and penalize the lazy ones. In this case, the diligent contractors receive the reward r and an additional bounty b, while each contractor contributes a share towards the total bounties and incurs a fixed fine f. Table 1 illustrates the resulting expected utility matrix. Note that for this mechanism to be meaningful, it is necessary that $c(q) < c(1) < r$.

Table 1: IOC Payoff Matrix (where $0 < k < n$)

Others / This	Diligent	Lazy
All diligent	$r - c(1)$	$rq - (f + b(n-1))(1-q) - c(q)$
k lazy	$r + b(1-q) - c(1)$	$rq - (f + \frac{b(n-k-1)}{k+1})(1-q) - c(q)$
All lazy	$r + b(1-q) - c(1)$	$r - c(q)$

The desired outcome of the game is all players choosing the diligent strategy. [6] proved that IOC is not 2-resilient. They also proved that for $n > 2$, if the boss sets the reward and bounty as $r(n-1)/(n-2) \geq b > r/(1-q)$, then IOC is $(n-1)$-stable. Moreover, they proved that IOC is $(n-1)$-immune.

Forwarding Dilemma (FD) [14]: The authors of [14] introduced the Forwarding Dilemma (FD) game as a model to study the forwarding behavior of flooded packets in wireless ad hoc networks. Network nodes are the players of FD. Each node receives the same flooded packet. Each player can either forward the packet or drop it. Two factors determine the utility of each player: the network gain factor g and the forwarding cost c. Table 2 shows the specific utilities for each player, considering their strategy and the strategies of others. It is essential to have $c < g$.

Table 2: FD Payoff Matrix

Others / This	Forward	Drop
All drop	$g - c$	0
At least one forward	$g - c$	g

The desirable strategies in this game are when one player forwards the packet, and the rest drops it. [6] proved that FD is not 2-resilient, is n-repellent, is not $(1,1)$-resistant, is not $(1,1)$-robust, is not 1-immune, and is not 1-stable.

4.2 Input Format

We implemented Gamu Blue in Python, using the Python interface to the Gambit[1] library. The source code is openly available on GitHub.

[1] `https://github.com/gambitproject/gambit`

The Gambit library is a powerful tool for game-theory analysis, offering a range of features. One of its key strengths is its implementation of various algorithms for computing Nash Equilibria, providing researchers and practitioners with efficient methods to analyze strategic interactions. Additionally, Gambit includes a user-friendly graphical user interface (GUI) that facilitates the examination of small games, allowing users to visualize game structures and explore their strategic dynamics. Moreover, Gambit offers a Python interface, enabling developers to leverage its game representations and develop custom game-theory algorithms. This flexibility and accessibility make the Gambit library a valuable resource for studying and understanding game-theoretic concepts and applications.

Gamu Blue accepts the NFG and AGG game representations defined by the Gambit library. The core difference is that AGG tends to be much more compact than NFG for highly structured games. The Gambit library's documentation contains a detailed explanation of both representations. Figures 1 and 2 show example input representations for IOC with $N = 3$, $cost(1) = 10$, $cost(q) = 5$, $q = 0.5$, $r = 20$, $b = 20$, and $f = 2.5$ in NFG and AGG formats, respectively. Figures 3 and 4 show example input representations for FD with $N = 3$, $g = 2$, and $c = 1$ in NFG and AGG formats, respectively.

```
NFG 1 R "IOC"
{"P1" "P2" "P3"}
{{"L" "D"} {"L" "D"} {"L" "D"}}
 15.0  15.0  15.0 20.0 -1.25 -1.25
-1.25  20.0 -1.25 20.0  20.0 -16.25
-1.25 -1.25  20.0 20.0 -16.25 20.0
-16.25 20.0  20.0 10.0  10.0  10.0
```

Figure 1: IOC Represented in NFG Format For 3 Players

```
#AGG
3 2 0
2 2 2
0 1 0 1 0 1
2 0 1 2 0 1
0 -16.25 -1.25 15.0
0  20.0   20.0 20.0
```

Figure 2: IOC Represented in AGG Format For 3 Players

```
NFG 1 R "FD"
{"P1" "P2" "P3"}
{{"D" "F"} {"D" "F"} {"D" "F"}}
0.0 0.0 0.0 1.0 2.0 2.0
2.0 1.0 2.0 1.0 1.0 2.0
2.0 2.0 1.0 1.0 2.0 1.0
2.0 1.0 1.0 1.0 1.0 1.0
```

Figure 3: FD Represented in NFG Format For 3 Players

```
#AGG
3 2 0
2 2 2
0 1 0 1 0 1
2 0 1 2 0 1
0 2.0 2.0 0.0
0 1.0 1.0 1.0
```

Figure 4: FD Represented in AGG Format For 3 Players

4.3 Timing Benchmarks

Experimental Setup. We conducted the experiments on a machine with an Intel Core i7-8750H 2.20GHz processor, 16GB of DDR4 RAM, and 512GB of SSD storage. We ran each algorithm on each input 100 times. We report the mean running times of these runs.

Figure 5 for k-resiliency shows that Algorithm 1 terminates quickly (less than one second for 12 players). These timings are due to neither IOC nor FD being 2-resilient, as shown by [6]. The timings have fluctuations because external operating system factors have a significant impact on this small scale.

[6] proved that IOC is $(n-1)$-repellent and FD is n-repellent. The timings in Figure 6 align with this, and they also show the exponential nature of Algorithm 1.

In Figure 7, we can see the rapid growth of Algorithm 2 timings in IOC as the number of players increases. However, Algorithm 2 does not seem to get slower on FD. The reason is that IOC is $(n-1)$-immune, while FD is not even 1-immune, as proven by [6], and hence the algorithm terminates early finding this fact.

Similarly in Figure 8, we can see the rapid growth of Algorithm 3 timings in IOC as the number of players increases. However, Algorithm 3 does not seem to get slower on FD. The reason is that FD is not $(1,1)$-robust, as proven by [6], while the experiments show that IOC is $(1, n-1)$-robust.

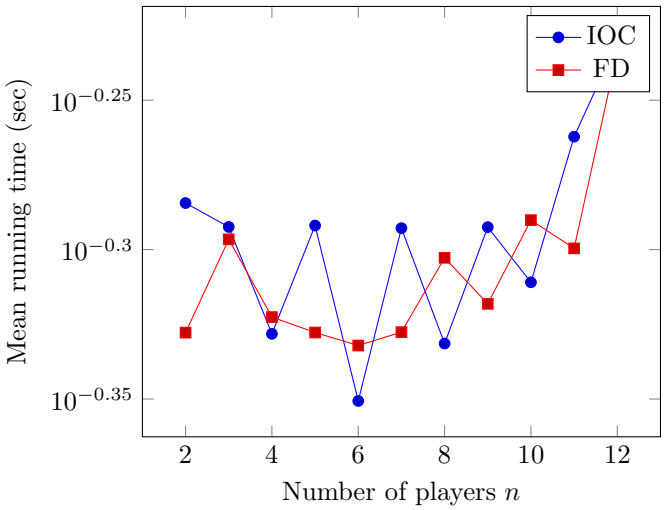


Figure 5: Experimental Performance of k-resiliency

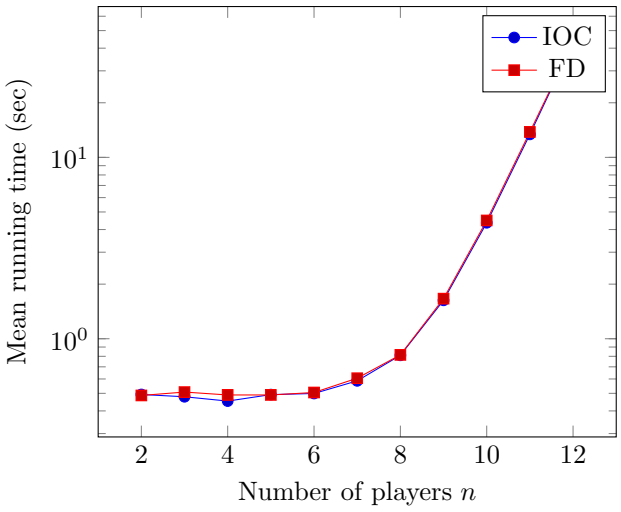


Figure 6: Experimental Performance of ℓ-repellence

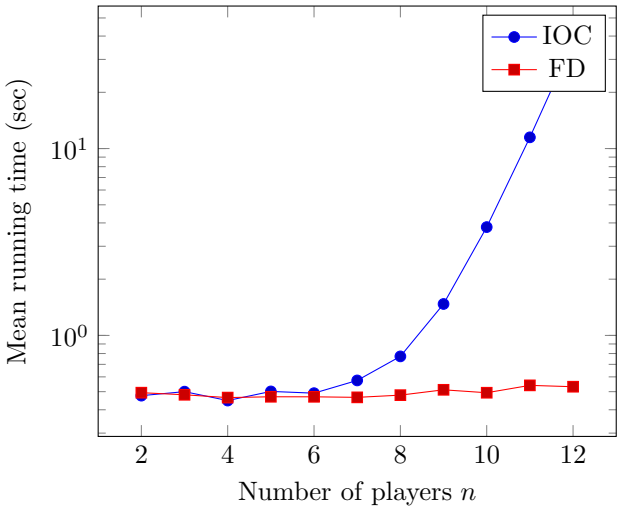


Figure 7: Experimental Performance of t-immunity

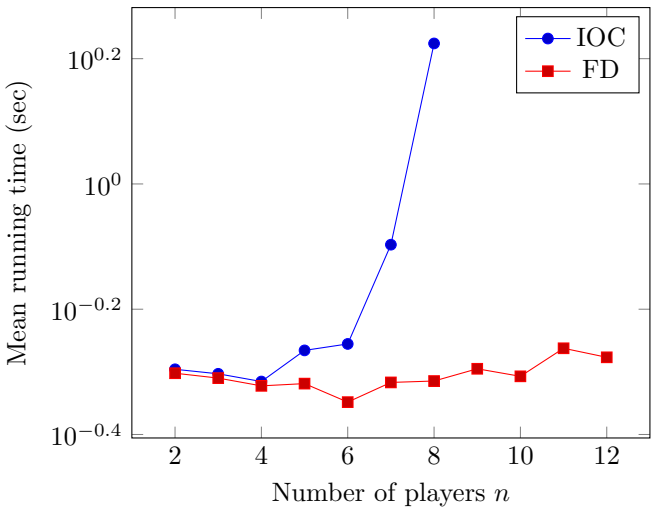


Figure 8: Experimental Performance of (k, t)-robustness

Again in Figure 9, we can see the rapid growth of Algorithm 3 timings in IOC as the number of players increases. However, Algorithm 3 does not seem to get slower on FD. The reason is that IOC is $(\ell, n-\ell)$-resistant for all $\ell \leq n-1$, while FD is not even $(1,1)$-resistant, as proven by [6].

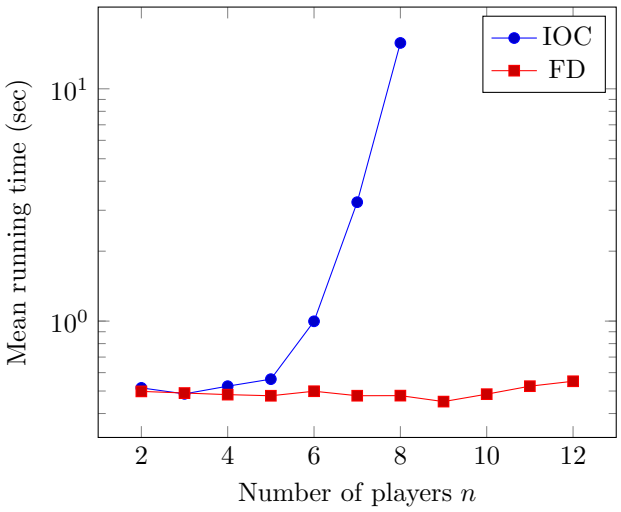


Figure 9: Experimental Performance of (ℓ, t)-resistance

In Figure 10, we can see the rapid growth of Algorithm 4 timings in IOC as the number of players increases. However, Algorithm 4 does not seem to get slower on FD. The reason is that IOC is $(n-1)$-stable, while FD is not even 1-stable, as proven by [6]. All these results confirm that our algorithms return the maximal k, ℓ, t, m values as soon as they are found.

5 Conclusion

We implemented a tool for computing equilibria definitions: k-resiliency, t-immunity, (k,t)-robustness, ℓ-repellence, (ℓ,t)-resistance, and m-stability. We also analyzed the worst-case time complexity of each of the algorithms. We conducted experiments on the Incentivized Outsourced Computation (IOC) and Forwarding Dilemma (FD) games. Our tool is available open source: github.com/CRYPTO-KU/GamuBlue-Game-Theory-Equilibrium-Finder

Acknowledgements

We thank TÜBİTAK (the Scientific and Technological Research Council of Turkey) project 119E088.

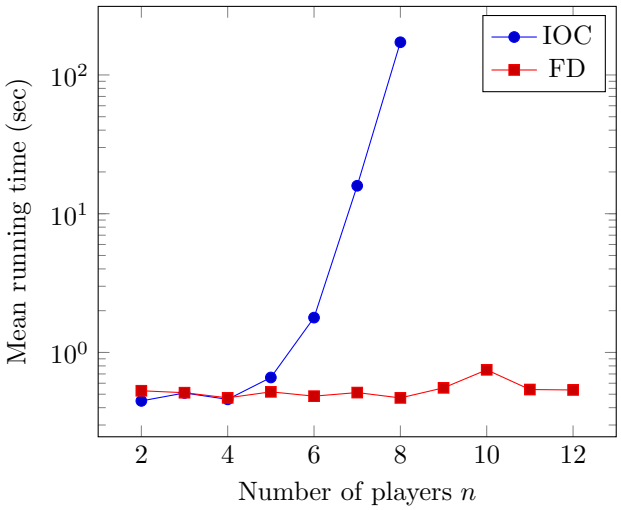


Figure 10: Experimental Performance of m-stability

References

[1] I. Abraham, D. Dolev, R. Gonen, and J. Halpern, "Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation," in *ACM PODC '06*.

[2] A. Küpçü, "Incentivized outsourced computation resistant to malicious contractors," *IEEE TDSC*, vol. 14, no. 6, pp. 633–649, 2017.

[3] S. Eidenbenz, V. S. A. Kumar, and S. Zust, "Equilibria in topology control games for ad hoc networks," in *ACM DIALM-POMC '03*.

[4] S. Kamara and A. Küpçü, "Dogfish: Decentralized optimistic game-theoretic file sharing," in *Applied Cryptography and Network Security*, 2018, pp. 696–714.

[5] Z. Tian, X. Gao, S. Su, J. Qiu, X. Du, and M. Guizani, "Evaluating reputation management schemes of internet of vehicles based on evolutionary game theory," *IEEE TVT*, vol. 68, no. 6, pp. 5971–5980, 2019.

[6] O. Bicer, B. Yildiz, and A. Küpçü, "M-stability: Threshold security meets transferable utility," in *ACM CCSW '21*.

[7] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya, "Incentivizing outsourced computation," in *ACM NetEcon '08*.

[8] A. Küpçü and R. Safavi-Naini, "Smart contracts for incentivized outsourcing of computation," in *ESORICS CBT*, 2021, pp. 245–261.

[9] I. Abraham, D. Dolev, R. Gonen, and J. Halpern, "Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation," in *ACM PODC '06*.

[10] J. Nash, *Non-cooperative games*. Cambridge University Press, 1989, p. 82–94.

[11] H. Peters, *Game Theory: A Multi-Leveled Approach*. Springer, 2015.

[12] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, 2009.

[13] X. Chen and X. Deng, "Settling the complexity of two-player nash equilibrium," in *IEEE FOCS'06*.

[14] M. Naserian and K. Tepe, "Game theoretic approach in routing protocol for wireless ad hoc networks," *Ad Hoc Networks*, vol. 7, no. 3, pp. 569–578, 2009.