Game-Theoretically Secure Distributed Protocols for Fair Allocation in Coalitional Games

T-H. Hubert Chan
The University of Hong Kong
Hong Kong, China
hubert@cs.hku.hk

Qipeng Kuang
The University of Hong Kong
Hong Kong, China
kuangqipeng@connect.hku.hk

Quan Xue
The University of Hong Kong
Hong Kong, China
csxuequan@connect.hku.hk

ABSTRACT

We consider game-theoretically secure distributed protocols for coalition games that approximate the Shapley value with small multiplicative error. Since all known existing approximation algorithms for the Shapley value are randomized, it is a challenge to design efficient distributed protocols among mutually distrusted players when there is no central authority to generate unbiased randomness. The game-theoretic notion of maximin security has been proposed to offer guarantees to an honest player's reward even if all other players are susceptible to an adversary.

Permutation sampling is often used in approximation algorithms for the Shapley value. A previous work in 1994 by Zlotkin et al. proposed a simple constant-round distributed permutation generation protocol based on commitment scheme, but it is vulnerable to rushing attacks. The protocol, however, can detect such attacks.

In this work, we model the limited resources of an adversary by a violation budget that determines how many times it can perform such detectable attacks. Therefore, by repeating the number of permutation samples, an honest player's reward can be guaranteed to be close to its Shapley value. We explore both high probability and expected maximin security. We obtain an upper bound on the number of permutation samples for high probability maximin security, even with an unknown violation budget. Furthermore, we establish a matching lower bound for the weaker notion of expected maximin security in specific permutation generation protocols. We have also performed experiments on both synthetic and real data to empirically verify our results.

KEYWORDS

Secure distributed protocols; Coalitional games; Shapley value

ACM Reference Format:
T-H. Hubert Chan, Qipeng Kuang, and Quan Xue. 2025. Game-Theoretically Secure Distributed Protocols for Fair Allocation in Coalitional Games. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

A coalitional game (*aka* cooperative game) involves n players in the set $\mathcal{N}$, and the utility function $v : 2^{\mathcal{N}} \to \mathbb{R}_+$ determines the reward $v(S)$ obtained by a coalition $S \subseteq \mathcal{N}$ when players cooperate together. An extensively researched problem in this context is the fair allocation of the reward among the n players.

The *Shapley value* is an allocation concept in coalitional games, which has many applications and can provide a fair and consistent framework to assess and distribute rewards among different agents, leading to a more equitable, sustainable, and inclusive system. For instance, it can be used to develop an axiomatic framework for attribution in online advertising [3], measure the significance of nodes in the context of network centrality [9] and develop a pricing scheme of personal recommendations [14]. Computing the exact Shapley value is #P-hard [12], so approximations are used in practice. We focus on the notion of *ϵ-multiplicative error*, where each player receives at least $(1 - \epsilon)$ fraction of its Shapley value, with $\epsilon > 0$ as a constant.

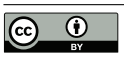
All known efficient algorithms for approximating the Shapley value require randomness, and one approach involves generating uniformly random permutations of all players. However, this study focuses on the *distributed setting* where no *trusted authority* provides randomness. Each player has its own local source of randomness. The objective is to design an efficient *protocol* for allocating rewards close to the Shapley value with a small multiplicative error, while accounting for players' incentives to deviate.

For distributed protocols, the game-theoretic concept of *maximin security* [11] is introduced to offer security guarantees from the perspective of an *honest* player, where all other players may be *susceptible* to an *adversary*. For coalitional games, the security guarantee can be (i) the expected reward received by a player has ϵ-multiplicative error, or (ii) the reward achieves this multiplicative error with high probability.

One previous attempt has been made to design a secure distributed protocol for returning an allocation in a coalitional game. Zlotkin et al. [28] proposed NaivePerm, in which players commit to uniformly random permutations of all players in the first round, opening the commitments in the second round. The composition of all the opened permutations determines each player's marginal contribution. Provided that all players must send their messages in each round at the same time, the expected reward of a player is its Shapley value.

However, if the adversary is *rushing* (i.e., in each round, it can act after seeing an honest player's message), then in some games, by instructing at most one susceptible player to abort, it can create a permutation such that the honest player get its smallest possible reward. Nevertheless, such an attack is limited to detected violations. This motivates the concept of *violation budget* that quantifies the number of violations that the adversary can make.

We also adopt the random permutation approach, and analyze the sampling complexity of random permutations, which we call *P-samples*. Our idea is that the number of P-samples used in a protocol



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

should depend on two factors: (1) ensuring a small multiplicative error with high probability and (2) mitigating bias from adversaries due to detected violations.

In this paper, we design maximin secure distributed protocols that return allocations that approximate the Shapley value. Moreover, we analyze upper and lower bounds on the sampling complexity under different violation budget models. Furthermore, our theoretical results are verified empirically by performing experiments on both real and synthetic data.

Related Work. Shapley value [27] satisfies symmetry, group rationality and additivity, and has become the *de facto* fairness notion. Shapley value also has applications areas such as evaluating the importance of data sources in a learning model [20] and assessing the contribution of data to database queries [13, 23, 26].

However, computing the Shapley value is intractable. Deng et al. [12] proved that calculating the Shapley value for weighted voting games is #P-complete. Elkind et al. [15] proved that it is NP-hard to decide whether the Shapley value of a certain player is 0 in the weighted voting game; hence, it is NP-hard to achieve constant multiplicative error for general games. Bachrach et al. [1] proved the impossibility of achieving super-polynomially small additive error with high probability if the randomized algorithm only samples the utility function for a polynomial number of times.

Supermodular coalitional game models the situation in which it is more beneficial for players to collaborate together, an example of which is multicast cost sharing [16]. Games not having supermodular utility functions may still be reduced to supermodular games; see [12, 18, 19] for instances. However, computing the Shapley value exactly is still hard for supermodular games. In fact, Liben-Nowell et al. [21, Theorem 6] showed that any randomized approach will still need at least $\Omega(\frac{1}{n\epsilon})$ oracle accesses to the utility function to achieve ϵ-multiplicative error. Similar to the result in [1], this means that a super-polynomially small ϵ would require a super-polynomially number of accesses to the utility function. Moreover, they also showed [21, Theorem 5] that any deterministic algorithm that achieves a multiplicative error of at most $\epsilon = \frac{1}{2n}$ must make at least a super-polynomial number of samples on the utility function. In contrast, for this value of ϵ, a polynomial number of samples is sufficient for randomized algorithms.

Approximation approaches by sampling random permutations have been well studied, which is known as *simple random sampling* [5, 28]. More refined analysis with respect to the ranks in random permutations, which is known as *stratified sampling* [17, 24], has also been studied. Finally, by re-interpreting the Shapley value as the expectation of a process that randomly samples subsets of players, multilinear sampling methods [25] have also been investigated. Other heuristic algorithms can be found in [8].

The sampling complexity of algorithms has also been extensively studied. For general games, a simple application of the Hoeffding inequality can give a sufficient number of permutation samples. However, note that each permutation sample involves $\Theta(n)$ oracle accesses to the utility function, but only two of which are relevant to a single player. To avoid this extra factor of $\Omega(n)$ in the sampling complexity on the utility function, Jia et al. [20] has considered a different sampling procedure that first approximates the differences of Shapley values between all pairs of players.

For the special case of supermodular games, Liben-Nowell et al. [21] used the Chebyshev's inequality to give an upper bound of the permutation sampling complexity to achieve multiplicative approximation. We show that this can be easily improved by using the stronger Chernoff Bound.

Zlotkin et al. [28] proposed a distributed protocol (which we call NaivePerm) that samples one random permutation, from which an allocation is derived. As shown in Section 3, just using one sample is not sufficient under our adversarial model.

The game-theoretic notion of *maximin security* has been first proposed by Chung et al. [11] to analyze how the reward for an honest player in a distributed protocol can be protected in an environment where other players may behave adversarially. Subsequently, this notion has been applied to distributed protocols for other well-known problems such as *leader election* [10] and *ordinal random assignment problem* [6]. Both upper and lower bounds on the round complexity have been analyzed in [10].

Additional Material. Some technical proofs are deferred to the full version [7]. Experiment results are given in Section 6.

2 PRELIMINARIES

2.1 Coalitional Game

A *coalitional game* is characterized by $(\mathcal{N}, v)$, where $\mathcal{N}$ is a set of $n = |\mathcal{N}|$ players and $v : 2^{\mathcal{N}} \to \mathbb{R}_+$ is a utility function indicating that a coalition S of players can obtain a revenue of $v(S)$.

Marginal Contribution. The *marginal contribution* of player i joining a subset $S \not\ni i$ is: $\mu_i(S) \coloneqq v(S \cup \{i\}) - v(S)$. Observe that μ_i is defined with respect to v. In this work, we assume that the utility function v is monotone, i.e., the marginal contribution μ_i is always non-negative.

Dividing Up the Reward in the Grand Coalition. From monotonicity, we have for all $S \subseteq \mathcal{N}$, $v(S) \leq v(\mathcal{N})$. An interesting question is how to divide up the received reward $v(\mathcal{N})$ among the players. Given a game $(\mathcal{N}, v)$, an *allocation* is a (non-negative) vector $x \in \mathbb{R}_+^{\mathcal{N}}$ such that $\sum_{i \in \mathcal{N}} x_i = v(\mathcal{N})$, i.e., x_i represents the profit allocated to player i.

Shapley Value [27]. This is an allocation vector whose definition is recalled as follows. For player i, its Shapley value is:

$$\phi_i = \frac{1}{n} \sum_{S \subseteq \mathcal{N} \setminus \{i\}} \binom{n-1}{|S|}^{-1} \mu_i(S) = \frac{1}{n!} \sum_{\sigma} \mu_i(P_i(\sigma)), \tag{1}$$

where the last summation is over all $n!$ permutations $\sigma : [n] \to \mathcal{N}$ of players, and $P_i(\sigma)$ is the set of *predecessors* of player i in the permutation σ.

Random Permutation Interpretation. One way to interpret Equation (1) is to distribute the total reward $v(\mathcal{N})$ through a random process. This process involves sampling a uniformly random permutation (*aka* P-sample) that determines the order in which players join the coalition one by one. The reward of a player is its marginal contribution when joining existing players already in the coalition. The Shapley value ϕ_i is exactly the expected reward of player i in this random process. The following parameter will be used to analyze the number of P-samples used in our protocols.

Max-to-Mean Ratio Γ. This parameter is determined by the utility function v. For player i, let $U_{\max,i} := \max_{S \subseteq \mathcal{N}\setminus\{i\}} \mu_i(S)$ be its maximum marginal contribution; its *max-to-mean ratio* is $\Gamma_i = \frac{U_{\max,i}}{\phi_i}$, where we use the convention $\frac{0}{0} = 1$. We denote $\Gamma := \max_{i\in\mathcal{N}} \Gamma_i$.

Expectation vs High Probability. Typical measure concentration results, such as Chernoff Bound (Fact 4.1), show that the number of samples needed to accurately estimate the mean increases linearly with the max-to-mean ratio Γ.

Source of Randomness. The above sampling methods assume that there is an unbiased source of randomness. In this work, we are interested in the scenario where there is no central authority to generate the randomness. Instead, randomness needs to be jointly generated from the players via distributed protocols.

2.2 Distributed and Security Model

We clarify the model of distributed protocols that we will employ.

Communication Model. We consider a *synchronized* communication model, in which each player can post messages to some broadcast channel (such as a ledger [2]) in *rounds*. We assume that the *Byzantine broadcast* problem has already been solved, i.e., a message posted by any player (even adversarial) in a round will be seen by all players at the end of that round.

Distributed Protocol Model. A distributed protocol specifies the behavior of each player in $\mathcal{N}$, each of which has its own independent source of randomness. In the context of a coalitional game, we assume oracle access to the utility function v.

Adversarial Model. We analyze the protocol from the perspective of some honest player i^*, where all other players in $\mathcal{N} \setminus \{i^*\}$ may be *susceptible* to the adversary. Specifically, for a susceptible player, the adversary can control its randomness, observe its internal state, and dictate its actions. We assume that the adversary is *rushing*, i.e., it can wait and see the messages from player i^* in a round before it decides the actions of the susceptible players in that round.

Commitment Scheme. Assuming the existence of one-way functions/permutations, there is a constant-round publicly verifiable commitment scheme [22] that is perfectly correct, perfectly binding, and concurrent non-malleable. The *commit* operation allows a player to construct a *commitment* c of some secret message m. The *open* operation allows the player to reveal the secret message m from previously committed c, where perfectly binding means that it is impossible to open the commitment to any other message different from m. For simplicity, we assume that each of the commitment operation and the open operation can be performed in a single round.

Ideal Commitment Scheme. For ease of presentation, we assume that the commitment scheme is also *perfectly hiding*. This means that the adversary cannot learn anything about the secret message from the commitment c. Other than this restriction, we allow the adversary to perform any other computation (even exponential in n).

In a real-world scheme, the commitment is only *computationally hiding*. Formally, for some security parameter λ, this would introduce an extra multiplicative factor $(1 - \mathsf{negl}(\lambda))$ in the reward of the honest player for some negligible function $\mathsf{negl}(\cdot)$. However, since we will be considering $(1-\epsilon)$-approximation for constant $\epsilon > 0$, this extra factor of $(1 - \mathsf{negl}(\lambda))$ may be absorbed into ϵ.

Protocol Violation. We shall see that because of the ideal commitment scheme, the only way an adversary can harm an honest player is to instruct a susceptible player to refuse opening some previously committed message, where such a *violation* can be detected. On the other hand, deviating from the protocol's description in sampling randomness cannot be detected in our model and is therefore not considered a violation. Below are violation models that we consider.

- *Violation Budget.* The violation budget C is the maximum number of violations that the adversary can make throughout the protocol, where C can either be known or unknown to the protocol.
- *Violation Rate.* The violation rate $f \in [0,1]$ means that for every $T > 0$, during the generation of the first T number of P-samples, there can be at most fT violations.

We will investigate the following notions of security for our protocols.

Definition 2.1 (Maximin Security for Shapley Value). Suppose Π is a (randomized) distributed protocol between players $\mathcal{N}$ in some coalitional game $(\mathcal{N}, v)$, where there is at least one honest user. The purpose of Π is to return an allocation that is close to the Shapley vector ϕ. Suppose that when Π is run against some adversary Adv, it always terminates with some output allocation vector $x \in \mathbb{R}_+^{\mathcal{N}}$, i.e., $\sum_{i\in\mathcal{N}} x_i = v(\mathcal{N})$.

- *Expectation.* We say that Π is ϵ-expected maximin secure against Adv if, for any honest player i^*, its expected allocation satisfies $\mathbf{E}[x_{i^*}] \geq (1-\epsilon)\cdot\phi_{i^*}$.
- *High probability.* We say that Π is (ϵ,δ)-maximin secure against Adv, if, for any honest player i^*, with probability at least $1-\delta$, its allocation is $x_{i^*} \geq (1-\epsilon)\cdot\phi_{i^*}$.
  Adaptive Security. In this case, the protocol Π returns an allocation x and also a parameter $\epsilon \in [0,1]$. We say that it is δ-adaptively maximin secure, if, with probability at least $1-\delta$, $x_{i^*} \geq (1-\epsilon)\cdot\phi_{i^*}$

2.3 Max-to-Mean Ratio Γ in Special Games

Lemma 2.2 (Γ Ratio for Monotone Games). *In a game where the marginal contribution of every player is non-negative,* $\Gamma \leq n \cdot \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}$.

In the full version [7], we give an example that attains the upper bound. Games with special utility functions have been studied.

Supermodular Games. In additional to non-negativitiy and monotonicity of v, a supermodular (or convex) game is associated with a supermodular utility function $v : 2^{\mathcal{N}} \to \mathbb{R}_+$. In other words, for all $S, T \subseteq \mathcal{N}$, $v(S) + v(T) \leq v(S \cup T) + v(S \cap T)$.

Rank Preference. For $j \in [n]$, define $U_j := \mathbf{E}[\mu_{i^*}(P_{i^*}(\sigma))|\sigma(i^*) = j]$, which is the expected reward of player i^* conditioning on it having rank j in σ. The following lemma justifies why rank 1 is the least preferable in supermodular games.

Lemma 2.3 (Monotonicity of Ranks). *For* $1 \leq j < n$, $U_j \leq U_{j+1}$.

Fact 2.4 (Γ Ratio for Supermodular Games). *For supermodular games,* $\Gamma \leq n$.

Proof. Since player i^* is in the most preferable rank n with probability $\frac{1}{n}$, we have $\phi_{i^*} = \mathbf{E}[U] \geq \frac{1}{n} \cdot U_{\max,i^*}$, where the inequality holds because $U \geq 0$. □

Example: Edge Synergy Game. The edge synergy game [12] was defined on a simple undirected graph, but we extend it to a weighted hypergraph $G = (\mathcal{N}, E, w)$, where each vertex in $\mathcal{N}$ is a player, $E \subseteq 2^{\mathcal{N}}$ and $w : E \to \mathbb{R}_+$ gives the hyperedge weights. The utility function v is defined as $v(S) := \sum_{e \in E: e \subseteq S} w(e)$, which is the sum of edge weights in the induced subgraph $G[S]$. See more details in the full version [7].

Fact 2.5. *The edge synergy game is supermodular and has max-to-mean ratio* $\Gamma \le \max_{e \in E} |e|$.

3 DISTRIBUTED PROTOCOLS BASED ON PERMUTATION SAMPLING

High Level Approach. Based on the random permutation approach in equation (1), we give an abstract protocol in Algorithm 1. It uses a sub-protocol GenPerm that returns a random permutation σ of $\mathcal{N}$, which we shall call a **P-sample**. Given a P-sample σ, we indicate the player at rank j by σ_j, and $\sigma_{<j}$ represents the set of players ranked lower than σ_j. In other words, $\sigma_{<j}$ can be defined as $\{\sigma_i | i < j\}$.

The sub-protocol GenPerm also returns other auxiliary information aux such as whether any player has been detected for violation.

Stopping Condition. To decrease the variance of the output and mitigate the effect of the adversary, we may use multiple numbers of P-samples and take the average allocation over all samples. The stopping condition stopcond specifies when the whole protocol should terminate.

Algorithm 1: Abstract Protocol to Return an Allocation

Input: A game $(\mathcal{N}, v)$, a stopping condition stopcond.
Output: An allocation $x \in \mathbb{R}_+^{\mathcal{N}}$.

$z \leftarrow \vec{0} \in \mathbb{R}_+^{\mathcal{N}}$, $R \leftarrow 0$
while stopcond = false **do**
  $(\sigma, \text{aux}) \leftarrow \texttt{GenPerm}(\mathcal{N})$
  `//Marginal contribution of rank j player in σ`
  for every $j \in [n]$, $z_{\sigma_j} \leftarrow z_{\sigma_j} + \mu_{\sigma_j}(\sigma_{<j})$
  $R \leftarrow R + 1$
  `//May update other variables according to aux`
return $x := \frac{1}{R} \cdot z \in \mathbb{R}_+^{\mathcal{N}}$

Remark 3.1 (Reward Decomposition). *In our analysis of Algorithm 1, it will be convenient to decompose the reward received by the honest player i^* in each P-sample as follows.*

Specifically, in the j-th P-sample, the reward received can be expressed as $X_j := Y_j - Z_j$, where Y_j is the reward received by player i^* had the adversary not caused any violation (if any), and Z_j can be interpreted as the amount of damage due to the adversary's violation in the j-th P-sample.

This allows us to analyze Y_j and Z_j separately. For instance, because there is at least one honest player, we can assume that the Y_j's are independent among themselves, and $\mathbf{E}[Y_j] = \phi_{i^*}$.

On the other hand, the random variables Z_j's can interact with the Y_j's and behave in a very complicated way. However, if we know that the adversary has a violation budget of C, then we can conclude that with probability 1, $\sum_j Z_j \le C U_{\max, i^*} = C\Gamma \cdot \phi_{i^*}$.

3.1 "Secure" Permutation Generation

We revisit a simple permutation generation protocol that was proposed by [28]. We paraphrase it in Algorithm 2, and call it NaivePerm, because it can easily be attacked by a rushing adversary in supermodular games(see Lemma 3.3). However, as we shall later see in Sections 4 and 5, the security property in Lemma 3.2 turns out to be sufficient for us to design good protocols.

Algorithm 2: NaivePerm

Input: Player set N.
Output: A random permutation σ of N, together with a collection Dev of detected violating players.

`//Commit phase:`
foreach player $i \in N$ **do**
  Uniformly sample a permutation $\sigma^{(i)}$ of N.
  Commit $\sigma^{(i)}$ and broadcast the commitment.
`//Open phase:`
foreach player $i \in N$ **do**
  Broadcast the opening for its previously committed $\sigma^{(i)}$.
Denote S as the collection of players i that have followed the commit and the open phases, i.e., its commitment and opened $\sigma^{(i)}$ have been verified.
Using the permutations generated by players in S, compute the composed permutation $\sigma \leftarrow \circ_{i \in S} \sigma^{(i)}$, using the predetermined composition order (such as the lexicographical order of players).
Dev $\leftarrow N \setminus S$
return (σ, Dev)

Lemma 3.2 (Security of NaivePerm). *Suppose in Algorithm 2, the strategy of an adversary* Adv *never causes violation. Then, assuming an ideal commitment scheme, Algorithm 2 returns a uniformly random permutation of players if there exists at least one honest player.*

Proof. Because of the ideal commitment scheme assumption, all players must follow the commit and the open phases to avoid violation being detected. Since there exists an honest player i, its sampled permutation $\sigma^{(i)}$ is uniformly random, whose commitment is assumed to leak no information. Therefore, other players' sampled permutations are independent of $\sigma^{(i)}$, and so, the resulting composed permutation is uniformly random. □

3.2 Attack NaivePerm in Supermodular Games

Recall that by Lemma 2.3, in supermodular games, the expected reward of a player is monotone to its rank in σ. The following Lemma shows NaivePerm can be attacked by a rushing adversary so that the honest player is always at the least favorable position.

Lemma 3.3 (Attacking NaivePerm Using at Most One Abort in Supermodular Games). *Suppose the game is supermodular and a rushing adversary controls $n - 1$ players. Then, by instructing at most 1 player to abort, it can always cause* NaivePerm *to return a permutation such that the remaining honest player i^* is at the least preferable position with probability 1.*

Proof. For simplicity, consider the case that the players are indexed by $[n]$, where $i^* = n$ is the only honest player. The adversary adopts the following strategy. Pick any cycle permutation τ, i.e., the permutation corresponds to a cyclic shift of the players $\mathcal{N}$. Note that by applying the shift n times, we have τ^n equal to the identity permutation.

Then, the j-th susceptible player will commit to τ^j. The key observation is that for $0 \le i \le n-1$, the n numbers $\sum_{j=1}^{n-1} j - i$ are distinct modulo n. This means that by omitting the permutation from at most 1 susceptible player, the adversary can simulate a cyclic shift for i positions, for any $0 \le i \le n-1$.

Therefore, for a rushing adversary, after the honest player i^* opens its committed permutation $\sigma^{(i^*)}$, the adversary can observe the rank of i^* in its permutation. Hence, the adversary can always instruct at most 1 susceptible player to abort opening its committed permutation such that the honest player is shifted to the least preferable position. □

Remark 3.4. *In Lemma 3.3, even if the single violating player is punished by sending it to the least preferable position instead of i^*, player i^* is still at the second least preferable position.*

More "Secure" Permutation Generation. Although NaivePerm has very weak security properties especially in supermodular games, as discussed in Section 4, it is still useful because it can detect violations. This feature allows us to achieve high probability maximin security even when the violation budget is unknown. In Section 5, we will investigate a more secure P-sample protocol and analyze its sampling complexity to achieve expected maximin secure for supermodular games. However, we will find that, surprisingly, the stronger security properties do not lead to a significant advantage in terms of the sampling complexity.

4 SAMPLING COMPLEXITY TO ACHIEVE HIGH PROBABILITY MAXIMIN SECURITY

In this section, we will use NaivePerm to generate P-samples in Algorithm 1. Even though we see in Lemma 3.3 that one P-sample from NaivePerm can easily be attacked, the fact that violation can be detected means that repeating enough number of P-samples can counter the effect of the adversary's limited violation budget. Another reason to have a larger number of P-samples is to achieve high probability maximin security based on measure concentration argument, such as the following variant of Chernoff Bound.

Fact 4.1 (Chernoff Bound). *Suppose for each $j \in [R]$, Y_j is sampled independently from the same distribution with support $[0, U_{\max}]$ such that $\Gamma = \frac{U_{\max}}{\mathbf{E}[Y_j]}$. Then, for any $0 < \epsilon < 1$,*

$$\mathbf{Pr}\left[\sum_{j\in[R]} Y_j \le (1-\epsilon)\cdot \mathbf{E}[\sum_{j\in[R]} Y_j]\right] \le \exp(-\tfrac{\epsilon^2 R}{2\Gamma}).$$

This readily gives a guarantee on the reward of a player when the adversary does not interfere.

Lemma 4.2 (Baseline Reward with No Adversary). *Suppose Algorithm 1 is run with* NaivePerm *and R number of P-samples such that there is no interference from the adversary. Then, for any $0 < \epsilon < 1$, the probability that player i^* receives less than $(1-\epsilon)\cdot\phi_{i^*}$ is at most $\exp(-\frac{\epsilon^2 R}{2\Gamma})$, which is at most δ when $R = \frac{2\Gamma}{\epsilon^2}\log\frac{1}{\delta}$.*

Recall that in a supermodular game, $\Gamma \le n$. By using this fact we can obtain a sampling complexity $O(n)$. We note that [21] has also analyzed the sampling complexity with no adversary in supermodular games. However, instead of using Chernoff Bound, they used the weaker Chebyshev's Inequality. As a result, their bound has a factor of $O(n^2)$, as opposed to $O(n)$ in our bound.

4.1 Warmup: Known Violation Budget

The simple case is that the violation budget C is known in advance. Because each violation can cause a damage of at most $U_{\max,i^*} \le \Gamma\cdot\phi_{i^*}$, it is straightforward to give a high probability statement. Recall that under the violation model, each unit of the violation budget can cause a susceptible player to violate the protocol once in a round. (In this case, actually it is not important whether a violation can be detected.)

Theorem 4.3 (Known Violation Budget). *For any $0 < \epsilon, \delta < 1$, when the violation budget C is known, (ϵ,δ)-maximin security can be achieved by setting the stopping condition* stopcond *as having the number of P-samples reaching: $R = \max\{\frac{8\Gamma}{\epsilon^2}\ln\frac{1}{\delta}, \frac{2C\Gamma}{\epsilon}\}$.*

Proof. For $j \in [R]$, we decompose the reward received by player i^* in the j-th P-sample as $X_j := Y_j - Z_j$, which is described in Remark 3.1. Recall that Y_j is the reward received by player i^* had the adversary not caused any violation (if any), and Z_j can be interpreted as the amount of damage due to the adversary's violation in the j-th P-sample.

By Lemma 4.2, $(1-\frac{\epsilon}{2})$-approximation can be achieved by $\sum_{j\in[R]} Y_j$ with probability at least $1-\delta$, if $R \ge \frac{8\Gamma}{\epsilon^2}\ln\frac{1}{\delta}$. In other words, under this range of R, with probability at least $1-\delta$, we have the lower bound for the sum $\sum_{j\in[R]} Y_j \ge (1-\frac{\epsilon}{2})\cdot R\phi_{i^*}$.

Now, the violation budget C is known, which means that, with probability 1, the adversary causes a total reward deduction $\sum_{j\in[R]} Z_j \le CU_{\max,i^*} \le C\Gamma\phi_{i^*}$, which is at most $\frac{\epsilon}{2}\cdot R\phi_{i^*}$ if we choose $R \ge \frac{2C\Gamma}{\epsilon}$. Therefore, if $R = \max\{\frac{8\Gamma}{\epsilon^2}\ln\frac{1}{\delta}, \frac{2C\Gamma}{\epsilon}\}$, (ϵ,δ)-maximin security is achieved. □

4.2 Unknown Violation Budget

In the case where violation budget is unknown, the stopping condition in Algorithm 1 may no longer specify the pre-determined number of P-samples. However, one advantage of NaivePerm is that it can detect whether a violation has occurred. Therefore, we can use the fraction of P-samples in which violation has occurred in the stopping condition.

Note that if the adversary has an unlimited violation budget, then the protocol may never terminate. Hence, in the most general case, the following lemma does not directly guarantee (ϵ,δ)-maximin security.

Lemma 4.4 (General Unknown Budget). *Suppose the adversary has some unknown violation budget (that may be unlimited). Furthermore, for $0 < \epsilon, \delta < 1$, Algorithm 1 is run with* NaivePerm *and the stopping condition* stopcond *is:*

- *the fraction of P-samples observed so far with detected violations is at most $\frac{\epsilon}{2\Gamma}$; AND*
- *the number of P-samples has reached at least: $R_0 := \frac{8\Gamma}{\epsilon^2}\cdot(\ln\frac{16\Gamma}{\epsilon^2} + \ln\frac{1}{\delta})$.*

Then, the probability that the protocol terminates and player i^* receives an outcome of less than $(1-\epsilon)\phi_{i^*}$ is at most δ.

Proof. For $R \geq R_0$, let δ_R be the probability of the bad event $\mathcal{E}_R$ that the protocol terminates after exactly R number of P-samples and the honest player i^* receives an outcome of less than $(1-\epsilon)\phi_{i^*}$. Our final goal is to show that $\sum_{R\geq R_0} \delta_R \leq \delta$.

Analyzing Fixed $\mathcal{E}_R$. We fix some R and analyze δ_R. Recall that for $j \in [R]$, we consider the reward decomposition $X_j := Y_j - Z_j$ as described in Remark 3.1. Observe that the bad event $\mathcal{E}_R$ implies both of the following have occurred:

- The number of P-samples with detected violation is at most $\frac{\epsilon R}{2\Gamma}$. Therefore, $\sum_{j\in[R]} Z_j \leq \frac{\epsilon R}{2\Gamma} \cdot U_{\max,i^*} \leq \frac{\epsilon}{2} \cdot R\phi_{i^*}$.
- Had the adversary not interfered, the sum is: $\sum_{j\in[R]} Y_j < (1-\frac{\epsilon}{2})R \cdot \phi_{i^*}$.

The first point is part of the stopping condition. If the second point does not happen, this means that $\sum_{j\in[R]}(Y_j - Z_j) \geq (1-\frac{\epsilon}{2})R\cdot\phi_{i^*} - \frac{\epsilon}{2}\cdot R\phi_{i^*} = (1-\epsilon)R\cdot\phi_{i^*}$, thereby violating $\mathcal{E}_R$.

Therefore, we have $\delta_R \leq \mathbf{Pr}\left[\sum_{j\in[R]} Y_j < (1-\frac{\epsilon}{2})R\cdot\phi_{i^*}\right] \leq \exp(-\frac{\epsilon^2 R}{8\Gamma})$, where the last inequality follows from Lemma 4.2.

Analyzing the Union of All Bad Events. Therefore, we have

$$\sum_{R\geq R_0} \delta_R \leq \frac{\exp\left(-\frac{\epsilon^2 R_0}{8\Gamma}\right)}{1-\exp\left(-\frac{\epsilon^2}{8\Gamma}\right)} \leq \frac{16\Gamma}{\epsilon^2}\exp\left(-\frac{\epsilon^2 R_0}{8\Gamma}\right),$$

where the last inequality follows because $1-e^{-t} \geq \frac{t}{2}$ for $t \in (0,1)$. By choosing $R_0 := \frac{8\Gamma}{\epsilon^2}\cdot(\ln\frac{16\Gamma}{\epsilon^2} + \ln\frac{1}{\delta})$, the last expression equals δ. This completes the proof. □

Finite Violation Budget. The adversary has some finite violation budget C that is unknown to the protocol. The stopping condition ensures that the protocol will terminate with probability 1. In this case, (ϵ,δ)-maximin security follows immediately.

Corollary 4.5 (Maximin Security for Finite Unknown Budget). *Suppose in Lemma 4.4, the adversary has some finite (but unknown) violation budget C. Let $0<\epsilon,\delta<1$ and R_0 be as defined in Lemma 4.4. Then, with probability 1, the protocol terminates after at most $\max\{R_0, \frac{2C\Gamma}{\epsilon}\}$ number of P-samples. Moreover, (ϵ,δ)-maximin security is achieved for an honest player.*

4.3 Unknown Violation Rate

The violation rate $f \in [0,1]$ means that for every $T>0$, during the generation of the first T number of P-samples, there can be at most fT violations. However, the protocol does not know f in advance.

Adaptive Maximin Security. The protocol has some target approximation parameter $\epsilon \in [0,1]$. However, whether this is achievable depends on the violation rate f and the max-to-mean ratio Γ.

Algorithm 3 outlines an adaptive algorithm that returns both an allocation x and an approximation parameter $\widehat{\epsilon}$. Theorem 4.6 describes the security property of Algorithm 3.

Theorem 4.6 (Adaptive Maximin Security for Violation Rate). *Suppose the adversary has some (unknown) violation rate $f \in [0,1]$. Furthermore, for any $0<\epsilon,\delta<1$, we execute Algorithm 3 using the* NaivePerm *approach.*

Algorithm 3: Abstract Adaptive Protocol to Return an Allocation

Input: A game $(\mathcal{N}, v)$, Γ, δ, ϵ.
Output: An allocation $x \in \mathbb{R}_+^{\mathcal{N}}$ and ϵ_k.

```
z, x ← 0⃗ ∈ ℝ_+^𝒩, R ← 0
k ← 0, V ← 0
//For all t ≥ 0, denote ε_t := 2^{-t}.
while ε_k > ε do
    //Here aux is the number of violations.
    (σ, aux) ← GenPerm(𝒩)
    //Marginal contribution of rank j player in σ
    for every j ∈ [n], z_{σ_j} ← z_{σ_j} + μ_{σ_j}(σ_{<j})
    R ← R + 1
    V ← V + |aux|
    if R ≥ (8Γ/ε_{k+1}^2) ln(2^{k+1}/δ) then
        if V/R > ε_{k+1}/(2Γ) then
            break
        else
            x ← (1/R) · z
            k ← k + 1
return (x, ε_k) //For any honest player i*, with
    probability at most δ, x_{i*} < (1 − ε_k)φ_{i*}.
```

Then, the protocol returns $(x,\widehat{\epsilon})$ such that $\widehat{\epsilon} \leq \max\{\epsilon, 4f\Gamma\}$ holds with probability 1. Moreover, for any honest player i^*, with probability at least $1-\delta$, its received reward satisfies $x_{i^*} \geq (1-\widehat{\epsilon})\phi_{i^*}$.

Proof. Let (x,ϵ_k) be the returned value. The algorithm terminates either when $\epsilon_k \leq \epsilon$ or $\frac{V}{R} > \frac{\epsilon_{k+1}}{2\Gamma}$. Since $f \geq \frac{V}{R}$ and $\epsilon_{k+1} = \frac{\epsilon_k}{2}$, the second inequality implies that $\epsilon_k < 4f\Gamma$.

Let $\mathcal{B}$ denote the bad event that the protocol terminates and player i^* receives a reward of less than $(1-\epsilon_k)\phi_{i^*}$. Let $\epsilon_t = \frac{1}{2^t}$. Let $\mathcal{B}_t$ denote the event that the protocol terminates with the value of variable k equals t, and player i^* receives a reward of less than $(1-\epsilon_t)\phi_{i^*}$. When $t=0$, $\epsilon_t = 1$. Since i^* will receive a non-negative reward, $\mathbf{Pr}[\mathcal{B}_0] = 0$, so we only focus on the cases when $t \geq 1$. It holds that $\mathcal{B} \subseteq \cup_{t\geq 1}\mathcal{B}_t$, therefore, $\mathbf{Pr}[\mathcal{B}] \leq \mathbf{Pr}[\cup_{t\geq 1}\mathcal{B}_t]$. Our goal is to show that:

$$\mathbf{Pr}[\cup_{t\geq 1}\mathcal{B}_t] \leq \delta.$$

Analyzing the probability of $\cup_{t\geq 1}\mathcal{B}_t$. We can follow a similar argument as in the Lemma 4.4 about analyzing $\mathcal{E}_R$. Let $R_t = \frac{8\Gamma}{\epsilon_t^2}\ln\frac{2^t}{\delta}$. Let $\mathcal{B}'_t$ denote the event that when NaivePerm has run R_t number of P-samples and the fraction of P-samples observed so far with detected violations is at most $\frac{\epsilon_t}{2\Gamma}$, player i^* receives a reward of less than $(1-\epsilon_t)\phi_{i^*}$. It's clear that $\mathcal{B}_t \subseteq \mathcal{B}'_t$, so $\mathbf{Pr}[\mathcal{B}_t] \leq \mathbf{Pr}[\mathcal{B}'_t]$.

Recall that for $j \in [R]$, we consider the reward decomposition $X_j := Y_j - Z_j$ as described in Remark 3.1. Observe that the event $\mathcal{B}'_t$ implies both of the following have occurred:

- The number of P-samples with detected violation is at most $\frac{\epsilon_t R_t}{2\Gamma}$. Therefore, $\sum_{j\in[R_t]} Z_j \leq \frac{\epsilon_t R_t}{2\Gamma}\cdot U_{\max,i^*} \leq \frac{\epsilon_t}{2}\cdot R_t\phi_{i^*}$.
- Had the adversary not interfered, the sum is:

$\sum_{j\in[R_t]} Y_j < (1-\frac{\epsilon_t}{2})R_t \cdot \phi_{i^*}$.

The first point is from the definition of $\mathcal{B}'_t$. If the second point does not happen, this means that $\sum_{j\in[R_t]}(Y_j - Z_j) \geq (1-\frac{\epsilon_t}{2})R_t \cdot \phi_{i^*} - \frac{\epsilon_t}{2} \cdot R_t\phi_{i^*} = (1-\epsilon_t)R_t \cdot \phi_{i^*}$, thereby violating $\mathcal{B}'_t$.

Therefore, we have $\mathbf{Pr}\left[\mathcal{B}'_t\right] \leq \mathbf{Pr}\left[\sum_{j\in[R_t]} Y_j < (1-\frac{\epsilon_t}{2})R_t \cdot \phi_{i^*}\right] \leq \exp(-\frac{\epsilon_t^2 R_t}{8\Gamma}) = \frac{\delta}{2^t}$, where the last inequality follows from Lemma 4.2.

Therefore, we have

$$\mathbf{Pr}\left[\mathcal{B}\right] \leq \mathbf{Pr}\left[\cup_{t\geq 1}\mathcal{B}_t\right] \leq \mathbf{Pr}\left[\cup_{t\geq 1}\mathcal{B}'_t\right] \leq \sum_{t\geq 1}\frac{\delta}{2^t} \leq \delta,$$

where the first inequality follows because $\mathcal{B} \subseteq \cup_{t\geq 1}\mathcal{B}_t$, the second inequality holds because $\mathcal{B}_t \subseteq \mathcal{B}'_t$, the third inequality is from the union bound. □

5 SAMPLING COMPLEXITY TO ACHIEVE EXPECTED MAXIMIN SECURITY

In this section, we give an upper bound of the sampling complexity to achieve expected maximin security for the baseline protocol using NaivePerm. The main result of this section is a lower bound on the sampling complexity even when we restrict to supermodular games and impose further conditions to help an honest player. The detailed version of this section is in the full version [7].

Model Assumptions. We consider an adversary with some known budget C. For analyzing the upper bound of sampling complexity, we consider using the same model as in Section 4.

For analyzing the lower bound of sampling complexity, we consider supermodular games, and replace NaivePerm with another P-sample protocol known as SeqPerm that has better security properties. Moreover, we consider the perpetual violation model in which detected violating players will be moved to the least preferable positions, thereby giving an advantage to an honest player.

Baseline Protocol. We instantiate the abstract protocol in Algorithm 1 by using NaivePerm to sample permutations. Moreover, the stopping condition can simply be having enough number of P-samples.

Lemma 5.1 (Upper Bound on Sampling Complexity). *By using* NaivePerm *in Algorithm 1 against an adversary with a known violation budget of C, for any $\epsilon > 0$, $R = \frac{\Gamma C}{\epsilon}$ number of P-samples is sufficient to achieve ϵ-expected maximin security.*

Lower Bound: Punish Violating Players in Supermodular Games. In Algorithm 1, if the violating players are given the least preferable positions, then the expected reward of an honest player i^* will not decrease in supermodular games. We call this violation model to be *perpetual*.

Lemma 5.2. *In Algorithm 1, if violating players are given least preferable positions, then the expected reward of an honest player i^* will not decrease in supermodular games.*

Replacing NaivePerm **with a More "Secure" P-Sample Protocol in Supermodular Games.** To rectify the attack on NaivePerm in Lemma 3.3, a simple idea is to generate a random permutation by deciding which player goes to which position sequentially, starting from the least preferable position. RandElim (described in the full version [7]) is a variant of Blum's protocol [4] that eliminates a uniformly random player from the active set one at a time, which goes to least preferable available remaining position.

Claim 5.3 (Probability of Elimination). *Assume that the commitment scheme in* RandElim *is ideal. When* RandElim *is run between a collection S of players, an honest player $i^* \in S$ is chosen to be eliminated with probability at most $\frac{1}{|S|}$, even if a Byzantine adversary controls all other players.*

Sequential Permutation Generation. We can generate a random permutation sequentially in n phases, where in each phase, the protocol RandElim is used to determine which player goes to the least preferable available position.

Algorithm 4: Sequential Permutation Generation SeqPerm

Input: Player set N.
Output: A random permutation $\sigma = (\sigma_1, \ldots, \sigma_n)$ of N, together with a collection Dev $\subseteq N$ of detected violating players.

```
S ← N, i ← 1
while S ≠ ∅ do
    (a, Dev) ← RandElim(S)
    if Dev ≠ ∅ then
        σ_i, ··· , σ_{i+|Dev|−1} ← players in Dev in lexicographical order
        i ← i + |Dev|
    else
        σ_i ← a
        i ← i + 1
    S ← S \ (Dev ∪ {a})
return (σ, Dev)
```

Claim 5.4 (Guarantee for Preferable Positions). *In the random permutation generated by Algorithm 4, for any $k \in [n]$, an honest player will be in one of the top k most preferable positions with probability at least $\frac{k}{n}$, even if a Byzantine adversary controls all other players.*

NaivePerm **vs** SeqPerm**.** It might seem that SeqPerm generate a more "secure" random permutation than NaivePerm. Hence, it would be somehow surprising that even when we replace NaivePerm with SeqPerm, there is a lower bound on the sampling complexity that asymptotically matches the baseline in Lemma 5.1.

Theorem 5.5 (Lower Bound). *Suppose* SeqPerm *is used to generate permutations in Algorithm 1. Then, for any large enough $n \geq 100$, there exists a supermodular game with n players such that for any $\frac{1}{n} \leq \epsilon \leq 0.01$, if the adversary has a perpetual violation budget of $1 \leq C \leq \epsilon n$, then at least $\frac{nC}{10\epsilon}$ number of P-samples is necessary to achieve ϵ-expected maximin security.*

6 EXPERIMENTS

In previous sections, we have proved upper and lower bounds on the number of P-samples to achieve several notions of maximin security. In this section, we conduct experiments to empirically verify our theoretical results. The experiments are run on the computer with

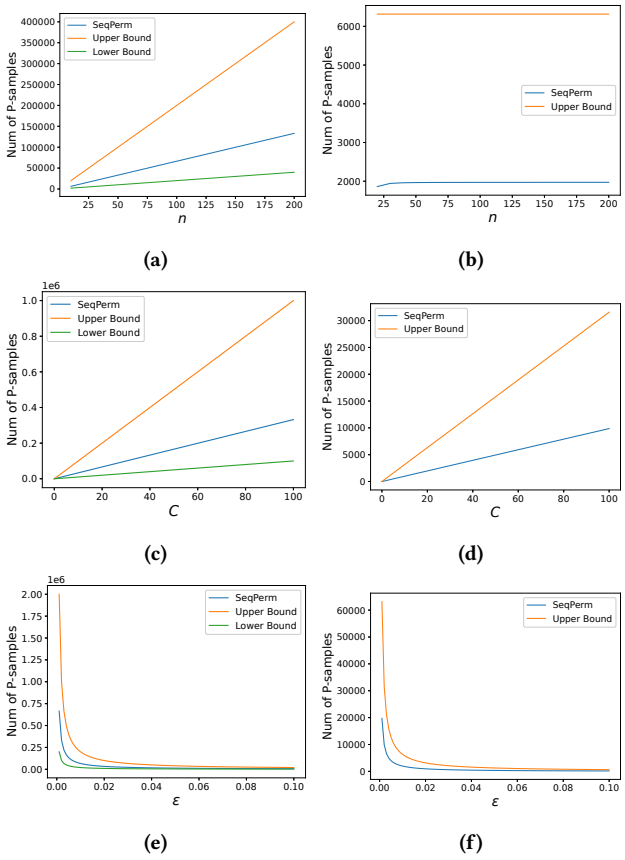


Figure 1: The number R of P-samples to reach ϵ-expected maximin security under different n, C and ϵ. The default values are $n = 100, C = 20, \epsilon = 0.01$. (1a), (1c) and (1e) are for the synthetic game; (1b), (1d) and (1f) are for edge synergy game on DBLP.

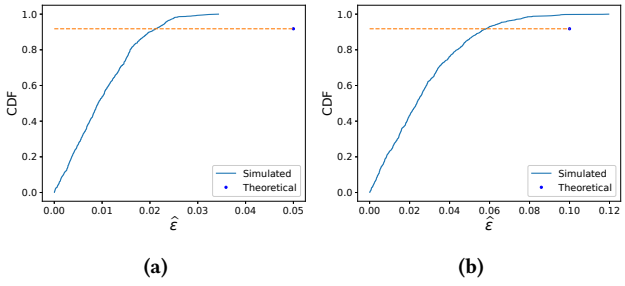


Figure 2: Empirical Cumulative Distribution Function of multiplicative error $\widehat{\epsilon}$ over 1000 simulations. (2a) is for the synthetic game, and (2b) is for the edge synergy game.

CPU i5-1240P, 1.70 GHz and a 16 GB RAM. The source code can be found in our public repository[1]. To simulate the adversary's strategy, we use a dynamic program approach to give the optimal adversarial strategy to attack SeqPerm, where the details are given in the full version [7].

Game Settings. We use two supermodular games with different values of the max-to-mean ratio Γ.

(1) The first game is based on Theorem 5.5, which is synthetic and characterized by the number n of players. Its definition is given in the full version [7] and has Γ equal to n.

(2) The second is the edge synergy game described in Section 2.3. We use a real-world hypergraph that is constructed from the collaboration network in DBLP, where each person is a vertex and the co-authors in a publication is a hyperedge. For an honest player, we pick an author, who in 2021, had 13 collaborators and 8 publications. To simulate other susceptible players, we include more players that represent scholars that have no collaboration with the honest player, finally resulting in a total of n players. In this case, $\Gamma = 3.15789$ and is independent of n.

We evaluate the performance of Algorithm 1 using `SeqPerm` to generate random permutations. Even though the protocols can be run with large number of players, the optimal adversarial strategy takes time exponential in n. Hence, we can only run experiments for small values of n up to 200.

Expected Maximin Security. We perform experiments to verify Lemma 5.1 and Theorem 5.5. We evaluate the number R of P-samples to reach ϵ-expected maximin security under the optimal adversarial strategy. We pick default values $n = 100, C = 20, \epsilon = 0.01$, and investigate how R varies with each of the three parameters.

As predicted, Figure 1 shows the results that R is proportional to C and inversely proportional to ϵ. In Figure 1a, we verify indeed that R is proportional to $\Gamma = n$ in the synthetic game; in Figure 1b, we see that R is independent of n, because $\Gamma = O(1)$ for the edge synergy game. Moreover, we see that $R \approx \frac{1}{3} \cdot \frac{C\Gamma}{\epsilon}$ falls between its lower bound and upper bound in the synthetic game. On the other hand, $R \approx 0.31 \cdot \frac{C\Gamma}{\epsilon}$ in the edge synergy game.

High Probability Maximin Security. We perform experiments to verify Theorem 4.3, Corollary 4.5 and Theorem 4.6. Although the protocol descriptions are different, one can check that, in each case, the number of P-samples is $R \approx \max\{O(\frac{\Gamma}{\epsilon^2} \ln \frac{1}{\delta}), O(\frac{C\Gamma}{\epsilon})\}$. Hence, we choose the parameters such that the two terms are the same.

For the synthetic game, we set $n = 100, C = 200, R = 8 \times 10^5, \epsilon = 0.05, \delta = 0.082$; for the edge synergy game, we set $n = 100, C = 100, R = 6316, \epsilon = 0.1, \delta = 0.082$. We ran 1000 simulations for each game to plot the empirical cumulative distribution of the multiplicative error $\widehat{\epsilon}$ for the honest player's Shapley value.

Figure 2 shows the empirical cumulative distribution of the multiplicative error $\widehat{\epsilon}$ in the 1000 simulations. The theoretical point $(\epsilon, 1-\delta)$ falls on the right of the curve, indicating that the simulated results are better than the theoretical bounds.

ACKNOWLEDGMENTS

This research was partially supported by the Hong Kong RGC grants 17203122 and 17202121.

[1] https://github.com/l2l7l9p/coalitional-game-protocol-expcode

REFERENCES

[1] Yoram Bachrach, Evangelos Markakis, Ariel D. Procaccia, Jeffrey S. Rosenschein, and Amin Saberi. 2008. Approximating power indices. In *AAMAS (2)*. IFAAMAS, 943–950.

[2] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. 2017. Bitcoin as a Transaction Ledger: A Composable Treatment. In *CRYPTO (1) (Lecture Notes in Computer Science, Vol. 10401)*. Springer, 324–356.

[3] Omar Besbes, Antoine Désir, Vineet Goyal, Garud Iyengar, and Raghav Singal. 2019. Shapley Meets Uniform: An Axiomatic Framework for Attribution in Online Advertising. In *WWW*. ACM, 1713–1723.

[4] Manuel Blum. 1982. Coin Flipping by Telephone - A Protocol for Solving Impossible Problems. In *COMPCON*. IEEE Computer Society, 133–137.

[5] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Comput. Oper. Res.* 36, 5 (2009), 1726–1730.

[6] T.-H. Hubert Chan, Ting Wen, Hao Xie, and Quan Xue. 2023. Game-Theoretically Secure Protocols for the Ordinal Random Assignment Problem. In *ACNS (Lecture Notes in Computer Science, Vol. 13906)*. Springer, 582–610.

[7] T-H. Hubert Chan, Qipeng Kuang, and Quan Xue. 2024. Game-Theoretically Secure Distributed Protocols for Fair Allocation in Coalitional Games. arXiv:2412.19192 [cs.GT] https://arxiv.org/abs/2412.19192

[8] Hugh Chen, Ian C. Covert, Scott M. Lundberg, and Su-In Lee. 2022. Algorithms to estimate Shapley value feature attributions. *CoRR* abs/2207.07605 (2022).

[9] Wei Chen and Shang-Hua Teng. 2017. Interplay between Social Influence and Network Centrality: A Comparative Study on Shapley Centrality and Single-Node-Influence Centrality. In *WWW*. ACM, 967–976.

[10] Kai-Min Chung, T.-H. Hubert Chan, Ting Wen, and Elaine Shi. 2021. Game-Theoretic Fairness Meets Multi-party Protocols: The Case of Leader Election. In *CRYPTO (2) (Lecture Notes in Computer Science, Vol. 12826)*. Springer, 3–32.

[11] Kai-Min Chung, Yue Guo, Wei-Kai Lin, Rafael Pass, and Elaine Shi. 2018. Game Theoretic Notions of Fairness in Multi-party Coin Toss. In *TCC (1) (Lecture Notes in Computer Science, Vol. 11239)*. Springer, 563–596.

[12] Xiaotie Deng and Christos H. Papadimitriou. 1994. On the Complexity of Cooperative Solution Concepts. *Math. Oper. Res.* 19, 2 (1994), 257–266.

[13] Daniel Deutch, Nave Frost, Benny Kimelfeld, and Mikaël Monet. 2022. Computing the Shapley Value of Facts in Query Answering. In *SIGMOD Conference*. ACM, 1570–1583.

[14] Paul Dütting, Monika Henzinger, and Ingmar Weber. 2010. How much is your personal recommendation worth?. In *WWW*. ACM, 1085–1086.

[15] Edith Elkind, Leslie Ann Goldberg, Paul W. Goldberg, and Michael J. Wooldridge. 2007. Computational Complexity of Weighted Threshold Games. In *AAAI*. AAAI Press, 718–723.

[16] Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. 2001. Sharing the Cost of Multicast Transmissions. *J. Comput. Syst. Sci.* 63, 1 (2001), 21–41.

[17] Liyang Han, Thomas Morstyn, and Malcolm McCulloch. 2021. Estimation of the shapley value of a peer-to-peer energy sharing game using multi-step coalitional stratified sampling. *International Journal of Control, Automation and Systems* 19, 5 (2021), 1863–1872.

[18] Samuel Ieong and Yoav Shoham. 2005. Marginal contribution nets: a compact representation scheme for coalitional games. In *EC*. ACM, 193–202.

[19] Kamal Jain and Vijay V. Vazirani. 2001. Applications of approximation algorithms to cooperative games. In *STOC*. ACM, 364–372.

[20] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. 2019. Towards Efficient Data Valuation Based on the Shapley Value. In *AISTATS (Proceedings of Machine Learning Research, Vol. 89)*. PMLR, 1167–1176.

[21] David Liben-Nowell, Alexa Sharp, Tom Wexler, and Kevin Woods. 2012. Computing Shapley Value in Supermodular Coalitional Games. In *COCOON (Lecture Notes in Computer Science, Vol. 7434)*. Springer, 568–579.

[22] Huijia Lin and Rafael Pass. 2015. Constant-Round Nonmalleable Commitments from Any One-Way Function. *J. ACM* 62, 1 (2015), 5:1–5:30.

[23] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. 2021. Query Games in Databases. *SIGMOD Rec.* 50, 1 (2021), 78–85.

[24] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. 2013. Bounding the Estimation Error of Sampling-based Shapley Value Approximation With/Without Stratifying. *CoRR* abs/1306.4265 (2013).

[25] Ramin Okhrati and Aldo Lipani. 2020. A Multilinear Sampling Algorithm to Estimate Shapley Values. In *ICPR*. IEEE, 7992–7999.

[26] Alon Reshef, Benny Kimelfeld, and Ester Livshits. 2020. The Impact of Negation on the Complexity of the Shapley Value in Conjunctive Queries. In *PODS*. ACM, 285–297.

[27] Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games* (1953), 307–317.

[28] Gilad Zlotkin and Jeffrey S. Rosenschein. 1994. Coalition, Cryptography, and Stability: Mechanisms for Coalition Formation in Task Oriented Domains. In *AAAI*. AAAI Press / The MIT Press, 432–437.