Multi-Party Computation Meets Game Theory

Summary of Secret Sharing

Yang Xiao
Johns Hopkins University
yxiao40@jh.edu

Xiecongyou Yang
Johns Hopkins University
xyang107@jh.edu

KEYWORDS

Game Theory, Multi-Party Computation, Secret Sharing

1 INTRODUCTION

This is a basic summary of two papers concerned about game theory applied to Multiparty Computation, which are "Rational Secret Sharing and Multiparty Computation: Extended Abstract"[6] and "Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation"[1].

2 MULTI-PARTY COMPUTATION BACKGROUND

2.1 Basic Structure of MPC

This section will be about a high-level overview of the problem of Multi-Party Computation (MPC) and a specific MPC protocol used with rational secret sharing protocol. The general notion of MPC is that there are n parties want to jointly compute a function $f(x_1, x_2, ..., x_n)$, where x_i is the private input of party i. The idea of this problem is that each party will finally get the output of this function, but won't learn any additional information about the input of other parties.

The basic structure [9] of a common solution for MPC could contain three main stages, which can be regard as a process of secret sharing. During the first stage, each party will send $n-1$ shares to other parties through a private and secure channel. These $n-1$ shares probably are related to its private input. In the second stage, all the parties will compute this function f locally. Generally, it can be done by evaluating the pre-agreed-upon arithmetic circuit representing f. The arithmetic circuit in here often consists of addition, scalar-multiplication and multiplication gates (AND, NOT and XOR). There are several different protocols used to deal with these computation. Anyway, at the end of the second stage, each parties will get a valid output, which is related to the final output of function f. In the final stage, which is also called reconstruction phase, each party need to share or broadcast its output generated after second stage. Then with all these shared outputs, each party can construct the final output of function f.

2.2 Security of MPC

Basically, we can found that the key point of MPC is security, meaning each party can learn the correct output of function f without revealing anything else. In order to define the security formally, a *ideal model* is created. In this *ideal model*, each party will send their private input to a completely trusted party. This trusted third-party just simply computes the result of function f and then return to all parties. But in the real world, there doesn't exist a trusted party, all parties have to compute this function by utilizing particular protocol. Therefore, a MPC protocol is secure if the parties will get the same output as they would in the ideal model and they won't learn any more information than they would with a trusted party in the ideal model.

One of the most important factors that can affect the security of MPC is adversaries, which can corrupt parties and protocols [9]. A *semi-honest adversary* is one who corrupts parties but follows the defined protocol. *Semi-honest parties* may try to learn as much as possible from the messages they receive from other parties. They are also considered *passive* or called *honest-but-curious*. Another more dangerous adversary is called *malicious (active) party*. These parties may deviate from the prescribed protocol in any way that they desire, with the goal of either influencing the computed output value in some way. They have all the powers of *semi-honest adversary*, but also can take any actions it wants during protocol execution.

The adversary can corrupt any number of parties out of the n parties. In order to solve MPC problem and define the security precisely, the proof of protocols will limit the number of corrupted parties to k, indicating that this protocol will keep secure as long as the number of corrupted parties is at most k.

2.3 Secret Sharing

The classic mechanism of secret sharing is Shamir's scheme (t-out-of-n secret sharing)[11], which allow someone to share secret s among n other agents. It assume the secret s lies in a finite field $\mathbb{D}$, with $\mathbb{D} > n$. The dealer chooses a random polynomial $g(x)$ of degree at most $t-1$ subject to the constraint $g(0) = s$, and gives the "share" $g(i)$ to party P_i. Any set of t parties can recover $g(x)$ by broadcasting their shares and interpolating the polynomial function; Furthermore, no set of fewer than t players can deduce any information about s.

2.4 Oblivious Transfer

Beside the secret sharing model, Oblivious Transfer (OT) [10] is another essential primitive for MPC. Given OT, one can build MPC without any additional assumptions and reversely, one can get a OT from MPC [7].

The standard definition of 1-out-of-2 OT involves two parties (Alice and Bob). Alice is the sender who has two message m_0 and m_1, and wants to ensure that receiver Bob only can get one of them without learn any information about another one. Bob will have a bit b and want to receive m_b, but have to ensure Alice won't know anything about his b. This protocol can be instantiated using RSA encryption

Supervised by Professor Michael Dinitz.

Project in Algorithmic Game Theory,
2022.

(The protocol of Even, Goldreich, and Lempel).
1-out-of-n oblivious transfer protocol can be generated from 1-out-of-2 oblivious transfer protocol. In specific, a sender has n messages, and a receiver has an index i. This receiver want to receive the i-th messages of sender without the sender learning i. Also, the sender want to ensure that the receiver can't get any information of other $n-1$ messages.

2.5 Goldreich-Micali-Wigderson (GMW) Protocol

The GMW protocol [2] is a solution for MPC via combining secret sharing protocol and Oblivious Transfer protocol. It uses a boolean-circuit representation for the function being computed and is secure against a semi-honest adversary controlling any number of corrupted parties. The protocol can be generlized to arithmetic circuits as well.
For a two-party Boolean version, we assume that there are two parties, P_1 and P_2. P_1 has a private input x and P_2 has another private input y. They want to compute the function $f(x, y)$ and the Boolean circuit representing this function is C.
In the first stage, which is similar to the common structure of MPC we mentioned before, P_1 will divide its input x to x_1 and x_2, which satisfy that $x_1 \oplus x_2 = x$. Symmetrically, P_2 need to generate y_1 and y_2 in the same way. Then P_1 will send x_2 to P_2 and P_2 send y_1 to P_1, which is the first phase of secret sharing. No one can recover the actual value of other's private input, which we can regard as a 2-out-of-2 secret sharing.
In the second stage, each parties need to evaluating C gate by gate, computing the output of C locally. The goal of this stage is we wish each parties will compute a output $f(x_i, y_i)$, then we can reconstruct the output $f(x, y)$ according to $f(x_1, y_1)$ and $f(x_2, y_2)$. Without loss of generality, we assume C consists of NOT, XOR and AND gates. Both NOT and XOR gates can be evaluated without interaction or communication between P_1 and P_2. For example, if a immediate value w is computed by $x \oplus y$, then we only need to ensure P_1 computes $x_1 \oplus y_1$ and P_2 computes $x_2 \oplus y_2$. In the final stage, we XOR these two values,$(x_1 \oplus y_1) \oplus (x_2 \oplus y_2)$, which is equal to $x \oplus y = w$. But for evaluating an AND gate, it's another story. Assume $w = x \wedge y$, we can get $x \wedge y = (x_1 \oplus x_2) \wedge (y_1 \oplus y_2) = (x_1 \wedge y_1) \oplus (x_1 \wedge y_2) \oplus (x_2 \wedge y_1) \oplus (x_2 \wedge y_2)$. According to this equation, we can find that P_1 can compute $(x_1 \wedge y_1)$ and P_2 can compute $(x_2 \wedge y_2)$ locally. But $(x_1 \wedge y_2)$ and $(x_2 \wedge y_1)$ are the problems here, since they can't get the actual value of x_2 and y_2 from each other, otherwise they can directly reconstruct the actual value of other's private input. So this is the place we need two 1-out-of-2 OT. In the first OT, P_1 is the sender and P_2 is the receiver. P_1 need to prepare two messages r_1 and $r_1 + x_1$ where r_1 is a random value, while P_2 send the value of y_2. Then P_2 will receive the value of $r_1 \oplus (x_1 \wedge y_2)$. According to the properties of OT, P_1 won't know anything about y_2 and P_2 won't know anything about r_1. Similarly, through another OT, p_1 can receive a value of $r_2 \oplus (y_1 \wedge x_2)$. Once these two OT process are finished. P_1 owns the value of $(x_1 \wedge y_1) \oplus r_1 \oplus (r_2 \oplus y_1 \wedge x_2)$ and P_2 owns the value of $(r_1 \oplus x_1 \wedge y_2) \oplus r_2 \oplus (x_2 \wedge y_2)$. The XOR result of these two value is $x \wedge y$. At the end, each party will have a share of every value along every wire in the circuit.
After evaluating all gates, they can reconstruct the function output by exchanging shares of the output gate value, which is reconstruction phase of secret sharing. Moreover, this two-party Boolean version can be generalized to more than two parties straightforwardly.

3 GAME THEORY BACKGROUND

3.1 Game Theory Notions and Settings

According to the definition of *Nash Equilibrium* (NE) and *Correlated Equilibrium* (CE), the class of NE corresponds to independent strategies of parties, while the class of CE corresponds to correlated strategies. However, in order to implement a given CE, we need a special 'correlated device'-so-called the *Mediator M* – who will sample a strategy profile $s = (s_1, s_2, ...s_n)$ from the joint distribution and privately recommend each action s_i to party P_i. Notice that each party P_i could not learn anything about the recommended actions of other parties beyond what could be implied about its own action s_i. The joint distribution is a correlated equilibrium if every player can achieve higher payoffs by playing the recommended action than what is possible using any given NE, or even what can be achieved by taking any convex combination of NE payoffs[9].

Game with incomplete information. Consider the scenario when each party has a private type t_i, the joint type vector $t = (t_1, t_2, ..., t_n)$ is drawn from a publicly known distribution. Furthermore, the utility function of each party P_i is determined by its type t_i in addition to their actions $s_1, s_2, ...s_n$. Therefore, a new kind of equilibrium called Bayesian Nash Equilibrium is generalized from the simple NE.

The mediated game generalized to this setting, in which each party have to send their types to the mediator M before receiving the recommended action. The mediator M samples a correlated strategy profile s based on the received type vector t and privately give the recommended action s_i to each party P_i. Remark that the expected canonical strategy for party P_i is to honestly report its type t_i to M, and then follow the recommended action s_i. However, party P_i could deviate its two-stage actions out of its own benefit: send a wrong type t_i' or not send a type at all to M, or change the recommended action from M. As a mediator might receive faulty types, a well-defined recommended actions sampling strategy for the mediator should specify the joint distribution x for every possible type $t = (t_1, t_2, ..., t_n)$, even out of the support of the joint distribution of type. More formally, the joint distribution x^t should be defined all over the set $\bigcup_i (T_i \cup \{\perp\})$, where $\perp$ represents a invalid type. Then with these description, the generalization of CE to games with incomplete information is straightfoward.

Collusions. The above discussion of algorithmic game theory all assumes the traditional non-cooperative setting, where agents are assumed not to form collusions. In contrast, *cooperative game theory* tries to model reasonable equilibrium concepts arising in scenarios where agents are allowed to form collusions[9]. However, traditional game theoretic treatment of such equilibrium are fairly weak. We are going to introduce stronger equilibria in later section.

3.2 Repeated Game

So far we considered only strategic games, where parties move in 'one-shot' (possibly with the help of mediator). Of course, these games are special cases of much more general *extensive form* games,

where a party can move in many rounds and whose payoffs depend on the entire run of the game. A *repeated game* is an extensive form game that consists of a number of repetitions of some base game (called a stage game). In a repeated game, a player will have to take into account the impact of his or her current action on the future actions of other players.

Further, repeated games can be divided into two categories: finite games and infinite games. In finite games with parameter k, which indicates total rounds of the stage game, all the players are aware that the stage game is going to be repeated k times. Equilibrium of finite games can often be solved by *backward induction*. Infinite games are those in which the game is being played an infinite number of times. A game with an infinite number of rounds is also equivalent (in terms of strategies to play) to a game in which the players in the game do not know for how many rounds the game is being played.

Even if the stage game is the same for finite games and infinite games, different outcomes (equilibrium) could be resulted by repeating the stage game a finite or an infinite times, as well as different optimal strategies. The *Folk Theorem*[3] suggests that if the players are patient enough and far-sighted, then repeated interaction can result in virtually any average payoff in an *SPE* equilibrium[8]. For example, in the one-shot Prisoner's Dilemma, both players choose to defect is a NE. While folk theorem says that, in the infinitely repeated version of the game, provided players are sufficiently patient, there is a Nash equilibrium such that both players cooperate on the equilibrium path. But if the game is repeated a known finite number of times, backward induction shows that both players will play the one-shot Nash equilibrium in each period, i.e. they will defect each time.

3.3 Non-Cooperative Computation

Non-cooperative computation is the joint computation of a function by self-motivated parties, where each of the parties possesses one of the inputs to the function. Each parties in NCC have to communicate their input to a trusted third-party (Mediator). NCC is a game theoretic concept concerned about whether the agents can be incentive to give their true private input to the trusted third-party. In order to introduce game theory into multi-party computation, specifically, into multi-party function evaluation, we have to assume that each party involved in this computation is 'rational'. Then we have to define a corresponding game, named *function evaluation game*. We assume that the parties' types t_i are their inputs to a function $f(t_1, ..., t_n)$. The action of each party P_i is to make their guess about the output s^* of function f. Now, in order to build appropriate utility functions, we have to make the assumption that each party P_i withes to compute f correctly with the highest priority. Then, given P_i the same success in learning the output, we assume P_i prefers as few parties as possible to be successful.

Now the game is set, certain equilibrium becomes interesting. In this part we are going to focus on the correlated equilibrium. Namely, each party gives its input t_i to a mediator M, who then recommends an action s_i^* for each party. Given that in our game, each party is trying to compute the actual value of f, a natural canonical form of mediator strategy is: evaluating the function f on the reported type vector t, then recommend each party to guess the resulting function value $s^* = f(t)$. Therefore, we can ask what should a function look like so that the above canonical mediator strategy is a correlated equilibrium for function evaluation game. We have to be careful that some parties might give wrong input to the mediator in order to prevent other parties to make their correct guess. So we assume that the mediator will send an error message to all the parties and let them decide by themselves what to play. Formally,

Definition 3.1. We say that a function f is non-cooperatively computable (NCC) with respect to utility functions $\{u_i\}$ (and a specific input distribution D) if the above canonical mediated strategy is a correlated equilibrium of the function evaluation game. Namely, it is in the parties' selfish interest to honestly report their true inputs to the mediator.[12]

It immediately follows that two classes of functions are clearly not NCC. A function f is called *dominated* if there exists an index i that t_i determines the value of f irrespective of the other inputs t_{-i}. Obviously, a rational P_i would never send its true value to the mediator, because it is able to reconstruct the value of f by itself while other players are not. Therefore, dominated functions are not NCC. Meanwhile, a function f is called *reversible* if for some index i, there exists another input t_i' and a function g such that for all other parties' input t_{-i} we have $g(f(t_i', t_{-i}), t_i) = f(t_i, t_{-i})$ and $f(t_i', t_{-i}) \neq f(t_i, t_{-i})$. With these property, function f guarantees that P_i has no risk in reporting t_i' instead of t_i, and P_i can recover the output from mediator to recover the true value of f by input its true value t_i.

In general, depending on properties of utility functions, there might be other non-NCC functions. However, if we slightly add another assumption that the risk of incorrectly guess the true value of f is far more greater than letting other players make correct guess, then these two classes of functions are the only non-NCC functions[9]. Formally,

Theorem 3.2. *Under the above assumption, a function f is NCC if and only if it is not dominated and not reversible.[12]*

4 RATIONAL SECRET SHARING

In this section, we are going to talk about (m, n) secret sharing, where there are n parties and it takes at least m parties' private secret to reconstruct the original secret s that the dealer holds, under the assumption that each party is 'rational' and follows a certain utility function that satisfies the properties we mentioned in section 3.3 that each party wants to correctly reconstruct the correct original secret s which the dealer holds, and that given the same success in learning the original secret s, each party prefers as few parties as possible to be successful. Under these assumptions, we are going to eliminate the practical feasibility of Shamir's protocol[11] in secret sharing and multi-party computation and describe two randomized practical mechanisms.

4.1 Nash Equilibrium

In this section, we are going to use the traditional definition of *game trees* in repeated games. Intuitively, the root of a tree represents the initial situation in the game, later nodes represents the results of

parties' each move. We assume that at each step, a party receives all the messages that were sent to it by other parties at the previous step, performs some computation, then sends some messages (probably none). Associated with each *run* (i.e. path in the forest that starts at a root and either is infinite or ends in a leaf node) is a tuple of real-valued utility $u = (u_1, u_2, ...u_n)$. Each node stores both the local state and previous actions of all parties. Actions are defined for each party as either send a message to other parties. Then, a strategy (or protocol) is defined as a function from local states to actions. Let $U_i(\sigma)$ denotes the expected utility of party P_i when plays strategy σ.

Although there might be many NE in this game, most of them are unreasonable. Here we are going to focus on one particular NE that is determined by iterated deletion of weakly-dominated strategy. Intuitively, we don't want any party to use a strategy that is weakly-dominated.

Formally, if S_i is a set of strategies for party P_i, we say that a strategy $\sigma \in S_i$ is weakly dominated by $\gamma \in S_i$ with respect to S_{-i} if for some strategy $\sigma_{-i} \in S_{-i}$ we have $u_i(\sigma_{-i}, \sigma_i) < u_i(\sigma_{-i}, \gamma)$, and for all strategy $\sigma_{-i} \in S_{-i}$ we have $u_i(\sigma_{-i}, \sigma_i) \leq u_i(\sigma_{-i}, \gamma)$. Therefore, if strategy σ is weakly dominated by γ with respect to S_{-i}, then party P_i would intuitively switch to use strategy γ instead of using σ since use strategy γ gives at least the same utility as σ and sometimes gives strictly greater utility. We say a strategy $\sigma \in S_i$ is weakly-dominated with respect to $S_1 \times S_2 \times ... \times S_n$, if there exists a strategy $\gamma \in S_i$ such that σ is weakly-dominated by γ.

Let $DOM_i(S_1 \times S_2 \times ... \times S_n)$ be all the strategies of party P_i that is weakly dominated. Given a game Γ, let S_i^0 consists of all strategies of party P_i in Γ. Now we are going to recursively define S_i^k. Suppose that we have now defined S_i^k for some k, which represents all the strategies of party P_i that survives k rounds of deletion. Let $S_i^{k+1} = S_i^k \setminus DOM_i(S_i^k \times ... \times S_n^k)$. Let $S_i^\infty = \cap_k S_i^k$. Thus S_i^∞ represents all the strategies that survives an arbitrary number of rounds of iterated deletion of weakly-dominated strategies.

We take a mechanism to be (Γ, σ) consisting a game and joint strategies for that game and say that (Γ, σ) is a *practical mechanism* if σ is a NE of game Γ that survives the iterated deletion of weakly-dominated strategies[6].

4.2 The Impossibility Result

4.2.1 Secret Sharing. [6] shows that there is no practical mechanism for secret sharing with a commonly-known bound on its running time, provided the reasonable assumptions about the preferences of partys (properties of utility functions we talked above). Formally,

Theorem 4.1. *(Halpern, Teague) Let M be a mechanism for m out of n secret sharing. After an arbitrary number of rounds of iterated deletion of weakly-dominated strategies, all deterministic strategies are deleted.*

4.2.2 Multi-Party Function Computation. [6] shows that there is also no practical mechanism for multi-party function computation such that any party learns the value of function. Formally,

Theorem 4.2. *(Halpern, Teague) If the parties' utility function satisfies the above properties, then there is no practical mechanism (Γ, σ) such that Γ is finite, and some party learns the value of function using σ.*

4.3 A Randomized Practical Mechanism for Secret Sharing

With the impossibility result in section 4.2, the only hope that we can get a practical mechanism for secret sharing lies in connecting personal benefit with social benefit and in using uncertainty about when the game will end in order to encourage each party to cooperate. Next, we are going to give a randomized protocol for 3-out-of-3 secret sharing.

(3,3)-secret sharing randomized protocol:

(1) Every party tosses a coin, and is supposed to send their secret if their coin lands on heads.
(2) Every party reveals their coin.
(3) If any party fails to send their secret even though their coin lands on heads or every party learns the original secret, protocol terminates. Otherwise, the dealer takes a new polynomial function with free coefficient to be the secret itself, issues new shares of the secret, and repeats the above steps.

Consider the incentive of each party that has tossed a tail, then it is better off for them to keep the secret and not send, because sending out their secret not only increase the chance that other parties learn the secret but also leads to immediate termination of the protocol. While now let's consider the incentive of each party that has tossed a head and is supposed to send their share. Normally, these parties don't have incentive to withhold their secret because that will leads to immediate termination of the protocol. Unless the party is gambling that other parties are both tossing heads and are being honest with sending their shares. Therefore, with probability of $\frac{1}{4}$ the party will eventually learn the secret by cheating, and there is no other parties learn the secret, but with probability of $\frac{3}{4}$, nobody learns the secret even including the cheating party itself. In order to make each party to be honest out of their own benefit, we have to make the inequality $\frac{1}{4}u_i(\text{only party } i \text{ learns the secret}) + \frac{3}{4}u_i(\text{nobody learns the secret}) < u_i(\text{everyone learns the secret})$, then party i will not be tempted to cheat. If party i's utility function does not satisfy this inequality, then the probability of heads and tails can be modified in order to make the inequality holds.

However, there is a flaw in this randomized protocol: even if every party is honest, there is a probability of $\frac{3}{8}$ that one of the party tosses a tail and the other two parties both tosses a head. Then in this specific iteration, the party with tail will learn the secret while the other two parties will not. And the party that learns the secret has obviously no incentive to be honest in the next iteration and never sends their secret before the protocol terminates. Now we can slightly modify the protocol to prevent such circumstance. Let the parties be 1, 2 and 3. For $i \in \{1, 2, 3\}$, let $i^+ = (i+1) \mod 3$, $i^- = (i-1) \mod 3$.

modified (3,3)-secret sharing randomized protocol:

(1) The dealer privately sends each party their share of secret.
(2) Each party i chooses a bit c_i such that with probability α, $c_i = 1$, with probability $1 - \alpha$, $c_i = 0$, and randomly (follows uniform distribution) chooses a bit $c_{(i,+)}$. Let $c_{(i,-)} = c_i \oplus c_{(i,+)}$. Party i sends $c_{(i,+)}$ to i^+, sends $c_{(i,-)}$ to i^-. Note that

this means i should receive $c_{(i^+,-)}$ from i^+ and $c_{(i^-,+)}$ from i^-.

(3) Each party i computes $p = c_{(i^-,+)} \oplus c_{(i^-,-)} \oplus c_{i^+} \oplus c_i = c_{i^-} \oplus c_{i^+} \oplus c_i = c_1 \oplus c_2 \oplus c_3$. Then if $p = c_i = 1$, party i sends its signed share to the others.

(4) if $p = 0$ and i received no secret shares, or if $p = 1$ and i received exactly one share (possibly from itself), the dealer is asked to take a new polynomial function with free coefficient to be the secret itself and repeat the protocol. Otherwise, party i stops the protocol.

Note that party i will stop the protocol only if either it has all three shares (other parties as well) or some party is cheating. We call this mechanism $M(\alpha)$. Intuitively, as the same argument above that we can modify α in order to make the inequality of each party's utility function holds, so that each party will play honestly for its own benefit. Formally,

Theorem 4.3. *(Halpern, Teague)[6] For all utility functions that satisfies our assumptions, if $n \geq 3$, then there exists an α^* such that $M(\alpha)$ is a practical mechanism for m out of n secret sharing for all $\alpha < \alpha^*$. Moreover, the expected running time of the recommended strategy in $M(\alpha)$ is $\frac{5}{\alpha^3}$.*

4.4 A Randomized Practical Mechanism for Multiparty Computation

Similar to the randomized protocol for secret sharing, we can also give a randomized practical mechanism for multi-party function computation. With the assumption of the existence of *one-way 1-1 functions*, [6] gives the following theorem:

Theorem 4.4. *If a function f is non-cooperatively computable, for $n \geq 3$, and parties' utility function satisfies the previous assumptions, then there is a randomized practical mechanism for computing f that runs in constant expected time.*

Recall that GMW protocol[4] provides a multi-party function protocol without assuming every party is rational. The key idea is to expand the objective function into circuits such that, at each stage of the protocol, the value of a node in the circuit is viewed as a secret, and all players have a share of that secret. Halpern et al.[6] gives a protocol that also uses this circuit simulation, while replaces the last step of secret sharing (where each party could actually learn the original secret) by the above randomized secret-sharing protocol. This protocol employs the same zero-knowledge and bit-commitment sub-protocols to force semi-honest behavior with the assumption of existence of one-way 1-1 function as Goldreich does. Follow the same procedure of iterated deletion of weakly-dominated strategies, this strategy survives an arbitrary number of rounds and therefore is a NE.

5 K-RESILIENCE NASH EQUILIBRIUM

Traditional Nash equilibrium only tolerates the deviation of only one agent. However, in practice, agent can form coalition. Therefore, there is another equilibrium called *k-resilient*, which means it can tolerate deviation by coalitions of size up to k. If a joint strategy $(\sigma_1, ..., \sigma_n)$ is *k-resilient*, for any coalition $|C| \leq k$ that deviates from the equilibrium, none of the agents in C do better than they do with $(\sigma_1, ..., \sigma_n)$. Let $N = \{1, ..., N\}$ be the set of players. Let S_i denote the set of possible strategies for player i and let $S = S_1 \times ... \times S_n$. We can formally define the *k-resilient* [1].

Definition 5.1. Given a non-empty set $C \subseteq N, \sigma_C \in \mathcal{S}_C$, is a group best response for C to $\sigma_{-C} \in \mathcal{S}_{-C}$ if, for all $\tau_C \in \mathcal{S}_C$ and all $i \in C$, we have

$$u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_{-C})$$

A joint strategy $\vec{\sigma} \in \mathcal{S}$ is a k-resilient equilibrium if , for all $C \subseteq N$ with $|C| \leq k$, σ_C is a group best response for C to σ_{-C}. A strategy is strongly resilient if it is k resilient for all $k \leq n - 1$.

We can easily found a 1-resilient equilibrium is a normal Nash equilibrium. In addition, this definition can imply tolerance of deviations where only a single player in the coalition does better and coalition-proof equilibrium tolerate only deviations where everyone in the coalition does better.

5.1 Resilient Secret Sharing With Mediator

To prove there exist a resilient secret sharing, they [1] firstly attempt to prove this game with mediators. The mediator in secret sharing can be a communication device. Consider a multistage game with T stages with complete recall. A mediator is a tuple $(\mathcal{I}_i^t, O_i^t, \mathcal{P}^t)$ for $i \in N$ and $t \in T$, where $\mathcal{I}_i^t$ is a set of inputs that player i can send at stage t, O_i^t is a set of outputs for player i at stage t, and $\mathcal{P}^t$ is a function from $\prod_{i \in N, r \leq t} \mathcal{I}_i^t$ and possibly some random bits r to $\prod_{i \in N} O_i^t$. Intuitively, each player need to send a input and the mediator computes a function of all messages ever sent and sends each player a piece of advice output at each stage t. Therefore, for a multistage game Γ and a mediator d, we can define a game Γ_d. Each stage t of Γ_d can consists of three phases. The first two phases are mentioned above (sending input and sending output). In the third phase, each player chooses a action or no action at all and the utilities of the players depend only on the actions made in Γ
Now we can define a protocol for the resilient secret sharing with mediator. The definition of secret sharing is still Shamir's scheme. There are two assumptions here.

(1) The initial shares given to each player are signed by the issuer and each other player can verify the signature of the issuer and no player can forge the signature (This assumption can be removed)

(2) Assume that players must output the secret they guessed at the end of the game (To formalize the intuition that players prefer getting the secret to not getting it)

A solution with a mediator is formally defined below, supposing that f is of degree $m - 1$ and without loss of generality, $f(0)$(the secret) is not 0.

(1) In stage 0, each agent i sends its secret share $f(i)$ to mediator
(2) The mediator computes f
(3) In stage $r \geq 0$, if the game is not over, the mediator
  (a) chooses $c \in 0, 1(Pr(c = 1)$ to be determined by $\alpha)$ and random polynomial g^r of degree $m - 1$ such that $g^r(0) = 0$
  (b) computes $h^r = cf + g^r$ and sends $h^r(i)$ to each agent i
(4) The players then share their shares and compute h^r.
  (a) if $h^r(0) \neq 0$, then $h^r(0) = f(0)$; the game is over
  (b) if $h^r(0) = 0$, the agents learn nothing; go to stage stage $r + 1$

When every player follows this protocol, they showed $\sigma_1, ..., \sigma_n$ is a k-resilient equilibrium, provided that α is chosen appropriately.

$$\alpha \leq min_{i \in N} \frac{u_i(N) - u_i(\emptyset)}{m_i - u_i(\emptyset)} \text{ and } k < m$$

$m_i = max_{A \subseteq N} u_i(1)$, and A is a space defined by $A_1 \times ... \times A_n$. For a set $I \subset N$, $A_I = \prod_{i \in I} A_i$ and $A_{-I} = \prod_{i \notin I} A_i$. Thus S_I is the strategy set of players in I. And Given $x \in A_I$ and $y \in A_{-I}$, let (x, y) be the tuple in A such that $(x, y)_i = x_i$ if $i \in I$ and $(x, y)_i = y_i$ otherwise. $u_i(A)$ is i's best-case expected utility if exactly the players in A learn the secret.

The brief proof is that first, there is no advantage to lying about the initial value, since shares are signed and liars will be caught. Secondly, there is no advantage to not revealing your share in phase 3 of round t. When $c = 0$ (with probability $1 - \alpha$), the game will stop and no player will learn the secret. If the game does not stop, the best outcome is to gain m_i. But only if $\alpha m_i + (1 - \alpha) u_i(\emptyset) > u_i(N)$, you can gains it from sending, which is impossible according to the definition of α. Thirdly, there also is no advantage to lying about your share. Since if $c = 0$, $h^r(0) \neq 0$, then other players wrongly stop playing, or if m is small enough the secret can be reconstructed anyway.

5.2 Resilient Secret Sharing Without Mediator

There are many games with mediators that can deal with secret sharing. But the advantage of the game we mentioned in last section is that it can be simulated by the players without a mediator, by using multiparty computation [1]. Still with assumption that the players are given signed shares, for each round t, the input of player i to multiparty computation consists of the following inputs (a and b are parameters of the protocol chosen by $a/2^b \leq min_{i \in N} \frac{u_i(N) - u_i(\emptyset)}{m_i - u_i(\emptyset)}$):

(1) The signed secret share that i received initially.
(2) A random bitstring c_i^t of length b, uniformly distributed in $\{0, 1, ..., 2^b\}$.
(3) A random degree $m - 1$ polynomial g_i^t such that $g_i^t(0) = 0$ and the coefficients of g_i^t are uniformly distributed in the underlying field F.
(4) A random bitstring b_i of length $\lceil log(|F|) \rceil$.

Let $c_t = 1$ if $\oplus_{i \in N} c_i^t \leq a 2^z$, otherwise $c_t = 0$. And let $g^t = (g_1^t + ... + g_n^t)$. Basically, the multiparty computation does essentially what the mediator did before:

(1) Checking that the shares sent are correctly signed.
(2) Compute the polynomial f that interpolates them.
(3) Check that $g_i^t(0) = 0$ for each random polynomial.
(4) If these properties hold, then compute $h^t = c^t \cdot f + g^t$.
(5) Output $(h^t(1) \oplus b_1, ..., h^t(n) \oplus b_n)$.

This entire process can be achieved through GMW protocol mentioned in Section 2, which uses m out of n secret sharing. The choice of m depends on the size k of coalitions we want to tolerate, whether we have cryptography available and whether we know the exact utility. A coalition member i will follow the protocol during the circuit evaluation of F for the similar reasons. Deviation can result in being caught and learning no information depends on $Pr(c = 1)$, and they cannot gain advantage in any case.

However, using MPC indicate that we using the cryptography, so there is a small probability that the cryptography will be broken. In order to define a guarantee as to what happens if the cryptography is broken, they attempt to weaken the notion of k-resilient mechanism [1].

Definition 5.2. $\Gamma, \vec{\sigma}$ is a ϵ-k-resilient mechanism for $\mathcal{F}$ if $\vec{\sigma}$ satisfies $\mathcal{F}$ but, for each coalition C such that $|C| \leq k$, σ_C is not necessarily a group best response to σ_{-C}, but it is a ϵ-best response: no one in the group can do more than ϵ better than they would using a best response. That is, for all C such $|C| \leq k$, for all $\tau_C \in \mathcal{S}_C$, and all $i \in C$, we have:

$$u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_{-C}) - \epsilon$$

Based on several propositions and previous theorems [6] [5] [4], they [1] proved the Theorem below:

Theorem 5.3. *Suppose that player's utilities is depending on outputs and the players can compute f with a trusted mediator*

(1) *If $3k < n$, there is a k-resilient multiparty computation protocol with expected running time 2.*
(2) *If $2k < n$ and utilities are known, there is a k-resilient multiparty computation protocol with expected running time 2.*
(3) *If $k < n$ and utilities are known, and 1-way functions exist, there is a k-resilient multiparty computation protocol with constant expected running time (depends on utilities) and ϵ error probability that the cryptography is "broken"*

6 T-IMMUNITY

In large systems, there are certainly some agents who won't respond to incentives, perhaps because they have "unexpected" utilities, are irrational, or have faulty computers. Whatever the reason, it's important to design protocols that can tolerate these unanticipated behavior, to prevent these nonstandard players from affecting the payoffs of the users with standard utilities. This property [1] is somewhat like the Byzantine agreement in distributed computing (Good agents reach agreement despite up to t faulty agents).

Definition 6.1. A joint strategy $\vec{\sigma}$ is a t-immune if for all $T \subseteq N$ with $|T| \leq t$, all $\vec{\tau}_C \in \mathcal{S}_T$, and all $i \notin T$, we have

$$u_i(\vec{\sigma}_{-T}, \vec{\tau}_T) \geq u_i(\vec{\sigma})$$

.

Then we can combine this property with the k-resilient before [1].

Definition 6.2. A joint strategy $\vec{\sigma} \in \mathcal{S}$ is (k,t)-robust if for all $C, T \subseteq N$ such that $C \bigcap T = \emptyset$, $|C| \leq k$, and $|T| \leq t$, for all $\vec{\tau} \in \mathcal{S}_T$, for all $\vec{\phi}_C \in S_C$, for all $i \in C$ we have

$$u_i(\vec{\sigma}_{-T}, \vec{\tau}_T) \geq u_i(\vec{\sigma}_{N-(C \cup T)}, \vec{\phi}_C, \vec{\tau}_T)$$

We can found normal Nash Equilibrium is equivalent to (1,0)-robustness. In general, (k,t)-robust equilibrium don't exist, however, it can be obtained with the help of mediators. Since up to t players can send arbitrary values, they assume that getting a value that in some sense is close to the true function value is good enough.

Theorem 6.3. *Suppose that it can compute f with a trusted mediator, an output is acceptable if it is obtained by applying f to an input that is "close" to the true input, and players' utilities depends just on outputs:*

(1) *If* $3(t+k) < n$*, then, assuming each player never prefers not to learn the secret, there exists a practical (k, t)-robust mechanism without a mediator for the multiparty computation of* f *with an expected running time of 2 rounds.*
(2) *If* $3t + 2k < n$*, then, assuming each player strictly prefers learning the secret to not learning it, there exists a mechanism without a mediator that takes a parameter* β *and is a practical (k, t)-robust mechanism for the multiparty computation of* f *with an expected running time of 2 rounds.*
(3) *If* $2(t+k) < n$*, then, assuming each player never prefers not to learn the secret and cryptography, for all* $\epsilon > 0$*, there exists a practical* ϵ*-(k, t)robust mechanism for multiparty computation with an expected running time of 2 rounds.*
(4) *If* $2t + k < n$*, then, assuming each player never prefers not to learn the secret and cryptography, for all* $\epsilon > 0$*, there exists a practical* ϵ*-(k, t)-robust mechanism for multiparty computation with an expected running time of* $O(1/\alpha)$ *rounds.*

7 SIMULATING COMMUNICATION EQUILIBRIUM VIA CHEAP TALK

If we can get a good outcome with a mediator, can we get the same outcome without mediator. According to the Section 5.2, we can see Multiparty Computation protocols is a decent approach. In the economics literature, punishment strategy can be used to simulate Nash Equilibrium. In fact, we do not always need to assume the existence of a punishment strategy. According to Theorem 6.3, given that k and t satisfy a appropriate bounds, we can realize the punishment strategy by using verifiable secret sharing and the GMW protocol.
More precisely, first we can define the (k,t) punishment strategy [1]:

Definition 7.1. A joint strategy $\vec{\rho} \in \mathcal{S}$ is (k,t)-punishment strategy with respect to $\vec{\sigma}$ if for all $C, T, P \subseteq N$ such that C, T, P are disjoint, $|C| \leq k$, $|T| \leq t$, and $|P| > t$ for all $\vec{\tau} \in \mathcal{S}_T$, for all $\vec{\phi}_C \in S_C$, for all $i \in C$ we have

$$u_i(\vec{\sigma}_{-T}, \vec{\tau}_T) \geq u_i(\vec{\sigma}_{N-(C\cup T\cup P)}, \vec{\phi}_C, \vec{\tau}_T, \vec{\rho}_P)$$

For any coalition C of at most k players and any set T of nonstandard players, as long as more than t players use the punishment strategy and the remaining players play their component of $\vec{\sigma}$, all the players in C are worse off than they would be had everyone not in T played $\vec{\sigma}$. The idea is that the thread of having more than t players use their component of $\vec{\rho}$ is enough to stop players in C form deviating from $\vec{\sigma}$.
Cheap Talk is a economical mechanism which can simulate the mediator. First, we assume that each pair of agents has a secure private channel to to communicate. Assume Γ_{CT} is a private channel Cheap-Talk (CT) extension of the game Γ if the mediator in Γ_{CT} acts as a private channel between every pair of players [1]. This Cheap Talk specifically, can be implemented through cryptography with public channel. To prove the upper bound and lower bound of k, t and n, some proofs exploit techniques for Byzantine Agreement can be used.

8 CONCLUSION

(1) Equilibrium notions should be more robust, and take fault tolerance into account.
(2) Cryptography technique can be helpful in achieving equilibrium.
(3) All these mechanisms are based on Synchronous System, but Asynchronous Systems is a more common in real life.

REFERENCES

[1] Ittai Abraham, Danny Dolev, Rica Gonen, and Joe Halpern. 2006. Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC '06)*. Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/1146381.1146393
[2] David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2018. *A pragmatic introduction to secure multi-party computation.* Now, the essence of knowledge, 40–41.
[3] James W Friedman. 1971. A non-cooperative equilibrium for supergames. *The Review of Economic Studies* 38, 1 (1971), 1–12.
[4] Goldreich. 2004. *Foundations of Cryptography, volume 2.* Cambridge University Press.
[5] O. Goldreich, S. Micali, and A. Wigderson. 1987. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*. Association for Computing Machinery, New York, NY, USA, 218–229. https://doi.org/10.1145/28395.28420
[6] Joseph Halpern and Vanessa Teague. 2004. Rational secret sharing and multiparty computation. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. 623–632.
[7] Joe Kilian. 1988. Founding Crytpography on Oblivious Transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC '88)*. Association for Computing Machinery, New York, NY, USA, 20–31. https://doi.org/10.1145/62212.62215
[8] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. 1995. *Microeconomic theory*. Vol. 1. Oxford university press New York.
[9] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. 2007. Algorithmic game theory, 2007. *Book available for free online* (2007).
[10] Michael O. Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. http://eprint.iacr.org/2005/187 Harvard University Technical Report 81 talr@watson.ibm.com 12955 received 21 Jun 2005.
[11] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
[12] Yoav Shoham and Moshe Tennenholtz. 2005. Non-cooperative computation: Boolean functions with correctness and exclusivity. *Theoretical Computer Science* 343, 1-2 (2005), 97–113.